



Continuous Delivery

Contents

Welcome to Continuous Delivery for Puppet Enterprise.....	5
Key concepts.....	6
Working with Git branches in Continuous Delivery for PE.....	6
Understanding the Continuous Delivery for PE workflow.....	7
Using the feature branch workflow.....	10
Getting started guide.....	13
Release notes and known issues.....	17
Continuous Delivery for PE release notes.....	17
Continuous Delivery for PE known issues.....	45
Puppet Application Manager.....	47
Welcome to Puppet Application Manager (PAM).....	48
Release notes.....	50
PAM release notes.....	50
Known issues.....	62
Architecture overview.....	62
PAM system requirements.....	66
Component versions in PAM releases.....	79
Install PAM.....	80
Install Puppet applications using PAM on a customer-supported Kubernetes cluster.....	81
PAM HA online installation.....	83
PAM HA offline installation.....	87
PAM standalone online installation.....	90
PAM standalone offline installation.....	93
Automate PAM and Puppet application online installations.....	95
Automate PAM and Puppet application offline installations.....	97
Uninstall PAM.....	101
Working with Puppet applications.....	101
Install applications via the PAM UI.....	102
Update a license for online installations.....	103
Update a license for offline installations.....	103
Upgrade an automated online application installation.....	103
Upgrade an automated offline application installation.....	104
Maintenance and tuning.....	105
Upgrading PAM on a Puppet-supported cluster.....	106
Upgrading PAM on a customer-supported cluster.....	110
Backing up PAM using snapshots.....	112
Migrating PAM data to a new system.....	114
Disaster recovery with PAM.....	120
Troubleshooting PAM.....	121
Install.....	126
Continuous Delivery for PE architecture.....	126
Supported PE versions and browsers.....	129
Deploy Continuous Delivery for PE.....	130

Deploy Continuous Delivery for PE offline.....	131
Upgrade.....	133
Upgrade paths.....	134
Upgrade Continuous Delivery for PE.....	134
Upgrade Continuous Delivery for PE offline.....	134
Upgrade an automated installation.....	136
Migrate 3.x data to 4.x.....	136
Configure and integrate.....	141
Analytics data collection.....	142
Integrate with Puppet Enterprise.....	143
Integrate with source control.....	146
Configure job hardware.....	153
Job hardware requirements.....	154
Configure job hardware.....	154
Add job hardware capabilities.....	155
Configure global shared job hardware.....	156
Add repositories.....	156
Configure SAML.....	157
Configure LDAP.....	158
Configure SMTP.....	161
Set up external PostgreSQL.....	162
Load balancing.....	162
Advanced configuration.....	163
Manage workspaces and access.....	167
Manage workspaces.....	167
Manage access.....	169
Create a user account.....	169
Add users and groups.....	170
Permissions reference.....	171
Use the root console.....	173
Manage personal access tokens.....	175
REST API.....	175
Authenticate public APIs.....	176
REST API tutorial.....	176
OpenAPI specification.....	177
Common API workflows.....	178
Test and deploy Puppet code.....	184
Test Puppet code with jobs.....	185
Analyze the impact of code changes.....	192
Configure impact analysis.....	192
Generate impact analysis reports.....	194
Run impact analysis on fewer nodes.....	197
Impact analysis limitations.....	198
Construct pipelines.....	199
Construct pipelines in the web UI.....	201
Construct pipelines from code.....	204

Deploy Puppet code.....	215
Built-in deployment policies.....	216
Custom deployment policies.....	220
Deploy code manually.....	222
Deploy module code.....	222
Require approval for deployments to protected Puppet environments.....	223
Review node inventory.....	224
Review activity.....	227
Support and troubleshooting.....	228
Troubleshooting Continuous Delivery for PE.....	228
Getting support.....	234
Copyright and trademark notices.....	239

Welcome to Continuous Delivery for Puppet Enterprise

Continuous Delivery for Puppet Enterprise (PE) is a tool for streamlining and simplifying the continuous integration and continuous delivery of your Puppet® code. Continuous Delivery for PE offers a prescriptive workflow to test and deploy Puppet code across environments.

Note: Puppet Application Manager (formerly called the platform admin console) deploys and manages Continuous Delivery for PE and other Puppet applications. Learn more in the [Puppet Application Manager](#) section of the documentation.

To harness the full power of Puppet Enterprise® (PE), you need a robust system for testing and deploying your Puppet code. Continuous Delivery for PE offers prescriptive, customizable workflows and intuitive tools for Puppet code testing, deployment, and impact analysis, helping you ship changes and additions with speed and confidence.

Important: Before you use the product and its documentation, review the [Copyright and trademark notices](#) on page 239.

Learn to use Continuous Delivery for PE	Other useful links
<p>Before you install</p> <ul style="list-style-type: none"> Release notes Known issues System requirements Supported PE versions and browsers on page 129 <p>Learn the basics</p> <ul style="list-style-type: none"> Getting started guide on page 13 Key concepts on page 6 <p>Install</p> <ul style="list-style-type: none"> Install Puppet Application Manager Install Continuous Delivery for PE If you are upgrading to the 4.x series from a version in the 3.x series: Migrate 3.x data to 4.x on page 136 <p>Configure and manage access</p> <ul style="list-style-type: none"> Configuring job hardware and integrate with source control and PE. User permissions and groups Creating and managing workspaces <p>Continuously deliver Puppet code</p> <ul style="list-style-type: none"> Create jobs to test code Preview potential impacts of new code Construct pipelines on page 199 Deploy Puppet code on page 215 	<p>Docs for related Puppet products</p> <ul style="list-style-type: none"> Puppet Enterprise Open source Puppet Puppet Development Kit <p>Why and how people use Continuous Delivery for PE</p> <ul style="list-style-type: none"> Critical Considerations for Continuous Delivery ebook <p>Get support</p> <ul style="list-style-type: none"> Support portal and knowledge base Upgrade your Support plan <p>Share and contribute</p> <ul style="list-style-type: none"> The Puppet community Puppet Forge Puppet's open source projects on GitHub

To send us documentation feedback or let us know about a docs error, use the feedback form at the bottom of each page.

- [Key concepts](#) on page 6
- [Getting started guide](#) on page 13

If you're using Continuous Delivery for Puppet Enterprise (PE) for the first time, you'll need to perform some initial workspace setup tasks and learn how to use the core features.

Key concepts

- [Working with Git branches in Continuous Delivery for PE](#) on page 6

Continuous Delivery for Puppet Enterprise (PE) watches for Puppet code changes you make in Git and helps you deploy those changes to your environment node groups. You and Continuous Delivery for PE work together, so it's important to understand what you need to do in Git, and what Git work Continuous Delivery for PE handles for you.

- [Understanding the Continuous Delivery for PE workflow](#) on page 7

Continuous Delivery for Puppet Enterprise (PE) helps you to deliver changes to your organization's infrastructure-as-code. Use this workflow to develop and deploy changes with Continuous Delivery for PE.

- [Using the feature branch workflow](#) on page 10

Use the feature branch workflow to safely and efficiently move new code from an isolated development environment, through initial testing and validation, and then into your organization's process for staging, review, and promotion to production. Adapt the steps of this Continuous Delivery for Puppet Enterprise (PE) workflow as needed for your team's requirements and best practices.

Working with Git branches in Continuous Delivery for PE

Continuous Delivery for Puppet Enterprise (PE) watches for Puppet code changes you make in Git and helps you deploy those changes to your environment node groups. You and Continuous Delivery for PE work together, so it's important to understand what you need to do in Git, and what Git work Continuous Delivery for PE handles for you.

Continuous Delivery for PE automates management of Git branches for Puppet environments, letting you focus on Git workflows for developing and deploying changes to your code.

You and your team own the `main` branch in Git, as well as any other development-focused branches you create, such as `feature`, `hotfix`, or `release` branches. In simple Git workflows, the `main` branch is designated as the branch to create pull requests against, tag releases off of, and treat as the leading edge of new development.

You must create new environment branches when you first [set up environment node groups](#). Environment branches are long-lived Git branches that control which code version is available in corresponding Puppet environments. Continuous Delivery for PE owns environment branches after you create them; it deploys changes to them and keeps them updated. You and your team don't change these branches directly because manual changes to environment branches are overwritten when an automated deployment occurs.

While you focus on committing to the `main` branch and merging pull requests, Continuous Delivery for PE works in your environment branches on your behalf. It makes sure your commits or changes are tested, deployed, and promoted to your environment node groups in accordance with your control repo pipelines and your manual deployment directives.

Continuous Delivery for PE also uses a small set of Git branches for bookkeeping, which are not part of the standard Git process. These branches are visible to users, and must be protected so that users cannot modify them. Continuous Delivery for PE manages these bookkeeping branches automatically.

GitLab users: *Pull request* is another term for *merge request*. For the sake of simplicity and consistency, the term *pull request (PR)* is used in the Continuous Delivery for PE web UI and documentation.

Organizing Puppet content in your source control repository

Most Puppet content that developers interact with is infrastructure-as-code, which is versioned and committed to a Git source-control repository.

When using Continuous Delivery for Puppet Enterprise (PE), every Git repository, regardless of its type or contents (control repo or module), works the same way and has the same key components:

- A main branch tracking the latest developing code.
- Tags to identify meaningful code versions.
- Short-lived feature branches created and deleted during the code development lifecycle.

Tip: There are many [Git 101](#) tutorials online that describe how to interact with standard Git repositories of this type and explain the basic branching and merging process.

Required repository permissions vary by source control provider, and these are configured when you [Integrate with source control](#) on page 146.

Control repos

In its purest form, a control repo is a hub for Puppet configuration content. Usually, however, the control repo serves as a hub for modularized content and contains some directly embedded content. Typically, embedded content includes the `role` module, the `profile` module, and Hieradata. In a pure control repo, this content is modularized rather than embedded.

The control repo contains a Puppetfile, which is the switchboard or ledger that makes the control repo a hub. The Puppetfile is a file that enumerates all of the modular content Puppet incorporates and makes available. Reading the Puppetfile can convey a good sense of all the Puppet content being used at a given site or Puppet implementation.

Tip: Visit the PE documentation for more information about [Managing environment content with a Puppetfile](#).

Depending on how much content is embedded in the control repo, the control repo might also contain some Puppet configuration files and possibly some directories for embedded content.

Puppet module repos

A Puppet module is a collection of content laid out in a particular way. A module can contain desired state code, Puppet Bolt® tasks or plans, Puppet extensions, or a variety of other things. Modules are the standard content container for everything Puppet.

Puppet knows how to consume modules automatically, provided they are placed somewhere in its `modulepath`.

Other non-module repos

Sometimes it's useful to partition off certain types of Puppet content for development purposes, even if the content is not technically a Puppet module. A prime example of this is Hieradata. Hieradata can be kept in a dedicated Git repository and evoked by the control repo using the Puppetfile, just like any standard Puppet content.

There are other use cases for partitioning non-module Git repositories, but they are rare. The commonality is that all such content is committed to a Git repository and referenced in the control repo's Puppetfile.

Important: Typically, Puppet doesn't consume non-module content automatically; therefore, correct consumption of this content requires some kind of configuration in one of the control repo's configuration files.

Understanding the Continuous Delivery for PE workflow

Continuous Delivery for Puppet Enterprise (PE) helps you to deliver changes to your organization's infrastructure-as-code. Use this workflow to develop and deploy changes with Continuous Delivery for PE.

Tip: Before you begin, review [Working with Git branches in Continuous Delivery for PE](#) on page 6.

Workflow phase 1: Developing changes

The general change development process is always the same regardless of whether the content lives in the control repo, a module repo, or a non-module repo. The change development process is driven through Git.

This is the general change development process:

1. Using Git, create a feature branch from the main branch. This feature branch is where you'll develop your change.
2. Work on your change. Edit files, run tests, and do anything else necessary to actualize, in code, the change you want to make. Which files you edit and how you test your change locally depends on the kind of content you're working on.
3. Using Git, commit all the files you've changed to your feature branch. You may commit frequently as you iterate, or you may not commit until you're fully satisfied with your work. It's up to you how much you involve Git in your process up to this point.
4. Submit a pull/merge request to merge your feature branch into the main branch. This may trigger a Continuous Delivery for Puppet Enterprise (PE) pipeline run (see **Note on continuous integration**, below).
5. After following your organization's CI or review process, you or someone else merges your feature branch into the main branch. Once the merge is complete, your change is ready to be put on the path to production, and you can safely delete your feature branch.

Important: Once merged into the main branch, your change is finished. However, a finished change is not the same as a deployed change. Finished changes are ready to be tested and deployed using the deployment process described in [Workflow phase 2: Deploying changes](#) on page 8.

Note on continuous integration

When you submit a pull/merge request against a Puppet content repo (a control repo, a module repo, or a non-module repo), the continuous integration (CI) pipeline configured in Continuous Delivery for Puppet Enterprise (PE) for that content repo runs and reports any success or failure results in your pull/merge request.

Pipelines in Continuous Delivery for PE are not cleanly separated into CI and CD. Instead, you see a single continuous pipeline. However, generally, everything in a Continuous Delivery for PE pipeline that occurs before the *pull request gate* is the CI portion of the pipeline, and is (usually) focused on validating that proposed changes pass requisite acceptance tests or governance requirements.

The CI portion of a Continuous Delivery for PE pipeline, therefore, aids the development workflow.

Related information

[Working with Git branches in Continuous Delivery for PE](#) on page 6

Continuous Delivery for Puppet Enterprise (PE) watches for Puppet code changes you make in Git and helps you deploy those changes to your environment node groups. You and Continuous Delivery for PE work together, so it's important to understand what you need to do in Git, and what Git work Continuous Delivery for PE handles for you.

[Construct pipelines](#) on page 199

Pipelines drive the continuous aspect of Continuous Delivery for Puppet Enterprise (PE). Constructing a pipeline involves defining work that must happen to ensure every new line of Puppet code is ready for deployment. Once your pipeline is set up, this work happens automatically each time the pipeline is triggered.

Workflow phase 2: Deploying changes

Once a proposed change is merged into the main branch of a Puppet content repo, it is time to deploy the new content version (containing your change) from the main branch.

Generically, deploying a change involves selecting a version of content, selecting a target environment, and applying the selected change to the selected environment. To do this in Continuous Delivery for Puppet Enterprise (PE), you:

1. Pick a version.
2. Pick a target environment.
3. Optionally, set some deployment parameters.
4. Click **Deploy**.

A *version* is any commit in the history of the main branch. There are no chronological requirements for picking a version to deploy. Deploying a version does not change the contents of the main branch or affect it in any way.

Environments

An *environment* is a set of nodes to which a content version can be deployed. You configure environments by [creating environment node groups](#) in the PE console. You must configure environment node groups prior to deploying any content versions. At minimum, the environment you want to target must be defined as an environment node group.

When defining environment node groups:

- Each node can only belong to a single environment node group.
- You can name the environment node groups anything that makes sense to your organization.
- Each environment node group must be assigned a unique `Puppet environment` value. Functionally, this relates to how Continuous Delivery for PE performs bookkeeping of the environment's content in Git (using the bookkeeping branches mentioned in [Working with Git branches in Continuous Delivery for PE](#) on page 6). Unique values ensure simple bookkeeping and that deployments do not affect nodes they are not supposed to.

Deploying changes from control repos

Because a control repo contains a Puppetfile (the switchboard or ledger to which all modularized content is attached), the way modularized content is deployed in a target environment depends on the Puppetfile deployed to that environment.

Modularized content can be attached to a Puppetfile in two ways:

- **Statically:** Modularized content cannot be independently deployed or updated.
- **Dynamically:** Modularized content can be deployed independent of the control repo using a different Continuous Delivery for PE pipeline or deployment action.

Static content can only be updated by developing a change directly in the control repo (as opposed to a module repo), which modifies the Puppetfile to update the static module and deploy the change to the control repo.

Tip: Visit the PE documentation for more information about [Managing environment content with a Puppetfile](#).

Deploying changes from module repos

To be able to deploy module changes from module repos, the control repo version deployed to the target environment must have a Puppetfile that defines this modularized content dynamically. If the target environment does not have such a Puppetfile, it is not possible to independently deploy the module changes.

Pipelines, or the path to production

Many organizations have a semi-linear sequence of target environments to which a change is deployed, terminating at the highest-value target environment. This final target is frequently called *production*.

Continuous Delivery for PE pipelines use automation to reflect this deployment sequence. Pipelines are a progression of deployments and other actions that, when triggered, run in the defined manner configured by the pipeline creator, including pauses, termination, or continuation after each step.

Pipelines can contain actions other than deployments. As mentioned in [Note on continuous integration](#), the first part of a Continuous Delivery for PE pipeline up to the *pull request gate* is typically CI validation and testing actions, rather than deployment actions. Deployment actions are usually not placed in the pipeline until after the *pull request gate*.

Pipelines are considered *continuous* because of when they run or how they are triggered. Usually, pipelines automatically start when new changes are merged to the main branch of a content repository (or when a pull request is submitted to the main branch).

It is common for deployment pipelines to automatically deploy validated changes to some pre-production environments, and then pause or wait for human approval before continuing to deploy to more sensitive, higher-tiered environments.

Using the feature branch workflow

Use the feature branch workflow to safely and efficiently move new code from an isolated development environment, through initial testing and validation, and then into your organization's process for staging, review, and promotion to production. Adapt the steps of this Continuous Delivery for Puppet Enterprise (PE) workflow as needed for your team's requirements and best practices.

Workflow phase 1: Develop and test code changes

The first phase of the feature branch workflow involves writing new code in a feature branch, then testing and validating that code.

Before you begin

Create a [regex branch pipeline](#) on page 202 for the control repo or module repo you're working on. This pipeline can be as simple or complex as you need it to be, but at minimum it must:

- Use commits as a pipeline trigger.
- Contain a deployment that uses the feature branch deployment policy.
- (Strongly recommended) Include jobs that perform syntax validation tests before the pipeline's deployment stage.

This is an example of a valid regex branch pipeline in the Continuous Delivery for Puppet Enterprise (PE) web UI:

The screenshot shows the 'Pipelines' management interface. At the top, there's a 'Pipelines' header with a 'Manage pipelines' link. Below that, a dropdown menu shows 'regex'. The pipeline trigger is set to 'Pull Request, Commit'. The pipeline consists of two stages:

- Lint/Parser validation**: Contains two jobs, both marked with a checkmark:
 - Job - control-repo-puppetfile-syntax-validate
 - Job - control-repo-template-syntax-validate
- Deploy feature environment**: Contains one job:
 - Policy: Feature branch policy
 - Instance: cdpe-delivery
 - Environment: n/a

Below the first stage, there is a green progress indicator and the text 'Auto promote: All succeeded'. To the right of this indicator is a 'Promote' button with a right-pointing arrow.

This is the same pipeline formatted as code:

```
pipelines:
  /feature_.*/:
    triggers:
```

```

- "PULL_REQUEST"
- "COMMIT"
stages:
- name: "Lint/Parser validation"
  steps:
  - type: "JOB"
    name: "control-repo-puppetfile-syntax-validate"
    concurrent_compilations: 0
    all_deployments: false
  - type: "JOB"
    name: "control-repo-template-syntax-validate"
    concurrent_compilations: 0
    all_deployments: false
  auto_promote: "all_succeeded"
- name: "Deploy feature environment"
  steps:
  - type: "DEPLOYMENT"
    name: "Feature branch deployment on cdpe-delivery"
    policy:
      name: "cd4pe_deployments::feature_branch"
      concurrent_compilations: 0
      all_deployments: false
      pe_server: "cdpe-delivery"
    auto_promote: false

```

1. In your source control system, navigate to the control repo or module repo you want to update.
2. Create a feature branch by create a new branch based on the main branch and naming it `feature_<BRANCHNAME>`. The `<BRANCHNAME>` can be a ticket number, fix description, or feature name.

Important: Continuous Delivery for PE automatically recognizes feature branches by their names. By default, the software uses `feature_.*` as the regular expression that triggers regex pipelines, so it's important that your feature branch name has the `feature_` prefix.

Tip: If you need to use a different naming convention for your control/module repo feature branches, you can change the regular expression under **Manage pipelines** in the Continuous Delivery for PE web UI.

3. On your feature branch, make your code changes.
4. When your work is complete, commit the changes to the feature branch and push the commit to your source control system. This commit triggers your regex branch pipeline. You can monitor the pipeline's progress in the Continuous Delivery for PE web UI.

When your pipeline is triggered, Continuous Delivery for PE runs the jobs and impact analysis tasks you set up. If the pipeline's promotion conditions are met (for example, if all jobs succeed), the pipeline starts a feature branch deployment.

How does a feature branch deployment work? Using Code Manager, Continuous Delivery for PE deploys your committed code to a Puppetenvironment with the same name as the branch that triggered the pipeline. If an environment with this name doesn't already exist (as is most likely the case), Continuous Delivery for PE creates it.

5. When the feature branch deployment is complete, run Puppet on a test node, specifying the special environment created by Continuous Delivery for PE. This Puppet run applies your changes to the test node so you can review them. You can run Puppet from the PE console or the command line:
 - a) From the PE console, navigate to the **Run Puppet** page and create a job with the following settings, and click **Run job**:
 - Job description: Enter a description for your test run.
 - Environment: Select **Select an environment for nodes to run in** and choose the environment that matches your feature branch name.
 - Schedule: Now.
 - Run Mode: Leave these options unchecked.
 - Inventory: Select **Node list** and add the name of your test node.
 - b) From the command line, run:

```
puppet job run --nodes <TEST_NODE_NAME> --environment
  <feature_BRANCHNAME>
```

Tip: [Running Puppet on demand from the CLI](#) provides more information on the `puppet job run` command.

6. When the Puppet run is complete, navigate to the test node and review the changes. If further work is needed, repeat steps 3 through 5 until your code is ready for deployment to production. When you're satisfied with your new code, move on to [Workflow phase 2: Review and merge to production](#) on page 12.

Workflow phase 2: Review and merge to production

In the second phase of the feature branch workflow, new code is reviewed, merged into the main branch, and deployed to production. Along the way, Continuous Delivery for Puppet Enterprise (PE) provides checks and safeguards to ensure new code is only sent to production nodes after being fully vetted.

Before you begin

Complete [Workflow phase 1: Develop and test code changes](#) on page 10 and make sure your main branch pipeline:

- Uses pull requests as a pipeline trigger.
- Includes a pull request (PR) gate.

1. In your source control system, create a pull request from your `feature_<BRANCHNAME>` branch to the main branch.
2. The pull request triggers your main branch pipeline to test your PR code against the jobs and impact analysis report tasks included in the pipeline up to the PR gate. You can monitor the pipeline's progress in the Continuous Delivery for PE web UI.
You can also see whether each pipeline stage succeeded or failed on the pull request page in your source control system.
3. Your team's stakeholders can review the results of the jobs and impact analysis reports generated by the pipeline. If the PR is approved, merge it into the main branch.

Note: After merging the PR, you can delete the original `feature_<BRANCHNAME>` branch from your source control system.

Merging a PR creates a commit to the main branch and automatically re-triggers the main branch pipeline. The pipeline starts over from the top, re-running all the jobs and impact analysis tasks to ensure the newly updated main branch code is fully tested and vetted. However, because a commit triggered this pipeline run, the pipeline proceeds past the PR gate and performs any tasks included in the pipeline after the PR gate, such as additional jobs or deployment to a staging environment.

4. Once the code is deployed to a staging environment, perform additional testing by running Puppet against test or staging nodes. Make sure to specify the staging environment. The scope of this testing is determined by your team or company best practices.
5. When testing is complete and the code changes are approved, promote the code to the production environment.

Usually, the main branch pipeline does not auto-promote code to the production environment. To trigger the final stage, a stakeholder must click **Promote** on the pipeline page in the Continuous Delivery for PE web UI. This kicks off deployment to the production environment unless the production environment is a protected environment.

Deployments to protected environments require review and approval from a member of a designated approval group before proceeding. These deployments remain in a pending state until an approval decision is provided. [Require approval for deployments to protected Puppet environments](#) on page 223 explains how to set up approval groups.

6. If necessary, run Puppet to apply the new code to the nodes in your production environment. Depending on which deployment policy you've used, a Puppet run might be included in the deployment Continuous Delivery for PE performs. Otherwise, you can trigger Puppet manually or wait for the next scheduled Puppet run.

When the Puppet run is complete, your code changes are officially deployed to production.

Getting started guide

If you're using Continuous Delivery for Puppet Enterprise (PE) for the first time, you'll need to perform some initial workspace setup tasks and learn how to use the core features.

Soon, you'll have a more streamlined, powerful, and flexible Puppet code delivery process.

Part 1: Install Puppet Application Manager and deploy Continuous Delivery for PE

First, install Puppet Application Manager (PAM) and then configure and deploy Continuous Delivery for Puppet Enterprise (PE).

1. Make sure your system meets the [PAM system requirements](#) on page 66.
2. Follow the instructions for your environment to [Install PAM](#) on page 80.
3. [Deploy Continuous Delivery for PE](#) on page 130.

Part 2: Create your user account and set up a workspace

If Continuous Delivery for Puppet Enterprise (PE) were a city, workspaces would be neighborhoods within that city. Your workspace is where you store and access resources such as control repos, pipelines, and jobs. When you're ready to collaborate, you can invite team members to join your workspace.

1. If you have not done so already, navigate to the Continuous Delivery for PE web UI URL (`https://<HOSTNAME>`) you received at the end of [Part 1](#).
2. Create your personal user account.
 - a) On the login page, click **Create an account**.
 - b) Fill in the registration form and create a username and password.
 - c) Click **Sign Up**.

Note: For almost all tasks you'll perform in Continuous Delivery for PE, you'll use your individual user account, not the root account. You only use the root account for special administrative tasks such as installation, designating super users, and deleting users.

3. Set up a workspace.
 - a) On the **Choose a workspace** screen, click **Add new workspace**.
 - b) Enter a name for your workspace and click **Create workspace**.

Part 3: Set up integrations

Next, integrate with your Puppet Enterprise (PE) instance and the source control system where you keep your Puppet code.

1. Follow the instructions to [Integrate with Puppet Enterprise](#) on page 143.
2. Follow the instructions to [Configure impact analysis](#) on page 192.
3. Follow the integration instructions for your source control system:
 - [Azure DevOps Services](#)
 - [Bitbucket Cloud](#)
 - [Bitbucket Server](#)
 - [GitHub](#)
 - [GitHub Enterprise](#)
 - [GitLab](#)

Part 4: Configure job hardware

Configure a node running a Puppet agent to be a job hardware server, where your code is tested before deployments.

1. Install a Puppet agent on the node you plan to use as job hardware. Instructions for [Installing agents](#) are in the Puppet Enterprise (PE) documentation.
2. Make sure your [Continuous Delivery user role in PE](#) has permission to run the `cd4pe_jobs::run_cd4pe_job` task.
3. Install the `puppetlabs-cd4pe_jobs` module from the Forge, which is required to run Continuous Delivery for PE jobs on your node.
 - a) Add the `puppetlabs-cd4pe_jobs` module to:
 - The Puppetfile for the production environment on the PE primary server that manages the agent node you've selected as job hardware.
 - The Puppetfile on the main branch of the control repo you will add to Continuous Delivery for PE in [Part 5](#).

A sample Puppetfile entry:

```
mod 'puppetlabs-cd4pe_jobs', '1.6.0'
```

Tip: Visit the PE documentation for more information about [Managing environment content with a Puppetfile](#).

- b) Run `puppet code deploy production --wait` to deploy the updated code to the production environment.
4. The job hardware server must be able to use the [pre-built jobs](#) included in Continuous Delivery for PE. Since these jobs are Docker-based, you must install and configure Docker on the node. You can find installation files and instructions on the [Docker website](#).
5. Tell Continuous Delivery for PE that this node is ready to be used as job hardware for Docker-based jobs by assigning it to a hardware capability. Capabilities organize your job hardware servers and ensure that jobs run on hardware with the right characteristics. Continuous Delivery for PE automatically creates a **Docker** hardware capability for you.
 - a) In the Continuous Delivery for PE web UI, click **Hardware**.
 - b) Locate the **Docker** capability and click **Edit**.
 - c) Select the PE instance that manages the node you've selected as job hardware, and then select your job hardware node. The selected node appears in the **Hardware with this capability** list.
 - d) Click **Save**.

Your job hardware node is configured and ready for use. You'll use it in [Part 8](#) when you run your pipeline for the first time.

Part 5: Add a control repo

A control repo in Continuous Delivery for Puppet Enterprise (PE) tracks the changes made on the active development branch of your source control system. When adding a control repo to Continuous Delivery for PE, it's important to connect the `main` Git branch.

Adding a repo in Continuous Delivery for PE configures a webhook to the repository in your source control system. The webhook reports new commit activity on the repository to Continuous Delivery for PE, enabling you to track code changes and take action (such as triggering pipelines or running individual tests on the new code).

1. In the Continuous Delivery for PE web UI, click **Control repos** and then click **Add control repo**.
2. Select the source control provider and repository.
3. Your repository's `main` branch is automatically selected, if one exists. If your control repo does not have a `main` branch, select **Create a main branch from an existing branch** and select your active development branch as the basis for the new `main` branch. Continuous Delivery for PE creates a `main` branch for you based on the existing branch you choose.



CAUTION: When working with Continuous Delivery for PE, only commit to the `main` branch and any feature branches (which are eventually merged back into the `main` branch). Do not push code changes to any other Git branches; this can create conflicts with Continuous Delivery for PE workflows.

4. Optional: Edit the control repo's **Display name** name as it appears in the Continuous Delivery for PE web UI. The name can contain only numbers, letters, dashes, and underscores.
5. Click **Add control repo**. The control repo appears in the list on the **Control repos** page.
6. Make sure you have added the `puppetlabs-cd4pe_jobs` module to the Puppetfile on the `main` branch of your control repo, as explained in [Part 4](#).

Part 6: Set up a pipeline

Continuous Delivery for Puppet Enterprise (PE) pipelines are made up of *stages* and *tasks*. Tasks include jobs to test code, deployments, and impact analysis. Stages group tasks into a series of sequential phases.

1. In the Continuous Delivery for PE web UI, click **Control repos** and click the name of the control repo you added in [Part 5](#).
2. You'll create a pipeline in the **Pipelines** section of the page. Click **Add default pipeline**.

Continuous Delivery for PE automatically creates a default pipeline for your `main` branch. This pipeline has three stages:

- The **Code validation stage** includes two tasks (jobs to test Puppet code).
- The **Impact analysis stage** includes an impact analysis task with a pull request gate (more on this later).
- The **Deployment stage**, which is where you'll add deployment instructions in [Part 8](#).

Part 7: Set up an environment node group

Next, set up a small environment node group to use for deployment testing and to demonstrate how Continuous Delivery for Puppet Enterprise (PE) deploys new code to PE-managed nodes.

This is only for testing. You can change the configuration later.

You can learn more [About environments](#) in the Puppet documentation and learn about [Managing environments with a control repository](#) in the PE documentation.

1. In your Git repository, create a new branch called **cd4pe_testing**. This represents the environment node group for the test.

2. On your PE primary server, run:

```
puppet-code deploy cd4pe_testing
```

Tip: For more information about the `puppet-code` command, read about [Triggering Code Manager on the command line](#) in the PE documentation. You can also use webhooks and scripts to trigger deployments.

3. In the PE console, click **Node groups**.
4. Click **Add group** and create a new node group with these specifications:
 - **Parent name:** Select **All Environments**
 - **Group name:** Enter `CD4PE test group`
 - **Environment:** Select `cd4pe_testing`
 - **Environment group:** Select this option.
 - **Description:** Enter any description, such as `For Continuous Delivery for PE testing`.
5. Click **Add**.
6. In the node groups list, click your new **CD4PE test group**. In the **Rules** tab, pin two or three test nodes to the node group. Make sure these nodes are not assigned to any other node groups in your PE installation. For more information, refer to [Add nodes to a node group](#) in the PE documentation.

Part 8: Deploy changes to your nodes

Once you've added a deployment to your pipeline, you can automatically move code changes to your nodes according to the deployment conditions you've set.

1. In the Continuous Delivery for PE web UI, in your control repo's pipeline, click **Add a deployment**.
2. Select your PE instance and then select the **CD4PE test group** node group you created in [Part 7](#).
3. Select **Direct deployment policy**. Don't set special conditions for this deployment.
4. Click **Add deployment to stage**. On the confirmation screen, click **Done**.
5. To trigger the deployment, you need to create a change to pass through to production. On the main branch of your Git repository, make a code change, such as updating a package version. Commit the change and then return to the Continuous Delivery for PE web UI to watch your pipeline in action.

When triggered by your commit, the pipeline automatically runs tests on your code, skips over the impact analysis stage (which isn't currently configured), and then stops, because the pipeline is configured to not automatically promote into the Deployment stage. You must click **Promote** to continue the pipeline run and launch the deployment. The **Events** section of the page logs the results of the pipeline run.

Part 9: Create an impact analysis report

Impact analysis reports show the potential impact that new Puppet code will have on the nodes and resources you're managing with Puppet Enterprise (PE). In Continuous Delivery for PE, you can add impact analysis tasks to your pipelines and you can generate impact analysis reports on demand.

Here, you'll generate an on-demand impact analysis report to review how a change impacts the nodes in the **CD4PE test group** node group you created in [Part 7](#).

1. Create a change for the report to analyze.
 - a) In your Git repository, create a feature branch from your main branch.
 - b) Make a code change on the feature branch, such as updating a package version.
 - c) Commit the change on the feature branch.
2. In the Continuous Delivery for PE web UI, click **Manual actions** and select **New impact analysis**.
3. In the **New impact analysis** window, select your feature branch, and select your new commit. Select your PE instance and the **CD4PE test group** node group.

4. Click **Analyze** to generate the report.

Continuous Delivery for PE generates a new catalog containing your commit and compares this new catalog to the current catalog for the nodes in the **CD4PE test group**.

5. Click **View impact analysis**. The report shows how the change on your feature branch would impact your nodes and resources if you merged it into the main branch.

Congratulations! By completing the getting started guide for Continuous Delivery for PE, you've become familiar with some of Continuous Delivery for PE's core features, and you have a basic understanding of how Continuous Delivery for PE helps you deploy Puppet code and preview the impact of code changes.

Release notes and known issues

New features, enhancements, resolved issues and known issues for the Continuous Delivery for Puppet Enterprise 4.x release series.

Security and vulnerability announcements are posted on [Puppet's Security and vulnerability announcements page](#).

- [Continuous Delivery for PE release notes](#) on page 17

These are the new features, enhancements, resolved issues, and deprecations for the Continuous Delivery for Puppet Enterprise (PE) 4.x release series.

- [Continuous Delivery for PE known issues](#) on page 45

These are the known issues for the Continuous Delivery for Puppet Enterprise (PE) 4.x release series.

Continuous Delivery for PE release notes

These are the new features, enhancements, resolved issues, and deprecations for the Continuous Delivery for Puppet Enterprise (PE) 4.x release series.

Note: [Migrate 3.x data to 4.x](#) on page 136 explains how to upgrade to the Continuous Delivery for PE 4.x series from a version in the 3.x series.

Version 4.35.0

Released 29 April 2025.

Added in this release:

- **Document how to set up HTTPS access for ADO via OAuth through the API.** Added documentation with [instructions on how to set up HTTPS access for ADO via OAuth through the REST API](#).
- **Security headers.** Added a more robust Content Security Policy and X-Frame-Options to Continuous Delivery for PE web server headers.

Resolved in this release:

- **LDAP user sessions and tokens return 403 until re-login.** Fixed an issue where LDAP user sessions and tokens would return 403 until re-login by improving logging during the background LDAP group sync task, as well as for LDAP user logins.
- **Migrate inventory UI from insights-charts to recharts.** Resolved [WS-2022-0322](#) by replacing the chart library.

Security notices:

- **CVE-2024-45337, CVE-2025-22869.** Updated `golang.org/x/crypto` to address these vulnerabilities.
- **CVE-2024-45338.** Updated `golang.org/x/net` to address this vulnerability.
- **CVE-2025-22868.** Updated `golang.org/x/oauth2` to address this vulnerability.

Version 4.34.0

Released 30 January 2025.

Resolved in this release:

- **Editing control repos that reference GitHub repos can cause deployments and Impact Analyses to fail with "RepositoryBranchRequest.repoId cannot be null or empty".** Fixed an issue where editing a GitHub control repo would cause deployments/IA runs to fail with an error if the repo was part of an organization rather than belonging to an individual user.
- **Impact Analysis fails with: Cannot invoke "String.endsWith(String)" because "filename" is null.** Fixed an issue where renaming files in an Azure DevOps repository (control repo or module) would cause IA runs to fail.

Security notices:

- **CVE-2023-39417 and CVE-2023-39418.** Updated PostgreSQL to 14.15 to address these vulnerabilities.

Version 4.33.0

Released 30 October 2024.

New in this release:

- **Added ability to edit a repo.** Added an edit button to the control repo and module list allowing you to update the owner, displayowner, or provider for that repo. An associated PATCH method was added to the `ControlRepoV1` and `ModuleV1` API endpoints to accomplish the same task.

Resolved in this release:

- **Updated Forgot Password screen.** Fixed the wording on the **Forgot Password** screen to enhance security.
- **Updated create Modules and create Control Repos API responses.** Fixed the API responses for create **Modules** and create **Control Repos** so they are consistent.
- **Alignment on Groups 'All permissions' check box.** Fixed an issue with the **All permissions** check box on the **Groups** permissions screen.

Security notices:

- **Security fixes.** Fixed several minor issues to improve overall security for Continuous Delivery for PE.

Version 4.32.0

Released 29 August 2024.

New in this release:

- **Postgres repack schedule is now configurable.** The schedule used to run the Postgres repack is now configurable. Use the **pg_repack Cronjob Run Time** setting in the Puppet Application Manager UI to configure this.
- **Added option to disable object store related objects.** Added a new option in the PAM console to disable the in-cluster object store. This is available once migration of the store data to Postgres is complete, which happens on the first upgrade to version 4.26.0 or later.
- **Added API endpoint to retrieve list of users.** Added a new API endpoint, `/v1/users/`, to retrieve a list of users.
- **Added API endpoint to retrieve user details.** Added a new API endpoint, `/v1/users/{userId}`, to retrieve a details about users.

Resolved in this release:

- **Security vulnerability with the username and password in URL path.** Fixed an issue with a security vulnerability that involved an API sending sensitive information as part of the URL path. We have replaced this call with a new endpoint that passes sensitive information as part of a request body in a more secure manner.

- **Prevent regex pipelines having impact analysis stages.** Fixed an issue where you could set an impact analysis stage on a regex pipeline when constructing the pipeline as code. Impact analysis stages are properly prevented on regex pipelines now.

Security notices:

- **Cross Origin Resource Sharing (CORS): Arbitrary Origin Trusted.** Resolved a security vulnerability related to Cross-Origin Resource Sharing (CORS). An Access-Control-Allow-Origin header is no longer set to the request's origin header on API calls, preventing any unintended exposure of resources to untrusted origins.

Version 4.31.0

Released 10 July 2024.

New in this release:

- **Data retention.** Added two new configuration options for data retention that allow you to [set a retention period](#) for historical pipeline and value reporting data.
- **Added a new PostgreSQL image.** Added a new PostgreSQL image which includes `pg_repack` for database maintenance. `pg_repack` runs daily to maintain the database.

Resolved in this release:

- **Nodes section is blank and has errors after restart.** Fixed an issue where the query service token was not properly rotated, causing the **Nodes** section to be blank and have errors. The query service token is now properly rotated even if Continuous Delivery for PE is not running when the rotation was scheduled, ensuring that node data is fetched properly.
- **Puppet Application Manager (PAM) trace logging sometimes failed for Continuous Delivery for PE jobs with UnknownFormatConversionException exceptions.** Fixed an issue where a percentage sign (%) in file names caused an issue when writing to Continuous Delivery for PE logs.

Version 4.30.2

Released 14 June 2024.

Resolved in this release:

- **Pipeline error: Cannot invoke "com.distelli.models.ControlRepoId.getDomain()" because "controlRepoId" is null.** Fixed an issue where custom deployments could not be used if the deployment was added to a pipeline through the UI.

Version 4.30.1

Released 22 May 2024.

Resolved in this release:

- **Update default facts displayed on Estate Reporting Node Table.** Updated **Estate Reporting Node** table on the **Nodes** screen to change the default facts displayed. Previously the table displayed the **ipaddress** and **operatingsystem** fact values in the two rightmost columns. These are legacy facts and those two columns have been updated to show the modern **networking.ip** and **os.name** facts instead.
- **Activity reporting screen showing labels instead of strings after upgrading.** Fixed an issue where the activity reporting screen displays incorrect labels after upgrading.
- **Direct Deployment Policy MAX_NODE_FAILURE value is being passed to the API from the UI as a string instead of an integer.** Fixed an issue where triggering a deployment with the **Direct Deployment Policy** option selected would fail when a `MAX_NODE_FAILURE` value is set.
- **SSLHandshakeException when trying to use Proxy for Azure DevOps integration.** Fixed an issue where trying to use a proxy for Azure DevOps integration would fail. Continuous Delivery for PE now correctly loads the CA certificate chain provided via the **External CA Certificate** setting in the PAM console.

- **Nodes section is blank and full of errors after every restart.** Fixed an issue where the **Nodes** section is blank or shows errors after a restart. The query service token now properly rotates even if Continuous Delivery for PE is not running when the rotation was scheduled, ensuring that node data is fetched properly.

Security notices:

- **CVE-2024-29857.** Updated bouncycastle to 1.78 to address this vulnerability.

Version 4.30.0

Released 7 May 2024.

Resolved in this release:

- **Unable to clone via SSH from ADO Cloud.** Upgraded Continuous Delivery for PE's SSH client to support RSA SHA256 and RSA SHA512 handshake algorithms. This was necessary to support SSH connections with ADO Cloud after Microsoft began phasing out support for SHA1 handshakes. Any other providers that also need the newer algorithms (such as GitLab deployed on Red Hat Enterprise Linux (RHEL) 9) are also now supported.
- **Removed authtokens API from OpenAPI specification.** The `/v1/authtokens` API has been replaced by `/v1/tokens`, which requires an authentication token in the header.
- **Duplicate GitLab integration causing a blank screen.** Fixed an issue loading the source control integrations screen caused by accidentally adding duplicate VCS providers.
- **After upgrading to 4.29.2 the commit link on Continuous Delivery for PE redirects to an invalid URL.** Fixed invalid commit links when using Azure DevOps as a source control provider.
- **Impact Analysis job link for a module goes to job for a control repo.** Fixed an issue navigating between the main control repo view and the Impact Analysis/Deployment screens.
- **Can create an invalid deployment stage.** Fixed an issue where it was possible to create a deployment stage pointing at an environment node group with the same name as the branch the pipeline runs from.
- **Module feature branch deployment stages are failing to deploy.** Fixed an issue where the module feature branch deployment stages failed to deploy. The missing control repo and branch fields have been added to the module deployment dialog for a regex pipeline.
- **HTTP 403 error when upgrading to Continuous Delivery for PE 4.29.2.** Fixed an issue in 4.29.x versions of Continuous Delivery for PE that required the user triggering a manual deployment to be either the owner of the workspace or a root user.
- **CD4PE_JOB_CONTEXT is unavailable unless a secret is added.** Fixed an issue where the output of pipeline job tasks were not displaying the value of the `CD4PE_JOB_CONTEXT` environment variable unless the job had a Secret added. The value of `CD4PE_JOB_CONTEXT` now displays regardless of whether a Secret is set or not.

Security notices:

- **CVE-2023-49569.** Updated go-git to address this vulnerability.
- **CVE-2023-1732.** Updated Cloudflare/circl to address this vulnerability.
- **CVE-2024-27304, CVE-2024-27289, CVE-2023-39325.** Updated Query Service to 1.8.16 to address these vulnerabilities.

Version 4.29.2

Released 21 March 2024.

Resolved in this release:

- **Missing Impact analysis (IA) report.** Fixed an issue with Impact Analysis and Azure DevOps where Continuous Delivery for PE falsely reports no impacted nodes.
- **Error trying to run an Impact Analysis for a Module.** Fixed an issue where users who created a deployment stage on a pipeline may see the Impact Analysis pipeline stage fail with the following error: `Cannot invoke "com.distelli.models.ControlRepoId.getDomain()" because "controlRepoId" is null`
- **Error in Continuous Delivery for PE Feature branch policy UI.** Fixed an error when editing a regex pipeline's deployment stage for a Bitbucket or GitHub control repo.

- **Pull request from Bitbucket Cloud not triggering Continuous Delivery for PE pipeline with pull request trigger.** Fixed an issue where pull requests to a Bitbucket cloud repository would not trigger a pipeline.
- **Triggering a regex pipeline against a branch with an existing pipeline runs that pipeline instead.** Fixed an issue where manually triggering a regex pipeline against a branch with an existing pipeline would cause the branch pipeline to run rather than the expected regex pipeline.
- **Unable to verify Global HTTP read timeout setting.** Increased the NGINX `proxy_read_timeout` from 60 seconds to 300 seconds to allow for longer running jobs.
- **Pressing <TAB> after entering user/email changes focus to Show/hide password instead of Input password.** Fixed a minor UI issue in the login screen where pressing <TAB> after entering the user name or email address would focus on the **Show/hide password** icon instead of the password input field.

Version 4.29.1

Released 21 February 2024.

New in this release:

- **Updating a pipeline now requires a new query parameter.** The `projectType` (`MODULE` or `CONTROL_REPO`) is now a required query parameter when updating pipelines with `/api/v1/pipelines-spec`.

Resolved in this release:

- **Unable to run deployments after creating or editing a pipeline.** Fixed an issue where deployments were not run for new or edited pipelines.
- **Unable to update a deployment on a regex pipeline.** Fixed an issue that prevented updates to a deployment on a regex pipeline.
- **Option to select an environment prefix in the Deployment dialog box.** Added the **SELECT AN ENVIRONMENT PREFIX** option to select a Puppet Enterprise environment prefix when creating a manual deployment or adding a deployment stage to a pipeline.
- **Unable to manually trigger a regex pipeline.** Fixed an issue where manually triggering a regex pipeline caused a "branch not found" error. Regex pipelines can now be triggered against branches matching the regex.
- **New Impact Analysis jobs cause list errors.** Manual Impact Analysis runs can now be triggered on code projects with custom names without causing an error with the tables on the **Control Repos** and **Modules** pages.
- **Unable to select a different view on the Nodes page.** Fixed an issue where selecting a different view on the **Nodes** page resulted in an error.
- **Continuous Delivery for PE approval emails not being sent after changing Bitbucket to GitLab.** Fixed an issue where approval notifications were not sent for deployments from GitLab projects in [subgroups](#).

Security notice:

- **CVE-2024-0567.** Updated the Debian Docker image to address this vulnerability.

Version 4.29.0

Released 8 February 2024.

New in this release:

- **Personal access token management.** You can now create authentication tokens to allow a user to enter their credentials once, then receive an alphanumeric token to access different services or parts of the system infrastructure. To manage personal access tokens, see [Manage personal access tokens](#).
- **OpenAPI support.** You can now fetch data and automate your workflows with the Continuous Delivery for Puppet Enterprise (PE) REST API. To get started using Continuous Delivery for PE public APIs, see [REST API](#).
- **Value reporting.** You can now view activity values across all the Puppet Enterprise (PE) instances integrated within a workspace in the **Activity** report. To view your activity in Continuous Delivery for PE, see [Activity reporting](#).
- **Refreshed Continuous Delivery for PE pipelines UI.** The Continuous Delivery for PE pipelines pages have a refreshed appearance.

Security notice:

- **CVE-2023-39325.** Updated several direct and indirect dependencies to address this vulnerability.

Version 4.28.0

Released 30 November 2023.

New in this release:

- **New node filter feature added to impact analysis.** A new node filter feature for Impact Analysis can be configured to run the analysis on a subset of impacted nodes. Nodes can be filtered by percentage of the number of nodes impacted by the change. See the [adding Impact Analysis step for your pipelines-as-code](#) to learn how to add this setting to your pipeline. Currently this setting is only available in pipelines-as-code. To enable pipelines-as-code, see [Construct pipelines from code](#).

Resolved in this release:

- **Fact charts do not always show the correct number of nodes when switching filters.** Fixed an issue in the node table so that the fact charts reflect the selected filters.

Security notice:

- **CVE-2023-36478.** Continuous Delivery for PE is not vulnerable, but we are now running the updated version of Jetty that addresses this vulnerability.

Version 4.27.1

Released 11 October 2023.

Resolved in this release:

- **Jobs fail with Null pointer exception on trigger events.** Fixed an issue where jobs in the first stage of a pipeline would occasionally fail due to a synchronization issue on the backend.

Version 4.27.0

Released 4 October 2023.

Resolved in this release:

- **Deleting workspace for LDAP user causes issues.** Fixed an issue where the `LDAPSyncFunction` task fails and the user is not able to log in to Continuous Delivery for PE. This happened when a deleted workspace was referenced in an LDAP Group Mapping.
- Upgraded the query service to version 1.8.14 to update the default configuration to support Continuous Delivery for PE 5.x.

Version 4.26.2

Released 7 September 2023.

Fixed in this release:

- **Pipeline jobs no longer intermittently fail with Postgres errors.** Fixed an issue where pipeline jobs randomly failed with Postgres errors in Continuous Delivery for PE.

Version 4.26.1

Released 23 August 2023.

New in this release:

- **Added new text field under the Config page in Puppet Application Manager called External CA Certificate.** The PAM UI now includes the **External CA Certificate** option under the **Config** page that allows you to supply a trusted SSL certificate for Continuous Delivery for PE to use when communicating with external services.

Version 4.26.0

Released 22 August 2023.

New in this release:

- **Large object store data is now stored in PostgreSQL.** Continuous Delivery for PE 4.26.0 now uses PostgreSQL for large object storage. New 4.26.0 installations use PostgreSQL from the outset. For existing users, large object data is migrated for you as part of the upgrade to version 4.26.0.

Note: For existing users, the data migration may cause the 4.26.0 upgrade process to take in excess of 15 minutes.

- **Refreshed the Continuous Delivery for PE Control Repos UI.** The Continuous Delivery for PE **Control Repos** pages have a refreshed appearance.
- **Refreshed the Continuous Delivery for PE Modules UI.** The Continuous Delivery for PE **Modules** pages have a refreshed appearance.

Resolved in this release:

- **Security fixes.** This release upgrades `okio-jvm` to version 3.4.0 to address CVE-2023-3635.

Version 4.25.1

Released 24 July 2023.

Resolved in this release:

- **Continuous Delivery for PE does not respect JVM HTTP proxy settings in certain scenarios.** We fixed an issue where Continuous Delivery for PE ignored JVM HTTP proxy settings when attempting to proxy requests for Bitbucket Server/Cloud, GitHub/GitHub Enterprise, and GitLab.

Version 4.25.0

Released 11 July 2023.

New in this release:

- **User interface improvements.** The following Continuous Delivery for PE pages have a refreshed appearance:
 - Page not found (404)
 - Runtime error
 - Forbidden (403)
 - **Forgot your password?**
 - **Create a user account**

Resolved in this release:

- **Continuous Delivery for PE default pipeline impact analysis fails with a non-actionable error message.** Updated the error message to make it more descriptive and useful when a pipeline with no deployment stage fails the impact analysis stage.
- **GetJobInstanceV1 returns control repo display name for GitLab.** Fixed an issue where links to a GitLab source control repository from the **Job details** screen wouldn't work if the control repo/module name did not match the GitLab repo name.
- **Security fixes.** Version 4.25.0 includes several security fixes, including:
 - Upgraded `gin-gonic` to version 1.9.1 to address CVE-2023-29401.
 - Upgraded `guava` to version 32.0.0-android to address CVE-2023-2976.

Version 4.24.1

Released 28 June 2023.

Resolved in this release:

- **Impact Analysis always shows zero changes.** We fixed an issue where module impact analysis would not correctly diff module versions when running against Puppet Enterprise deployed with the extra-large architecture. This fix improves logging in module impact analysis around module version diffing and shortens the time it takes the module impact analysis to report a failure when it is unable to compare module versions between environments.
- **Documentation improvements.** Added a new section to clarify how to [Integrate with Azure DevOps Server on prem](#) on page 149.

Version 4.24.0

Released 31 May 2023.

New in this release:

- **Added AzureDevOps support.** Continuous Delivery for PE now supports ADO Server (on prem) integration.
- **Refreshed Continuous Delivery for PE user login UI.** The Continuous Delivery for PE user login pages have a refreshed appearance.

Resolved in this release:

- **Unable to access job details page for jobs under code validation stage.** Job detail links from code validation jobs now correctly redirect to the appropriate job details pages, rather than timing out.
- **Security fixes.** Version 4.24.0 includes several security fixes, including:
 - Upgraded the query service to version 1.8.12 to protect against potential race conditions in the backend query service.
 - Upgraded jackson to version 2.15.0 to address CVE-2022-1471 in snakeyaml.

Version 4.23.1

Released 2 May 2023.

Resolved in this release:

- **Unable to access Job details pages for jobs in the Code validation stage.** We fixed an issue where, after installing or upgrading to version 4.23.0 of Continuous Delivery for PE, the **Job details** pages for jobs in the **Code validation stage** of the pipeline were no longer available.
- The version 4.23.1 `.airgap` bundle has an updated version of `puppet-dev-tools` that fixes CVE-2023-27533 and CVE-2023-27536.

Version 4.23.0

Released 18 April 2023.

New in this release:

- **Refreshed Continuous Delivery for PE job hardware UI.** The Continuous Delivery for PE job hardware pages have a refreshed appearance.
- **Refreshed Continuous Delivery for PE Job details UI.** The Continuous Delivery for PE **Job details** page has a refreshed appearance.

Resolved in this release:

- **Module subset modal prefix not working.** The Module subset modal on the **Edit Permissions** screen now filters modules by prefix when typing.
- **Continuous Delivery for PE misidentifies pull requests coming from a Bitbucket Server.** Fixed an issue where webhook events for Bitbucket Server repositories would incorrectly detect a pull request as coming from a forked repository.
- **Unable to list commits when branch name contains a "/".** Commits are now properly listed when a branch name contains a "/".

- **Creating a new job template with a blank description causes an error.** The **Job details** page now updates properly when a job template is created with a blank (`null`) description.

Version 4.22.2

Released 21 March 2023.

Resolved in this release:

- **Users unable to manually update token for Puppet Enterprise integration.** We fixed an issue where users were unable to manually update their Puppet Enterprise token in the UI.
- **Disabling Vault before deploying Continuous Delivery for PE 4.22.0 causes it to crash.** We fixed an issue where disabling **Enable Vault** in the Puppet Application Manager (PAM) UI prior to the initial deployment of Continuous Delivery for PE 4.22.0 causes Continuous Delivery for PE to crash. You can now disable Vault prior to the initial deployment of Continuous Delivery for PE 4.22.0.
- The version 4.22.2 `.airgap` bundle has an updated version of `puppet-dev-tools` that appropriately sets the tag at 4.x.

Version 4.22.1

Released 16 March 2023.

Resolved in this release:

- **Console doesn't appear after upgrading.** IPv6 was enabled by default, which caused NGINX to fail to start on non-IPv6 environments. IPv6 is now dynamically enabled only on IPv6 environments to prevent this.

Version 4.22.0

Released 8 March 2023.

New in this release:

- **Puppet Application Manager (PAM) UI now has an option to disable Vault.**

The PAM UI now includes the **Enable Vault** option under **Advanced configuration and tuning** that allows you to remove Vault. **Enable Vault** is selected by default to allow for migrations on older Continuous Delivery for PE releases (4.19.0 and older).

Vault can be safely removed on all new installations from this release on. For users upgrading from version 4.19.0 or older, you must complete at least one successful deployment of version 4.20.0 or newer to safely remove Vault using the **Enable Vault** option in version 4.22.0.

- **Refreshed Continuous Delivery for PE Groups Settings UI.** The Continuous Delivery for PE workspace Groups Settings pages have a refreshed appearance.
- **Refreshed Continuous Delivery for PESource Control Integrations Settings UI.** The Continuous Delivery for PE Source Control Integrations Settings pages have a refreshed appearance.

Resolved in this release:

- **Login flow for root broken when no workspaces exist.** An infinite redirect loop was possible when users logged in as root and there were no workspaces in Continuous Delivery for PE.
- **Concurrent catalog compilation throttling is now reset on Continuous Delivery for PE restart.** Impact analysis would hang when the maximum number of concurrent compilations on each PE instance was reached.
- **404 error when accessing a node from the node table.** Selecting a node from the node table on the **Nodes** page would result in a 404 error.
- The version 4.22.0 `.airgap` bundle has an updated version of `puppet-dev-tools` that includes PDK 2.6.1 and a fix addressing CVE-2023-23916.

- **Security fixes.** Version 4.22.0 includes several security fixes, including:
 - Upgrading `crypto` to `v0.0.0-20211209193657-4570a0811e8b` to address CVE-2022-27191.
 - Upgrading `gin-gonic` to 1.9 to address CVE-2022-41723.
 - Upgrading `postgresql` to 42.4.1 to address CVE-2022-41946.

Removed in this release:

- **Support for versions of Kubernetes prior to 1.21.** Kubernetes version 1.21 or higher is now required for Continuous Delivery for PE and Comply.

Version 4.21.1

Released 6 February 2023.

Resolved in this release:

- **Broken links associated with renaming workspaces.** This fix resolves the following issues:
 - Broken links in the UI that pointed to the old name of a workspace that had been renamed.
 - Broken links from adding or editing job hardware capabilities from the root console.
- **Regex pipeline is triggered by webhook from GitLab when the branch is deleted.** Regex pipelines are no longer triggered after merging the GitLab feature branch PRs and deleting the feature branches.

Version 4.21.0

Released 25 January 2023.

New in this release:

- **Documentation improvements.** Beginning with 4.18.1 and ending with this release, we made quality of life changes to the Continuous Delivery for PE documentation. This primarily consisted of reorganizing some pages in a more logical manner and renaming some pages with overly-long titles. We rolled out a few changes with each release and enabled redirects where necessary.

Changes in this release include:

- Moved [Test Puppet code with jobs](#) on page 185, [Analyze the impact of code changes](#) on page 192, [Construct pipelines](#) on page 199, and [Deploy Puppet code](#) on page 215 under [Test and deploy Puppet code](#) on page 184. There was no change to the structure of these sections other than some minor renaming.
- Renamed **Structuring a .cd4pe.yaml file** to [.cd4pe.yaml file structure](#) on page 206.
- Renamed **Validating your .cd4pe.yaml file** to [.cd4pe.yaml file validation](#) on page 209.
- Renamed **Limitations of impact analysis** to [Impact analysis limitations](#) on page 198.

If you have any questions or comments about these changes, please use the feedback form at the bottom of this page to get in touch with our documentation team. Refer to prior release notes for details about previous changes.

Resolved in this release:

- **Environment status for cancelled impact analysis tasks was incorrect.** Previously, if you cancelled an impact analysis task, the task was actually cancelled, but the **Status** for each environment would not update after cancellation. If you examined the impact analysis details in the Web UI, it would seem as though impact analysis was still running, when it was actually stopped. Now the environment statuses correctly update after impact analysis is cancelled.
- **OOMKilled error after upgrade.** We fixed an issue where the `cd4pe-migrate-object-store` job was OOMKilled on application upgrade, which also caused Continuous Delivery for PE to get stuck initializing.
- **Pipeline statuses weren't communicated to AzureDevOps.** Continuous Delivery for PE sends pipeline status information to AzureDevOps for pull requests.
- **Rapid navigation broke some pages.** Occasionally, leaving a page before it was fully loaded caused an error.

- **Security fixes.** Version 4.21.0 includes several security fixes, including:
 - Upgrading `react-hot-loader` to version 4.13.1 to address CVE-2022-37601 and CVE-2022-37603.
 - Upgrading the query service to version 1.8.8 to address CVE-2022-32149 and CVE-2022-27664.
 - Upgrading `express` to version 4.18.2 to address CVE-2022-24999.
 - Removing `rbac` and `rbac-init` images from airgapped bundles for Continuous Delivery for PE.
 - Upgrading `jackson` to version 2.14.0 to address CVE-2022-41854 and CVE-2022-38752 in `snakeyaml`.

Version 4.20.0

Released 15 November 2022.

New in this release:

- **Refreshed Puppet Enterprise integration UI.** The Puppet Enterprise integration settings screens have a refreshed appearance.
 - Integrations are organized in card format, instead of table format.
 - The general process to [Integrate with Puppet Enterprise](#) on page 143 or [set up protected environments](#) has not changed, but labels and placement of some fields, icons, and buttons have changed.
 - We added an error message to the **Protected environments** section that appears if you don't have permission to manage protected environments for a particular PE integration.
- **Compiler maintenance mode.** You can [Enable compiler maintenance mode](#) on page 166 to force code deployments to skip unavailable or offline compilers and replicas.
- **Job secrets stored in PostgreSQL.** Job secrets are now stored in PostgreSQL instead of Vault. Upon upgrade to 4.20.0 or later, your existing secrets are automatically migrated from Vault to PostgreSQL. After this migration, Vault is no longer used.
- **Search all branches when managing pipelines as code.** You can now search all branches when selecting a branch to [manage pipelines as code](#). Previously, the list of branches was truncated and wasn't searchable.
- **Delete SMTP configuration.** You can now delete unwanted [SMTP configurations](#).
- **Impact analysis ignores patch fact generation script files.** The `pe_patch_fact_generation.ps1` and `pe_patch_fact_generation.sh` files are now excluded from impact analysis reports because these files always report a change of `n/a` to `n/a`. This created unnecessary clutter in the reports.
- **Documentation improvements.** Beginning with 4.18.1, we've begun to make quality of life changes to the Continuous Delivery for PE documentation. This primarily consists of reorganizing some pages in a more logical manner and renaming some pages with overly-long titles. We'll roll out a few changes with each release and enable redirects where necessary.

Changes in this release include:

- Moved [Configure impact analysis](#) on page 192 under [Analyze the impact of code changes](#) on page 192 so that all impact analysis pages are grouped together.
- Renamed **Managing teams and team members** to [Manage workspaces and access](#) on page 167.
- Renamed **Configuring and adding integrations** to [Configure and integrate](#) on page 141.
- Renamed **Advanced Continuous Delivery for PE configuration** to [Advanced configuration](#) on page 163.

If you have any questions or comments about these changes, please use the feedback form at the bottom of this page to get in touch with our documentation team. Refer to prior release notes for details about previous changes.

Resolved in this release:

- **Fixed nested facts in Node Table.** Nested facts now display correctly in the **Node Table** in **Node Inventory**.
- **Corrected error message.** When running impact analysis on a module repo, if the specified branch is missing, the error message correctly refers to the repo as a module, instead of a control repo.
- **Impact analysis runs could fail in environments with proxies.** We fixed an issue where impact analysis runs triggered by webhooks would fail in environments with proxies that use OpenTelemetry tooling.
- **Security fixes.** This release upgraded `curl-proxy` to 1.86.2 to address CVE-2022-40674 and CVE-2022-37434.

Version 4.19.0

Released 4 October 2022.

New in this release:

- **Improved impact analysis performance and removed redundant setting.** We improved impact analysis performance when calculating impacted nodes. As a result of this, we removed the **Environments path** setting since it is no longer required to improve impact analysis performance. If you enabled this setting, this setting is removed during the upgrade, and you do not need to take any additional steps.
- **Repository menu limited to 10 results.** When [adding repositories](#), the **Repository** dropdown menu shows a maximum of 10 repositories, and you can type a repository name to refresh the results. In this release, we added a message to the dropdown menu clarifying the results limit.
- **Return to Login page from Forgot password page.** We added a link back to the **Login** page from the **Forgot password** page.
- **Improved error messages.** We improved the error messages shown on the **Hardware** page when Continuous Delivery for PE can't connect to the selected PE integration.
- **Documentation improvements.** Beginning with 4.18.1, we've begun to make quality of life changes to the Continuous Delivery for PE documentation. This primarily consists of reorganizing some pages in a more logical manner and renaming some pages with overly-long titles. We'll roll out a few changes with each release and enable redirects where necessary.

If you have any questions or comments about these changes, please use the feedback form at the bottom of this page to get in touch with our documentation team.

No changes were included with 4.19.0. Refer to the 4.18.1 release notes for details about the latest changes.

Resolved in this release:

- **Impact analysis queued indefinitely.** We fixed an issue that sometimes caused impact analysis tasks to get stuck in the queued state indefinitely.
- **Couldn't change a job's hardware capabilities.** We fixed an issue that prevented you from selecting or deselecting **Hardware capabilities** when creating or editing [jobs](#).
- **Browser-specific issues.** We resolved a couple of error-handling issues that occurred in some browsers when running Continuous Delivery for PE in Kubernetes.
 - When editing workspace settings, the **SSH key** tab shows a warning if the workspace doesn't have an SSH key and gives you the option to add one.
 - When [adding repositories](#), if webhook setup fails, you'll get a specific pop-up message, rather than a generic error.
- **Links to impact analysis documentation.** In the Continuous Delivery for PE web UI, a few links to impact analysis documentation were broken. We've fixed those.
- **Incomplete list of Azure DevOps projects.** When [adding repositories](#) from Azure DevOps Services, Continuous Delivery for PE can now show an unlimited number of projects. Previously, there was a cap on the maximum number of projects shown, and you would only hit the limit if you had access permission for a significant number of projects.
- **Security fixes.** This release upgraded `curl-proxy` to 1.83.0 to address CVE-2021-22946, CVE-2022-22576, CVE-2022-27781, and CVE-2022-27782.

Version 4.18.1

Released 27 September 2022.

New in this release:

- **Documentation improvements.** With this release, we're beginning to make quality of life changes to the Continuous Delivery for PE documentation. This primarily consists of reorganizing some pages in a more logical

manner and renaming some pages with overly-long titles. We'll roll out a few changes with each release and enable redirects where necessary.

If you have any questions or comments about these changes, please use the feedback form at the bottom of this page to get in touch with our documentation team.

Changes in this release include:

- Moved [Key concepts](#) on page 6 and the [Getting started guide](#) on page 13 under [Welcome to Continuous Delivery for Puppet Enterprise](#) on page 5.
- Renamed **Set up Continuous Delivery for PE** to [Install](#) on page 126.
- Renamed **Upgrading Continuous Delivery for PE** to [Upgrade](#) on page 133.
- Moved [Migrate 3.x data to 4.x](#) on page 136 under [Upgrade](#) on page 133, since migration is part of the upgrade process.
- Moved [Analytics data collection](#) on page 142 under [Configure and integrate](#) on page 141, which matches the placement of a similar page in the PE documentation.
- Moved [Review node inventory](#) on page 224 further down in the table of contents.

Resolved in this release:

- **GitLab integration configuration issue.** Resolved an issue preventing users from configuring GitLab integrations to use HTTP or HTTPS for cloning.

Version 4.18.0

Released 8 September 2022.

New in this release:

- **Java 17 upgrade.** This release includes an upgrade to Java 17, which deprecated some JVM args.

Important: If you are passing any custom JVM args, make sure these are compatible with Java 17.

Resolved in this release:

- **Impact analysis details page crashes.** The impact analysis details page no longer crashes if the filter set in the URL does not exist.

Version 4.17.0

Released 11 August 2022.

Resolved in this release:

- **Impact analysis detects Hiera `.eyaml` files.** Previously, impact analysis only detected changes in Hiera files with the `yaml` extension. Impact analysis now also detects changes in Hiera files with the `eyaml` extension.
- **Corrected a database issue with pipeline event data.** We fixed an issue where triggering a pipeline caused incorrect pipeline event data to be recorded in the database for all pipelines.
- **Security fixes.** Version 4.17.0 includes several fixes related to security, including:
 - Upgraded `pam-utils` to address several CVEs.
 - Fixes to address CVE-2022-31197, CVE-2022-30591, and CVE-2020-29587.

Version 4.16.1

Released 14 July 2022.

Resolved in this release:

- **The Group user attribute wasn't respected when querying LDAP group membership.** We fixed an issue where Continuous Delivery for PE incorrectly assumed the **Group user attribute** value was the same as the user's **Distinguished name**. This caused failure of group sync tasks and prevented deletion of LDAP group mappings.

- **You can delete LDAP group mappings after removing an LDAP configuration.** Previously, if you removed an LDAP configuration before removing the associated group mappings, the group mappings were orphaned and impossible to delete.

Version 4.16.0

Released 12 July 2022.

New in this release:

- **You must manually update webhooks after changing the backend service endpoint.** To prevent unexpected and undesired changes to webhooks, Continuous Delivery for PE no longer automatically updates your configured webhooks when you change the `CD4PE_BACKEND_SERVICE_ENDPOINT`. Instead, a warning message is logged asking you to [Update webhooks](#) on page 153 manually through the web UI.
- **Automatic prefix selection when there is only one prefix available.** If you use environment prefixes, when you add a deployment stage to a pipeline and there is only one prefix option available, Continuous Delivery for PE automatically selects that prefix. This enhancement resolves an issue where it was unclear that you needed to select an option when there was only one possible choice.
- **Minor UI change.** We removed the Gravatar icon from the job **Details** page.

Resolved in this release:

- **Multiple pipelines-as-code issues, including unchanged pipelines suddenly missing (entirely or partially), unexpected duplicated pipelines, unfamiliar pipelines, or the Pipelines page loads very slowly.** Webhooks were causing [pipelines-as-code](#) to be rebuilt more often than necessary and retaining extra, unnecessary pipeline data.

Important: If you experience any of the above issues, you need to forcefully reload your `.cd4pe.yaml` file, as explained in this Support article: [Pipelines-as-code issues in Continuous Delivery for PE 3.0.0 to 4.15.1](#).

Version 4.15.1

Released 14 June 2022.

Resolved in this release:

- The **SMTP port** setting is no longer ignored if you enable TLS for your SMTP configuration.

Version 4.15.0

Released 7 June 2022.

New in this release:

- **(Experimental) Run impact analysis on fewer nodes.** If an environment has a lot of nodes, it might take a long time for impact analysis to run. It is possible to only analyze a subset of your total nodes, but there are tradeoffs. We've described a process you could use to [Run impact analysis on fewer nodes](#) on page 197. If this feature interests you, please let us know what you think.
- **Improved source control integration field validation.** When configuring GitLab, GitHub Enterprise, or Bitbucket Server integrations, the field validations are more robust and provide more useful error messages.

Resolved in this release:

- **Pipeline runs triggered by pull requests continue to respect PR gates after pipeline promotion.** Previously, promoting a pipeline that was triggered by a pull request could discard the PR event type association. This caused any subsequent manual stage promotions to ignore their respective PR gates, because the pipeline run incorrectly appeared as if it were triggered by a commit (instead of a PR). Now, pipelines triggered by PRs maintain their PR event type association throughout the entire pipeline run, and manual pipeline promotions respect PR gates when the pipeline is originally triggered by a PR.

Version 4.14.0

Released 5 May 2022.

Resolved in this release:

- **Endpoints properly display errors.** We fixed an issue where some endpoints couldn't correctly display errors in certain circumstances.
- **Jobs running longer than 20 minutes no longer fail with exit code 1.** Lengthy jobs now continue running as expected.
- **Deploying Continuous Delivery for PE to an OpenShift cluster resulted in pod failures.** To prevent an error where OpenShift can't find `/sbin/nologin`, a copy of `/usr/sbin/nologin` is made to `/sbin/nologin` in the new `pam-utils` container.
- **Security fixes.** Version 4.14.0 includes several fixes related to security, including:
 - Fixes to address CVE-2022-0778, CVE-2022-1271, CVE-2022-1233, and CVE-2020-36518.
 - Updated teams-ui webpack to v5 to address a security issue with a child dependency of v4.

Version 4.13.0

Released 5 April 2022.

New in this release:

- **Improved proxy handling.** Continuous Delivery for PE now uses relative, rather than absolute, lookup paths to construct all URLs in the UI. Previously, configuring a proxy required changing the Continuous Delivery for PE service endpoint to avoid CORS errors. This improvement is a more complete resolution to the fix included in version 4.12.1.
- **Accessibility improvements.** Version 4.13.0 introduces several improvements for accessibility in Continuous Delivery for PE, including:
 - Removed leading + symbols from some buttons, which could cause incongruity between the accessible name and visible label.
 - On the PE settings page, the icons to remove a protected environment and change a token's lifetime now have tooltips correctly describing their functions.
 - The exit icon now has a tooltip.
 - On the SSH key settings page, the **More actions** icon now has a tooltip explaining the contents of the menu associated with this button.

Resolved in this release:

- **Fixed navigation for super users without workspace membership.** A super user who is not a member of any workspace is directed to the root console after logging in. When a super user accesses a workspace they are not a member of, the navigation menu loads correctly.
- **Multi-node clusters don't prevent draining nodes with StatefulSets.** We modified `PodDisruptionBudget` to allow multi-node Continuous Delivery for PE clusters to drain all but the last node.
- **Security fixes.** Version 4.13.0 includes several bug fixes related to security, including:
 - The kurl-proxy and minIO containers have the latest OS patches.
 - Fixes that address CVE-2021-43858 and CVE-2022-0839.
 - Upgraded OpenTelemetry to upgrade a Commons-io dependency.
 - The build process removes test keys such as those left by the `public-encrypt` package and Bolt installation.

Version 4.12.1

Released 7 March 2022.

Resolved in this release:

- Fixed a reverse proxy configuration issue that caused CORS errors when users tried to login.

Version 4.12.0

Released 2 March 2022.

Important: Puppet Application Manager (PAM) version 1.64.0 is now available. To avoid a failed to pull: deployment method for chart vault has changed error, upgrade PAM to version 1.64.0 **before** upgrading Continuous Delivery for PE to version 4.12.0.

New in this release:

- **Secrets management in Continuous Delivery for PE.** You can add secrets to Continuous Delivery for PE jobs, which jobs use while running. To learn more, go to [Test Puppet code with jobs](#) on page 185.
- **Continuous Delivery for PE now supports Kubernetes 1.19 to 1.24.** Kubernetes 1.17 and 1.18 are no longer supported.
- **Usability improvements.** Version 4.12.0 introduces several improvements to the design and usability of Continuous Delivery for PE, including:
 - The **Nodes** page has improved button placement and text.
 - The **Nodes** page has a new view selector that allows you to pick which view you want to see.
 - Custom view names now become the page title when selected.
 - You can edit an existing view or save a new view by clicking **Save** or **Edit** in the drop down.

Resolved in this release:

- **Fixed the impact analysis filter for failed nodes.** This also fixed the **Compilation failures** link on the impact analysis details view.
- **You can use .cd4pe.yaml files over 500 lines with Bitbucket Server.**
- **The Disable MinIO option is no longer available in standalone installs.** This option only applies to HA installs.
- This release contains fixes that address CVE-2021-43527.

Version 4.11.5

Released 22 February 2022.

Resolved in this release:

- Fixed an issue where running impact analysis against a Bitbucket Cloud control repository detected no changes.

Version 4.11.4

Note: Due to an issue discovered after release, we retracted version 4.11.3.

Released 14 February 2022.

Resolved in this release:

- Fixed an issue where you could not run Continuous Delivery for PE jobs without Docker hardware. Now, non-Docker-based jobs run directly on the job runner machine.
- This version's `.airgap` bundle includes an updated version of `puppet-dev-tools`.

Version 4.11.2

Released 2 February 2022.

Resolved in this release:

- Fixed an issue where you could not add a control repo or module with Bitbucket Cloud.
- This release contains fixes that address CVE-2022-21724.

Version 4.11.1

Released 20 January 2022.

Resolved in this release:

- Installations that use a legacy or high availability (HA) architecture for Puppet Application Manager no longer receive a `Job cd4pe-migrate-object-store is invalid` error when upgrading to the 4.11.x series.

Version 4.11.0

Released 20 January 2022.

New in this release:

- **Compound filters on the Nodes page.** You can now build multi-element filters that use logical operators (**and** and **or**) to answer complex queries about your nodes. For more information about creating and using compound filters, go to [Create filters to focus on specific node sets](#).
- **Endpoint configuration clarified.** When you [Deploy Continuous Delivery for PE](#) on page 130, you can use a NodePort or an Ingress for your webhook and local container registry endpoints. Previously, the Ingress option was only available if you had previously set it. Now, you can always choose from both options.
- **Retrieve impact analysis CSV files through an API call.** You can now reach the `getImpactAnalysisCsvV1` endpoint from the Continuous Delivery for PE deployments module. This means your custom deployment policies can use this endpoint to retrieve impact analysis CSV exports.
- **Custom deployment policy logging.** You can add custom deployment events with message parameters to your custom deployment policies. These appear as arbitrary log messages and, ultimately, on the web UI. This facilitates debugging when creating custom deployment policies.
- **Usability improvements.** Version 4.11.0 introduces several improvements to the design and usability of Continuous Delivery for PE, including:
 - The ability to search for nodes in the table by name on the **Nodes** page.
 - To improve reliability of snapshot restores, PostgreSQL now initially listens on localhost during startup.

Resolved in this release:

- **Improved impact analysis report filtering, searching, and pagination.** When an impact analysis report has multiple pages, searching and filtering refreshes pagination.
- **Export impact analysis report functionality restored.** From the web UI, you can export CSV files of your impact analysis reports again.
- **Improved error handling when changing user email addresses.** Changing a user's email address to the user's existing email address no longer triggers an error. Attempting to change a user's email address to an email address belonging to another user returns an error message explaining that another user is using this email address.
- **SMTP "from" address defaults to root account's email.** If the **Send emails from this address** field is empty, Continuous Delivery for PE now uses the email address associated with the root user.
- **Impact analysis succeeds on GitHub repos with different repo and display names.** Impact analysis tasks are now performed correctly on GitHub repositories where the repo name does not match the display name set in Continuous Delivery for PE.
- **The Continuous Delivery for PE container restarts successfully.** The database migration lock is now automatically removed when the container stops, allowing for a successful restart without manually removing the lock.

Version 4.10.5

Released 20 December 2021.

Resolved in this release:

- This release upgrades the included version of Apache Log4j to 2.17.0.

Version 4.10.4

Released 17 December 2021.

Resolved in this release:

- This release upgrades the included version of Apache Log4j to 2.16.0.

Version 4.10.3

Released 10 December 2021.

Resolved in this release:

- This release contains fixes that address [CVE-2021-44228](#).

Version 4.10.2

Released 9 December 2021.

Resolved in this release:

- When listing Bitbucket Server branches, the first result is no longer omitted from the list.
- Continuous Delivery for PE no longer attempts to set up SSH cloning for GitLab integrations unless explicitly instructed to do so.
- You can now successfully re-add an integration to Bitbucket Server or to GitLab using SSH.
- The value set for **Global HTTP write timeout (seconds)** in the **Advanced configuration and tuning** section of the **Config** page in Puppet Application Manager is now also used as the value for the `CD4PE_MODULE_DEPLOY_READ_TIMEOUT` environment variable in deployment tasks. The default value is 120 seconds.
- An issue with database connections has been resolved, and Continuous Delivery for PE now renders pages as expected without requiring you to periodically restart the application.

Version 4.10.1

Released 11 November 2021.

Resolved in this release:

- Requests to the `/v1/auth/tokens` endpoint are now processed correctly.

Version 4.10.0

Released 9 November 2021.

Important:

This version includes a security fix to Continuous Delivery for PE that requires new authentication tokens for all PE integrations. As part of the upgrade process to version 4.10.0, Continuous Delivery for PE attempts to automatically rotate the tokens for all your integrated PE instances. In cases where tokens can not be successfully rotated by the software, you must complete the token rotation process manually.

After upgrading to version 4.10.0, go to the **Message Center** in the Continuous Delivery for PE web UI for a custom report on the state of your PE tokens and instructions for performing any required manual steps.

Special note for users of PE version 2021.x: The security update revokes **all** tokens assigned to the Continuous Delivery user in version 2021.x. You must regenerate and reconnect all PE tokens assigned to this user.

New in this release:

- **Environments path setting for impact analysis.** To improve impact analysis performance for users with certain PE configurations, a new **Environments path** setting is now available in the **Impact analysis credentials** section

of each Puppet Enterprise instance's credentials. Users who have configured PE to use lockless deploys MUST NOT set the environments path.

Resolved in this release:

- PuppetDB queries are now modified in order to improve impact analysis performance.
- Support bundles are now analyzed correctly and do not throw errors. Upgrade to Puppet Application Manager 1.49.0 or a newer version to apply this fix.
- Support bundle collection now requires less memory for environments experiencing heavy usage.
- Cross-version Puppet Development Kit (PDK) dependencies are now included in Continuous Delivery for PE, so PDK jobs no longer fail in offline (airgapped) environments.

Version 4.9.0

Released 8 September 2021.

New in this release:

- **Impact analysis tasks run in parallel on multiple PE instances.** When an impact analysis task is triggered to run on multiple PE instances, the task now runs simultaneously on each instance rather than waiting for one instance to finish before starting on the next.
- **Usability improvements.** Version 4.9.0 introduces several improvements to the design and usability of Continuous Delivery for PE, including:
 - The **Users** page is updated with a cleaner, more streamlined design.

Resolved in this release:

- Clicking **Documentation** in the web UI now correctly directs you to the 4.x documentation set.
- When enabled, the HTTP health check for load balancers now operates as expected.

Version 4.8.2

Released 31 August 2021.

Important: You must upgrade to version 4.8.2 before installing Puppet Enterprise 2021.3 or 2019.8.8. Version 4.8.2 resolves a PuppetDB issue that prevented the generation of new fact charts on the **Nodes** page.

Resolved in this release:

- Issues with the query service and the interaction between the **Nodes** page and PuppetDB are now resolved.

Version 4.8.1

Released 24 August 2021.

Resolved in this release:

- An endpoint that was accidentally removed in version 4.8.0 is now restored.

Version 4.8.0

Released 10 August 2021.

New in this release:

- **Configure snapshot timeouts.** You can now configure the length of time that Puppet Application Manager spends attempting to back up Continuous Delivery for PE components when creating a snapshot. For more information, go to [Adjust the timeout period for snapshots](#) on page 163.

Resolved in this release:

- When impact analysis tasks are run on a compiler, the resulting report now shows the list of impacted nodes.
- The installation preflight check now correctly requires 50 GB of storage for Ceph.

- You can now successfully restore Continuous Delivery for PE from a snapshot on legacy installations of Puppet Application Manager.

Version 4.7.2

Released 26 July 2021.

Resolved in this release:

- The web UI no longer attempts to fetch remotely hosted fonts, and now loads correctly for installations in offline (airgapped) environments.
- A bug caused database backups in new installations of versions 4.7.0. and 4.7.1 to silently fail. New installations of Continuous Delivery for PE now correctly back up and restore the contents of the PostgreSQL database.
- Users running legacy installations of Puppet Application Manager version 1.44.1 can now successfully upgrade from Continuous Delivery for PE version 4.4.2 or older to the current version.

Version 4.7.1

Released 12 July 2021.

Resolved in this release:

- The LDAP group mappings list now displays up to 200 group mappings.
- If multiple LDAP group mappings use the same LDAP group name and RBAC group name, you can now successfully delete one group mapping without deleting all group mappings that share these names.

Version 4.7.0

Released 8 July 2021.

New in this release:

- **Fact charts.** You can now see visual representations of Facter fact values on all nodes across the infrastructure you've integrated with Continuous Delivery for PE. The new **Fact charts** section of each view on the **Nodes** page displays the distribution of unique values across your inventory for your selected facts. We've included four fact charts to get you started, and you can [build custom fact charts](#) for the facts that are relevant to your business goals.
- **Usability improvements.** Version 4.7.0 introduces several improvements to the design and usability of Continuous Delivery for PE, including:
 - Several web UI pages have been updated with a cleaner, more streamlined design.
 - The **Users** page now shows the complete list (up to 1,000 users) of users in a workspace.

Resolved in this release:

- New Azure DevOps integrations can now be set up successfully.
- If an empty (memberless) LDAP group map is added to Continuous Delivery for PE, other previously added LDAP group maps now sync correctly.
- When a new workspace is created, jobs in that workspace now default to running on workspace hardware.
- Node filter results are now correctly returned for fact names that use dot notation.
- A change to a saved view created by removing a filter can now be saved.

Version 4.6.1

Released 16 June 2021.

Resolved in this release:

- Login attempts after upgrading to Continuous Delivery for PE 4.6.0 or higher with an older license no longer fail.
- Setting up an external PostgreSQL database no longer requires a separate configuration for the estate reporting service. The estate reporting service now defaults to sharing the Continuous Delivery for PE database. For more information, see [Set up external PostgreSQL](#).

Version 4.6.0

Released 3 June 2021.

New in this release:

- **Promote permission.** Previously, the permission to manually promote changes through pipeline stages was included in the Edit permission for control repos and modules. The Promote permission is now separate from the Edit permission, and you can grant or deny these permissions to groups as needed.

Note: The new Promote permission has been automatically assigned to any group that was assigned the Edit permission on control repos or on modules in versions prior to 4.6.0.

- **Set group permissions on a subset of control repos.** You can now create groups that have permissions on only a subset of the control repos in your workspace.
- **Export impact analysis data.** You can now download the data generated by an impact analysis task. Click **Export** on the impact analysis report page to generate a CSV file of the data.
- **LDAP group login filtering.** You now have the option to enable login filtering for your LDAP configuration. If login filtering is turned on, only those LDAP users who are included in mapped LDAP groups are able to log into Continuous Delivery for PE.
- **Increased default memory limits.** In order to support higher out-of-the-box load, the default memory configuration for Continuous Delivery for PE now uses higher default memory limits while starting with the same base memory use.
- **Run multiple Puppet applications on the same cluster.** You can now run multiple supported applications (currently Continuous Delivery for PE version 4.6.0 and newer, and Puppet Comply version 1.0.4 and newer) on a single instance of Puppet Application Manager. Find more information in the [Working with Puppet applications](#) section of the Puppet Application Manager documentation.
- **Usability improvements.** Version 4.6.0 introduces several improvements to the design and usability of Continuous Delivery for PE, including:
 - Several web UI pages have been updated with a cleaner, more streamlined design.
 - Improved error messaging when a webhook cannot be automatically set up for a newly added control repo or module.
 - Support bundles now note whether services are unavailable because the **We're migrating an existing Continuous Delivery for PE 3.x instance** option is enabled.
 - The certificate preflight check now accepts a wildcard certificate as valid.

Resolved in this release:

- Custom deployment policies no longer require environment branches.
- If a workspace has no owner, the **Workspaces** page in the root console now loads correctly so that you can reassign the workspace to a new owner.
- Information about global shared hardware is now correctly displayed when you navigate to the **Hardware** page in the root console from the individual workspace's **Hardware** page.

Version 4.5.2

Released 11 May 2021.

Resolved in this release:

- If SAML is enabled for your Continuous Delivery for PE installation, a **Log in using single sign-on** option is now shown on the login screen, and the pod no longer falls into a restart loop.
- Docker runtime arguments are no longer passed if a job previously run on workspace hardware is updated to run on global shared hardware.

Version 4.5.1

Released 27 April 2021.

Resolved in this release:

- The links provided in deployment approval emails now resolve correctly.

Version 4.5.0

Released 22 April 2021.

Important:

- Continuous Delivery for PE version 4.5.0 includes architectural changes that alter the paths of some page URLs and might break previously generated links to pull requests and other pages in your source control.
- If you use the Bolt tasks included in the `puppetlabs-cd4pe` module, upgrade the module to version 3.1.0 in your Bolt project.
- **Optional.** A new version of the platform admin console is available with support for full (instance-level) snapshots. Learn more in the [platform admin console release notes](#). If you'd like to use this feature to back up Continuous Delivery for PE, upgrade to the latest version of the platform admin console.

New in this release:

- **Configure the Bolt PCP read timeout period.** To prevent job run timeouts caused by file sync delays, you can now adjust the Bolt Puppet Communications Protocol (PCP) timeout period. Learn more in [Adjust the timeout period for jobs](#) on page 164.
- **Reduced resource requirements for high availability (HA) installations.** Services that can run multiple replicas now default to running on two replicas in an HA cluster rather than three. This change maintains the former level of failure resistance while reducing resource requirements.
- **Usability improvements.** Version 4.5.0 introduces several improvements to the design and usability of the web UI and platform admin console, including:
 - Display text on the **Config** page has been updated to clarify the purpose and operation of the optional and advanced configuration sections.

Resolved in this release:

- The correct CA certificate is now passed to agents when switching between certificate generation methods and redeploying the application.
- Graphs shown on the application dashboard in the platform admin console no longer double-count resource use for pods using containerd.

Version 4.4.2

Released 13 April 2021.

Important: Version 4.4.2 includes several fixes that impact how Continuous Delivery for PE interacts with GitLab repositories that use nested groups (also called subgroups). In order to take advantage of these fixes, you must delete and re-add any control repos or modules in Continuous Delivery for PE created in version 4.4.1 or earlier that connect to GitLab repositories that use nested groups.

Resolved in this release:

- Webhooks between Continuous Delivery for PE and GitLab repositories that use nested groups now correctly trigger pipeline runs.
- Links on control repo and module details pages to GitLab repositories that use nested groups now resolve correctly.
- When selecting a GitLab repository in the Continuous Delivery for PE web UI, the list of results is now correctly filtered by the selected organization or user.

Version 4.4.1

Released 29 March 2021.

Resolved in this release:

- The impact analysis details page for modules now appears as expected.
- The deployment details page for modules now appears as expected.
- Continuous Delivery for PE now interprets errors from Code Manager correctly, and impact analysis runs are no longer impacted by parsing errors.
- The version 4.4.1 `.airgap` bundle includes an updated version of `puppet-dev-tools`.

Version 4.4.0

Released 11 March 2021.

New in this release:

- **Save and share favorite node table views.** You can now save the custom versions (views) of the node table that you create by using filters and columns to zero in on the data that's most relevant to your work. When you've created a view that you want to save and share with the members of your workspace, click **Save view**. You can see a list of all saved views for your workspace, mark your personal favorites for quick access, and switch between your favorite saved views from the **Nodes** page. For more information, see [Save custom node table views](#).

Note: If you're using an external PostgreSQL database with Continuous Delivery for PE, this new feature creates the need to configure an estate reporting database. Find more information in [Set up external PostgreSQL](#).

- **Built-in user groups for new workspaces.** Newly created workspaces now include three built-in user groups: Administrators, Operators, and Viewers. See the [Permissions reference](#) for details on the permissions included in each built-in user group.
- **Streamlined workflow for adding users to a workspace.** As part of the process of adding a new user to a workspace, you are now prompted to assign the user to one or more user groups (either the new built-in user groups or those you've created).
- **Configure login attempt limits.** You can now configure the number of unsuccessful login attempts that a user can make on Continuous Delivery for PE before their account is locked, as well as the length of time the account is locked and the length of time before the login attempt counter resets. For more information, see [Configure login attempt limits](#) on page 165.
- **OpenTelemetry.** You now have the option to use OpenTelemetry to perform distributed tracing on your Continuous Delivery for PE installation. OpenTelemetry configuration options are available on the **Config** page in the platform admin console.

Important: When using OpenTelemetry, you can choose to export the gathered data to your logs, to Jaeger over gRPC, or via OTLP. Be aware that if you choose the logging exporter option, the size of your Continuous Delivery for PE logs increases significantly. OpenTelemetry data you collect is not shared with Puppet, except in one specific case: if, while using the logging exporter, you generate and send a support bundle to Puppet, the support bundle contains OpenTelemetry data for your installation.

- **Preflight check improvements.** Preflight checks now verify that schedulable CPU and memory capacity are available for performing upgrades, and that the system is running Kubernetes version 1.17.0 or newer.
- **Usability improvements.** Version 4.4.0 introduces several improvements to the design and usability of the web UI and platform admin console, including:
 - You'll no longer see an option to reset your password on the login screen if LDAP is enabled for your installation.
 - A logout option is now available on the 403 error screen.
 - The **Config** page in the platform admin console has been streamlined in order to help you locate the configuration settings relevant to your installation.
 - To improve readability, the dashboard charts in the platform admin console displaying CPU usage and memory usage now only show data for the top five pods.

Resolved in this release:

- Clicking on the **Modules** breadcrumb at the top of a module's details page no longer results in a 404 error.
- When you update your CA certificate in the platform admin console, the change now takes effect immediately.
- The option to set a PuppetDB connection timeout period has been added back to the **Config** page in the platform admin console.

Version 4.3.3

Released 23 February 2021.

Resolved in this release:

- The integration between Azure DevOps and Continuous Delivery for PE now works as expected.
- Continuous Delivery for PE now deploys correctly if the root account email address entered on the **Config** page in the platform admin console contains uppercase letters.
- Ownership of a workspace can now be successfully transferred to a new owner whose username contains uppercase letters.

Version 4.3.2

Released 3 February 2021.

Note: Based on the results of ongoing internal testing along with feedback from users, we increased our recommended minimum system resource requirements for Continuous Delivery for PE 4.x. See [system requirements](#) for the current guidance.

New in this release:

- **List and filter your nodes by structured fact values.** You can now add columns displaying structured fact values in dot notation format (such as `docker.Architecture`, `ec2_metadata.hostname`, or `loadaverages.15m`) to your node table. In addition, you now have the option to use the values within your structured facts when creating a fact value filter on the **Nodes** page.

Resolved in this release:

- Webhooks for GitLab repositories that exist in nested groups now correctly trigger pipelines.
- Webhooks for Bitbucket Cloud control repos and modules that were added to Continuous Delivery for PE versions 4.2.0 and later now correctly trigger pipelines.
- Invalid characters are no longer present in the repository organization field for Bitbucket Cloud control repos, and jobs now clone these repositories correctly.
- Unnecessary repeated `The requested range is not satisfiable` errors are no longer included in the application log.
- Jobs included in pipeline stages no longer fail when attempting to download the control repo and job scripts.

Removed in this release:

- **Support for Puppet Enterprise version 2018.1.** PE 2018.1 reached the end of its support lifecycle on 31 January 2021.

Version 4.3.1

Due to an issue discovered after release, we retracted version 4.3.0. Version 4.3.1 is now the first version in the 4.3.x series.

Released 26 January 2021.

Note: A new version of the platform admin console was released on 7 December 2020. Please review the [release notes](#) and upgrade to the latest version of the platform admin console before upgrading Continuous Delivery for PE to version 4.3.1.

New in this release:

- **Support for Red Hat Enterprise Linux (RHEL) 8 and CentOS 8.** You can now run Continuous Delivery for PE on RHEL version 8 and CentOS version 8.
- **Ceph replaces MinIO for object storage.** Continuous Delivery for PE 4.x now uses Ceph for object storage instead of MinIO. New 4.3.1 installations use Ceph from the outset. For existing 4.x users, MinIO information is migrated to Ceph for you as part of the upgrade to version 4.3.1. To support this change, Ceph replication status is now collected as part of the support bundle.

Note: For existing 4.x users, the data migration to Ceph may cause the 4.3.1 upgrade process to take in excess of 15 minutes. Monitor the progress of the data export phase of the migration by running `kubectl logs job/cd4pe-migrate-object-store-v2 -c export` and watching the logs for a message similar to `Done. Downloaded 12990574 bytes in 63.0 seconds, 201.22 KB/s`. Next, monitor the data import phase of the migration by running `kubectl logs job/cd4pe-migrate-object-store-v2` and watching for a message similar to `Done. Uploaded 12990574 bytes in 267.8 seconds, 47.37 KB/s`. When both the export and import phases are shown as done in the logs, the migration is complete.

- **Default job timeout period increased.** The default job timeout period is now 30 minutes. This change reduces the chance that complex jobs time out before completion. See [Adjust the timeout period for jobs](#) on page 164 to learn more.
- **Usability improvements.** Version 4.3.1 introduces several improvements to the design and usability of the web UI, including:
 - The delete module icon is now correctly labeled.
 - Control repo icons are displayed when selecting a custom deployment policy for a deployment.

Resolved in this release:

- When logging in, users are now correctly directed to the last workspace they visited.
- Long branch names no longer overlap event status indicators in the **Events** area.
- Users are now less likely to encounter Docker Hub rate limits.
- The object storage migration process is now more robust and issues found in version 4.3.0 have been resolved.
- If an impact analysis task is canceled in a pipeline stage with the "any completed" auto-promotion criteria set, the pipeline run now stops at the canceled stage and does not continue.

Security notice:

- **CVE-2020-7946.** Source control tokens were displayed in plain text when trace-level logging was enabled. This issue has been resolved.
- **CVE-2020-27218.** An Eclipse Jetty vulnerability has been resolved.
- **CVE-2020-29361, CVE-2020-29362, and CVE-2020-29363.** The version of PostgreSQL included in Continuous Delivery for PE has been upgraded to resolve CVE-2020-29361, CVE-2020-29362, and CVE-2020-29363.

Version 4.2.4

Released 17 December 2020.

Resolved in this release:

- An issue with the webhooks for GitLab-based modules that were first added to Continuous Delivery for PE version 4.2.0 or newer has been resolved. Pipeline runs for these modules are now triggered correctly.

Version 4.2.3

Released 17 November 2020.

Resolved in this release:

- Webhooks now correctly trigger pipelines for GitLab repositories with names that include spaces or other unusual characters.

- The platform admin console now rate limits authentication attempts to prevent brute force attacks.

Note: Rate limiting does not currently apply to the Continuous Delivery for PE application web UI.

- This version includes an upgrade of PostgreSQL to version 12.5.

Note: The upgrade causes PostgreSQL to restart. In most cases, the downtime is expected to last less than a minute.

Version 4.2.2

Released 12 November 2020.

Resolved in this release:

- Impact analysis tasks on modules now manage prefixed environments correctly.
- This version includes an update to MinIO that addresses critical issues.

Note: This upgrade causes the MinIO service to be temporarily unavailable. In most cases, the downtime only lasts a few minutes.

Version 4.2.1

Released 5 November 2020.

Resolved in this release:

- Jobs no longer fail when triggered by pull requests from Bitbucket Cloud or Bitbucket Server repositories.
- The Bolt tasks included in the `puppetlabs-cd4pe` module version 3.0.1 and newer no longer fail with a `Connection reset by peer` error when run against Continuous Delivery for PE version 4.x.

Important: You must upgrade the `puppetlabs-cd4pe` module to version 3.0.1 or later in order to use its Bolt tasks.

Version 4.2.0

Released 3 November 2020.

New in this release:

- **Available memory setting.** A new setting on the **Config** page in the platform admin console lets you tune the total memory available to the Continuous Delivery for PE application. For more on the **Memory available for CD4PE** setting, see [Adjust available memory](#) on page 163.
- **Removal of harmful terminology.** Documentation for this release replaces the term “PE master” with “PE primary server,” and the term “master branch” with “main branch”. When adding a new control repo or module, Continuous Delivery for PE now looks for a “main” branch instead of a “master” branch. These changes are part of a company-wide effort to [remove harmful terminology from our products](#).

Resolved in this release:

- The eventual consistency deployment policy now runs more rapidly.
- Code Manager deployments triggered by Continuous Delivery for PE are now automatically retried if certain transient failures occur.
- PostgreSQL logs no longer include errors from health checks.
- If your workspace is connected to multiple PE instances with identically named nodes on each instance, the **Nodes** page now correctly reports the details of all identically named nodes.
- Impact analysis tasks are now case-insensitive when processing resource names.

- The LDAP group user attribute setting is now correctly applied when querying LDAP groups that use a custom attribute to identify members.

Important: If your installation previously used a group user attribute setting other than `dn`, you must set the group user attribute to `dn` in the root console after upgrading to version 4.2.0. Failure to do this breaks your installation's ability to correctly perform LDAP group lookups.

Security notice:

- **CVE-2020-25649.** A `jackson-databind` vulnerability has been resolved.
- **CVE-2020-15250.** A `JUnit4` vulnerability has been resolved.
- **CVE-2020-13956.** An Apache `HttpClient` vulnerability has been resolved.
- **Sonatype-2020-0926.** A security scanner may have detected a vulnerability in Continuous Delivery for PE version 4.1.x. However, Continuous Delivery for PE does not exercise the vulnerable code path and is not vulnerable.

Version 4.1.3

Released 15 October 2020.

Resolved in this release:

- Jobs now run successfully on pull requests opened from forked copies of source control repositories. This fix applies to all supported source control providers except Bitbucket Cloud and Bitbucket Server, which do not support pull requests from forks.
- Job logs are now shown correctly for all jobs run in a high availability environment.
- Continuous Delivery for PE no longer attempts to update webhooks on every startup if you have a backend URL that does not end with a trailing slash, or if you've used the webhook update tool in the root console. This fix means that GitHub and GitHub Enterprise no longer receive webhook payloads in an invalid format.
- Network policies now no longer restrict egress, supporting deployment of Continuous Delivery for PE on clusters that use tools such as Calico as a container network interface.
- You can now successfully enable TLS for the webhook proxy on port 8000. In offline installations, the local registry is now exposed on port 9001 for job hardware agents. Requests to these ports no longer time out.

Version 4.1.2

Released 8 October 2020.

Note: To upgrade to version 4.1.2 from version 4.0.1 or 4.0.0, you must first [Upgrade Continuous Delivery for PE](#) on page 134 and then [upgrade the platform admin console](#). Offline users, please see [Upgrade Continuous Delivery for PE offline](#) on page 134.

Resolved in this release:

- If your load balancer requires HTTP health checks, you can now opt into using Ingress settings that do not require Server Name Indication (SNI) for `/status`. Enable this setting in the **Customize endpoints** section of the **Config** tab in the platform admin console.
- Preflight checks for offline installations no longer hang with an `ImagePullBackoff` error on initial setup.
- Long-running deployments and jobs no longer fail with a `504 upstream request timeout` error.

Version 4.1.1

Released 29 September 2020.

Note: To upgrade to version 4.1.1, you must first [Upgrade Continuous Delivery for PE](#) on page 134 and then [upgrade the platform admin console](#). Offline users, please see [Upgrade Continuous Delivery for PE offline](#) on page 134.

New in this release:

- **Filter the Nodes page.** You can now apply custom filter combinations to your nodes table and zero in on the node data that's most relevant to your work. Available filters include fact value, most recent node change status, operating system, PE server, node group, and no-op status.
- **Snapshots.** Snapshots are point-in-time backups of your Continuous Delivery for PE deployment, which can be used to roll back to a previous state. You can create snapshots manually or set up a schedule to capture them automatically. To get started, see [Configure rollback snapshots](#).



CAUTION: Snapshots are a beta feature. As such, they may not be fully documented or work as expected; please explore them at your own risk.

- **Simplified port configuration for new installations.** The webhook service now defaults to HTTP on port 8000 and can be switched to HTTPS on the same port. In new offline installations, the local registry is exposed on port 9001 for job hardware agents. No action is required for existing installations that use webhook or registry hostnames; existing configurations work as they did previously.

Resolved in this release:

- Snapshots now successfully save to Amazon S3. In order to save your snapshots to an Amazon S3 bucket, you must upgrade the platform admin console to the latest version after upgrading to Continuous Delivery for PE version 4.1.1. See [Upgrade the platform admin console](#) for instructions.
- When exporting node table data, occasional failed queries to PuppetDB are now retried automatically, and no longer result in a failed export.

Version 4.0.1

Released 14 September 2020.

Resolved in this release:

- The export functionality on the 4.x **Nodes** page now works correctly.
- The container no longer hangs indefinitely in some circumstances after the host is rebooted.
- Network security rules now restrict inter-service communications.
- Local registry credentials are now stored as secrets.
- Certificate validation preflight checks now correctly refer to the local registry during offline installations.

Version 4.0.0

Released 25 August 2020.

New in this release:

- **New installer and administration platform.** The new Continuous Delivery for PE 4.x platform introduces a streamlined experience for installation, upgrades, license management, troubleshooting, and more. Use the new **platform admin console** to configure, monitor, upgrade to new versions in the 4.x series, back up, restore, and deploy your Continuous Delivery for PE installation.
- **Migrate your 3.x data to a 4.x installation.** To upgrade to the Continuous Delivery for PE 4.x series from a version in the 3.x series, see [Migrate 3.x data to 4.x](#) on page 136.
- **Update webhooks.** The new **Webhooks** tool in the root console updates your source control webhooks to point to the current installation. Use this tool as part of the 3.x to 4.x migration process, or any time you change the location of your Continuous Delivery for PE installation.

Removed in this release:

- **Continuous Delivery agent on job hardware.** Support for the Continuous Delivery agent was deprecated in version 3.4.0. Puppet agent-based job hardware is still supported.
- **Support for external Amazon DynamoDB and MySQL databases.** Support for external Amazon DynamoDB and MySQL databases was deprecated in version 3.1.0.

- **Support for external object storage.** The 4.x series replaces external Artifactory and Amazon S3 object storage with a built-in highly available object storage system.

Continuous Delivery for PE known issues

These are the known issues for the Continuous Delivery for Puppet Enterprise (PE) 4.x release series.

New OAuth apps cannot be created in Azure for Continuous Delivery for PE

As of April 2025, Microsoft has moved to using Entra ID to create Azure DevOps OAuth applications, which is not supported by Continuous Delivery for PE. OAuth applications created prior to April 2025 continue to function.

Impact Analysis is unable to handle Rich Data in Puppet Enterprise 2023.8

When running against Puppet Enterprise 2023.8.1 and later, Impact Analysis reports resources with Rich Data types (e.g., Deferred, Sensitive, etc.) as changed even when no changes have been made to those resources, due to a mismatch in data serialization formats. These false positives cause noise in the IA report, but can be ignored.

Multi-node Kubernetes clusters fail in 4.30.0

Incorrect access modes for a new PVC can cause Multi-Attach Error for Volume for users with multi-node Kubernetes clusters when scaling up replicas or using a RollingUpdate strategy for upgrade. If you use multi-node Kubernetes clusters, do not upgrade to 4.30.0.

Creating or editing a pipeline with deployments causes the deployments to stop running

Using the Continuous Delivery for PE 4.29.0 or 5.3.0 web UI to create or edit a pipeline that contains a deployment causes the pipeline to stop running its deployments. Jobs and Impact Analyses continue to work and pipelines continue to run, but deployments are skipped. This does not affect pipelines that do not contain deployments, pipelines managed with code, or versions of Continuous Delivery for PE prior to, or later than, 4.29.0 or 5.3.0.

Optional environment variables are not being passed to Docker containers

On the **Jobs** page, environment variables that are added to the **Optional configuration** section are unavailable in the Docker container. To work around this issue, [add a secret to the job](#) with the environment data you want to use. You can then reference the variable it creates to access your data.

A Puppet Enterprise (2023.4 or later) module is not compatible with Ruby 3.2

A module included with Puppet Enterprise 2023.4 or later is not compatible with Ruby 3.2. While this does not affect Puppet Enterprise users, it can affect users integrating Continuous Delivery for Puppet Enterprise (PE) with these versions of PE. If you are integrating Continuous Delivery for PE with Puppet Enterprise 2023.4 or later, install a version of the module compatible with newer Ruby versions using the following command:

```
puppet module upgrade puppetlabs-cd4pe_jobs
```

Continuous Delivery for Puppet Enterprise (PE) fails when used with Puppet 8 (Puppet Enterprise 2023.4 or later)

Puppet 8 (included with Puppet Enterprise 2023.4 or later) has removed legacy facts by default. Continuous Delivery for PE relies on the legacy fact `operatingsystem` in order to develop a list of systems that are available to be used as job hardware. If you are running Continuous Delivery for PE 5.x with Puppet Enterprise 2023.4 or later, you need to add the following to `puppet.conf`:

```
include_legacy_facts=true
```

Users trying to login using SAML SSO could see a 405 Method Not Allowed

The SAML redirect URL changed in version 4.22.0 to: `<YOUR_CD4PE_WEB_UI_ENDPOINT>/cd4pe/saml-auth`. Versions of Continuous Delivery for PE prior to 4.22.0 should continue to use: `<YOUR_CD4PE_WEB_UI_ENDPOINT>/saml-auth`.

When upgrading Continuous Delivery for PE and Puppet Application Manager, you lose connectivity to your cluster

The issue occurs either during or after an upgrade to Continuous Delivery for PE and Puppet Application Manager. In each scenario you eventually lose connectivity to your clusters in Puppet Application Manager and Kubernetes. You get errors that images are missing, but `containerd` and `kubelet` are running.

To address this issue, follow the instructions in [this knowledge base article](#).

Installing on 4 CPUs can hang with the new cd4pe pod stuck in a "Pending" status

Standalone installations on systems with 4 CPUs have an issue where upgrades and configuration changes can hang with the new `cd4pe` pod stuck in a "Pending" status.

To work around the issue temporarily delete a non-critical pod. For example, to delete the Prometheus pod use `kubectl delete pod -n monitoring -l app.kubernetes.io/instance=k8s`.

Once 4.22.0 is deployed, the best long term solution is to deselect **Enable Vault** in the Puppet Application Manager (PAM) UI under **Advanced configuration and tuning**. Vault is not used in Continuous Delivery for PE 4.22.0 and freeing the resources it consumes allows 4 CPU installations to upgrade without this problem.

Upgrading to 4.12.0 can delete the cd4pe Ingress

In busier Kubernetes clusters, upgrading to Continuous Delivery for PE version 4.12.0 can delete the `cd4pe` Ingress. If the application is unavailable after upgrading and `kubectl get ingress cd4pe` returns an empty list, redeploy the deployed version to recreate the Ingress.

Impact analysis can require additional disk space

Impact analysis branches are not always pruned after impact analysis is complete. This creates orphaned environments that fill up the `/etc/puppetlabs/code/environments` directory on the PE primary server. If you are experiencing storage capacity issues with impact analysis, increase the storage capacity on the `/etc/puppetlabs/code/environments` mount.

Impact analysis tasks fail when using the satellite_pe_tools module

When using the `satellite_pe_tools` module with Continuous Delivery for PE, impact analysis tasks fail with a "Internal Server Error: org.jruby.exceptions.RuntimeError: (Error) PuppetDB not configured, please provide facts with your catalog request" error. This issue occurs because the API endpoint used to collect facts during impact analysis tasks errors if the `fact_terminus` parameter is set to `satellite` or any value other than `puppetdb`. This issue is resolved in the versions of Puppet Server included in PE versions 2021.4 and 2019.8.9.

Impact analysis tasks fail when using Puppet Enterprise versions 2021.2 or 2019.8.7

Impact analysis fails with a `R10K::Module::Forge cannot handle option 'default_branch_override'` error. If you're using PE version 2021.2 or 2019.8.7, you must update the `pe-r10k` package by [following the instructions in this Puppet Support article](#) to continue to use impact analysis. After you update the package, you can update to future versions of PE using the installer as normal.

Preflight check failure when using Puppet Application Manager versions 1.19 or 1.20

Puppet Application Manager versions 1.19 and 1.20 display an `Analyzer Failed: invalid analyzer` error during the preflight checks when deploying a new version of Continuous Delivery for PE. This error relates to

analyzers supported by version 1.24 and newer versions of the Puppet Application Manager. The error can be safely ignored. To resolve the failure and take advantage of the new preflight checks, [upgrade Puppet Application Manager](#) to the latest version.

Deployments might time out on node groups with complex rules

When using a built-in deployment policy other than the eventual consistency policy to deploy changes to a node group with highly complex rules, the deployment times out in some cases.

A PE instance cannot be integrated if `dns_alt_names` is not set on the master certificate

If the Puppet master certificate for your PE instance does not have `dns_alt_names` configured, attempting to integrate the instance with Continuous Delivery for PE fails with a `We could not successfully validate the provided credentials against the Code Manager Service` error. The master certificate must be regenerated before PE is integrated with Continuous Delivery for PE. For instructions, see [Regenerate master certificates](#) in the PE documentation.

Jobs fail when using chained SSL certificates on Windows

If you are using Continuous Delivery for PE with SSL configured to use chained certificates, attempts to run jobs on Windows job hardware fail.

Custom deployment policies aren't initially shown for new control repos

When your first action in a newly created control repo is to add a deployment to a pipeline, any custom deployment policies stored in the control repo aren't shown as deployment policy options. To work around this issue, click **Built-in deployment policies** and then click **Custom deployment policies** to refresh the list of available policies.

Puppet Application Manager

Before you can begin using Continuous Delivery for PE, you must install Puppet Application Manager (PAM). PAM is an administrative console that provides tools for managing Continuous Delivery for PE and other Puppet applications.

Note: Puppet Application Manager was previously known as the *platform admin console* in earlier versions of the Continuous Delivery for PE documentation.

What does PAM do?

The PAM installation process sets up a managed Kubernetes cluster (or, if you prefer, adds PAM to your existing cluster). Continuous Delivery for PE runs on this Kubernetes cluster, and PAM manages the cluster for you.

In the PAM UI, you can configure Continuous Delivery for PE, monitor the cluster's activity, perform upgrades, and back up your installation.

How do I use PAM to deploy Continuous Delivery for PE?

Once the cluster is ready, upload your Continuous Delivery for PE license and provide any needed configuration details about your installation in the PAM UI. You can then deploy the latest version of Continuous Delivery for PE with one click whenever you're ready.

- [Welcome to Puppet Application Manager \(PAM\)](#) on page 48

Puppet Application Manager is an administrative console where you can install, access, and manage your Puppet applications. It is also where you can go to access upgrades to new Puppet applications releases.

- [Architecture overview](#) on page 62

Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

- [PAM system requirements](#) on page 66

You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

- [Component versions in PAM releases](#) on page 79

These tables show the versions of components included in recent Puppet Application Manager (PAM) releases.

- [Install PAM](#) on page 80

You can install Puppet-supported Puppet Application Manager on a single node or in an HA configuration. Both online and offline install packages are available. You can also install it on an existing Kubernetes cluster.

- [Working with Puppet applications](#) on page 101

You can install and upgrade Puppet applications using the Puppet Application Manager UI.

- [Maintenance and tuning](#) on page 105

Follow these guidelines when you're tuning or performing maintenance on a node running Puppet Application Manager (PAM).

- [Upgrading PAM on a Puppet-supported cluster](#) on page 106

Upgrade Puppet Application Manager (PAM) on a Puppet-supported cluster to take advantage of new features and bug fixes, and to upgrade your cluster to the latest version of Kubernetes when one is available.

- [Upgrading PAM on a customer-supported cluster](#) on page 110

Upgrade Puppet Application Manager (PAM) on your own Kubernetes cluster to take advantage of new features and bug fixes.

- [Backing up PAM using snapshots](#) on page 112

Snapshots are point-in-time backups of your Puppet Application Manager (PAM) deployment, which can be used to roll back to a previous state or restore your installation into a new cluster for disaster recovery.

- [Migrating PAM data to a new system](#) on page 114

By using a snapshot, you can migrate your data to a new Puppet Application Manager (PAM) instance.

- [Disaster recovery with PAM](#) on page 120

It is important to prepare your system and regularly capture full snapshots. This backs up your data and makes it easier to restore your system if disaster recovery is needed.

- [Troubleshooting PAM](#) on page 121

Use this guide to troubleshoot issues with your Puppet Application Manager installation.

Related information

[Deploy Continuous Delivery for PE](#) on page 130

After installing Puppet Application Manager (PAM), specify your initial configuration setting and deploy Continuous Delivery for Puppet Enterprise (PE) for the first time.

[Deploy Continuous Delivery for PE offline](#) on page 131

Use these instructions to install Continuous Delivery for Puppet Enterprise (PE) in an airgapped or offline environment where the Continuous Delivery for PE host server does not have direct access to the internet.

Welcome to Puppet Application Manager (PAM)

Puppet Application Manager is an administrative console where you can install, access, and manage your Puppet applications. It is also where you can go to access upgrades to new Puppet applications releases.

Useful links:

Puppet Application Manager docs links	Other useful places
<p>Before you install</p> <ul style="list-style-type: none"> Release notes System requirements <p>Install Puppet Application Manager</p> <ul style="list-style-type: none"> PAM standalone online installation on page 90 PAM standalone offline installation on page 93 PAM HA online installation on page 83 PAM HA offline installation on page 87 <p>Upgrading, disaster recovery, and troubleshooting</p> <ul style="list-style-type: none"> Upgrading PAM on a Puppet-supported cluster on page 106 Backing up PAM using snapshots on page 112 Disaster recovery with PAM on page 120 Troubleshooting PAM on page 121 	<p>Docs for related Puppet products</p> <ul style="list-style-type: none"> Continuous Delivery for PE Comply <p>Get support</p> <ul style="list-style-type: none"> Support Upgrade your support plan <p>Share and contribute</p> <ul style="list-style-type: none"> Engage with the Puppet community Puppet Forge Open source projects from Puppet on GitHub <p>External resources</p> <ul style="list-style-type: none"> Getting started with Kubernetes Off-The-Shelf software (KOTS)

PAM UI

The Puppet Application Manager (PAM) UI provides administration functionality where you can access and manage your Puppet applications.

PAM console menu

Use the console menu at the top of the Puppet Application Manager UI to manage Puppet Application Manager itself. It has three tabs of interest to us:

- Use the **Dashboard** tab to:
 - Manage your applications
 - See version history
 - Set application configuration settings
 - Access support bundles for troubleshooting
 - Manage licenses
 - View files
 - Configure registry settings
- Use the **Cluster Management** tab to view current information on the nodes in your cluster. You can also use this tab to drain, and add nodes to your cluster.
- Use the **Snapshots** tab to create point-in-time backups of your deployment, which can be used to roll back to a previous state, or restore your installation into a new cluster for disaster recovery. For more information, see [Backing up PAM using snapshots](#) on page 112.

You can also use the console menu to **Add a new application** and to log out of Puppet Application Manager.

Application monitoring graphs

When you have Prometheus installed, the **Dashboard** tab has an **Application** sub-tab that provides several simplified graphs for tracking overall health of the system.

- **Node CPU Usage (%)** shows when hosts are getting overwhelmed (high % usage).
- **Node Memory Usage (%)** shows when hosts are reaching full memory capacity that may result in processes being killed due to out-of-memory errors.

- **Node Available Storage (%)** shows when hosts are running out of storage. At 15%, pods may start to be evicted or reads/writes on databases are paused until more storage is made available.
- **Volume Available Storage (%)** shows when application persistent volumes are getting full (low %) that may lead to problems with a particular application. Note that -

Note: As of the 30 June 2021 Puppet Application Manager release, the monitoring/Prometheus-Kubernetes pods limit their storage use and are expected to never fall below 10% available storage.

Puppet Application Manager HA architectures include Prometheus and Grafana. Metrics about how the system is working are sent to Prometheus, and can be displayed with Grafana. Grafana credentials are printed during install, or can be retrieved later with the following command:

```
kubectl -n monitoring get secret grafana-admin -o go-template='{{index .data "admin-user" | base64decode}}:{{index .data "admin-password" | base64decode}}'
```

Related information

[Backing up PAM using snapshots](#) on page 112

Snapshots are point-in-time backups of your Puppet Application Manager (PAM) deployment, which can be used to roll back to a previous state or restore your installation into a new cluster for disaster recovery.

Release notes

PAM release notes

These are the new features, enhancements, resolved issues, and deprecations for Puppet Application Manager.

Restriction: Because kURL does not support upgrading more than two Kubernetes minor release versions at once, if you're upgrading from an older version of PAM, you might need to follow a specific upgrade path to avoid failures. For example, PAM version 1.80.0 uses Kubernetes version 1.21.x, so you can upgrade up to PAM 1.91.3 (Kubernetes version 1.23.x), but not to PAM 1.94.0 (Kubernetes version 1.24.x). To determine the specific upgrade path for your installation, please check the [table of Kubernetes versions](#) for each version of PAM.

4 March 2025 (Puppet Application Manager 1.112.4-r1)

Note: Customers using the `pam_firewall` module must upgrade the module to version 1.0.5 prior to upgrading PAM to 1.112.4-r1.

New in this release:

- **Component upgrades to address security issues.** This version upgrades the following:

Note: Before updating, ensure MinIO has 10GB of free space.

- kURL: v2025.02.14-0
- containerd: 1.7.25
- Flannel: 0.26.4
- Velero: 1.15.2
- Metrics Server: 0.7.2
- Prometheus: 0.80.0-69.2.0
- OpenEBS: 4.2.0
- MinIO: 2024-11-07T00-52-20Z

Resolved in this release:

- This release contains an update to containerd 1.7.25, which resolves a memory leak.

22 October 2024 (Puppet Application Manager 1.112.4)

Note: Customers using the `pam_firewall` module must upgrade the module to version 1.0.5 prior to upgrading PAM to 1.112.4.

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.30.5.

Important upgrade information: The upgrade process takes place on all nodes, upgrading Kubernetes to version 1.30.5 on each. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.

Additionally, please keep in mind that [kURL can only be upgraded two minor versions at a time](#) on page 125.

- **Component upgrades to address security issues.** This version upgrades the following:

Note: Before updating, ensure MinIO has 10GB of free space.

- KOTS: 1.112.4
- kURL: v2024.09.26-0
- containerd: 1.6.33
- Flannel: 0.25.6
- Project Contour: 1.30.0
- Velero: 1.14.0
- Metrics Server: 0.6.4
- ekco: 0.28.7
- Prometheus: 0.76.1-62.6.0
- Registry: 2.8.3
- OpenEBS: 4.1.0
- MinIO: 2024-08-26T15-33-07Z
- Rook: 1.12.8
- Goldpinger: 3.10.0-6.2.0

23 July 2024 (Puppet Application Manager 1.110.0)

New in this release:

- **Component upgrades to address security issues.** This version upgrades the following:

Note: Before updating, ensure MinIO has 10GB of free space.

- KOTS: 1.110.0
- kURL: v2024.07.02-0
- containerd: 1.6.32
- Flannel: 0.25.4
- Project Contour: 1.29.0
- Velero: 1.13.2
- Metrics Server: 0.6.4
- ekco: 0.28.7
- Prometheus: 0.74.0-59.0.0
- Registry: 2.8.3
- OpenEBS: 4.0.0
- MinIO: 2024-05-10T01-41-38Z
- Rook: 1.12.8
- Goldpinger: 3.10.0-6.2.0

21 May 2024 (Puppet Application Manager 1.109.0)

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.28.9.

Important upgrade information: The upgrade process takes place on all nodes, upgrading Kubernetes to version 1.28.9 on each. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.

Additionally, please keep in mind that [kURL can only be upgraded two minor versions at a time](#) on page 125.

- **Component upgrades to address security issues.** This version upgrades the following:

Note: Before updating, ensure MinIO has 10GB of free space.

- KOTS: 1.109.0
- kURL: v2024.05.03-0
- containerd: 1.6.31
- Flannel: 0.25.1
- Project Contour: 1.28.3
- Velero: 1.13.2
- Metrics Server: 0.6.4
- ekco: 0.28.6
- Prometheus: 0.73.1-58.1.1
- Registry: 2.8.3
- OpenEBS: 4.0.0
- MinIO: 2024-04-06T05-26-02Z
- Rook: 1.12.8
- Goldpinger: 3.10.0-6.2.0

26 March 2024 (Puppet Application Manager 1.108.0)

New in this release:

- **Component upgrades to address security issues.** This version upgrades the following:

Note: Before updating, ensure MinIO has 10GB of free space.

- KOTS: 1.108.0
- kURL: v2024.02.23-0
- containerd: 1.6.28
- Flannel: 0.24.2
- Project Contour: 1.27.0
- Velero: 1.12.3
- Metrics Server: 0.6.4
- ekco: 0.28.4
- Prometheus: 0.71.2-56.6.0
- OpenEBS: 3.10.0
- MinIO: 2024-02-17T01-15-57Z

13 February 2024 (Puppet Application Manager 1.107.0)

New in this release:

- **Component upgrades to address security issues.** This version upgrades the following:

Note: Before updating, ensure MinIO has 10GB of free space.

- KOTS: 1.107.0
- kURL: v2024.01.09-0
- containerd: 1.6.26
- Flannel: 0.24.0
- Project Contour: 1.27.0
- Registry: 2.8.3
- Velero: 1.12.2
- ekco: 0.28.4
- Prometheus: 0.70.0-55.0.0
- OpenEBS: 3.10.0
- MinIO: 2024-01-01T16-36-33Z
- Rook: 1.12.8

7 November 2023 (Puppet Application Manager 1.103.3)

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.28.2.

Important upgrade information: The upgrade process takes place on all nodes, and first upgrades Kubernetes to version 1.27.6 before upgrading to version 1.28.2 on each. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.

Additionally, please keep in mind that [kURL can only be upgraded two minor versions at a time](#) on page 125.

- **Component upgrades to address security issues.** This version upgrades the following:

Note: Before updating, ensure MinIO has 10GB of free space.

- KOTS: 1.103.3
- kURL: v2023.10.26-0
- containerd: 1.6.24
- Flannel: 0.22.3
- Project Contour: 1.26.1
- Registry: 2.8.3
- Velero: 1.12.1
- OpenEBS: 3.9.0
- MinIO: 2023-10-16T04-13-43Z
- Rook: 1.12.6

26 September 2023 (Puppet Application Manager 1.102.2)

New in this release:

- **Migrated from Weave to Flannel.** Flannel has replaced Weave as the Kubernetes CNI on Puppet-supported clusters, as Weave is no longer supported. The installation has additional interactive prompts to support this change.

Important upgrade information:

- IPv6 and dual-stack networks are not supported on Flannel.
- Pod-to-pod networking now depends on UDP port 8472 being open instead of ports 6783 and 6784.

- **Added a host preflight.** Added a host preflight in the installer to stop installation if the installer detects the presence of a default REJECT rule in the FORWARD chain of iptables.

Important upgrade information: This is a known issue with the Flannel installation. To check for a REJECT rule in the FORWARD chain of iptables, run:

```
iptables -vL FORWARD
```

If there are any REJECT rules, those rules must be removed prior to the upgrade. They can be restored afterwards.

- **Component upgrades to address security issues.** This version upgrades the following:

Note: Before updating standalone installations, ensure there is at least 10GB of free space in `/var/openebs` to allow for migration of MinIO in this release.

- KOTS: 1.102.2
- kURL: v2023.09.15-0
- containerd: 1.6.22
- Weave: REMOVED
- Flannel: 0.22.2
- Project Contour: 1.25.2
- Velero: 1.11.1
- Kubernetes Metrics Server: 0.6.4
- ekco: 0.28.3
- Prometheus: 0.68.0-51.0.0
- OpenEBS: 3.8.0
- MinIO: 2023-09-04T19-57-37Z
- Rook: 1.12.3

Note: If you are using the [firewall module](#) to manage your PAM install, you must update it to version 1.0.4 to support this PAM release.

18 July 2023 (Puppet Application Manager 1.100.3)

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.26.6.

Important upgrade information: The upgrade process takes place on all nodes, and first upgrades Kubernetes to version 1.25 before upgrading to version 1.26.6 on each. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.

Additionally, please keep in mind that [kURL can only be upgraded two minor versions at a time](#) on page 125.

- **Component upgrades to address security issues.** This version upgrades, adds, and removes the following:

Note: Before updating, ensure MinIO has 10GB of free space.

- KOTS: 1.100.3
- kURL: v2023.06.27-0
- Prometheus: 0.65.2-46.8.0
- OpenEBS: 3.7.0
- MinIO: 2023-06-19T19-52-50Z
- Rook: 1.11.8

Note: If you are using the [firewall module](#) to manage your PAM install, you must update it to version 1.0.3 to support this PAM release.

8 June 2023 (Puppet Application Manager 1.99.0)

New in this release:

- **Component upgrades to address security issues.** This version upgrades the following:

Note: Before updating, ensure MinIO has 10GB of free space.

- KOTS: 1.99.0
- kURL: v2023.05.22-0
- containerd: 1.6.21
- Weave: 2.8.1-20230417
- Project Contour: 1.25.0
- Registry: 2.8.2
- Velero: 1.11.0
- ekco: 0.27.1
- Prometheus: 0.65.1-45.28.0
- OpenEBS: 3.6.0
- MinIO: 2023-05-18T00-05-36Z
- Rook: 1.11.5
- Goldpinger: 3.7.0-6.0.1

Note: For offline HA installs the Rook update in this release can cause significant downtime (around 4 hours) while downloading additional files. It is possible to [do some of this prior to upgrading](#) Puppet Application Manager from 1.97.0 to 1.99.0 to decrease the downtime.

25 April 2023 (Puppet Application Manager 1.97.0)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of OpenEBS to version 3.5.0, an upgrade of kURL to v2023.04.11-0, an upgrade of containerd to 1.6.20, an upgrade of Weave to version 2.8.1-20230324, an upgrade of Project Contour to version 1.24.3, an upgrade of ekco to 0.26.5, an upgrade of Velero to version 1.10.2, an upgrade of the Prometheus bundle to version 0.63.0-45.9.1, and upgrade of Kubernetes Metrics Server to version 0.6.3, an upgrade of KOTS to 1.97.0, an upgrade of MinIO to version 2023-03-24T21-41-23Z, and an upgrade of Goldpinger to 3.7.0-5.6.0.

Note: Before updating, ensure MinIO has 10GB of free space.

Deprecated in this release:

- **force-reapply-addons flag.** Starting with Puppet Application Manager 1.97.0, the `force-reapply-addons` flag is deprecated and generates a warning on use. This flag is only required when upgrading to a Puppet Application Manager version prior to 1.97.0.

28 February 2023 (Puppet Application Manager 1.94.0)

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.24.10.

Important upgrade information: The upgrade process takes place on all nodes, and first upgrades Kubernetes to version 1.24.10 on each. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.

Additionally, because [kURL can only be upgraded two minor versions at a time](#) on page 125, if you're on PAM version 1.80.0 or earlier, you must upgrade to PAM 1.81.1 before upgrading to PAM 1.94.0.

- This release also includes component upgrades to address security issues and general bug fixes.

10 January 2023 (Puppet Application Manager 1.91.3)

New in this release:

- **Component upgrades to address security issues and support RHEL 8.7.** This version upgrades the following:

Note: Before updating, ensure MinIO has 10GB of free space.

- KOTS: 1.91.3
- MinIO: 2022-10-20T00-55-09Z
- OpenEBS: 3.3.0
- Prometheus: 0.60.1-41.7.3
- ekco: 0.26.1
- Velero: 1.9.4
- Project Contour: 1.23.1
- kURL: v2022.12.12-0
- Weave: 2.8.1-20221122
- Goldpinger: 3.7.0-5.5.0

28 September 2022 (Puppet Application Manager 1.81.1)

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.23.9.

Important upgrade information: The upgrade process takes place on all nodes, and first upgrades Kubernetes to version 1.22 before upgrading to version 1.23.9. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.

Additionally, because [kURL can only be upgraded two minor versions at a time](#) on page 125, if you're upgrading from PAM version 1.56.0 or earlier, you must upgrade to PAM 1.80.0 before upgrading to PAM 1.81.1.

For legacy installations, Kubernetes remains on version 1.19.15. If you're not sure which installation type you're running, see [How to determine your version of Puppet Application Manager](#).

16 August 2022 (Puppet Application Manager 1.80.0)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version upgrades containerd to 1.4.13, KOTS to 1.80.0, ekco to 0.19.6, and Goldpinger to 3.5.1-5.2.0.

Resolved in this release:

- Fixed an issue where legacy encryption keys didn't load properly during snapshot restores.

2 August 2022 (Puppet Application Manager 1.76.2)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of OpenEBS to version 3.2.0, an upgrade of Weave to version 2.8.1-20220720, an upgrade of Project Contour to version 1.21.1, and an upgrade of MinIO to version 2022-07-17T15-43-14Z.

Note: Before updating, ensure MinIO has 10GB of free space.

20 July 2022 (Puppet Application Manager 1.76.1)

New in this release:

- **Support for Red Hat Enterprise Linux version 8.6.** Beginning with version 1.76.1, PAM can be successfully installed on systems running Red Hat Enterprise Linux version 8.6.
- **More log data is now retained.** To ensure that you and our Support team have the data you need in debugging scenarios, the size of the pod logs has been increased from 10 files of 10MiB each to 10 files of 50MiB each. This change increases the storage used in `/var/log/pods` by 400MiB.
- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of Velero to version 1.9.0 and an upgrade of the Prometheus bundle to version 0.57.0-36.2.0.
- **Other component upgrades.** This version also includes an upgrade of Registry to version 2.8.1 and an upgrade of MinIO to version 2022-07-06T20-29-49Z.

Note: Before updating, ensure MinIO has 10GB of free space.

Resolved in this release:

- Velero pods no longer get stuck in a pending state when creating a snapshot to be saved to internal storage on a Puppet-supported cluster.

23 June 2022 (Puppet Application Manager 1.72.1)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of ekco to version 0.19.2 and an upgrade of kURL to v2022.06.17-0.

26 May 2022 (Puppet Application Manager 1.70.1)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of Project Contour to version 1.21.0, an upgrade of Velero to version 1.8.1, and an upgrade of the Prometheus bundle to version 0.56.2-35.2.0.

Resolved in this release:

- Image garbage collection in Kubernetes installer-created clusters (embedded clusters) no longer removes images outside of the application's dedicated registry namespace.
- The **Deploy** button is now present in newly updated versions after the configuration is updated from the previously deployed version.
- Legends are now shown properly for the performance graphs on the dashboard.

12 April 2022 (Puppet Application Manager 1.68.0)

New in this release:

- **Install a specific version of an application.** When installing a Puppet application using the automated installation method, you now have the option to specify the application's version by passing the `--app-version-label=<version>` flag to the `kubectl kots install` command. For more information, go to [Automate PAM and Puppet application online installations](#) on page 95.
- **Status reporting improvements.** The status reporting tools can now detect when an application is being upgraded.
- **Component upgrades to address CVEs.** To address various CVEs in Envoy, this version includes an upgrade of Project Contour to version 1.20.1.
- **Other component upgrades.** This version includes an upgrade of KOTS to version 1.68.0, which enables Kubernetes audit event logging by default and adds a 1 GB storage requirement for `/var/log/apiserver`.

Resolved in this release:

- During image garbage collection, images still in use by the cluster are no longer in danger of being deleted from the private registry in a Kubernetes installer-created cluster.

1 March 2022 (Puppet Application Manager 1.64.0)

Resolved in this release:

- Diffs are now shown correctly in the PAM UI.
- The OpenSSL package is no longer a prerequisite for successful installation on newer Red Hat Enterprise Linux 7 systems.
- You can now successfully install Puppet Application Manager on Red Hat Enterprise Linux 8 systems without the need to force-install the kurl-local audit-libs library.

17 February 2022 (Puppet Application Manager 1.62.0)

Important: Version 1.0.2 of the `puppetlabs/pam_firewall` module is now available. To avoid conflicts, upgrade the module **before** upgrading Puppet Application Manager to version 1.62.0.

New in this release:

- **Kubernetes version upgrade.** For standalone and HA installations, this version includes an upgrade of Kubernetes to version 1.21.8.

Important upgrade information: The upgrade process takes place on all nodes, and first upgrades Kubernetes to version 1.20 before upgrading to version 1.21.8. For a three-node cluster, you can expect the upgrade process to take around an hour. Confirmations are required during the upgrade process.

For legacy installations (installed before May 2021), this version includes an upgrade of Kubernetes to version 1.19.15.

Tip: See [How to determine your version of Puppet Application Manager](#) if you're not sure which installation type you're running.

- **Prometheus enabled on standalone architecture.** Beginning with version 1.62.0 Prometheus is enabled by default on all new and existing standalone Puppet Application Manager installations. Prometheus requires an additional 350m CPU and 500MiB of memory, so ensure your system is properly sized before upgrading. Prometheus is an optional component; if you need to disable it to conserve resources, see [Optional components](#) on page 124.
- **Automatic certificate rotation.** By default, the self-signed certificates used by Project Contour and Envoy expire after one year. This version includes an update that auto-rotates those certificates before they expire.
- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of containerd to version 1.4.12.
- **Other component upgrades.** This version includes an upgrade of KOTS to version 1.62.0.

Deprecated in this release:

- **Legacy architecture.** The legacy architecture, which was the version of Puppet Application Manager available for installation prior to May 2021, is now deprecated. (See [How to determine your version of Puppet Application Manager](#) if you need to confirm whether you're running the legacy architecture.) The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. Puppet will continue to update legacy architecture components other than Kubernetes until 30 June 2022. If security advisories against Kubernetes 1.19 arise, the remediation path is to migrate to one of the newer architectures by following the instructions in [Migrating PAM data to a new system](#) on page 114.

Important: Before beginning the migration process from a legacy deployment you must upgrade to PAM version 1.62.0 with the `force-reapply-addons` flag included in the upgrade command. Find upgrade instructions at [PAM legacy upgrades](#) on page 109 and [PAM offline legacy upgrades](#) on page 109.

30 November 2021 (Puppet Application Manager 1.56.0)

This release includes an upgrade of KOTS to version 1.56.0, which adds the following improvements:

- **Improved support bundles:** Adds an option to upload a support bundle directly from Puppet Application Manager.
- **Improved troubleshooting:** Adds detailed information on failing pods to the **Troubleshoot** tab.

6 October 2021 (Puppet Application Manager 1.52.1)

New in this release:

- **Improved statuses.** More granular status levels are now available from the **Application** tab.
- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of Kubernetes to 1.19.15.
- **Other component upgrades.** This version includes an upgrade of KOTS to version 1.52.1.

Resolved in this release:

- Generating a support bundle no longer results in unusually high memory use.
- Preflight check logs post to info level for progress messages and to error level for error messages.

25 August 2021 (Puppet Application Manager 1.49.0)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of Kubernetes to 1.19.13, an upgrade of Project Contour to 1.18.0, and an upgrade of Velero to 1.6.2.
- **Goldpinger.** High availability architectures now include Goldpinger, which aids the debugging of network issues.
- **containerd upgrade.** This version includes an upgrade of containerd to version 1.4.6, and removes the need to use the `force-reapply-addons` option when upgrading.
- **Other component upgrades.** This version includes an upgrade of KOTS to version 1.49.0, an upgrade of ekco to 0.11.0, an upgrade of Prometheus to 0.49.0, and an upgrade of Rook to 1.5.12.

30 June 2021 (Puppet Application Manager 1.44.1)

New in this release:

- **Certificate auto-rotation for standalone architecture.** Certificates are now automatically rotated for the Kubernetes API and Puppet Application Manager UI in the standalone architecture. With this change, certificate auto-rotation is now supported in all Puppet Application Manager architectures.
- **Rook upgrades.** This version includes an upgrade of Rook in the high availability architecture to 1.5.11 and the version of Rook in the legacy architecture to 1.0.4-14.2.21. These upgrades address a vulnerability in Ceph components (CVE-2021-20288).
- **Prometheus upgrade.** This version includes an upgrade of Prometheus in the high availability and legacy architectures to 0.48.1. Additionally, Prometheus disk usage is now limited in order to preserve the storage space required for the usage charts on the **Application** tab.
- **Other component upgrades.** This version includes an upgrade of KOTS to version 1.44.1, an upgrade of Project Contour to version 1.15.1, and an upgrade of Weave to version 2.8.1.

Resolved in this release:

- Snapshots can now successfully use the **Other S3-Compatible Storage** option as the storage destination.

To apply this update, add the `force-reapply-addons` option during upgrade. For example:

```
curl <url> | bash -s force-reapply-addons
```

26 May 2021

New in this release:

- **runC.** The version of runC has been upgraded to v1.0.0-rc95 to address CVE-2021-30465.

Known issues in this release:

- Running the KOTS installer with the `airgap` and `kurl-registry-ip` flags results in an error.
As a workaround (if you do not have any applications already installed in the cluster), delete the registry service, recreate the registry service IP and then re-run the installation script with the `kurl-registry-ip` flag.

10 May 2021 (Puppet Application Manager 1.40.0)

New in this release:

- Distinct architectures for standalone and high availability deployments of the Puppet Application Manager platform. Standalone supports lower system requirements and resolves inherent flaws in using Ceph on a single node. High availability uses an updated version of Rook for faster, more reliable distributed storage.

Note: It is not possible currently to upgrade to these architectures from existing installations. However, migrating applications between them is on the roadmap for a future release.

- The previous architecture is maintained as the legacy configuration. This version includes an upgrade of Kubernetes to 1.19.10; this upgrade process upgrades through Kubernetes 1.18, and happens on all nodes. It can take ~1 hour to do for a 3-node cluster, and requires confirmations during that period. It also includes an upgrade of Project Contour to version 1.14.1, adds Metrics Server 0.4.1, an upgrade of ekco to 0.10.1, and an upgrade of Prometheus to 2.26.0.

For more information on legacy upgrades, see [PAM legacy upgrades](#) on page 109.

15 April 2021 (Puppet Application Manager 1.38.0)

New in this release:

- **Snapshots.** Puppet Application Manager now supports full (instance-level) snapshots, which can be used for application rollbacks and disaster recovery. For more information, see [Backing up Puppet Application Manager using snapshots](#).
- **Component upgrades.** This version includes an upgrade of KOTS to version 1.38.0.

17 February 2021 (Puppet Application Manager 1.29.3)

New in this release:

- **Support for Ubuntu 20.04.** You can now run Puppet Application Manager on Ubuntu 20.04.
- **Component upgrades.** This version includes an upgrade of Prometheus to version 2.22.1 and Prometheus Operator to version 0.44.1, an upgrade of KOTS to version 1.29.3, an upgrade of Project Contour to version 1.12.0, and an upgrade of ekco to version 0.10.0.

3 February 2021 (Puppet Application Manager 1.29.2)

New in this release:

- **Component upgrades.** This version includes an upgrade of KOTS to version 1.29.2, an upgrade of Project Contour to version 1.11.0, and an upgrade of `containerd` to version 1.4.3.

Resolved in this release:

- During their initial preflight checks, new installations now pull images successfully and no longer report a `Failed to pull image` error.

7 December 2020

New in this release:

- **Support for Red Hat Enterprise Linux (RHEL) 8 and CentOS 8.** You can now run Puppet Application Manager on RHEL version 8 and CentOS version 8. To support this change, `containerd` is now used independently of Docker during the installation process.

- **Component upgrades.** This version includes an upgrade of Kubernetes to version 1.17.13.

Related information

[Upgrading PAM on a Puppet-supported cluster](#) on page 106

Upgrade Puppet Application Manager (PAM) on a Puppet-supported cluster to take advantage of new features and bug fixes, and to upgrade your cluster to the latest version of Kubernetes when one is available.

[Backing up PAM using snapshots](#) on page 112

Snapshots are point-in-time backups of your Puppet Application Manager (PAM) deployment, which can be used to roll back to a previous state or restore your installation into a new cluster for disaster recovery.

Known issues

These are the known issues for Puppet Application Manager (PAM).

Restarting a PAM node on v1.103.3, v1.107.0, and v1.108.0 does not successfully bring back up all pods

A known Kubernetes issue impacts PAM versions v1.103.3, v1.107.0, and v1.108.0. Rebooting a node can result in pods incorrectly changing phases, causing them to remain at states such as 0/1 Completed and 0/1 Error. This can lead to pod crashloops and application outages. To work around this issue, delete the errant pods, which forces them to restart. Check the pods afterwards to ensure they come up as “Running”. This issue is resolved in PAM version 1.109.0

PAM versions 1.72.1 and older cannot be installed on RHEL 8.6+ systems

A known issue in kURL prevents Puppet Application Manager versions 1.72.1 and older from successfully installing on Red Hat Enterprise Linux (RHEL) version 8.6 and newer versions. To work around this issue, install or upgrade to Puppet Application Manager version 1.76.1 or a newer version, which support RHEL version 8.6.

Velero fails if network file system (NFS) snapshot storage is misconfigured

In Puppet Application Manager version 1.64.0 and newer versions, changes to the configuration of snapshot storage on a network file system (NFS) is appended to Velero containers, rather than replaced. This means that if NFS snapshot storage is misconfigured, attempts to fix the configuration do not correct the problem. This issue manifests as a failure of Velero to start up.

OpenSSL package required for newer RHEL 7 systems with PAM 1.62.0

Attempts to install Puppet Application Manager version 1.62.0 or older on newer Red Hat Enterprise Linux (RHEL) 7 systems fail unless the OpenSSL package is present on the system before installation. To work around this issue, run `yum install openssl` and then re-run the PAM installation script.

Package updates with yum or DNF fail after upgrading PAM

If you are unable to run `yum update` or `dnf upgrade` after a PAM upgrade, run one of the following commands to clean up a temporary module added by PAM:

```
yum module reset kurl.local
```

or

```
dnf module reset kurl.local
```

Architecture overview

Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

PAM can run on Puppet-supported or customer-supported Kubernetes clusters. Due to potential variations in the architecture of customer-supported clusters, the architecture overview provided on this page assumes PAM is running on Puppet-supported clusters. For more information on installing on a customer-supported Kubernetes cluster, see [Install Puppet applications using PAM on a customer-supported Kubernetes cluster](#) on page 81.

Terminology

Throughout this documentation, we use a few terms to describe different roles nodes can take:

- **Primary** - A primary node runs core Kubernetes components (referred to as the Kubernetes control plane) as well as application workloads. At least three primaries are required to support high availability for Puppet Application Manager. These are also sometimes referred to as *masters*.
- **Secondary** - A secondary node runs application workloads. These are also sometimes referred to as *workers*.

Puppet Application Manager is built on the KOTS (Kubernetes off-the-Shelf) project, and we occasionally use its CLI tools (`kubect1`, `kots`) to manage the installation.

Standalone architecture

Standalone is optimized for limited resources, storing data directly on disk. If you need to remove optional components like Prometheus and Grafana to decrease resource utilization, see [Optional components](#) on page 124. While additional compute capacity can be added through secondary nodes, this does not provide increased resilience as data is only stored on the node where a component service runs.

For information on migrating data from standalone to HA deployments, see [Migrating data between two systems with different architectures](#) on page 119.

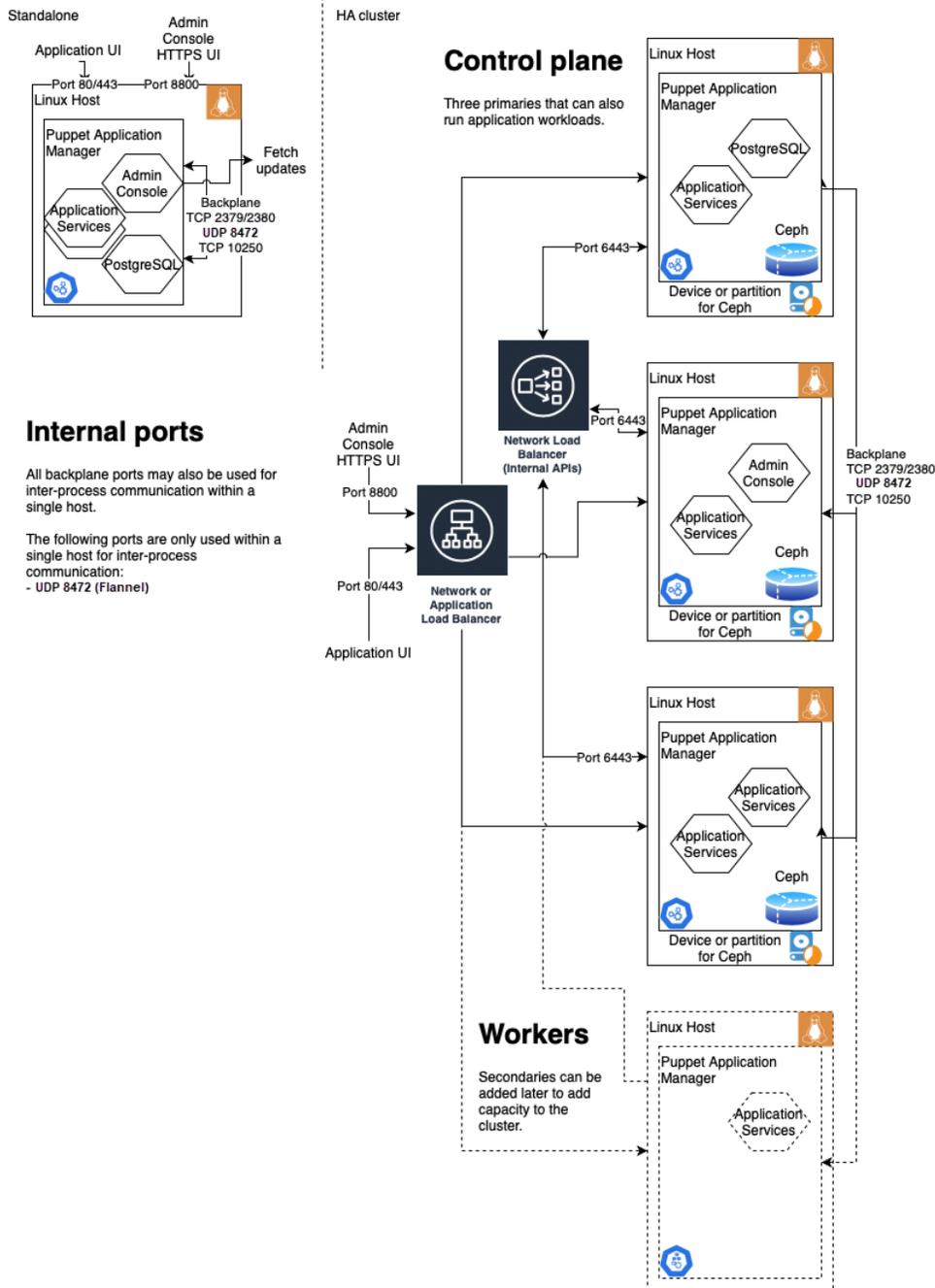
HA architecture

A high availability (HA) architecture provides high availability for scheduling application services during failure and uses Ceph for distributed storage in case of node failure. Individual applications may still experience some loss of availability (up to 10 minutes) if individual services do not have replicas and need to be rescheduled. For more information, see [Reduce recovery time when a node fails](#) on page 123. An HA implementation requires a cluster of three primary nodes. Additional compute capacity can be added through secondary nodes.

The HA architecture installs Prometheus and Alertmanager. These are used to provide system monitoring in the Puppet Application Manager UI. Prometheus and Alertmanager are unauthenticated on ports 30900 and 30903, and you are recommended to control access to these ports via firewall rules. For information on how to remove Prometheus and Alertmanager, see [Optional components](#) on page 124.

Puppet Application Manager architectures

The following diagram and lists outline some of the core components involved in standalone and HA architectures and how they communicate. For a detailed list of ports used by Puppet Application Manager, refer to the **Cluster port requirements** sections of the [PAM system requirements](#) on page 66. For firewall information, refer to [Web URL and port requirements for firewalls](#) on page 78.



Standalone architecture

Puppet Application Manager

Lives on a cluster within a Linux host.

The PAM application includes the admin console, application services, and PostgreSQL.

PAM communicates out of the Linux host to fetch updates.

UI ports

The application UI communicates on 80/443 to the Linux host.

The admin console HTTPS UI communicates on 8800 to the Linux host.

Backplane and internal ports

Backplane ports include 8472 (UDP) and 10250 (TCP).

Backplane ports can also be used within a single host for inter-process communication.

These ports are only used within a single host for inter-process communication (Flannel): 8472 (UDP)

Additional default ports

30900: Prometheus UI

30902: Grafana UI

30903: Alertmanager UI

HA cluster architecture

Control plane (primaries)

Multiple primaries that can also run application workloads.

Structured as clusters within Linux hosts with a device or partition for Ceph.

Each primary hosts PAM and can run application services in addition to supporting either PostgreSQL or the admin console.

Workers (secondaries)

Can be added later to add capacity for running application workloads.

Structured as clusters within Linux hosts.

Network or Application Balancer

The balancer communicates out to the control plane (primaries) and workers (secondaries).

Receives admin console HTTPS UI communication over 8800.

Receives application UI communication over 80/443.

Network load balancer internal APIs communicate with primaries and secondaries over 6443.

To learn about setting up health checks for your load balancer, go to [Load balancer health checks](#) on page 105.

Backplane and internal ports

Backplane ports include 2379/2380 (TCP), 8472 (UDP), and 10250 (TCP).

Backplane ports can also be used within a single host for inter-process communication.

These ports are only used within a single host for inter-process communication (Flannel): 8472 (UDP)

Additional default ports

30900: Prometheus UI

30902: Grafana UI

30903: Alertmanager UI

UNSUPPORTED: Legacy architecture

Note: The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. The legacy architecture reached the end of its support lifecycle on **30 June 2022**, and Puppet no longer updates legacy architecture components.

The Puppet Application Manager legacy architecture reflects an older configuration that used Ceph 1.0 which hosted data directly on the file system. Installing the legacy architecture is no longer supported.

For information on upgrading to a newer version of the legacy architecture, see [PAM legacy upgrades](#) on page 109 and [PAM offline legacy upgrades](#) on page 109.

For information on migrating data from a legacy architecture to a standalone or HA architecture, go to our Support Knowledge Base instructions:

- [Migrate to a supported PAM architecture for Continuous Delivery for PE](#)
- [Migrate to a supported PAM architecture for Comply](#)

Related information

[Reduce recovery time when a node fails](#) on page 123

If a node running a non-replicated service like PostgreSQL fails, expect some service downtime.

[Install Puppet applications using PAM on a customer-supported Kubernetes cluster](#) on page 81

Use these instructions to install Puppet Application Manager and any Puppet applications on an existing Kubernetes cluster.

[PAM legacy upgrades](#) on page 109

The legacy architecture is no longer supported. However, if you have not yet migrated to a supported architecture, you can use this method to upgrade Puppet Application Manager (PAM).

[PAM offline legacy upgrades](#) on page 109

The legacy architecture is no longer supported. However, if you have not yet migrated to a supported architecture, you can use this method to upgrade Puppet Application Manager (PAM) on offline nodes.

[Troubleshooting PAM](#) on page 121

Use this guide to troubleshoot issues with your Puppet Application Manager installation.

PAM system requirements

You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

Customer-supported cluster hardware requirements

The following Kubernetes distributions are supported:

- Google Kubernetes Engine
- AWS Elastic Kubernetes Service

If you use a different distribution, contact [Puppet Support](#) for more information on compatibility with PAM.

Application requirements:

Application	CPU	Memory	Storage	Ports
Continuous Delivery for Puppet Enterprise (PE)	3 CPU	8 GB	280 GB	Ingress, NodePort 8000 Note: NodePort is configurable
Puppet Comply®	7 CPU	7 GB	35 GB	Ingress, NodePort 30303 Note: NodePort is configurable

Make sure that your Kubernetes cluster meets the minimum requirements:

- Kubernetes version 1.24-1.26.
- A default storage class that can be used for relocatable storage.
- A standard Ingress controller that supports websockets (we have tested with Project Contour and NGINX).
- We currently test and support Google Kubernetes Engine (GKE) clusters.

Cluster ports: In addition to the NodePorts used by your Puppet applications, make sure that TCP port 443 is open for your ingress controller.

Puppet-supported HA cluster hardware requirements

A high availability (HA) configuration uses multiple servers to provide availability in the event of a server failure. A majority of servers must be available to preserve service availability. Below are suggested configurations for each application.

Continuous Delivery for Puppet Enterprise (PE)

Three servers (referred to as primaries during installation) with the following minimum requirements:

CPU	Memory	Storage	Open ports
6 CPU	10 GB	<p>100 GB on an unformatted storage device.</p> <p>1 GB for <code>/var/log/apiserver</code> for Kubernetes audit logs.</p> <p>An additional 140 GB for <code>/var/lib</code>. You can use separate filesystems if necessary, but it is not a requirement to do so. For your reference, here is how the usage is roughly divided:</p> <ul style="list-style-type: none"> • 2 GB for <code>/var/lib/etcd</code> • 10 GB for <code>/var/lib/rook</code> (plus buffer) • 32 GB for <code>/var/lib/kubelet</code> • 80 GB for <code>/var/lib/containerd</code> <p>Note: The storage backend prefers the file system inhabited by <code>/var/lib/rook</code> to remain below 70% utilization.</p> <p>SSDs (or similarly low-latency storage) are recommended for <code>/var/lib/etcd</code> and <code>/var/lib/rook</code>.</p>	<p>TCP: 80, 443, 2379, 2380, 6443, 8000, 8800, and 10250</p> <p>UDP: 8472</p>

Puppet Comply

Three servers (referred to as primaries during installation) with the following minimum requirements:

CPU	Memory	Storage	Open ports
7 CPU	10 GB	<p>100 GB on an unformatted storage device.</p> <p>1 GB for <code>/var/log/apiserver</code> for Kubernetes audit logs.</p> <p>An additional 140 GB for <code>/var/lib</code>. You can use separate filesystems if necessary, but it is not a requirement to do so. For your reference, here is how the usage is roughly divided:</p> <ul style="list-style-type: none"> • 2 GB for <code>/var/lib/etcd</code> • 10 GB for <code>/var/lib/rook</code> (plus buffer) • 32 GB for <code>/var/lib/kubelet</code> • 80 GB for <code>/var/lib/containerd</code> <p>Note: The storage backend prefers the file system inhabited by <code>/var/lib/rook</code> to remain below 70% utilization.</p> <p>SSDs (or similarly low-latency storage) are recommended for <code>/var/lib/etcd</code> and <code>/var/lib/rook</code>.</p>	<p>TCP: 80, 443, 2379, 2380, 6443, 8800, 10250, and 30303</p> <p>UDP: 8472</p>

Continuous Delivery for Puppet Enterprise (PE) and Puppet Comply

Three servers (referred to as primaries during installation) with the following minimum requirements:

CPU	Memory	Storage	Open ports
8 CPU	13 GB	<p>150 GB on an unformatted storage device.</p> <p>1 GB for <code>/var/log/apiserver</code> for Kubernetes audit logs.</p> <p>An additional 140 GB for <code>/var/lib</code>. You can use separate filesystems if necessary, but it is not a requirement to do so. For your reference, here is how the usage is roughly divided:</p> <ul style="list-style-type: none"> • 2 GB for <code>/var/lib/etcd</code> • 10 GB for <code>/var/lib/rook</code> (plus buffer) • 32 GB for <code>/var/lib/kubelet</code> • 80 GB for <code>/var/lib/containerd</code> <p>Note: The storage backend prefers the file system inhabited by <code>/var/lib/rook</code> to remain below 70% utilization.</p> <p>SSDs (or similarly low-latency storage) are recommended for <code>/var/lib/etcd</code> and <code>/var/lib/rook</code>.</p>	<p>TCP: 80, 443, 2379, 2380, 6443, 8000, 8800, 10250, and 30303</p> <p>UDP: 8472</p>

For a detailed example of an HA configuration running Continuous Delivery for PE and Puppet Comply, see [Example of an HA cluster that supports CDPE and Comply](#).

Networking requirements

Gigabit Ethernet (1GbE) and a latency of less than 10 milliseconds (ms) between cluster members is sufficient for most deployments. For more information on networking for specific Puppet Application Manager components, see the documentation for [Ceph](#), and [etcd](#).

Cluster port requirements

Puppet Application Manager (PAM) uses the following ports in an HA cluster architecture:

Category	Port	Protocol	Purpose	Source
Puppet application ports	443	TCP	Web UI Relies on Server Name Indication to route requests to the application.	Browser
Continuous Delivery for Puppet Enterprise (PE) ports	8000	TCP	Webhook service	Source control

Category	Port	Protocol	Purpose	Source
Puppet Comply ports	30303	TCP	Communication with Puppet Enterprise (PE)	PE instance
Platform ports	2379, 2380	TCP	High availability (HA) communication Only needs to be open between the cluster's primary nodes.	etcd on the Kubernetes host.
	6443	TCP	Kubernetes API Might be useful to expose to workstations.	Admin workstation
	8472	UDP	Kubernetes networking - Flannel	Kubernetes host
	8800	TCP	PAM	Admin browser
	9001	TCP	Internal registry in offline installs only. Requires configuring an Ingress to use this port.	Kubernetes host
	9090	TCP	Rook CSI RBD Plugin Metrics	Kubernetes host
	10250	TCP	Kubernetes cluster management Only communicates in one direction, from a primary to other primaries and secondaries.	Kubernetes host

Additionally, these ports are configured by default: 30900 (Prometheus UI), 30902 (Grafana UI), and 30903 (Alertmanager UI)

For Kubernetes-specific information, refer to [Networking Requirements in the Kurl documentation](#).

IP address range requirements

Important: Puppet Application Manager must be installed on nodes with static IP assignments because IP addresses cannot be changed after installation.

Ensure that IP address ranges `10.96.0.0/22` and `10.32.0.0/22` are locally accessible. See [Resolve IP address range conflicts](#) for instructions.

Note: The minimum size for CIDR blocks used by PAM are:

- `/23` for pod and service CIDRs
- Default of `/22` is recommended to support future expansion

Antivirus and antimalware considerations

Antivirus and antimalware software can impact PAM and its applications or prevent them from functioning properly.

To avoid issues, exclude the following directories from antivirus and antimalware tools that scan disk write operations:

- `/var/lib/rook`
- `/var/lib/kubelet`
- `/var/lib/containerd`

Firewall modules

If you use the `puppetlabs/firewall` module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the `puppetlabs/pam_firewall` module before installing Puppet Application Manager.

If you've already installed PAM, apply the `pam_firewall` module and then restart the `kube-proxy` service to recreate its iptables rules by running the following on a primary:

```
systemctl restart kubelet
      kubectl -n kube-system delete pod -l k8s-app=kube-proxy
      kubectl -n kube-flannel delete pod -l app=flannel
```

For more information, see the PAM [firewall module](#).

Supported operating systems

Puppet Application Manager and the applications it supports can be installed on these operating systems:

Operating system	Supported versions
Amazon Linux	2
CentOS	7.4, 7.5, 7.6, 7.7, 7.8, 7.9 8.0, 8.1, 8.2, 8.3, 8.4
Oracle Linux	7.4, 7.5, 7.6, 7.7, 7.8, 7.9 8.0, 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8
Red Hat Enterprise Linux (RHEL)	7.4, 7.5, 7.6, 7.7, 7.8, 7.9 8.0, 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8 9.0, 9.1, 9.2

Operating system	Supported versions
Rocky Linux	9.0, 9.1, 9.2
Ubuntu (General availability kernels)	18.04 20.04 22.04

Puppet-supported standalone hardware requirements

Here are the suggested configurations for standalone installations.

Continuous Delivery for Puppet Enterprise (PE)

CPU	Memory	Storage	Open ports
4 CPU	8 GB	220 GB for <code>/var/lib</code> and <code>/var/opensbs</code> This is primarily divided among: <ul style="list-style-type: none"> • 2 GB for <code>/var/lib/etcd</code> • 32 GB for <code>/var/lib/kubelet</code> • 80 GB for <code>/var/lib/containerd</code> • 100 GB for <code>/var/opensbs</code> 	TCP: 80, 443, 2379, 2380, 6443, 8000, 8800, and 10250 UDP: 8472

Puppet Comply

CPU	Memory	Storage	Open ports
7 CPU	7 GB	220 GB for <code>/var/lib</code> and <code>/var/opensbs</code> This is primarily divided among: <ul style="list-style-type: none"> • 2 GB for <code>/var/lib/etcd</code> • 32 GB for <code>/var/lib/kubelet</code> • 80 GB for <code>/var/lib/containerd</code> • 100 GB for <code>/var/opensbs</code> 	TCP: 80, 443, 2379, 2380, 6443, 8800, 10250, and 30303 UDP: 8472

Cluster port requirements

Puppet Application Manager (PAM) uses the following ports in a standalone architecture:

Category	Port	Protocol	Purpose	Source
Puppet application ports	442	TCP	Web UI Relies on Server Name Indication to route requests to the application.	Browser
Continuous Delivery for Puppet Enterprise (PE) ports	8000	TCP	Webhook service	Source control
Puppet Comply ports	30303	TCP	Communication with Puppet Enterprise	PE instance
Platform ports	6443	TCP	Kubernetes API Might be useful to expose to workstations.	Admin workstation
	8472	UDP	Kubernetes networking - Flannel	Kubernetes host
	8800	TCP	PAM	Admin browser
	9001	TCP	Internal registry in offline installs only. Requires configuring an Ingress to use this port.	Kubernetes host
	10250	TCP	Kubernetes cluster management Only communicates in one direction, from a primary to other primaries and secondaries.	Kubernetes host

Additionally, these ports are configured by default: 30900 (Prometheus UI), 30902 (Grafana UI), and 30903 (Alertmanager UI)

For Kubernetes-specific information, refer to [Networking Requirements in the Kurl documentation](#).

IP address range requirements

Important: Puppet Application Manager must be installed on nodes with static IP assignments because IP addresses cannot be changed after installation.

Ensure that IP address ranges `10.96.0.0/22` and `10.32.0.0/22` are locally accessible. See [Resolve IP address range conflicts](#) for instructions.

Note: The minimum size for CIDR blocks used by PAM are:

- `/24` for pod and service CIDRs
- Default of `/22` is recommended to support future expansion

Antivirus and antimalware considerations

Antivirus and antimalware software can impact PAM and its applications or prevent them from functioning properly.

To avoid issues, exclude the following directories from antivirus and antimalware tools that scan disk write operations:

- `/var/openlibs`
- `/var/lib/kubelet`
- `/var/lib/containerd`

Firewall modules

If you use the `puppetlabs/firewall` module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the `puppetlabs/pam_firewall` module before installing Puppet Application Manager.

If you've already installed PAM, apply the `pam_firewall` module and then restart the `kube-proxy` service to recreate its iptables rules by running the following on a primary:

```
systemctl restart kubelet
      kubectl -n kube-system delete pod -l k8s-app=kube-proxy
      kubectl -n kube-flannel delete pod -l app=flannel
```

For more information, see the PAM [firewall module](#).

Detailed hardware requirements

For additional compute capacity, you can horizontally scale HA and standalone architectures by adding secondary nodes. During installation, only add secondaries after setting up all primaries.

You can add secondaries to HA and standalone architectures; however in standalone architectures, secondaries do not increase availability of the application, and data storage services are pinned to the host they start on and cannot be moved.

Here are the baseline requirements to run cluster services on primaries and secondaries. Any Puppet applications require additional resources on top of these requirements.

Node type	CPU	Memory	Storage	Open ports
Primary	4 CPU	7 GB	<p>At least 50 GB on an unformatted storage device in addition to application-specific storage (below) for the Ceph storage backend. This can be satisfied by multiple devices if more storage is needed later, but should be balanced across primaries.</p> <p>1 GB for <code>/var/log/apiserver</code> for Kubernetes audit logs.</p> <p>An additional 140 GB for <code>/var/lib</code>. You can use separate filesystems if necessary, but it is not a requirement to do so. For your reference, here is how the usage is roughly divided:</p> <ul style="list-style-type: none"> • 2 GB for <code>/var/lib/etcd</code> • 10 GB for <code>/var/lib/rook</code> (plus buffer) • 32 GB for <code>/var/lib/kubelet</code> • 80 GB for <code>/var/lib/containerd</code> <p>Note: Ceph storage backend prefers the file system inhabited by <code>/var/lib/rook</code> to remain below 70% utilization.</p> <p>SSDs (or similarly low-latency storage) are recommended for <code>/var/lib/etcd</code> and <code>/var/lib/rook</code>.</p>	<p>TCP: 80, 443, 2379, 2380, 6443, 8800, and 10250</p> <p>UDP: 8472</p>
Secondary	1 CPU	1.5 GB	<p>1 GB for <code>/var/log/apiserver</code> for Kubernetes audit logs.</p> <p>120 GB for <code>/var/lib</code>. You can use separate filesystems if necessary, but it is not a requirement to do so. For your reference, here is how the usage is roughly divided:</p> <ul style="list-style-type: none"> • 32 GB for <code>/var/lib/kubelet</code> • 80 GB for <code>/var/lib/containerd</code> 	

Applications are composed of multiple smaller services, so you can divide CPU and memory requirements across multiple servers. The listed ports can be accessed from all primaries and secondaries, but only need to be exposed on nodes you include in your load balancer. Apply application-specific storage to all primary nodes.

Application-specific requirements:

Application	CPU	Memory	Storage	Ports
Continuous Delivery for Puppet Enterprise (PE)	3 CPU	8 GB	50 GB	80, 443, 8000
Puppet Comply	7 CPU	7 GB	50 GB	80, 443, 30303

The minimum recommended size for a secondary node is 4 CPU and 8 GB of memory to allow some scheduling flexibility for individual services.

Example of an HA cluster capable of running Continuous Delivery for PE and Comply

An HA cluster capable of running both Continuous Delivery for Puppet Enterprise (PE) and Puppet Comply requires 10 CPU and 15 GB of application-specific memory in addition to per-node baselines. You can create a cluster from 4 CPU, 8 GB nodes. Each primary uses all CPU and 7 GB of memory for cluster services, providing 0 CPU and 1 GB of memory for application workloads; each secondary uses 1 CPU and 1.5 GB of memory for cluster services, providing 3 CPU and 6.5 GB of memory for application workloads. Create the cluster as follows:

- Three primaries provide an excess of 3 GB of memory for application workloads. Each primary must have 150 GB of storage in an unformatted, unpartitioned storage device for Ceph and 140 GB of storage for `/var/lib`.
- Three secondaries provide an excess of 9 CPU and 19.5 GB of memory for application workloads. Each secondary must have 120 GB of storage for `/var/lib`.

This diagram illustrates the suggested three-node configuration for a cluster capable of running Continuous Delivery for Puppet Enterprise (PE) and Puppet

Web URL and port requirements for firewalls

Puppet Application Manager interacts with external web URLs for a variety of installation, configuration, upgrade, and deployment tasks. Puppet Application Manager uses the following web URLs for internal and outbound network traffic.

Category	URLs
Puppet Application Manager and platform	<ul style="list-style-type: none"> • get.replicated.com • registry.replicated.com • proxy.replicated.com • api.replicated.com • k8s.kurl.sh • kurl-sh.s3.amazonaws.com • replicated.app • registry-data.replicated.com
Container registries	<ul style="list-style-type: none"> • gcr.io • docker.io • index.docker.io • registry-1.docker.io • auth.docker.io • production.cloudflare.docker.com • quay.io
Puppet Enterprise	<ul style="list-style-type: none"> • pup.pt • forgeapi.puppet.com • pm.puppetlabs.com • amazonaws.com • s3.amazonaws.com • rubygems.org

For information about containers and firewalls, refer to the [Networking Requirements in the Kurl documentation](#).

Firewall modules

If you use the [puppetlabs/firewall](#) module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the [puppetlabs/pam_firewall](#) module before installing Puppet Application Manager.

If you've already installed PAM, apply the `pam_firewall` module and then restart the `kube-proxy` service to recreate its iptables rules by running the following on a primary:

```
systemctl restart kubelet
      kubectl -n kube-system delete pod -l k8s-app=kube-proxy
      kubectl -n kube-flannel delete pod -l app=flannel
```

For more information, see the PAM [firewall module](#).

Supported browsers

The following browsers are supported for use with the Puppet Application Manager UI:

Browser	Supported versions
Google Chrome	Current version as of release
Mozilla Firefox	Current version as of release
Microsoft Edge	Current version as of release
Apple Safari	Current version as of release

Component versions in PAM releases

These tables show the versions of components included in recent Puppet Application Manager (PAM) releases.

Component	PAM 1.11.2	PAM 1.12.4	PAM 1.11.0	PAM 1.10.9	PAM 1.10.8	PAM 1.10.7	PAM 1.10.3	PAM 1.10.2	PAM 1.10.0	PAM 1.9.9	PAM 1.9.7	PAM 1.9.4	PAM 1.9.1	PAM 1.8.1	PAM 1.8.0	PAM 1.7.6	PAM 1.7.2	PAM 1.7.0	PAM 1.6.8
Kubernetes	1.30.5	1.28.9	1.28.9	1.28.2	1.28.2	1.28.2	1.26.6	1.26.6	1.24.10	1.24.10	1.23.9	1.23.9	1.21.8	1.21.8	1.21.8	1.21.8	1.21.8	1.21.8	1.21.8
KOTS	1.12.4	1.12.4	1.11.0	1.10.9	1.10.8	1.10.7	1.10.3	1.10.2	1.10.0	1.9.9	1.9.7	1.9.4	1.9.1	1.8.1	1.8.0	1.7.6	1.7.2	1.7.0	1.6.8
kURL	2025.002.4	2024.002.2	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0	2023.002.0
Weave	N/A	N/A	N/A	N/A	N/A	N/A	REMOVED	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1
Flannel	0.26.4	0.25.4	0.25.4	0.25.4	0.25.4	0.25.4	0.22.3	0.22.3	N/A										
Project Contour	1.30.0	1.30.0	1.29.0	1.28.3	1.27.0	1.27.0	1.26.11	1.25.2	1.25.0	1.25.0	1.25.0	1.24.3	1.24.0	1.23.11	1.22.0	1.21.11	1.21.11	1.21.0	1.21.0
Registry	2.8.3	2.8.3	2.8.3	2.8.3	2.8.3	2.8.3	2.8.2	2.8.2	2.8.2	2.8.2	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.8.1	2.7.1	2.7.1
Prometheus bundle	0.80.0	0.79.0	0.74.0	0.73.0	0.73.0	0.73.0	0.76.0	0.75.0	0.75.0	0.75.0	0.75.0	0.75.0	0.75.0	0.75.0	0.75.0	0.75.0	0.75.0	0.75.0	0.75.0
containerd	1.7.5	1.6.33	1.6.32	1.6.31	1.6.28	1.6.26	1.6.24	1.6.22	1.6.21	1.6.21	1.6.21	1.6.20	1.5.11	1.4.13	1.4.13	1.4.13	1.4.12	1.4.12	1.4.12
Velerd	0.15.2	0.14.0	0.13.2	0.13.2	0.12.3	0.12.2	0.12.11	0.11.11	0.11.0	0.11.0	0.10.2	0.9.5	0.9.4	0.9.1	0.9.0	0.9.0	0.9.0	0.8.1	0.8.1
ekco	0.28.7	0.28.7	0.28.7	0.28.6	0.28.4	0.28.4	0.28.3	0.28.3	0.27.1	0.27.1	0.26.5	0.26.3	0.26.3	0.26.3	0.20.0	0.19.0	0.19.0	0.19.0	0.16.0

- [PAM HA offline installation](#) on page 87

Use these instructions to install Puppet Application Manager (PAM) in an air-gapped or offline environment where the Puppet Application Manager host server does not have direct access to the internet.

- [PAM standalone online installation](#) on page 90

The Puppet Application Manager (PAM) installation process sets up the application manager (with a simple Kubernetes installation for container orchestration) for you and installs the application on the single-node cluster.

- [PAM standalone offline installation](#) on page 93

Use these instructions to install Puppet Application Manager (PAM) in an offline environment where the Puppet Application Manager host server does not have direct access to the internet.

- [Automate PAM and Puppet application online installations](#) on page 95

During a fresh online installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

- [Automate PAM and Puppet application offline installations](#) on page 97

During a fresh offline installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

- [Uninstall PAM](#) on page 101

Different uninstall procedures are required for Puppet-supported and customer-supported clusters

Related information

[Architecture overview](#) on page 62

Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

Install Puppet applications using PAM on a customer-supported Kubernetes cluster

Use these instructions to install Puppet Application Manager and any Puppet applications on an existing Kubernetes cluster.

Before you begin

1. If you haven't already done so, [install kubectl](#).
2. Puppet Application Manager is expected to work on any certified Kubernetes distribution that meets the following requirements. We validated and support:

- Google Kubernetes Engine
- AWS Elastic Kubernetes Service

If you use a different distribution, contact [Puppet Support](#) for more information on compatibility with PAM.

3. Make sure your Kubernetes cluster meets the minimum requirements:

- Kubernetes version 1.24-1.26.
- A default storage class that can be used for relocatable storage.
- A standard Ingress controller that supports websockets (we have tested with Project Contour and NGINX).
- We currently test and support Google Kubernetes Engine (GKE) clusters.

Note: If you're using self-signed certificates on your Ingress controller, you must ensure that your job hardware nodes trust the certificates. Additionally, all nodes that use Continuous Delivery for PE webhooks must trust the certificates, or SSL checking must be disabled on these nodes.

Important: If you are installing Puppet Comply on Puppet Application Manager, the ingress controller must be configured to allow request payloads of up to 32 MB. Ingress controllers used by Amazon EKS commonly default to a 1 MB maximum — this causes all report submissions to fail.

The ingress must have a generous limit for total connection time. Setting the connection timeout to `infinity` in conjunction with an idle timeout is recommended.

4. If you are setting up Puppet Application Manager behind a proxy server, the installer supports proxies configured via HTTP_PROXY/HTTPS_PROXY/NO_PROXY environment variables.

Restriction: Using a proxy to connect to external version control systems is currently not supported.

Installation takes several (mostly hands-off) minutes to complete.

1. Install the KOTS (Kubernetes off-the-shelf software) plugin on a workstation that has kubectl access to the cluster. Your kubectl configuration must have sufficient privileges to create cluster-level roles and permissions:

```
curl https://kots.io/install | bash
```

2. If you are performing an offline install, ensure the required images are available in a local registry.

- a) Download the release assets matching the CLI version using the following command:

```
curl -LO https://github.com/replicatedhq/kots/releases/download/v
$(kubectl kots version | head -n1 | cut -d' ' -f3)/kotsadm.tar.gz
```

- b) Extract the images and push them into a private registry. Registry credentials provided in this step must have push access. These credentials are not stored anywhere or reused later.

```
kubectl kots admin-console push-images ./kotsadm.tar.gz
<private.registry.host>/puppet-application-manager \
--registry-username <rw-username> \
--registry-password <rw-password>
```

- c) Install Puppet Application Manager using images pushed in the previous step. Registry credentials provided in this step only need to have read access, and they are stored in a Kubernetes secret in the current namespace. These credentials are used to pull the images.

```
kubectl kots install puppet-application-manager \
--kotsadm-namespace puppet-application-manager \
--kotsadm-registry <private.registry.host> \
--registry-username <ro-username> \
--registry-password <ro-password>
```

Note: If you are setting up Puppet Application Manager behind a proxy server, add the `--copy-proxy-env` flag to this command to copy the proxy-related environment values from your environment.

- d) You can use similar commands to upload images from the application bundle to your registry to continue to use read-only access when pulling images. Use the same registry namespace (`puppet-application-manager`) to pull application images.

```
kubectl kots admin-console push-images ./<application-release>.airgap
<private.registry.host>/puppet-application-manager \
--registry-username <rw-username> \
--registry-password <rw-password>
```

3. To perform an online install of Puppet Application Manager on your cluster, run the following commands from a workstation that has kubectl access to the cluster.

```
kubectl kots install puppet-application-manager --namespace <target
namespace>
```

This installs Puppet Application Manager on the cluster and sets up a port forward on the ClusterIP.

4. Navigate to `http://localhost:8800` and follow the prompts to be guided through the process of uploading a license for the application, configuring a local registry (for offline installs), checking to make sure your infrastructure meets system requirements, and configuring the application.

Note: If you are performing an offline install, download the application bundle and provide it when prompted.

Tip: Clusters like GKE often restrict ports to 30000-32767. The webhook for Continuous Delivery for PE defaults to port 8000. To update this port to something in the allowed range, when configuring the application, use the following steps:

- a. On the Puppet Application Manager **Dashboard** page, under **Config > Optional configuration**, select **View options for using a proxy or external load balancer**.
 - b. Enter a new value for **Webhook service port**.
5. To configure your installation further, click **Config**. On this tab, you can configure a public hostname, root user, and other settings. These are written as Kubernetes secrets in the deployment manifests. For information on how to configure your application, see the documentation for that application:
 - [Configure Continuous Delivery for PE in an online environment](#)
 - [Configure Comply in an online environment](#)
 6. To use cert-manager, in the **Customize endpoints** section, select **I have cert manager** and in the annotations section, add yours. For example:

```
kubernetes.io/ingress.class: nginx
cert-manager.io/cluster-issuer: letsencrypt-prod
```

7. When you are happy with your configuration, click **Save config** to deploy the application.

Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For general information, go to [Install applications via the PAM UI](#) on page 102.

For more information on installing Continuous Delivery for PE online, see [Install Continuous Delivery for PE](#).

For more information on installing Comply online, see [Install Comply online](#).

Related information

[Upgrade PAM on a customer-supported online cluster](#) on page 110

Upgrading Puppet Application Manager (PAM) on a customer-supported online Kubernetes cluster can be done with a single command.

[Upgrade PAM on a customer-supported offline cluster](#) on page 111

Upgrading Puppet Application Manager (PAM) on a customer-supported offline Kubernetes cluster requires a few simple kubectl commands.

PAM HA online installation

The Puppet Application Manager (PAM) installation process creates a Kubernetes cluster for you and walks you through installing your Puppet application on the cluster.

Before you begin

1. Review the [Puppet Application Manager system requirements](#).
2. Note that Swap is not supported for use with this version of Puppet Application Manager (PAM). The installation script attempts to disable Swap if it is enabled.

3. (Optional) If necessary, prepare additional steps related to SELinux and Firewalld:

The PAM installation script disables SELinux and Firewalld by default. If you want to keep SELinux enabled, append the `-s preserve-selinux-config` switch to the PAM install command. This may require additional configuration to adapt SELinux policy to the installation.

If you want to keep Firewalld enabled:

- a. Make sure Firewalld is installed on your system.
- b. To prevent the installation from disabling Firewalld, provide a patch file to the PAM install command using `-s installer-spec-file=patch.yaml`, where `patch.yaml` is the name of your patch file. For reference, here's an example patch file that enables Firewalld during installation, starts the service if it isn't running, and adds rules to open relevant ports:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
sysctl -p"]
  firewalldCmds:
    - ["--permanent", "--zone=trusted", "--add-interface=flannel.1"]
    - ["--zone=external", "--add-masquerade"]
    # SSH port
    - ["--permanent", "--zone=public", "--add-port=22/tcp"]
    # HTTPS port
    - ["--permanent", "--zone=public", "--add-port=443/tcp"]
    # Kubernetes etcd port
    - ["--permanent", "--zone=public", "--add-port=2379-2830/tcp"]
    # Kubernetes API port
    - ["--permanent", "--zone=public", "--add-port=6443/tcp"]
    # Flannel Net port
    - ["--permanent", "--zone=public", "--add-port=8472/udp"]
    # CD4PE Webhook callback port (uncomment line below if needed)
    # - ["--permanent", "--zone=public", "--add-port=8000/tcp"]
    # KOTS UI port
    - ["--permanent", "--zone=public", "--add-port=8800/tcp"]
    # CD4PE Local registry port (offline only, uncomment line below if
needed)
    # - ["--permanent", "--zone=public", "--add-port=9001/tcp"]
    # Kubernetes component ports (kubelet, kube-scheduler, kube-
controller)
    - ["--permanent", "--zone=public", "--add-port=10250-10252/tcp"]
    # Reload firewall rules
    - ["--reload"]
  bypassFirewalldWarning: true
  disableFirewalld: false
  hardFailOnFirewalld: false
  preserveConfig: false
```

4. Ensure that IP address ranges `10.96.0.0/22` and `10.32.0.0/22` are locally accessible. See [Resolve IP address range conflicts](#) on page 122 for instructions.

Note: The minimum size for CIDR blocks used by Puppet Application Manager are:

- **Standalone** - /24 for pod and service CIDRs
- **HA** - /23 for pod and service CIDRs
- Default of /22 is recommended to support future expansion

5. If you are setting up Puppet Application Manager behind a proxy server, the installer supports proxies configured via `HTTP_PROXY/HTTPS_PROXY/NO_PROXY` environment variables.

Restriction: Using a proxy to connect to external version control systems is currently not supported.

6. Set all nodes used in your HA implementation to the UTC timezone.
7. If you use the [puppetlabs/firewall](#) module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the [puppetlabs/pam_firewall](#) module before installing Puppet Application Manager.

This installation process results in a Puppet Application Manager instance that is configured for high availability. Installation takes several minutes (mostly hands-off) to complete.

For more context about HA components and structure, refer to the **HA architecture** section of the [Architecture overview](#) on page 62.

1. Install and configure a load balancer (or two if you want to segment internal and external traffic - for more information, see [Architecture overview](#) on page 62). Round-robin load balancing is sufficient. For an HA cluster, the following is required:
 - A network (L4, TCP) load balancer for port 6443 across primary nodes. This is required for Kubernetes components to continue operating in the event that a node fails. The port is only accessed by the Kubernetes nodes and any admins using `kubectl`.
 - A network (L4, TCP) or application (L7, HTTP/S) load balancer for ports 80, and 443 across all primaries and secondaries. This maintains access to applications in event of a node failure. Include 8800 if you want external access to the Puppet Application Manager UI.

Note: Include port 8000 for webhook callbacks if you are installing Continuous Delivery for PE.

- From the command line of your first primary node, run the installation script:

```
curl -sSL https://k8s.kurl.sh/puppet-application-manager | sudo bash
```

Tip: If you're installing Puppet Application Manager behind a proxy server, using `sudo` might cause the installation to fail. Try running the command as root and replace `sudo bash` with `bash`.

Note: An unformatted, unpartitioned storage device is required.

By default this installation automatically uses devices (under `/dev`) matching the pattern `vd[b-z]`. Attach a device to each host. Only devices that match the pattern, and are unformatted, are used.

If necessary, you can override this pattern by providing a patch during installation; append `-s installer-spec-file=patch.yaml` to the installation command.

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  rook:
    blockDeviceFilter: "sd[b-z]"
```

- When prompted for a load balancer address, enter the address of the DNS entry for your load balancer.
- The installation script prints the address and password (only shown once, so make careful note of it) for Puppet Application Manager:

```
---
Kotsadm: http://<PUPPET APPLICATION MANAGER ADDRESS>:8800
Login with password (will not be shown again): <PASSWORD>
---
```

Note: If you lose this password or wish to change it, see [Reset the PAM password](#) on page 122 for instructions.

- When the installation script is complete, run `bash -l` to reload the shell.

Tip: If the installation script fails, run the following and upload the results to the Puppet Support team:

```
kubectl support-bundle https://kots.io
```

If you're installing as the root user, run the command directly:

```
/usr/local/bin/kubectl-support_bundle https://kots.io
```

- Add two additional primary nodes to the installation by following the instructions in the install script:

```
To add MASTER nodes to this installation, run the following script on your
other nodes:
curl -sSL
https://k8s.kurl.sh/puppet-application-manager-unstable/join.sh
| sudo bash -s kubernetes-master-address=...
```

If you want to keep SELinux enabled, append the `-s preserve-selinux-config` switch to the install command.

- Add the two new nodes to your load balancer.

5. Navigate to the Puppet Application Manager UI using the address provided by the installation script (`http://<PUPPET_APPLICATION_MANAGER_ADDRESS>:8800`) and follow the prompts.

The Puppet Application Manager UI is where you manage Puppet applications. You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets application system requirements.

Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For more information, see [Install applications via the PAM UI](#) on page 102.

For more information on installing Continuous Delivery for PE online, see [Install Continuous Delivery for PE](#).

For more information on installing Comply online, see [Install Comply online](#).

Related information

[Reset the PAM password](#) on page 122

As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

[PAM system requirements](#) on page 66

You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

[Resolve IP address range conflicts](#) on page 122

When installing Puppet Application Manager, IP address ranges `10.96.0.0/22` and `10.32.0.0/22` must not be used by other nodes on the local network.

[Architecture overview](#) on page 62

Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

[Using sudo behind a proxy server](#) on page 125

Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

PAM HA offline installation

Use these instructions to install Puppet Application Manager (PAM) in an air-gapped or offline environment where the Puppet Application Manager host server does not have direct access to the internet.

Before you begin

1. Review the [Puppet Application Manager system requirements](#).
2. Note that Swap is not supported for use with this version of Puppet Application Manager (PAM). The installation script attempts to disable Swap if it is enabled.
3. (Optional) If necessary, prepare additional steps related to SELinux and FirewallD:

The PAM installation script disables SELinux and FirewallD by default. If you want to keep SELinux enabled, append the `-s preserve-selinux-config` switch to the PAM install command. This may require additional configuration to adapt SELinux policy to the installation.

If you want to keep FirewallD enabled:

- a. Make sure FirewallD is installed on your system.
- b. To prevent the installation from disabling FirewallD, provide a patch file to the PAM install command using `-s installer-spec-file=patch.yaml`, where `patch.yaml` is the name of your patch file. For reference, here's an example patch file that enables FirewallD during installation, starts the service if it isn't running, and adds rules to open relevant ports:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
```

```

metadata:
  name: patch
spec:
  firewallldConfig:
    firewallld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf && sysctl -p"]
  firewallldCmds:
    - ["--permanent", "--zone=trusted", "--add-interface=flannel.1"]
    - ["--zone=external", "--add-masquerade"]
    # SSH port
    - ["--permanent", "--zone=public", "--add-port=22/tcp"]
    # HTTPS port
    - ["--permanent", "--zone=public", "--add-port=443/tcp"]
    # Kubernetes etcd port
    - ["--permanent", "--zone=public", "--add-port=2379-2830/tcp"]
    # Kubernetes API port
    - ["--permanent", "--zone=public", "--add-port=6443/tcp"]
    # Flannel Net port
    - ["--permanent", "--zone=public", "--add-port=8472/udp"]
    # CD4PE Webhook callback port (uncomment line below if needed)
    # - ["--permanent", "--zone=public", "--add-port=8000/tcp"]
    # KOTS UI port
    - ["--permanent", "--zone=public", "--add-port=8800/tcp"]
    # CD4PE Local registry port (offline only, uncomment line below if
    needed)
    # - ["--permanent", "--zone=public", "--add-port=9001/tcp"]
    # Kubernetes component ports (kubelet, kube-scheduler, kube-
    controller)
    - ["--permanent", "--zone=public", "--add-port=10250-10252/tcp"]
    # Reload firewall rules
    - ["--reload"]
  bypassFirewallldWarning: true
  disableFirewallld: false
  hardFailOnFirewallld: false
  preserveConfig: false

```

4. Ensure that IP address ranges 10.96.0.0/22 and 10.32.0.0/22 are locally accessible. See [Resolve IP address range conflicts](#) on page 122 for instructions.

Note: The minimum size for CIDR blocks used by Puppet Application Manager are:

- **Standalone** - /24 for pod and service CIDRs
- **HA** - /23 for pod and service CIDRs
- Default of /22 is recommended to support future expansion

5. Ensure that the nodes can resolve their own hostnames, through either local host mapping or a reachable DNS server.
6. Set all nodes used in your HA implementation to the UTC timezone.
7. If you use the [puppetlabs/firewall](#) module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the [puppetlabs/pam_firewall](#) module before installing Puppet Application Manager.
8. If you're restoring a backup from a previous cluster, make sure you include the `kurl-registry-ip=<YOUR_IP_ADDRESS>` installation option. For more information, see [Migrating PAM data to a new system](#) on page 114.

This installation process results in a basic Puppet Application Manager instance that is configured for optional high availability. Installation takes several minutes (mostly hands-off) to complete.

For more context about HA components and structure, refer to the **HA architecture** section of the [Architecture overview](#) on page 62.

1. Install and configure a load balancer (or two if you want to segment internal and external traffic - for more information, see [Architecture overview](#) on page 62). Round-robin load balancing is sufficient. For an HA cluster, the following is required:
 - A network (L4, TCP) load balancer for port 6443 across primary nodes. This is required for Kubernetes components to continue operating in the event that a node fails. The port is only accessed by the Kubernetes nodes and any admins using `kubectl`.
 - A network (L4, TCP) or application (L7, HTTP/S) load balancer for ports 80, and 443 across all primaries and secondaries. This maintains access to applications in event of a node failure. Include 8800 if you want external access to the Puppet Application Manager UI.

Note: Include port 8000 for webhook callbacks if you are installing Continuous Delivery for PE.

2. From a workstation with internet access, download the cluster installation bundle (note that this bundle is ~4GB):

```
https://k8s.kurl.sh/bundle/puppet-application-manager.tar.gz
```

3. Copy the installation bundle to your primary and secondary nodes and unpack it:

```
tar xzf puppet-application-manager.tar.gz
```

4. Run the installation command:

```
cat install.sh | sudo bash -s airgap
```

Note: An unformatted, unpartitioned storage device is required.

By default this installation automatically uses devices (under `/dev`) matching the pattern `vd[b-z]`. Attach a device to each host. Only devices that match the pattern, and are unformatted, are used.

If necessary, you can override this pattern by providing a patch during installation; append `-s installer-spec-file=patch.yaml` to the installation command.

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  rook:
    blockDeviceFilter: "sd[b-z]"
```

- a) When prompted for a load balancer address, enter the address of the DNS entry for your load balancer.
- b) The installation script prints the address and password (only shown once, so make careful note of it) for Puppet Application Manager:

```
---
Kotsadm: http://<PUPPET APPLICATION MANAGER ADDRESS>:8800
Login with password (will not be shown again): <PASSWORD>
---
```

Note: If you lose this password or wish to change it, see [Reset the PAM password](#) on page 122 for instructions.

5. Add two additional primary nodes to your offline installation using the instructions provided in the install script:

```
To add MASTER nodes to this installation, copy and unpack this bundle on
your other nodes, and run the following:
cat ./join.sh | sudo bash -s airgap
kubernetes-master-address=...
```

6. Add the two new nodes to your load balancer.
7. Navigate to the Puppet Application Manager UI using the address provided by the installation script (`http://<PUPPET_APPLICATION_MANAGER_ADDRESS>:8800`) and follow the prompts.

The Puppet Application Manager UI is where you manage Puppet applications. You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets application system requirements.

Follow the instructions for installing your Puppet applications on Puppet Application Manager. For more information, see [Install applications via the PAM UI](#) on page 102.

For more information on installing Continuous Delivery for PE offline, see [Install Continuous Delivery for PE in an offline environment](#).

For more information on installing Comply offline, see [Install Comply offline](#).

Related information

[Reset the PAM password](#) on page 122

As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

[PAM system requirements](#) on page 66

You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

[Resolve IP address range conflicts](#) on page 122

When installing Puppet Application Manager, IP address ranges `10.96.0.0/22` and `10.32.0.0/22` must not be used by other nodes on the local network.

[Architecture overview](#) on page 62

Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

[Using sudo behind a proxy server](#) on page 125

Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

PAM standalone online installation

The Puppet Application Manager (PAM) installation process sets up the application manager (with a simple Kubernetes installation for container orchestration) for you and installs the application on the single-node cluster.

Before you begin

1. Review the [Puppet Application Manager system requirements](#).
2. Note that Swap is not supported for use with this version of Puppet Application Manager (PAM). The installation script attempts to disable Swap if it is enabled.

3. (Optional) If necessary, prepare additional steps related to SELinux and Firewalld:

The PAM installation script disables SELinux and Firewalld by default. If you want to keep SELinux enabled, append the `-s preserve-selinux-config` switch to the PAM install command. This may require additional configuration to adapt SELinux policy to the installation.

If you want to keep Firewalld enabled:

- a. Make sure Firewalld is installed on your system.
- b. To prevent the installation from disabling Firewalld, provide a patch file to the PAM install command using `-s installer-spec-file=patch.yaml`, where `patch.yaml` is the name of your patch file. For reference, here's an example patch file that enables Firewalld during installation, starts the service if it isn't running, and adds rules to open relevant ports:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
sysctl -p"]
  firewalldCmds:
    - ["--permanent", "--zone=trusted", "--add-interface=flannel.1"]
    - ["--zone=external", "--add-masquerade"]
    # SSH port
    - ["--permanent", "--zone=public", "--add-port=22/tcp"]
    # HTTPS port
    - ["--permanent", "--zone=public", "--add-port=443/tcp"]
    # Kubernetes etcd port
    - ["--permanent", "--zone=public", "--add-port=2379-2830/tcp"]
    # Kubernetes API port
    - ["--permanent", "--zone=public", "--add-port=6443/tcp"]
    # Flannel Net port
    - ["--permanent", "--zone=public", "--add-port=8472/udp"]
    # CD4PE Webhook callback port (uncomment line below if needed)
    # - ["--permanent", "--zone=public", "--add-port=8000/tcp"]
    # KOTS UI port
    - ["--permanent", "--zone=public", "--add-port=8800/tcp"]
    # CD4PE Local registry port (offline only, uncomment line below if
needed)
    # - ["--permanent", "--zone=public", "--add-port=9001/tcp"]
    # Kubernetes component ports (kubelet, kube-scheduler, kube-
controller)
    - ["--permanent", "--zone=public", "--add-port=10250-10252/tcp"]
    # Reload firewall rules
    - ["--reload"]
  bypassFirewalldWarning: true
  disableFirewalld: false
  hardFailOnFirewalld: false
  preserveConfig: false
```

4. Ensure that IP address ranges `10.96.0.0/22` and `10.32.0.0/22` are locally accessible. See [Resolve IP address range conflicts](#) on page 122 for instructions.
5. If you use the [puppetlabs/firewall](#) module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the [puppetlabs/pam_firewall](#) module before installing Puppet Application Manager.

This installation process results in a basic Puppet Application Manager instance. Installation takes several (mostly hands-off) minutes to complete.

1. From the command line of your node, run the installation script:

```
curl -sSL https://k8s.kurl.sh/puppet-application-manager-standalone | sudo
bash
```

Tip: If you're installing Puppet Application Manager behind a proxy server, using `sudo` might cause the installation to fail. Try running the command as root (use command `sudo su -`) and replace `sudo bash` with `bash`.

- a) When the installation script prints the Puppet Application Manager address and password, make a careful note of these credentials:

```
---
Kotsadm: http://<PUPPET APPLICATION MANAGER ADDRESS>:8800
Login with password (will not be shown again): <PASSWORD>
---
```

Note: If you lose this password or wish to change it, see [Reset the PAM password](#) on page 122 for instructions.

- b) When the installation script is complete, run `bash -l` to reload the shell.

Tip: If the installation script fails, run the following and upload the results to the Puppet Support team:

```
kubectl support-bundle https://kots.io
```

If you're installing as the root user, run the command directly:

```
/usr/local/bin/kubectl-support_bundle https://kots.io
```

2. Navigate to the Puppet Application Manager UI using the address provided by the installation script (`http://<PUPPET APPLICATION MANAGER ADDRESS>:8800`) and follow the prompts.

The Puppet Application Manager UI is where you manage Puppet applications. You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets application system requirements.

Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For more information, see [Install applications via the PAM UI](#) on page 102.

For more information on installing Continuous Delivery for PE online, see [Install Continuous Delivery for PE](#).

For more information on installing Comply online, see [Install Comply online](#).

Related information

[Reset the PAM password](#) on page 122

As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

[PAM system requirements](#) on page 66

You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

[Resolve IP address range conflicts](#) on page 122

When installing Puppet Application Manager, IP address ranges `10.96.0.0/22` and `10.32.0.0/22` must not be used by other nodes on the local network.

[Architecture overview](#) on page 62

Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

[Using sudo behind a proxy server](#) on page 125

Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

PAM standalone offline installation

Use these instructions to install Puppet Application Manager (PAM) in an offline environment where the Puppet Application Manager host server does not have direct access to the internet.

Before you begin

1. Review the [Puppet Application Manager system requirements](#).
2. Note that Swap is not supported for use with this version of Puppet Application Manager (PAM). The installation script attempts to disable Swap if it is enabled.
3. (Optional) If necessary, prepare additional steps related to SELinux and Firewalld:

The PAM installation script disables SELinux and Firewalld by default. If you want to keep SELinux enabled, append the `-s preserve-selinux-config` switch to the PAM install command. This may require additional configuration to adapt SELinux policy to the installation.

If you want to keep Firewalld enabled:

- a. Make sure Firewalld is installed on your system.
- b. To prevent the installation from disabling Firewalld, provide a patch file to the PAM install command using `-s installer-spec-file=patch.yaml`, where `patch.yaml` is the name of your patch file. For reference, here's an example patch file that enables Firewalld during installation, starts the service if it isn't running, and adds rules to open relevant ports:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
sysctl -p"]
  firewalldCmds:
    - ["--permanent", "--zone=trusted", "--add-interface=flannel.1"]
    - ["--zone=external", "--add-masquerade"]
    # SSH port
    - ["--permanent", "--zone=public", "--add-port=22/tcp"]
    # HTTPS port
    - ["--permanent", "--zone=public", "--add-port=443/tcp"]
    # Kubernetes etcd port
    - ["--permanent", "--zone=public", "--add-port=2379-2830/tcp"]
    # Kubernetes API port
    - ["--permanent", "--zone=public", "--add-port=6443/tcp"]
    # Flannel Net port
    - ["--permanent", "--zone=public", "--add-port=8472/udp"]
    # CD4PE Webhook callback port (uncomment line below if needed)
    # - ["--permanent", "--zone=public", "--add-port=8000/tcp"]
    # KOTS UI port
    - ["--permanent", "--zone=public", "--add-port=8800/tcp"]
    # CD4PE Local registry port (offline only, uncomment line below if
needed)
    # - ["--permanent", "--zone=public", "--add-port=9001/tcp"]
    # Kubernetes component ports (kubelet, kube-scheduler, kube-
controller)
```

```

- ["--permanent", "--zone=public", "--add-port=10250-10252/tcp"]
# Reload firewall rules
- ["--reload"]
bypassFirewalldWarning: true
disableFirewalld: false
hardFailOnFirewalld: false
preserveConfig: false

```

4. Ensure that IP address ranges 10.96.0.0/22 and 10.32.0.0/22 are locally accessible. See [Resolve IP address range conflicts](#) on page 122 for instructions.
5. Ensure that the nodes can resolve their own hostnames, through either local host mapping or a reachable DNS server.
6. If you use the [puppetlabs/firewall](#) module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the [puppetlabs/pam_firewall](#) module before installing Puppet Application Manager.
7. If you're restoring a backup from a previous cluster, make sure you include the `kurl-registry-ip=<YOUR_IP_ADDRESS>` installation option. For more information, see [Migrating PAM data to a new system](#) on page 114.

This installation process results in a basic Puppet Application Manager instance. Installation takes several (mostly hands-off) minutes to complete.

1. From a workstation with internet access, download the cluster installation bundle (note that this bundle is ~4GB):

```
https://k8s.kurl.sh/bundle/puppet-application-manager-standalone.tar.gz
```

2. Copy the installation bundle to the host node and unpack it:

```
tar xzf puppet-application-manager-standalone.tar.gz
```

3. Run the installation command:

```
cat install.sh | sudo bash -s airgap
```

- a) The installation script prints the address and password (only shown once, so make careful note of it) for Puppet Application Manager:

```

---
Kotsadm: http://<PUPPET APPLICATION MANAGER ADDRESS>:8800
Login with password (will not be shown again): <PASSWORD>
---

```

Note: If you lose this password or wish to change it, see [Reset the PAM password](#) on page 122 for instructions.

4. Navigate to the Puppet Application Manager UI using the address provided by the installation script (`http://<PUPPET APPLICATION MANAGER ADDRESS>:8800`) and follow the prompts.

The Puppet Application Manager UI is where you manage Puppet applications. You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets application system requirements.

Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For more information, see [Install applications via the PAM UI](#) on page 102.

For more information on installing Continuous Delivery for PE offline, see [Install Continuous Delivery for PE in an offline environment](#).

For more information on installing Comply offline, see [Install Comply offline](#).

Related information

[Reset the PAM password](#) on page 122

As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

[PAM system requirements](#) on page 66

You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

[Resolve IP address range conflicts](#) on page 122

When installing Puppet Application Manager, IP address ranges `10.96.0.0/22` and `10.32.0.0/22` must not be used by other nodes on the local network.

[Architecture overview](#) on page 62

Puppet Application Manager (PAM) runs on Kubernetes. We provide several supported configurations for different use cases.

[Using sudo behind a proxy server](#) on page 125

Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

Automate PAM and Puppet application online installations

During a fresh online installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

Before you begin

Ensure that your system meets the [PAM system requirements](#) on page 66.

1. Install Puppet Application Manager. For detailed instructions, see [PAM HA online installation](#) on page 83.

2. Define the configuration values for your Puppet application installation, using Kubernetes YAML format.

```
apiVersion: kots.io/v1beta1
kind: ConfigValues
metadata:
  name: app-config
spec:
  values:
    accept_eula:
      value: has_accepted_eula
  annotations:
    value: "ingress.kubernetes.io/force-ssl-redirect: 'false'"
  hostname:
    value: "<HOSTNAME>"
  root_password:
    value: "<ROOT ACCOUNT PASSWORD>"
```

Tip: View the keyword names for all settings by clicking **View files > upstream > config.yaml** in Puppet Application Manager.

Replace the values indicated:

- Replace <HOSTNAME> with a hostname you want to use to configure an Ingress and to tell job hardware agents and web hooks how to connect to it. You might need to configure your DNS to resolve the hostname to your Kubernetes hosts.
- Replace <ROOT ACCOUNT PASSWORD> your chosen password for the application root account. The root account is used to administer your application and has full access to all resources and application-wide settings. This account must NOT be used for testing and deploying control repositories or modules.
- **Optional.** These configuration values disable HTTP-to-HTTPS redirection, so that SSL can be terminated at the load balancer. If you want to run the application over SSL only, change the `force-ssl-redirect` annotation to `true`.
- **Optional.** If your load balancer requires HTTP health checks, you can now enable Ingress settings that do not require Server Name Indication (SNI) for `/status`. To enable this setting, add the following to the config values statement:

```
enable_lb_healthcheck:
  value: "1"
```

Note: The automated installation automatically accepts the Puppet application end user license agreement (EULA). Unless Puppet has otherwise agreed in writing, all software is subject to the terms and conditions of the Puppet Master License Agreement located at <https://puppet.com/legal>.

3. Write your license file and the configuration values generated in step 1 to the following locations:
 - Write your license file to `./replicated_license.yaml`
 - Write your configuration values to `./replicated_config.yaml`
4. Add the Puppet application definition to Puppet Application Manager with the license file and configuration values, passing in the Puppet Application Manager password you set in step 4:

```
kubectl kots install <APPLICATION NAME> --namespace default --shared-
password <YOUR CHOSEN PASSWORD> --port-forward=false \
  --license-file ./replicated_license.yaml --config-values ./
replicated_config.yaml
```

Note: If you want to install a specific version of the application, include the `--app-version-label=<VERSION>` flag in the install command.

5. Wait five minutes to allow the software time to process the change.

6. Navigate to `http://<NODE_IP_ADDRESS>:8800` and log in with the Puppet Application Manager password.

Your configuration values are applied, and if preflight checks have passed, the application is deployed and in the process of starting up.

The application's status on the **Application** tab is shown as **Missing** for several minutes while deployment is underway. To monitor the deployment's progress, run `kubectl get pods --watch`.

When the deployment is complete, the application status changes to **Ready**.

7. Update your DNS or `/etc/hosts` file to include the hostname you chose during configuration.
8. Installation is now complete! Navigate to `https://<HOSTNAME>` and sign into Puppet application.

Related information

[PAM HA online installation](#) on page 83

The Puppet Application Manager (PAM) installation process creates a Kubernetes cluster for you and walks you through installing your Puppet application on the cluster.

[Upgrade an automated online application installation](#) on page 103

If you installed a Puppet application following the automated online installation instructions, run a script to upgrade to the latest version.

Automate PAM and Puppet application offline installations

During a fresh offline installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

Before you begin

Ensure that your system meets the [PAM system requirements](#) on page 66.

Related information

[PAM HA offline installation](#) on page 87

Use these instructions to install Puppet Application Manager (PAM) in an air-gapped or offline environment where the Puppet Application Manager host server does not have direct access to the internet.

[Upgrade an automated offline application installation](#) on page 104

If you installed a Puppet application following the automated offline installation instructions, run a script to upgrade to the latest version.

Automate PAM and Puppet application offline installations on Puppet-supported clusters

1. Install Puppet Application Manager. For detailed instructions, see [PAM HA offline installation](#) on page 87.

- Define the configuration values for your Puppet application installation, using Kubernetes YAML format.

```
apiVersion: kots.io/v1beta1
kind: ConfigValues
metadata:
  name: app-config
spec:
  values:
    accept_eula:
      value: has_accepted_eula
  annotations:
    value: "ingress.kubernetes.io/force-ssl-redirect: 'false'"
  hostname:
    value: "<HOSTNAME>"
  root_password:
    value: "<ROOT ACCOUNT PASSWORD>"
```

Tip: View the keyword names for all settings by clicking **View files > upstream > config.yaml** in Puppet Application Manager.

Replace the values indicated:

- Replace <HOSTNAME> with a hostname you want to use to configure an Ingress and to tell job hardware agents and web hooks how to connect to it. You might need to configure your DNS to resolve the hostname to your Kubernetes hosts.
- Replace <ROOT ACCOUNT PASSWORD> your chosen password for the application root account. The root account is used to administer your application and has full access to all resources and application-wide settings. This account must NOT be used for testing and deploying control repositories or modules.
- Optional.** These configuration values disable HTTP-to-HTTPS redirection, so that SSL can be terminated at the load balancer. If you want to run the application over SSL only, change the `force-ssl-redirect` annotation to `true`.
- Optional.** If your load balancer requires HTTP health checks, you can now enable Ingress settings that do not require Server Name Indication (SNI) for `/status`. To enable this setting, add the following to the config values statement:

```
enable_lb_healthcheck:
  value: "1"
```

Note: The automated installation automatically accepts the Puppet application end user license agreement (EULA). Unless Puppet has otherwise agreed in writing, all software is subject to the terms and conditions of the Puppet Master License Agreement located at <https://puppet.com/legal>.

- Write your license file and the configuration values generated in the previous step to the following locations:
 - Write your license file to `./replicated_license.yaml`
 - Write your configuration values to `./replicated_config.yaml`
- Download the application bundle:

```
curl -L <APPLICATION BUNDLE URL> -o <APPLICATION BUNDLE FILE>
```

- Copy the application bundle to your primary and secondary nodes and unpack it:

```
tar xzf ./<APPLICATION BUNDLE FILE>
```

- Run the application install command on your primary node. Replace the `<YOUR CHOSEN PASSWORD>` , `<APPLICATION NAME>`, `<APPLICATION BUNDLE FILE>` values in the example below with your own values:

```
KOTS_PASSWORD=<YOUR CHOSEN PASSWORD>
kubectl kots install <APPLICATION NAME> --namespace default --shared-
password $KOTS_PASSWORD --license-file ./license.yaml --config-
values ./config.yaml --airgap-bundle ./<APPLICATION BUNDLE FILE> --port-
forward=false
# wait several minutes for the application to deploy; if it doesn't show
up, preflights or another error might have occurred
```

Note: If you want to install a specific version of the application, include the `--app-version-label=<VERSION>` flag in the install command.

Automate PAM and Puppet application offline installations on customer-supported clusters

Before you begin

- If you haven't already done so, [install kubectl](#).
- Puppet Application Manager is expected to work on any certified Kubernetes distribution that meets the following requirements. We validated and support:
 - Google Kubernetes Engine
 - AWS Elastic Kubernetes Service

If you use a different distribution, contact [Puppet Support](#) for more information on compatibility with PAM.

- Make sure your Kubernetes cluster meets the minimum requirements:
 - Kubernetes version 1.24-1.26.
 - A default storage class that can be used for relocatable storage.
 - A standard Ingress controller that supports websockets (we have tested with Project Contour and NGINX).
 - We currently test and support Google Kubernetes Engine (GKE) clusters.

Note: If you're using self-signed certificates on your Ingress controller, you must ensure that your job hardware nodes trust the certificates. Additionally, all nodes that use Continuous Delivery for PE webhooks must trust the certificates, or SSL checking must be disabled on these nodes.

Important: If you are installing Puppet Comply on Puppet Application Manager, the ingress controller must be configured to allow request payloads of up to 32 MB. Ingress controllers used by Amazon EKS commonly default to a 1 MB maximum — this causes all report submissions to fail.

The ingress must have a generous limit for total connection time. Setting the connection timeout to `infinity` in conjunction with an idle timeout is recommended.

- If you are setting up Puppet Application Manager behind a proxy server, the installer supports proxies configured via `HTTP_PROXY/HTTPS_PROXY/NO_PROXY` environment variables.

Restriction: Using a proxy to connect to external version control systems is currently not supported.

1. Define the configuration values for your Puppet application installation, using Kubernetes YAML format.

```
apiVersion: kots.io/v1beta1
kind: ConfigValues
metadata:
  name: app-config
spec:
  values:
    accept_eula:
      value: has_accepted_eula
  annotations:
    value: "ingress.kubernetes.io/force-ssl-redirect: 'false'"
  hostname:
    value: "<HOSTNAME>"
  root_password:
    value: "<ROOT ACCOUNT PASSWORD>"
```

Tip: View the keyword names for all settings by clicking **View files > upstream > config.yaml** in Puppet Application Manager.

Replace the values indicated:

- Replace <HOSTNAME> with a hostname you want to use to configure an Ingress and to tell job hardware agents and web hooks how to connect to it. You might need to configure your DNS to resolve the hostname to your Kubernetes hosts.
- Replace <ROOT ACCOUNT PASSWORD> your chosen password for the application root account. The root account is used to administer your application and has full access to all resources and application-wide settings. This account must NOT be used for testing and deploying control repositories or modules.
- **Optional.** These configuration values disable HTTP-to-HTTPS redirection, so that SSL can be terminated at the load balancer. If you want to run the application over SSL only, change the `force-ssl-redirect` annotation to `true`.
- **Optional.** If your load balancer requires HTTP health checks, you can now enable Ingress settings that do not require Server Name Indication (SNI) for `/status`. To enable this setting, add the following to the config values statement:

```
enable_lb_healthcheck:
  value: "1"
```

Note: The automated installation automatically accepts the Puppet application end user license agreement (EULA). Unless Puppet has otherwise agreed in writing, all software is subject to the terms and conditions of the Puppet Master License Agreement located at <https://puppet.com/legal>.

2. Write your license file and the configuration values generated in the previous step to the following locations:
 - Write your license file to `./replicated_license.yaml`
 - Write your configuration values to `./replicated_config.yaml`
3. Download the application bundle:

```
curl -L <APPLICATION BUNDLE URL> -o <APPLICATION BUNDLE FILE>
```


- [Install applications via the PAM UI](#) on page 102

The process of adding an application once you've installed Puppet Application Manager is simple.

- [Update a license for online installations](#) on page 103

If you have performed online installation of an application, you can use the Puppet Application Manager UI to update your license.

- [Update a license for offline installations](#) on page 103

If you have performed offline installation of an application, you can use the Puppet Application Manager UI to update your license.

- [Upgrade an automated online application installation](#) on page 103

If you installed a Puppet application following the automated online installation instructions, run a script to upgrade to the latest version.

- [Upgrade an automated offline application installation](#) on page 104

If you installed a Puppet application following the automated offline installation instructions, run a script to upgrade to the latest version.

Install applications via the PAM UI

The process of adding an application once you've installed Puppet Application Manager is simple.

Important: Ensure you are using the following Puppet application versions if you want to add more than one Puppet application via the Puppet Application Manager UI:

Application	Version
Continuous Delivery for Puppet Enterprise	4.6.0 or later
Comply	1.0.4 or later

For information on installing Puppet applications via the command line, see [Automate PAM and Puppet application online installations](#) on page 95 and [Automate PAM and Puppet application offline installations](#) on page 97.

To install a Puppet application using the Puppet Application Manager UI:

1. Log into the Puppet Application Manager UI, and click **Add a new application**.
 - If you have not added a Puppet application before you are prompted to upload a license.
 - If you have already added a Puppet application, click **Add a new application**.
2. Upload your `replicated_license.yaml` file when requested.

Note: Once the license file is installed, if offline installations are enabled, you are presented with an option to proceed with an offline setup.

Add the following information to install an offline application:

- **Hostname** - the hostname you want to use to configure an Ingress and to tell job hardware agents and web hooks how to connect to it. You might need to configure your DNS to resolve the hostname to your Kubernetes hosts.

Important: The hostname must be unique for each application you install.

- **Username/Password** - The username and password for the application root account. The root account is used to administer your application and has full access to all resources and application-wide settings. This account must NOT be used for testing and deploying control repositories or modules.
- **Registry namespace** - the registry namespace for the application, e.g. *CD4PE* or *Comply*.
- **Airgap bundle** - upload the relevant application bundle tarball. Click **Continue**.

3. Add any additional required information that is presented on the **Config** page. Configure any other settings on the page relevant to your installation, such as external databases, customized endpoints, a load balancer, or TLS certificates. Click **Save Config** when you are done.

Saving your new configuration settings prompts the creation of a new application version.

4. Click **Go to new version**, which redirects you to the **Version history** tab. The newly created version is shown in the **All versions** section of the page.
5. Monitor the new version's preflight checks. The **Running Checks** indicator is shown on the screen while your system is checked to make sure your cluster meets minimum system requirements. When the preflight check is complete:
 - If the status is **Checks Failed**, click **View preflights**. Correct the issues and click **Re-run**. Repeat this step as needed.

Important: Do not move on until all preflight checks pass.

- If the status is **Ready to Deploy**, move on to the next step.
6. Once the version is ready to deploy, click **Deploy**. On the **Application** tab, monitor the application for readiness. The application's status is shown as **Missing** for several minutes while deployment is underway. To monitor the deployment's progress, run `kubectl get pods --watch`.
When the deployment is complete, the application status changes to **Ready**.
 7. Navigate to `https://<HOSTNAME>` (using the hostname you entered on the **Config** screen) and sign into your application.

Update a license for online installations

If you have performed online installation of an application, you can use the Puppet Application Manager UI to update your license.

To update the license for an online application:

1. Log in to Puppet Application Manager, click the **License** tab, and then **Sync License**.
2. On the **Version history** tab, click **Deploy**.

Puppet Application Manager adds “License Change” as the deployment cause on the **Version history** tab.

Update a license for offline installations

If you have performed offline installation of an application, you can use the Puppet Application Manager UI to update your license.

To update the license for an offline application:

1. Ask your Puppet sales representative to email you an updated license file.
2. Log in to Puppet Application Manager, click the **License** tab.
3. Drag and drop or upload the updated license file provided by your Puppet sales representative.
4. On the **Version history** tab, click **Deploy**.

Puppet Application Manager adds “License Change” as the deployment cause on the **Version history** tab.

Upgrade an automated online application installation

If you installed a Puppet application following the automated online installation instructions, run a script to upgrade to the latest version.

Important: Ensure that you are following an approved upgrade path for the application you want to upgrade. For more information, check the relevant application documentation.

1. From the command line of your primary (control plane) node, get the *application slug* for the application you want to upgrade:

```
kubectl kots --namespace <NAMESPACE> get apps
```

Replace <NAMESPACE> with the name of the namespace in which you installed PAM (usually default).

2. Run the upgrade script:

```
kubectl kots upstream upgrade <APPLICATION SLUG> --namespace <NAMESPACE> --deploy
```

Replace <APPLICATION SLUG> with the relevant application slug for the application you want to upgrade.

Replace <NAMESPACE> with the name of the namespace in which you installed PAM (usually default).

3. Wait five minutes to allow the software time to process the change.
4. Navigate to `http://<NODE IP ADDRESS>:8800` and log in with the Puppet Application Manager password.

If preflight checks have passed, the upgraded application is deployed and in the process of starting up. To monitor the deployment's progress, run:

```
kubectl get pods --watch
```

Related information

[Automate PAM and Puppet application offline installations](#) on page 97

During a fresh offline installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

Upgrade an automated offline application installation

If you installed a Puppet application following the automated offline installation instructions, run a script to upgrade to the latest version.

Important: Ensure that you are following an approved upgrade path for the application you want to upgrade. For more information, check the relevant application documentation.

1. Download the application bundle you want to upgrade to. Copy to your primary node.
2. From the command line of your primary (control plane) node, get the *application slug* for the application you want to upgrade:

```
kubectl kots --namespace <NAMESPACE> get apps
```

Replace <NAMESPACE> with the name of the namespace in which you installed PAM (usually default).

3. Run the upgrade script:

```
kubectl kots upstream upgrade <APPLICATION SLUG> --airgap-bundle ./<APPLICATION BUNDLE FILE> --kotsadm-namespace <REGISTRY NAMESPACE> --namespace <NAMESPACE> --deploy
```

- Replace <APPLICATION SLUG> with the relevant application slug for the application you want to upgrade.
 - Replace <APPLICATION BUNDLE FILE> with the name of the application bundle file.
 - Replace <REGISTRY NAMESPACE> with your Registry namespace where images are uploaded.
 - Replace <NAMESPACE> with the name of the namespace in which you installed PAM (usually default).
4. Wait five minutes to allow the software time to process the change.

5. Navigate to `http://<NODE IP ADDRESS>:8800` and log in with the Puppet Application Manager password.

If preflight checks have passed, the upgraded application is deployed and in the process of starting up. To monitor the deployment's progress, run:

```
kubectl get pods --watch
```

Related information

[Automate PAM and Puppet application offline installations](#) on page 97

During a fresh offline installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

Maintenance and tuning

Follow these guidelines when you're tuning or performing maintenance on a node running Puppet Application Manager (PAM).

How to look up your Puppet Application Manager architecture

If you're running PAM on a Puppet-supported cluster, you can use the following command to determine your PAM architecture version:

```
kubectl get installer --sort-by=.metadata.creationTimestamp -o
  jsonpath='{.items[-1:].metadata.name}' ; echo
```

Depending on which architecture you used when installing, the command returns one of these values:

- **HA architecture:** `puppet-application-manager`
- **Standalone architecture:** `puppet-application-manager-standalone`
- **Legacy architecture:** Any other value, for example, `puppet-application-manager-legacy`, `cd4pe`, or `comply`

Rebooting PAM nodes

Where possible, avoid rebooting or shutting down a PAM node. Shutting down an HA PAM node incorrectly could result in storage volume corruption and the loss of data.

For tasks such as package updates or security patches, where you must perform a reboot or shut down, follow the procedure below to gracefully shut down the node and ensure that it is drained correctly.

To reboot a node:

1. Shut down services using Ceph-backed storage:

```
/opt/ekco/shutdown.sh
```

2. If you're using a high availability (HA) cluster, drain the node:

```
kubectl drain <NODE NAME> --ignore-daemonsets --delete-local-data
```

3. Reboot the node.

Load balancer health checks

To set up health checks for the load balancer that your Puppet Application Manager (PAM) applications are running behind, set up rules for these applications and services.

Application/service	URL/port	Notes
Puppet application. For example, Continuous Delivery for Puppet Enterprise or Puppet Comply	<code>https://<CDPE HOSTNAME>:443/status</code>	Although Puppet applications might expose other ports (Continuous Delivery for PE exposes ports 443, 80, and 8000), 443 is the HTTPS endpoint, and is the best port to use for health checks.
Puppet Application Manager (PAM)	<code>https://<KUBERNETES PRIMARY IP>:8800/healthz</code>	
External load balancer endpoint	Port 6443 or <code>https://<KUBERNETES PRIMARY IP>:6443/livez</code>	For information on setting up a TCP probe on an external load balancer endpoint, consult the kURL load balancer documentation .
Local container registry (for offline installations)	<code>https://<KUBERNETES PRIMARY IP>:9001</code>	

Load balancing

The following load balancer requirements are needed for a HA install:

- A network (L4, TCP) load balancer for port 6443 across primary nodes. This is required for Kubernetes components to continue operating in the event that a node fails. The port is only accessed by the Kubernetes nodes and any admins using `kubectl`.
- A network (L4, TCP) or application (L7, HTTP/S) load balancer for ports 80, and 443 across all primaries and secondaries. This maintains access to applications in event of a node failure. Include 8800 if you want external access to the Puppet Application Manager UI.

Note: Include port 8000 for webhook callbacks if you are installing Continuous Delivery for PE.

Important: If you are using application load balancing, be aware that Ingress items use Server Name Indication (SNI) to route requests, which may require additional configuration with your load balancer. If your load balancer does not support SNI for health checks, enable **Enable load balancer HTTP health check** in the Puppet Application Manager UI **Config** page .

Upgrading PAM on a Puppet-supported cluster

Upgrade Puppet Application Manager (PAM) on a Puppet-supported cluster to take advantage of new features and bug fixes, and to upgrade your cluster to the latest version of Kubernetes when one is available.

There are four possible upgrade types for Puppet Application Manager installations:

- **Online** - For standalone or HA installations with a connection to the internet.
- **Offline** - For air-gapped standalone or HA installations without a connection to the internet.
- **Online legacy** - For standalone or HA installations created prior to April 2021 with a connection to the internet.
- **Offline legacy** - For air-gapped standalone or HA installations created prior to April 2021 without a connection to the internet.

Restriction: You cannot use the upgrade process to move from a legacy deployment to a non-legacy deployment, or from standalone to HA, or vice versa. If you wish to change architecture types, see [Migrating PAM data to a new system](#) on page 114.

How to look up your Puppet Application Manager architecture

If you're running PAM on a Puppet-supported cluster, you can use the following command to determine your PAM architecture version:

```
kubectl get installer --sort-by=.metadata.creationTimestamp -o
  jsonpath='{.items[-1:].metadata.name}' ; echo
```

Depending on which architecture you used when installing, the command returns one of these values:

- **HA architecture:** `puppet-application-manager`
- **Standalone architecture:** `puppet-application-manager-standalone`
- **Legacy architecture:** Any other value, for example, `puppet-application-manager-legacy`, `cd4pe`, or `comply`

Upgrade PAM online

Upgrade Puppet Application Manager (PAM) to take advantage of new features and bug fixes, and to upgrade your cluster to the latest version of Kubernetes when one is available.

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at [Backing up PAM using snapshots](#) on page 112.

If you are upgrading from a version of PAM that used Weave (versions 1.100.3 and earlier) to a version of PAM that uses Flannel (versions 1.102.2 and later), pod-to-pod networking now depends on UDP port 8472 being open instead of ports 6783 and 6784.

Note: Starting with Puppet Application Manager 1.97.0, the `force-reapply-addons` flag is deprecated and generates a warning on use. If you are upgrading to a version prior to 1.97.0, you need to add the `force-reapply-addons` flag to the `bash` command using the `-s` flag.

1. On your first primary node, rerun the installation script, passing in any arguments you included when installing for the first time:

For standalone deployments, use:

```
curl -sSL https://k8s.kurl.sh/puppet-application-manager-standalone | sudo
  bash
```

For HA deployments, use:

```
curl -sSL https://k8s.kurl.sh/puppet-application-manager | sudo bash
```

2. If a new version of Kubernetes is available, the installer notes upgrade scripts to run on other nodes in an HA cluster.

The installer also pauses before draining nodes as part of the Kubernetes upgrade. The node draining process can take several minutes to complete, during which time application workloads are stopped or migrated to other systems. This migration may cause several minutes of downtime while databases are rescheduled.

Related information

[Using sudo behind a proxy server](#) on page 125

Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

Upgrade PAM offline

Users operating in environments without direct access to the internet must use the links below to upgrade to the latest version of Puppet Application Manager (PAM).

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at [Backing up PAM using snapshots](#) on page 112.

If you are upgrading from a version of PAM that used Weave (versions 1.100.3 and earlier) to a version of PAM that uses Flannel (versions 1.102.2 and later), pod-to-pod networking now depends on UDP port 8472 being open instead of ports 6783 and 6784.

Note: Starting with Puppet Application Manager 1.97.0, the `force-reapply-addons` flag is deprecated and generates a warning on use. If you are upgrading to a version prior to 1.97.0, you need to add the `force-reapply-addons` flag in **Step 3** to the bash command after `-s airgap`.

To upgrade Puppet Application Manager:

1. From a workstation with internet access, download the latest version of the installation bundle that is relevant for your installation type:

For standalone installations, enter the following command (note that this bundle is ~4GB):

```
curl -LO https://k8s.kurl.sh/bundle/puppet-application-manager-standalone.tar.gz
```

For HA installations, enter the following command (note that this bundle is ~4GB):

```
curl -LO https://k8s.kurl.sh/bundle/puppet-application-manager.tar.gz
```

2. Copy the installation bundle to your primary and secondary nodes and unpack it:

For standalone installations, use:

```
tar xzf puppet-application-manager-standalone.tar.gz
```

For HA installations, use:

```
tar xzf puppet-application-manager.tar.gz
```

3. Manually load the images from the installation bundle:

```
cat tasks.sh | bash -s load-images
```

4. On your primary node, rerun the installation script, passing in any arguments you included when installing for the first time:

```
cat install.sh | sudo bash -s airgap
```

Note: This script issues a prompt to run the `task.sh` and `upgrade.sh` scripts on your secondary nodes. Use the versions of these scripts from the downloaded bundle in step 2.

5. If a new version of Kubernetes is available, the installer systems provide upgrade scripts to run on other nodes in an HA cluster. The installer also pauses before draining nodes as part of the Kubernetes upgrade. Node draining is performed as part of a Kubernetes upgrade.

The node draining process can take several minutes to complete, during which time application workloads are stopped or migrated to other systems. This migration may cause several minutes of downtime while databases are rescheduled.

When the deployment is complete, sign into Puppet Application Manager- `http://<PUPPET_APPLICATION_MANAGER_ADDRESS>:8800` - and verify that the new version number is displayed in the bottom left corner of the web UI.

PAM legacy upgrades

The legacy architecture is no longer supported. However, if you have not yet migrated to a supported architecture, you can use this method to upgrade Puppet Application Manager (PAM).

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at [Backing up PAM using snapshots](#) on page 112.

Legacy architecture is no longer supported: The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. The legacy architecture reached the end of its support lifecycle on **30 June 2022**, and Puppet no longer updates legacy architecture components. For information on migrating data from a legacy architecture to a standalone or HA architecture, go to our Support Knowledge Base instructions:

- [Migrate to a supported PAM architecture for Continuous Delivery for PE](#)
- [Migrate to a supported PAM architecture for Comply](#)

Restriction: It is not possible to upgrade from an online legacy install to a new offline install configuration. Similarly, upgrades from an offline legacy configuration to a new online install are not supported.

To upgrade a legacy version of Puppet Application Manager on nodes with internet access:

1. On your node (or control plane node if you have a HA deployment), rerun the installation script, passing in any arguments you included when installing for the first time:

- For standalone installs:

```
curl -sSL https://k8s.kurl.sh/puppet-application-manager-legacy | sudo
  bash -s force-reapply-addons
```

- For HA installs:

```
curl -sSL https://k8s.kurl.sh/puppet-application-manager-legacy | sudo
  bash -s ha force-reapply-addons
```

2. If a new version of Kubernetes is available, the systems provide upgrade scripts to run on each node in your cluster.

Node draining is performed as part of a Kubernetes upgrade. The node draining process can take several minutes to complete.

Note: During the Kubernetes upgrade process, nodes are not able to properly route network connections. If you have a HA deployment, make sure you have load balancers or a multi-node fail-over process in place, or schedule downtime before upgrading.

PAM offline legacy upgrades

The legacy architecture is no longer supported. However, if you have not yet migrated to a supported architecture, you can use this method to upgrade Puppet Application Manager (PAM) on offline nodes.

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at [Backing up PAM using snapshots](#) on page 112.

Legacy architecture is no longer supported: The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. The legacy architecture reached the end of its support lifecycle on **30 June 2022**, and Puppet no longer updates legacy architecture components. For information on migrating data from a legacy architecture to a standalone or HA architecture, go to our Support Knowledge Base instructions:

- [Migrate to a supported PAM architecture for Continuous Delivery for PE](#)
- [Migrate to a supported PAM architecture for Comply](#)

Restriction: It is not possible to upgrade from an online legacy install to a new offline install configuration. Similarly, upgrades from an offline legacy configuration to a new online install are not supported.

To upgrade Puppet Application Manager on nodes without a connection to the internet:

1. From a workstation with internet access, download the latest version of the cluster installation bundle (note that this bundle is ~4GB):

```
https://k8s.kurl.sh/bundle/puppet-application-manager-legacy.tar.gz
```

2. Copy the installation bundle to your primary and secondary Puppet Application Manager nodes and unpack it:

```
tar xzf puppet-application-manager-legacy.tar.gz
```

3. Rerun the installation script. Don't forget to pass in any additional arguments you included when installing for the first time you installed the product:

For standalone installs use:

```
cat install.sh | sudo bash -s airgap force-reapply-addons
```

For HA installs use:

```
cat install.sh | sudo bash -s airgap ha force-reapply-addons
```

Note: During the upgrade process, follow any prompts to run commands on your other cluster nodes.

When the deployment is complete, sign into Puppet Application Manager and verify that the new version number is displayed in the bottom center of the web UI.

Upgrading PAM on a customer-supported cluster

Upgrade Puppet Application Manager (PAM) on your own Kubernetes cluster to take advantage of new features and bug fixes.

There are two possible upgrade types for customer-supported Puppet Application Manager deployments:

- **Online** - For installations with a connection to the internet.
- **Offline** - For air-gapped installations without a connection to the internet.

Upgrade PAM on a customer-supported online cluster

Upgrading Puppet Application Manager (PAM) on a customer-supported online Kubernetes cluster can be done with a single command.

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at [Backing up PAM using snapshots](#) on page 112.

To upgrade Puppet Application Manager on a customer-supported online cluster:

1. Upgrade kubectl KOTS:

```
curl https://kots.io/install | bash
```

2. Issue the following KOTS command:

```
kubectl kots admin-console upgrade --namespace <target namespace>
```

Tip: Run the `kubectl kots admin-console upgrade -h` command for more usage information.

Upgrade PAM on a customer-supported offline cluster

Upgrading Puppet Application Manager (PAM) on a customer-supported offline Kubernetes cluster requires a few simple kubectl commands.

Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at [Backing up PAM using snapshots](#) on page 112.

To upgrade Puppet Application Manager on a customer-supported offline cluster, perform the following steps from a workstation that has kubectl access to the cluster:

1. Upgrade kubectl KOTS:

```
curl https://kots.io/install | bash
```

2. Ensure the required images are available in your local registry. Download the release assets matching the CLI version using the following command:

```
curl -LO https://github.com/replicatedhq/kots/releases/download/v$(kubectl kots version | head -n1 | cut -d' ' -f3)/kotsadm.tar.gz
```

3. Extract the images and push them to your private registry. Registry credentials provided in this step must have push access. These credentials are not stored anywhere or reused later.

```
kubectl kots admin-console push-images ./kotsadm.tar.gz
  <private.registry.host>/puppet-application-manager \
--registry-username <rw-username> \
--registry-password <rw-password>
```

4. After you push the images to your private registry, execute the upgrade command with registry read-only credentials:

```
kubectl kots upgrade puppet-application-manager \
--kotsadm-namespace puppet-application-manager \
--kotsadm-registry <private.registry.host> \
--registry-username <ro-username> \
--registry-password <ro-password> \
--namespace <target namespace>
```

Backing up PAM using snapshots

Snapshots are point-in-time backups of your Puppet Application Manager (PAM) deployment, which can be used to roll back to a previous state or restore your installation into a new cluster for disaster recovery.

Related information

[Using sudo behind a proxy server](#) on page 125

Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

Full and partial snapshots

There are two options available when you're creating a snapshot for your Puppet Application Manager (PAM) deployment, full snapshots (also known as instance snapshots) and partial (or application) snapshots. For full disaster recovery, make sure you've configured and scheduled regular full snapshots stored on a remote storage solution such as an S3 bucket or NFS share.

Full snapshots offer a comprehensive backup of your PAM deployment, because they include the core PAM application together with the Puppet applications you've installed in your PAM deployment. You can use a full snapshot to restore your PAM deployment and all of your installed Puppet applications to a previous backup. For example, you could use a full snapshot to revert an undesired configuration change or a failed upgrade, or to migrate your PAM deployment to another Puppet-supported cluster.

Partial snapshots are available from the PAM console, but are limited in their usefulness. To restore from a partial snapshot, you must already have an installed and functioning version of PAM. A functioning PAM installation is needed because the option to restore a partial snapshot can only be accessed from the **Snapshots** section of the PAM admin console.

Partial snapshots only back up the Puppet application you specified when you configured the snapshot, for example, Continuous Delivery for Puppet Enterprise, or Puppet Comply. They do not back up the underlying PAM deployment. Partial snapshots are sometimes useful if you want to roll back to a previous version of a specific Puppet application that you've installed on your PAM deployment, but are far less versatile than full snapshots. To make sure that you have all disaster recovery options available to you, use a full snapshot wherever possible.

Configure snapshots

Before using snapshots, select a storage location, set a snapshot retention period, and indicate whether snapshots are created manually or on a set schedule.

Important: Disaster recovery requires that the store backend used for backups is accessible from the new cluster. When setting up snapshots in an *offline* cluster, make sure to record the registry service IP address with the following command:

```
kubectl -n kurl get svc registry -o jsonpath='{.spec.clusterIP}'
```

Be sure to record the value returned by this command as it is required when creating a new cluster to restore to as part of [Disaster recovery with PAM](#) on page 120.

1. In the upper navigation bar of the Puppet Application Manager UI, click **Snapshots > Settings & Schedule**.
2. The snapshots feature uses <https://velero.io>, an open source backup and restore tool. Click **Check for Velero** to determine whether Velero is present on your cluster, and to install it if needed.

3. Select a destination for your snapshot storage and provide the required configuration information. You can choose to set up snapshot storage in the PAM UI or on the command line. Supported destinations are listed below. We recommend using an external service or NFS, depending on what is available to you:

- Internal storage (default)
- Amazon S3
- Azure Blob Storage
- Google Cloud Storage
- Other S3-compatible storage
- Network file system (NFS)
- Host path

Amazon S3 storage

If using the PAM UI, provide the following information:

Field	Description
Bucket	The name of the AWS bucket where snapshots are stored.
Region	The AWS region the bucket is available in.
Path	Optional. The path within the bucket where all snapshots are stored.
Use IAM instance role?	If selected, an IAM instance role is used instead of an access key ID and secret.
Access key ID	Required only if not using an IAM instance role. The AWS IAM access key ID that can read from and write to the bucket.
Access key secret	Required only if not using an IAM instance role. The AWS IAM secret access key that is associated with the access key ID.

If using the command line, run the appropriate command:

Not using an IAM instance role:

```
kubectl kots velero configure-aws-s3 access-key --access-key-id <string>
--bucket <string> --path <string> --region <string> --secret-access-key
<string>
```

Using an IAM instance role:

```
kubectl kots velero configure-aws-s3 instance-role --bucket <string> --
path <string> --region <string>
```

Azure Blob Storage

If using the PAM UI, provide the following information:

Field	Description
Note: Only connections via service principals are currently supported.	
Bucket	The name of the Azure Blob Storage container where snapshots are stored.
Path	Optional. The path within the container where all snapshots are stored.
Subscription ID	Required only for access via service principal or AAD Pod Identity. The subscription ID associated with the target container.
Tenant ID	Required only for access via service principal . The tenant ID associated with the Azure account of the target container.
Client ID	Required only for access via service principal . The client ID of a Service Principle with access to the target container.
Client secret	Required only for access via service principal . The Client Secret of a Service Principle with access to the target container.

4. Click **Update storage settings** to save your storage destination information.

Depending on your chosen storage provider, saving and configuring your storage provider might take several minutes.

5. Optional: To automatically create new snapshots on a schedule, select **Enable automatic scheduled snapshots** on the **Full snapshots (instance)** tab. (If desired, you can also set up a schedule for capturing partial (application-only) snapshots.)

You can schedule a new snapshot creation for every hour, day, or week, or you can create a custom schedule by entering a cron expression.

6. Finally, set the retention schedule for your snapshots by selecting the time period after which old snapshots are automatically deleted. The default retention period is one month.

Note: A snapshot's retention period cannot be changed once the snapshot is created. If you update the retention schedule, the new retention period applies only to snapshots created after the update is made.

7. Click **Update schedule** to save your changes.

Snapshots are automatically created according to your specified schedule and saved to the storage location you selected. You can also create an unscheduled snapshot at any time by clicking **Start a snapshot** on the **Dashboard** or on the **Snapshots** page.

Roll back changes using a snapshot

When necessary, you can use a snapshot to roll back to a previous version of your Puppet Application Manager set-up without changing the underlying cluster infrastructure.

To roll back changes:

1. In console menu of the Puppet Application Manager UI, click **Snapshots > Full Snapshots (Instance)**.
2. Select the snapshot you wish to roll back to from the list of available snapshots and click **Restore from this**

backup 

3. Follow the instructions to complete either a partial restore or a full restore.

A full restore is useful if you need to stay on an earlier version of an application and want to disable automatic version updates. Otherwise, a partial restore is the quicker option.

Migrating PAM data to a new system

By using a snapshot, you can migrate your data to a new Puppet Application Manager (PAM) instance.

Data migration prerequisites

In order to perform a data migration, your system must be configured as follows:

- On the original system, Puppet Application Manager (PAM) must be configured to support **Full Snapshots (Instance)**. For instructions on configuring snapshots, see [Backing up PAM using snapshots](#) on page 112.
- Velero must be configured to use an external snapshot destination accessible to both the old and new clusters, such as S3 or NFS.
- Both the old and new clusters must have the same connection status (online or offline). Migrating from offline to online clusters or vice versa is not supported.
- For offline installs, both the old and new clusters must use the same version of PAM.
- Upgrade to the latest version of PAM on both the old and new clusters before you begin.

Migrating data between two systems with the same architecture

To perform data migration between two systems using the same architecture (from standalone to standalone, or from HA to HA), you must create a new cluster to migrate to, then follow the process outlined below to recover your instance from a snapshot.

Before you begin

Review the requirements in [Data migration prerequisites](#) on page 114.

Important: If you are migrating from a legacy architecture, go to our Support Knowledge Base instructions for migrating to a supported architecture for your Puppet application:

- [Migrate to a supported PAM architecture for Continuous Delivery for PE](#)
- [Migrate to a supported PAM architecture for Comply](#)

1. On the original system, find the version of kURL your deployment is using by running the following command. Save the version for use in step 3.

```
kubectl get configmap -n kurl kurl-current-config -o  
  jsonpath="{.data.kurl-version}" && echo
```

2. Get the installer spec section by running the command appropriate to your PAM installation type:

Tip: See [How to determine your version of Puppet Application Manager](#) if you're not sure which installation type you're running.

- HA installation: `kubectl get installers puppet-application-manager -o yaml`
- Standalone installation: `kubectl get installers puppet-application-manager-standalone -o yaml`
- Legacy installation: `kubectl get installers puppet-application-manager-legacy -o yaml`

The command's output looks similar to the following. The spec section is shown in bold in the example below. Save your spec section for use in step 3.

```
# kubectl get installers puppet-application-manager-standalone -o yaml
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

    {"apiVersion":"cluster.kurl.sh/v1beta1","kind":"Installer","metadata":{"annotations":{"creationTimestamp":null,"name":"puppet-application-manager-standalone","namespace":"default"},"spec":{"containerd":{"version":"1.4.12"},"contour":{"version":"1.18.0"},"ekco":{"version":"0.16.0"},"kotsadm":{"applicationSlug":"puppet-application-manager","version":"1.64.0"},"kubernetes":{"version":"1.21.8"},"metricsServer":{"version":"0.4.1"},"minio":{"version":"2020-01-25T02-50-51Z"},"openebs":{"isLocalPVEnabled":true,"localPVStorageClassName":"default","version":"2.6.0"},"prometheus":{"version":"0.49.0-17.1.1"},"registry":{"version":"2.7.1"},"velero":{"version":"1.6.2"},"weave":{"podCidrRange":"/22","version":"2.8.1"}},"status":{}}
    creationTimestamp: "2021-06-04T00:05:08Z"
    generation: 4
    labels:
      velero.io/exclude-from-backup: "true"
    name: puppet-application-manager-standalone
    namespace: default
    resourceVersion: "102061068"
    uid: 4e7f1196-5fab-4072-9399-15d18dcc5137
spec:
  containerd:
    version: 1.4.12
  contour:
    version: 1.18.0
  ekco:
    version: 0.16.0
  kotsadm:
    applicationSlug: puppet-application-manager
    version: 1.64.0
  kubernetes:
    version: 1.21.8
  metricsServer:
    version: 0.4.1
  minio:
    version: 2020-01-25T02-50-51Z
  openebs:
    isLocalPVEnabled: true
    localPVStorageClassName: default
    version: 2.6.0
  prometheus:
    version: 0.49.0-17.1.1
  registry:
    version: 2.7.1
  velero:
    version: 1.6.2 © 2024 Puppet, Inc., a Perforce company
  weave:
```

3. On a new machine, create a file named `installer.yaml` with the following contents, replacing `<SPEC>` and `<KURL VERSION>` with the information you gathered in the previous steps.

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  <SPEC>
  kurl:
    installerVersion: "<KURL VERSION>"
```

Important: If you are running PAM version 1.68.0 or newer, the kURL installer version might be included in the `spec` section. If this is the case, omit the `kurl:` section from the bottom of the `installer.yaml` file. There must be only one `kurl:` entry in the file.

Tip: Spacing is critical in YAML files. Use a YAML file linter to confirm that the format of your file is correct.

Here is an example of the contents of the `installer.yaml` file:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
spec:
  containerd:
    version: 1.4.12
  contour:
    version: 1.18.0
  ekco:
    version: 0.16.0
  kotsadm:
    applicationSlug: puppet-application-manager
    version: 1.64.0
  kubernetes:
    version: 1.21.8
  metricsServer:
    version: 0.4.1
  minio:
    version: 2020-01-25T02-50-51Z
  openebs:
    isLocalPVEnabled: true
    localPVStorageClassName: default
    version: 2.6.0
  prometheus:
    version: 0.49.0-17.1.1
  registry:
    version: 2.7.1
  velero:
    version: 1.6.2
  weave:
    podCidrRange: /22
    version: 2.8.1
  kurl:
    installerVersion: "v2022.03.11-0"
```

4. Build an installer using the `installer.yaml` file. Run the following command:

```
curl -s -X POST -H "Content-Type: text/yaml" --data-binary
"@installer.yaml" https://kurl.sh/installer |grep -o "[^/]*$"
```

The output is a hash. Carefully save the hash for use in step 5.

5. Install a new cluster. To do so, you can either:

- a. Point your browser to `https://kurl.sh/<HASH>` (replacing `<HASH>` with the hash you generated in step 4) to see customized installation scripts and information.
- b. Follow the appropriate PAM documentation.
 - **For online installations:** Follow the steps in [PAM HA online installation](#) on page 83 or [PAM standalone online installation](#) on page 90, replacing the installation script with the following:

```
curl https://kurl.sh/<HASH> | sudo bash
```

- **For offline installations:** Follow the steps in [PAM HA offline installation](#) on page 87 or [PAM standalone offline installation](#) on page 93, replacing the installation script with the following:

```
curl -LO https://k8s.kurl.sh/bundle/<HASH>.tar.gz
```

When setting up a new *offline* cluster as part of disaster recovery, add `kurl-registry-ip=<IP>` to the install options, replacing `<IP>` with the value you recorded when setting up snapshots.

Note: If you do not include the `kurl-registry-ip=<IP>` flag, the registry service will be assigned a new IP address that does not match the IP on the machine where the snapshot was created. You must align the registry service IP address on the new offline cluster to ensure that the restored configuration can pull images from the correct location.

Important: Do not install any Puppet applications after the PAM installation is complete. You'll recover your Puppet applications later in the process.

6. To recover using a snapshot saved to a **host path**, ensure user/group 1001 has full access on all nodes by running:

```
chown -R 1001:1001 /<PATH/TO/HOSTPATH>
```

7. Configure the new cluster to connect to your snapshot storage location. Run the following to see the arguments needed to complete this task:

```
kubectl kots -n default velero configure-{hostpath,nfs,aws-s3,other-s3,gcp} --help
```

8. Run `kubectl kots get backup` and wait for the list of snapshots to become available. This might take several minutes.
9. Start the restoration process by running `kubectl kots restore --from-backup <BACKUP NAME>`. The restoration process takes several minutes to complete. When the PAM UI is available, use it to monitor the status of the application.

Note: When restoring, wait for all restores to complete before making any changes. The following command waits for pods to finish restoring data from backup. Other pods may not be ready until updated configuration is deployed in the next step:

```
kubectl get pod -o json | jq -r '.items[] | select(.metadata.annotations."backup.velero.io/backup-volumes" | .metadata.name' | xargs kubectl wait --for=condition=Ready pod --timeout=20m
```

This command requires the [jq](#) CLI tool to be installed. It is available in most Linux OS repositories.

10. After the restoration process completes, save your config and deploy:

- a) From the PAM UI, click **Config**.
- b) (Optional) If the new cluster's hostname is different from the old one, update the **Hostname**.
- c) Click **Save Config**.
- d) Deploy the application. You must save your config and deploy even if you haven't made any changes.

Note: If you have installed Continuous Delivery for PE and changed the hostname, you need to update the webhooks that connect Continuous Delivery for PE with your source control provider. For information on how to do this, see [Update webhooks](#).

Migrating data between two systems with different architectures

To perform data migration between two systems using different PAM architectures (from standalone to HA, or from HA to standalone), you must create a new cluster to recover to, then follow the process outlined below to recover your instance from a snapshot.

Before you begin

Review the requirements in [Data migration prerequisites](#) on page 114.

Important: If you are migrating from a legacy architecture, go to our Support Knowledge Base instructions for migrating to a supported architecture for your Puppet application:

- [Migrate to a supported PAM architecture for Continuous Delivery for PE](#)
- [Migrate to a supported PAM architecture for Comply](#)

1. On the original system, find the version of kURL your deployment is using by running the following command. Save the version for use in step 2.

```
kubectl get configmap -n kurl kurl-current-config -o
jsonpath="{.data.kurl-version}" && echo
```

2. Set up a new cluster to house the recovered instance, following the system requirements for your applications.

Important: Do not install any Puppet applications after the PAM installation is complete. You'll recover your Puppet applications later in the process.

- Install PAM using the version of kURL you retrieved earlier:

- For online installs:

```
curl -sSL https://k8s.kurl.sh/version/<VERSION STRING>/puppet-
application-manager | sudo bash <-s options>
```

- For offline installs:

```
curl -O https://k8s.kurl.sh/bundle/version/<VERSION STRING>/puppet-
application-manager.tar.gz
```

- When setting up a new *offline* cluster as part of disaster recovery, add `kurl-registry-ip=<IP>` to the install options, replacing `<IP>` with the value you recorded when setting up snapshots.

Note: If you do not include the `kurl-registry-ip=<IP>` flag, the registry service will be assigned a new IP address that does not match the IP on the machine where the snapshot was created. You must align the registry service IP address on the new offline cluster to ensure that the restored configuration can pull images from the correct location.

- To recover using a snapshot saved to a **host path**, ensure user/group 1001 has full access on all nodes by running:

```
chown -R 1001:1001 /<PATH/TO/HOSTPATH>
```

- Configure the new cluster to connect to your snapshot storage location. Run the following to see the arguments needed to complete this task:

```
kubectl kots -n default velero configure-{hostpath,nfs,aws-s3,other-s3,gcp} --help
```

- Run `kubectl kots get backup` and wait for the list of snapshots to become available. This might take several minutes.
- Start the restoration process by running `kubectl kots restore --from-backup <BACKUP NAME>`. The restoration process takes several minutes to complete. When the PAM UI is available, use it to monitor the status of the application.

Note: When restoring, wait for all restores to complete before making any changes. The following command waits for pods to finish restoring data from backup. Other pods may not be ready until updated configuration is deployed in the next step:

```
kubectl get pod -o json | jq -r '.items[] |
  select(.metadata.annotations."backup.velero.io/backup-volumes")
  | .metadata.name' | xargs kubectl wait --for=condition=Ready pod --
  timeout=20m
```

This command requires the [jq](#) CLI tool to be installed. It is available in most Linux OS repositories.

- After the restoration process completes, save your config and deploy:
 - From the PAM UI, click **Config**.
 - (Optional) If the new cluster's hostname is different from the old one, update the **Hostname**.
 - Click **Save Config**.
 - Deploy the application. You must save your config and deploy even if you haven't made any changes.

Note: If you have installed Continuous Delivery for PE and changed the hostname, you need to update the webhooks that connect Continuous Delivery for PE with your source control provider. For information on how to do this, see [Update webhooks](#).

Disaster recovery with PAM

It is important to prepare your system and regularly capture full snapshots. This backs up your data and makes it easier to restore your system if disaster recovery is needed.

Prepare your system to support future disaster recovery

To make sure your system is equipped to help you recover from a potential system failure, you must:

- Configure Puppet Application Manager (PAM) to support **Full Snapshots (Instance)**. For instructions on configuring snapshots, see [Backing up PAM using snapshots](#) on page 112.
- Configure Velero to use an external snapshot destination that is accessible to both your current cluster and future new clusters, such as S3 or NFS.

- Disaster recovery requires that the store backend used for backups is accessible from the new cluster. When setting up snapshots in an *offline* cluster, use the following command to record the registry service IP address:

```
kubectl -n kurl get svc registry -o jsonpath='{.spec.clusterIP}'
```

Make a record of the value returned by this command, because you'll need it to create a new cluster to restore to as part of disaster recovery.

- Run the latest version of PAM. Disaster recovery is only available on systems running PAM version 1.44.1 or newer.

Disaster recovery process

To perform a disaster recovery, you must create a new cluster to recover to and then recover your instance from a snapshot.

1. Find the version of kURL your original deployment was using.

If you have access to the original cluster, you can use this command:

```
kubectl get configmap -n kurl kurl-current-config -o jsonpath="{.data.kurl-version}" && echo
```

If you aren't able to run the command, you remember your PAM version, and you were on version 1.68.0 or later, you can look up the kURL version in the [Component versions in PAM releases](#) on page 79 table.

If you don't remember your PAM version or you were on a version earlier than 1.68.0, you need to contact your technical account manager or Support for assistance.

2. If you have access to the original cluster, follow the steps for [Migrating data between two systems with the same architecture](#) on page 114.

If your original cluster is completely offline and inaccessible, you'll need to contact your technical account manager or Support for assistance.

Restriction:

Your old and new clusters must have the same connection status (online or offline). Disaster recovery from an offline to an online cluster (or vice versa) is not supported.

Additionally, for offline installs, both the old and new clusters must use the same PAM version.

Troubleshooting PAM

Use this guide to troubleshoot issues with your Puppet Application Manager installation.

How to look up your Puppet Application Manager architecture

If you're running PAM on a Puppet-supported cluster, you can use the following command to determine your PAM architecture version:

```
kubectl get installer --sort-by=.metadata.creationTimestamp -o jsonpath='{.items[-1:].metadata.name}' ; echo
```

Depending on which architecture you used when installing, the command returns one of these values:

- **HA architecture:** puppet-application-manager
- **Standalone architecture:** puppet-application-manager-standalone
- **Legacy architecture:** Any other value, for example, puppet-application-manager-legacy, cd4pe, or comply

Resolve IP address range conflicts

When installing Puppet Application Manager, IP address ranges `10.96.0.0/22` and `10.32.0.0/22` must not be used by other nodes on the local network.

Note: The minimum size for CIDR blocks used by Puppet Application Manager are:

- **Standalone** - /24 for pod and service CIDRs
- **HA** - /23 for pod and service CIDRs
- Default of /22 is recommended to support future expansion

To resolve IP address range conflicts, create a `patch.yaml` file and add the `installer-spec-file=patch.yaml` argument when running the installation script (see below):

1. If you use IP addresses internally that overlap `10.32.0.0/22`, add the following to your `patch.yaml` file (`10.40.0.0/23` used here as an example range):

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  flannel:
    podCIDR: 10.40.0.0/23
    podCIDRRange: "/22"
```

2. If you use IP addresses internally that overlap `10.96.0.0/22`, add the following to your `patch.yaml` file (`10.100.0.0/23` used here as an example range):

```
spec:
  ...
  kubernetes:
    serviceCIDR: 10.100.0.0/23
    serviceCIDRRange: "/23"
```



CAUTION: The `podCIDR` and `serviceCIDR` ranges must not overlap.

3. Once your `patch.yaml` file is set up, add the `installer-spec-file=patch.yaml` argument when you run the installation script:

```
cat install.sh | sudo bash -s airgap installer-spec-file=patch.yaml
```

Remember: Add the `installer-spec-file=patch.yaml` argument any time you re-run the installation script, such as when reinstalling to upgrade to a new version.

Related information

[Using sudo behind a proxy server](#) on page 125

Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

Reset the PAM password

As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

1. To reset the Puppet Application Manager password, run the following command as the root user:

```
kubectl kots reset-password default
```

The system prompts you to enter a new password of your choosing.

2. If the command fails with an unknown command "kots" for "kubectl" error, it's because /usr/local/bin is not in the path. To address this error, either update the path to include /usr/local/bin, or run this command:

```
/usr/local/bin/kubectl-kots reset-password default
```

Update the PAM TLS certificate

A self-signed TLS certificate secures the connection between your browser and Puppet Application Manager (PAM). Once the initial Puppet Application Manager setup process is complete, you can upload new certificates by enabling changes to the installation's Kubernetes secrets.

Use this process if you chose not to add a TLS certificate when installing Puppet Application Manager, or if you need to update your existing TLS certificate.

1. Enable changes to your installation's kotsadm-tls Kubernetes secret by running:

```
kubectl -n default annotate secret kotsadm-tls acceptAnonymousUploads=1
```

2. Restart the kurl-proxy pod to deploy the change by running:

```
kubectl delete pods $(kubectl get pods -A | grep kurl-proxy | awk '{print $2}')
```

3. Once the kurl-rpxy pod restarts and is back up and running, navigate to `https://<HOSTNAME>:8800/tls` and upload your new TLS certificate.

Reduce recovery time when a node fails

If a node running a non-replicated service like PostgreSQL fails, expect some service downtime.

How much downtime depends on the following factors:

- Timeout for communication between Kubernetes services (at least one minute to mark the node as unreachable).
- Timeout for the ekco service to determine that pods need to be rescheduled. The default is five minutes after node is marked unreachable.
- Time to restart services (at least two minutes, possibly up to five minutes, if there are complex dependencies).

The ekco service can be configured to reschedule pods more quickly by configuring the installation with a patch.yaml similar to the following:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  ekco:
    nodeUnreachableToleration: 1m
```

Apply the patch during an install or upgrade by including `installer-spec-file=patch.yaml` as an install option.

Important: This patch needs to be included during all future upgrades to avoid resetting the option.

PAM components

Puppet Application Manager (PAM) uses a range of mandatory and optional components.

Support services

- Database: PostgreSQL (single instance) - <https://www.postgresql.org/>
- Object storage: previously MinIO - <https://min.io>, now Ceph - <https://ceph.io>
- tlser for basic TLS cert management - <https://github.com/puppetlabs/tlser>
- kurl_proxy for HTTPS proxying outside the Ingress (ports besides 80/443): https://github.com/replicatedhq/kots/tree/v1.36.1/kurl_proxy

Kubernetes components

- Networking (CNI): Flannel - <https://github.com/flannel-io/flannel>
- Storage (CSI): Rook - <https://rook.io>, Ceph - <https://ceph.io>
- Ingress: Project Contour - <https://projectcontour.io>
- Kubernetes Cluster: kURL - <https://kurl.sh>
- Embedded kURL Cluster Operator: ekco - <https://github.com/replicatedhq/ekco>
- Admin Console: KOTS - <https://kots.io>
- Snapshots: Velero - <https://velero.io>, Restic - <https://restic.net>
- Monitoring: Prometheus - <https://prometheus.io>
- Registry: Docker Registry - <https://docs.docker.com/registry/>

Optional components

Prometheus (+Grafana) and Velero (+Restic) are optional components:

- Prometheus+Grafana uses 112m/node + 600m CPU, 200MiB/node + 1750MiB RAM
- Velero+Restic uses 500m/node + 500m CPU, 512MiB/node + 128MiB RAM

If you do not need these optional components, they can be omitted from the initial install and further upgrades with a patch similar to the following:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  prometheus:
    version: ''
  velero:
    version: ''
```

Important: This patch needs to be included during upgrades to avoid adding the components later.

If you want to remove optional components that are already installed, use the following command:

```
kubectl delete ns/monitoring ns/velero
```

Generate a support bundle

When seeking support, you might be asked to generate and provide a support bundle. This bundle collects a large amount of logs, system information and application diagnostics.

To create a support bundle:

1. In Puppet Application Manager UI, click **Troubleshoot** > **Generate a support bundle**.

2. Select a method for generating the support bundle:
 - **Generate the bundle automatically.** Click **Analyze <APPLICATION NAME>** (<APPLICATION NAME> is replaced in the UI by the name of the Puppet application you have installed), and Puppet Application Manager generates the bundle for you and uploads it to the **Troubleshoot** page.
 - **Generate the bundle manually.** Click the prompt to generate a custom command for your installation, then run the command on your cluster. Follow the prompts to upload the bundle to Puppet Application Manager.
3. Review the collected data before forwarding it to Puppet, as it may contain sensitive information that you wish to redact.
4. Return to the **Troubleshoot** page, download the newly created support bundle, and send it to your Puppet Support contact.

Create a support bundle from the command line

If installation of the Puppet Application Manager, or upload of an app, on an embedded kURL cluster fails, it may not be possible to access the UI to generate a support bundle.

You can generate a support bundle by using the default kots.io spec. To do this, run the following command:

```
kubectl support-bundle https://kots.io
```

On an offline server, you can copy the default kots.io spec by using the following command:

```
curl -o spec.yaml https://kots.io -H 'User-agent:Replicated_Troubleshoot/v1beta1'
```

The spec can then be uploaded to the server. Use the local spec by running:

```
kubectl support-bundle /path/to/spec.yaml
```

If the Puppet Application Manager UI is working and the app is installed, you can use:

```
kubectl support-bundle http://<server-address>:8800/api/v1/troubleshoot/<app-slug>
```

If the app is not installed but the Puppet Application Manager UI is running:

```
kubectl support-bundle http://<server-address>:8800/api/v1/troubleshoot
```

If you do not already have the support-bundle kubectl plugin installed, install it by using the command below:

```
curl https://krew.sh/support-bundle | bash
```

Or by installing [krew2](#) and running:

```
kubectl krew install support-bundle
```

Using sudo behind a proxy server

Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

kURL can only be upgraded two minor versions at a time

Because kURL does not support upgrading more than two Kubernetes minor release versions at once, if you're upgrading from an older version of PAM, you might need to follow a specific upgrade path to avoid failures. For example, PAM version 1.80.0 uses Kubernetes version 1.21.x, so you can upgrade up to PAM 1.91.3 (Kubernetes

version 1.23.x), but not to PAM 1.94.0 (Kubernetes version 1.24.x). To determine the specific upgrade path for your installation, please check the [table of Kubernetes versions](#) for each version of PAM.

Attempting to upgrade too far at once returns the following error message: The currently installed kubernetes version is <CURRENT_VERSION>. The requested version to upgrade to is <INVALID_TARGET_VERSION>. Kurl can only be upgraded two minor versions at time. Please install <VALID_TARGET_VERSION> first.

Install

The initial installation and set up process for Continuous Delivery for Puppet Enterprise (PE) includes understanding the architecture, selecting an appropriate TLS configuration, configuring and deploying Continuous Delivery for PE, ensuring your PE version and browser are supported, and configuring your analytics data collection preferences.

Note: [Migrate 3.x data to 4.x](#) on page 136 explains how to upgrade to the Continuous Delivery for PE 4.x series from a version in the 3.x series.

- [Continuous Delivery for PE architecture](#) on page 126

Continuous Delivery for Puppet Enterprise (PE) communicates with your PE installation, your source control system, the servers you've designated as job hardware, and the browser you use to connect to the web UI and Puppet Application Manager (PAM).

- [Supported PE versions and browsers](#) on page 129

The Continuous Delivery for Puppet Enterprise (PE) 4.x series is designed for use with currently-supported versions of PE and current versions of major web browsers.

- [Deploy Continuous Delivery for PE](#) on page 130

After installing Puppet Application Manager (PAM), specify your initial configuration setting and deploy Continuous Delivery for Puppet Enterprise (PE) for the first time.

- [Deploy Continuous Delivery for PE offline](#) on page 131

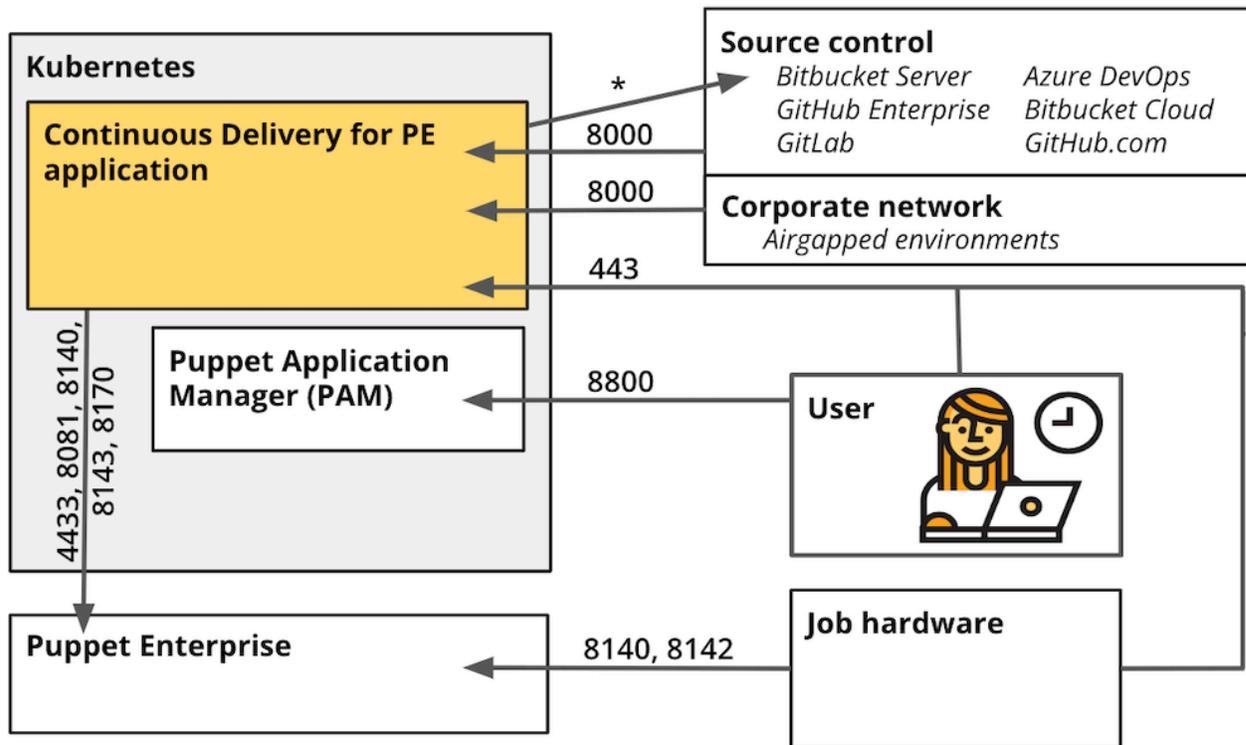
Use these instructions to install Continuous Delivery for Puppet Enterprise (PE) in an airgapped or offline environment where the Continuous Delivery for PE host server does not have direct access to the internet.

Continuous Delivery for PE architecture

Continuous Delivery for Puppet Enterprise (PE) communicates with your PE installation, your source control system, the servers you've designated as job hardware, and the browser you use to connect to the web UI and Puppet Application Manager (PAM).

This diagram shows the architecture and port requirements for a Continuous Delivery for PE 4.x installation.

Important: Continuous Delivery for PE uses TCP (Transmission Control Protocol) connections.



Port	Use
* (variable)	On this port, Continuous Delivery for PE makes API requests to, and clones from, source control over HTTPS or SSH. The specific port number depends on your source control integration .
443	On this port, Continuous Delivery for PE job hardware servers communicate with the Continuous Delivery for PE application, and users access the Continuous Delivery for PE application's web UI over HTTPS.
4433	The Continuous Delivery for PE application uses this PE port to communicate with the node classifier and the PE console (for authentication).
8000	This is the default port where source control provider webhooks send traffic to Continuous Delivery for PE. You can change this port in PAM under Optional configuration .
8081	The Continuous Delivery for PE application uses this PE port to send queries to PuppetDB.
8140	The Continuous Delivery for PE application and Continuous Delivery for PE job hardware servers use this PE port to communicate with Puppet Server.
8142	Continuous Delivery for PE job hardware servers and PE communicate through Puppet Agent on this port.
8143	The Continuous Delivery for PE application uses this PE port to communicate with Puppet Orchestrator.
8170	The Continuous Delivery for PE application uses this PE port to communicate with Code Manager.

Port	Use
8800	PAM's web UI accepts HTTPS traffic from users on this port.

You can configure ports 4433, 8081, 8140, 8143, and 8170 in the PE integration settings.

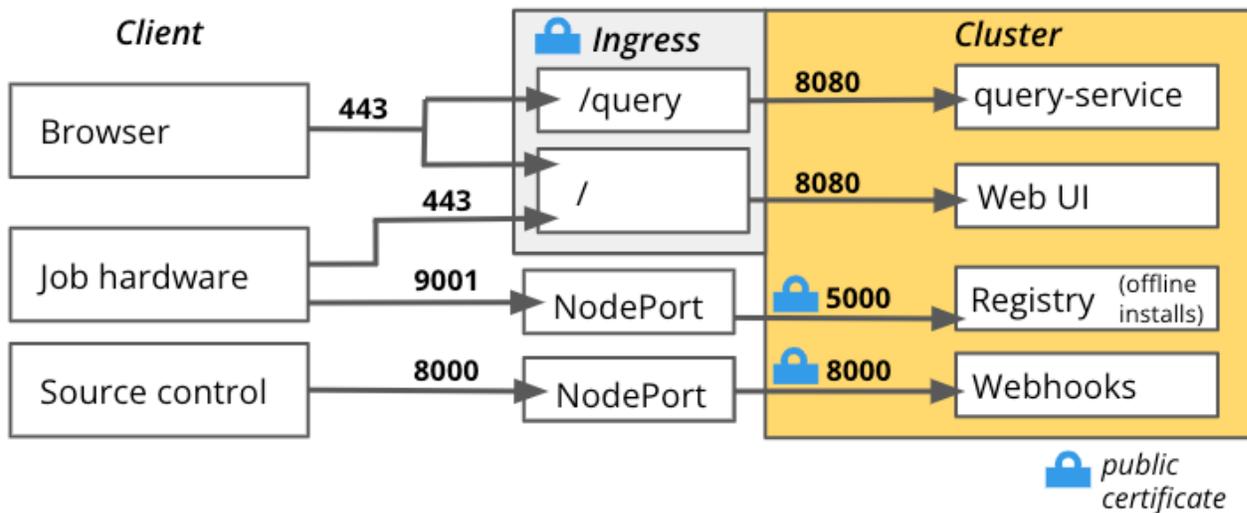
For additional information about each Continuous Delivery for PE port's source and destination, refer to the [PAM system requirements](#) on page 66. For more information about PE ports, refer to PE documentation, such as the [PE Firewall configuration diagrams](#).

TLS configuration

You can choose from several TLS configuration options when installing Continuous Delivery for Puppet Enterprise (PE). Select the installation architecture that best meets your security needs and limitations.

Basic installation (default configuration)

This installation architecture works for single-node clusters and multi-node clusters that utilize automatic load balancing or are set up for Ingress load balancing, such as with Google Kubernetes Engine (GKE). The webhook callback listens on port 8000. This architecture uses public certificates for machine-to-machine communication, and it's common to use TLS configured at the Ingress, whether in manual-entry, self-signed, or certificate manager form.

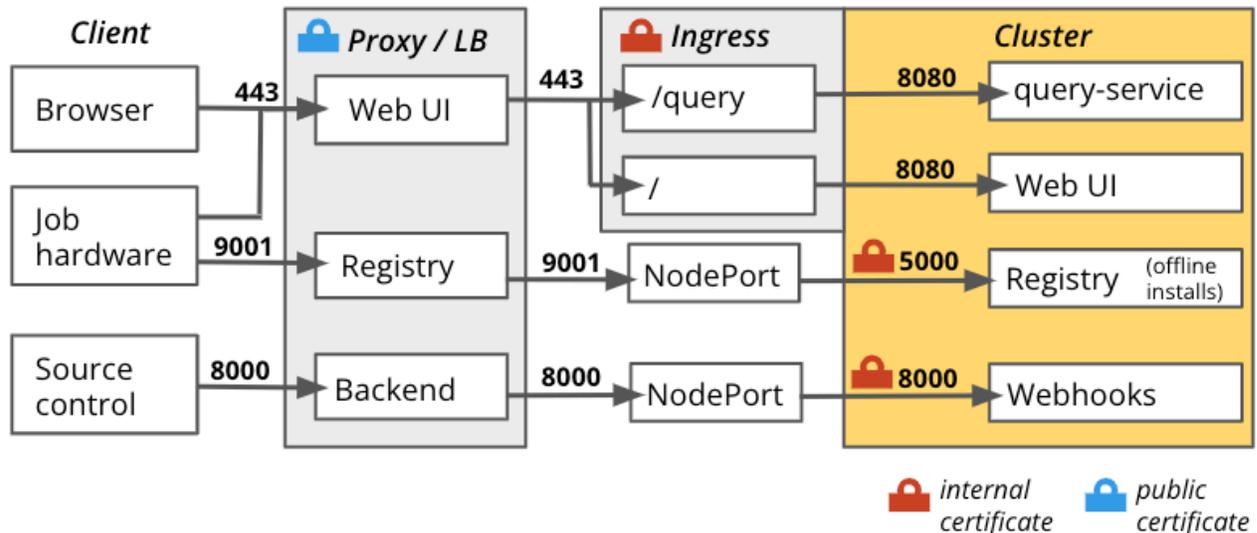


Port	Use
443	Continuous Delivery for PE job hardware servers communicate with the Continuous Delivery for PE application over this port, and users access the Continuous Delivery for PE web UI over HTTPS on this port.
5000	In offline installations, NodePort communicates with the registry on this port.
8000	This is the default port where source control provider webhooks send traffic to Continuous Delivery for PE. You can change this port in Puppet Application Manager (PAM) under Optional configuration .
8080	Ingress forwards traffic to the Continuous Delivery for PE web UI and communicates with <code>query-service</code> on this port.

Port	Use
9001	Continuous Delivery for PE job hardware servers communicate with NodePort on this port.

Installation with a proxy or load balancer using enhanced TLS

This installation architecture uses public certificates for external proxy TLS termination and internal certificates for machine-to-machine communication. The Ingress is used for routing rules to the various services within Continuous Delivery for PE. You must configure the backend and web UI endpoints separately from the Ingress hostnames.



Port	Use
443	On this port, Continuous Delivery for PE job hardware servers communicate with the Continuous Delivery for PE application, and users access the Continuous Delivery for PE web UI over HTTPS.
5000	In offline installations, NodePort communicates with the registry on this port.
8000	This is the default port where source control provider webhooks send traffic to Continuous Delivery for PE. You can change this port in Puppet Application Manager (PAM) under Optional configuration .
8080	Ingress forwards traffic to the Continuous Delivery for PE web UI and communicates with <code>query-service</code> on this port.
9001	Continuous Delivery for PE job hardware servers communicate with the proxy/load balancer registry and NodePort on this port.

Supported PE versions and browsers

The Continuous Delivery for Puppet Enterprise (PE) 4.x series is designed for use with currently-supported versions of PE and current versions of major web browsers.

Supported PE versions

You can use the latest version of Continuous Delivery for Puppet Enterprise (PE) with all currently-supported versions of PE.

Currently-supported PE versions are described on the [PE support lifecycle](#) page, and you can find the recommended [PE upgrade paths in the PE documentation](#).

Important: You must upgrade to Continuous Delivery for PE version 4.8.2 before installing PE version 2021.3, 2019.8.8, or newer versions in their respective release series. Version 4.8.2 resolves a PuppetDB issue that prevented generating new fact charts on the **Nodes** page.

Support might vary for older versions of Continuous Delivery for PE, depending on which PE versions existed and were supported at the time of release.

Supported browsers

You can use the Continuous Delivery for PE web UI with these browsers.

Browser	Supported versions
Google Chrome	Current version as of release
Mozilla Firefox	Current version as of release
Microsoft Edge	Current version as of release
Apple Safari	Current version as of release

Deploy Continuous Delivery for PE

After installing Puppet Application Manager (PAM), specify your initial configuration setting and deploy Continuous Delivery for Puppet Enterprise (PE) for the first time.

Before you begin

[Install PAM](#) on page 80.

These steps are for hosts with internet access. For airgapped installations, go to [Deploy Continuous Delivery for PE offline](#) on page 131.

- In PAM, upload your Continuous Delivery for PE license and follow the prompts to set up SSL certificates and make sure your infrastructure meets Continuous Delivery for PE system requirements.
- To configure your Continuous Delivery for PE installation, click **Config**, and:
 - Enter a hostname for the Continuous Delivery for PE installation.
 - Set a password for the Continuous Delivery for PE root user account.
 - Select where to host the webhook (for source control integration) and local container registry (in Puppet-supported offline installs) endpoints.
 - Use a Nodeport:** All Kubernetes hosts listen on this NodePort and forward traffic to the appropriate endpoint.
 - Use an Ingress with a hostname:** The endpoint listens on ports 80 (HTTP) and 443 (HTTPS) for requests using the specified hostname.
 - Configure any other settings on this page relevant to your installation, such as external databases, customized endpoints, a load balancer, or [TLS certificates](#).
 - When you've finished editing the configuration, click **Save config**. Saving new configuration settings initiates creation of a new Continuous Delivery for PE version.
- Click **Go to new version**, which redirects you to the **Version history** tab, and locate the new version in the **All versions** section of the page.

4. Monitor the new version's preflight checks. The **Running Checks** indicator is visible while Continuous Delivery for PE checks your system to make sure your cluster meets minimum system requirements. When the preflight checks are complete:

- If the status is **Checks Failed**, click **View preflights**, correct the issues, and click **Re-run**. Repeat this step as needed.

Important: Do not move on until all preflight checks pass.

- If the status is **Ready to Deploy**, move on to the next step.

5. Once the version is ready to deploy, click **Deploy** and monitor the application's status on the **Application** tab.

During deployment, the application is in **Missing** status for several minutes. You can run `kubectl get pods --watch` to monitor the deployment's progress.

When deployment is complete, the application's status changes to **Ready**.

6. Navigate to `https://<HOSTNAME>` (using the hostname you entered on the **Config** screen) and sign into Continuous Delivery for PE.

If you're upgrading to the 4.x series from a version in the 3.x series, see [Migrate 3.x data to 4.x](#) on page 136.

If you're new to Continuous Delivery for PE, see the [Getting started guide](#) on page 13 to learn about the software's core features and workflows.

Deploy Continuous Delivery for PE offline

Use these instructions to install Continuous Delivery for Puppet Enterprise (PE) in an airgapped or offline environment where the Continuous Delivery for PE host server does not have direct access to the internet.

Before you begin

[Install Puppet Application Manager \(PAM\)](#).

1. In PAM, upload your Continuous Delivery for PE license and follow the prompts to set up SSL certificates and make sure your infrastructure meets Continuous Delivery for PE system requirements.

2. When prompted, upload an `.airgap` bundle for the most recent version of Continuous Delivery for PE. These are the available bundles:

Version	Release date	Airgap bundle
4.35.0	29 April 2025	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.35.0.airgap
4.34.0	30 January 2025	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.34.0.airgap
4.33.0	30 October 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.33.0.airgap
4.32.0	29 August 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.32.0.airgap
4.31.0	10 July 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.31.0.airgap
4.30.2	14 June 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.30.2.airgap
4.30.1	22 May 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.30.1.airgap
4.30.0	7 May 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.30.0.airgap
4.29.2	21 March 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.29.2.airgap
4.29.1	21 February 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.29.1.airgap
4.29.0	8 February 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.29.0.airgap
4.28.0	30 November 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.28.0.airgap
4.27.1	11 October 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.27.1.airgap
4.27.0	4 October 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.27.0.airgap
4.26.2	7 September 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.26.2.airgap
4.26.1	23 August 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.26.1.airgap
4.26.0	22 August 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.26.0.airgap
4.25.1	24 July 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.25.1.airgap
4.25.0	11 July 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.25.0.airgap
4.24.1	28 June 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.24.1.airgap
4.24.0	31 May 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.24.0.airgap
4.23.1	2 May 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.23.1.airgap
4.23.0	18 April 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.23.0.airgap

3. To configure your Continuous Delivery for PE installation, click **Config**, and:
 - a) Enter a hostname for the Continuous Delivery for PE installation.
 - b) Set a password for the Continuous Delivery for PE root user account.
 - c) Select where to host the webhook (for source control integration) and local container registry (in Puppet-supported offline installs) endpoints.
 - **Use a Nodeport:** All Kubernetes hosts listen on this NodePort and forward traffic to the appropriate endpoint.
 - **Use an Ingress with a hostname:** The endpoint listens on ports 80 (HTTP) and 443 (HTTPS) for requests using the specified hostname.
 - d) Configure any other settings on the page relevant to your installation, such as external databases, customized endpoints, a load balancer, or TLS certificates.
 - e) When you've finished editing the configuration, click **Save config**. Saving new configuration settings initiates creation of a new Continuous Delivery for PE version.
4. Click **Go to new version**, which redirects you to the **Version history** tab, and locate the new version in the **All versions** section of the page.
5. Monitor the new version's preflight checks. The **Running Checks** indicator is visible while Continuous Delivery for PE checks your system to make sure your cluster meets minimum system requirements. When the preflight checks are complete:
 - If the status is **Checks Failed**, click **View preflights**, correct the issues, and click **Re-run**. Repeat this step as needed.

Important: Do not move on until all preflight checks pass.

 - If the status is **Ready to Deploy**, move on to the next step.
6. Once the version is ready to deploy, click **Deploy** and monitor the application's status on the **Application** tab. During deployment, the application is in **Missing** status for several minutes. You can run `kubectl get pods --watch` to monitor the deployment's progress.

When deployment is complete, the application's status changes to **Ready**.
7. Navigate to `https://<HOSTNAME>` (using the hostname you entered on the **Config** screen) and sign into Continuous Delivery for PE.

If you're upgrading to the 4.x series from a version in the 3.x series, see [Migrate 3.x data to 4.x](#) on page 136.

If you're new to Continuous Delivery for PE, see the [Getting started guide](#) on page 13 to learn about the software's core features and workflows.

Upgrade

We regularly release new versions of Continuous Delivery for Puppet Enterprise (PE). Upgrading to the current version ensures you have the latest features, fixes, and improvements.

- [Upgrade paths](#) on page 134

These are valid upgrade paths for Continuous Delivery for Puppet Enterprise (PE).

- [Upgrade Continuous Delivery for PE](#) on page 134

Check for, download, and deploy updates from the **Version history** tab in Puppet Application Manager (PAM). These instructions apply to Continuous Delivery for Puppet Enterprise (PE) installations in online environments.

- [Upgrade Continuous Delivery for PE offline](#) on page 134

If your host doesn't have direct internet access, you must download and install an airgap bundle to upgrade to the latest version of Continuous Delivery for Puppet Enterprise (PE).

- [Upgrade an automated installation](#) on page 136

If you used the Puppet Application Manager automated installation feature to install Continuous Delivery for Puppet Enterprise (PE), you can run a script to upgrade to the latest version.

- [Migrate 3.x data to 4.x](#) on page 136

The Continuous Delivery for PE 4.x series runs on a managed Kubernetes cluster, so you must migrate your 3.x series data to the new installation.

Upgrade paths

These are valid upgrade paths for Continuous Delivery for Puppet Enterprise (PE).

If you're on version...	Upgrade to version...	Notes
4.30.0	You're up to date!	
4.x (any version in the series)	4.30.0 (current version)	You can safely upgrade from any 4.x version to any newer 4.x version.
3.13.8	4.x (any version in the series)	You must upgrade to the latest 3.13.x version in order to access the 3.x to 4.x data migration tool , and then you can upgrade past 3.13.x.
3.13.3 or earlier	3.13.8	

Upgrade Continuous Delivery for PE

Check for, download, and deploy updates from the **Version history** tab in Puppet Application Manager (PAM). These instructions apply to Continuous Delivery for Puppet Enterprise (PE) installations in online environments.

1. In PAM, click **Version history**.
2. Click **Check for updates**.

Tip: You can click **Configure automatic updates**  to automatically check for updates hourly, every four hours, daily, weekly, or at a custom interval.

3. If there is an update, PAM downloads it and performs preflight checks to make sure your cluster meets system requirements for the new version. Click **View preflight** to review the results.
4. When you're ready to upgrade to the new version of Continuous Delivery for PE, click **Deploy**.

When deployment is complete, sign in to Continuous Delivery for PE and verify that the new version number is shown in the bottom left corner of the web UI.

Upgrade Continuous Delivery for PE offline

If your host doesn't have direct internet access, you must download and install an airgap bundle to upgrade to the latest version of Continuous Delivery for Puppet Enterprise (PE).

1. Download the `.airgap` bundle for the latest version (or the version you want to upgrade to, depending on your [upgrade path](#)). These are the available bundles:

Version	Release date	Airgap bundle
4.35.0	29 April 2025	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.35.0.airgap
4.34.0	30 January 2025	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.34.0.airgap
4.33.0	30 October 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.33.0.airgap
4.32.0	29 August 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.32.0.airgap
4.31.0	10 July 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.31.0.airgap
4.30.2	14 June 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.30.2.airgap
4.30.1	22 May 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.30.1.airgap
4.30.0	7 May 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.30.0.airgap
4.29.2	21 March 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.29.2.airgap
4.29.1	21 February 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.29.1.airgap
4.29.0	8 February 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.29.0.airgap
4.28.0	30 November 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.28.0.airgap
4.27.1	11 October 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.27.1.airgap
4.27.0	4 October 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.27.0.airgap
4.26.2	7 September 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.26.2.airgap
4.26.1	23 August 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.26.1.airgap
4.26.0	22 August 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.26.0.airgap
4.25.1	24 July 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.25.1.airgap
4.25.0	11 July 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.25.0.airgap
4.24.1	28 June 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.24.1.airgap
4.24.0	31 May 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.24.0.airgap
4.23.1	2 May 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.23.1.airgap
4.23.0	18 April 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.23.0.airgap

2. In Puppet Application Manager (PAM), click **Version history** > **Upload new version**, and upload the `.airgap` bundle. The upload can take several minutes.
3. When you're ready to upgrade to the new version of Continuous Delivery for PE, click **Deploy**.

When deployment is complete, sign into Continuous Delivery for PE and verify that the new version number is shown in the bottom left corner of the web UI.

Upgrade an automated installation

If you used the Puppet Application Manager automated installation feature to install Continuous Delivery for Puppet Enterprise (PE), you can run a script to upgrade to the latest version.

1. From the command line of your primary (control plane) node, run the upgrade script:

```
kubectl kots upstream upgrade cd4pe --namespace default --deploy
```

2. Wait five minutes to allow the software time to process the change.
3. Navigate to `http://<NODE_IP_ADDRESS>:8800` and log in with the Puppet Application Manager (PAM) password.

If preflight checks passed, the upgraded application is deployed and in the process of starting up. Run `kubectl get pods --watch` to monitor the deployment's progress.

When the deployment is complete, sign into Continuous Delivery for PE and verify that the new version number is shown in the bottom left corner of the web UI.

Related information

[Automate PAM and Puppet application online installations](#) on page 95

During a fresh online installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

[Automate PAM and Puppet application offline installations](#) on page 97

During a fresh offline installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

Migrate 3.x data to 4.x

The Continuous Delivery for PE 4.x series runs on a managed Kubernetes cluster, so you must migrate your 3.x series data to the new installation.

The migration process has four parts:

1. Upgrade your existing 3.x installation to version 3.13.8.
2. Create a new installation of Continuous Delivery for PE version 4.x.
3. From the version 3.x root console, run the migration task.
4. Restart the 4.x installation, which is now populated with your 3.x data.

What is migrated?

The migration task **copies** and migrates to 4.x:

- The database, which contains all your Continuous Delivery for PE 3.x installation's information, users, integrations, history, and artifacts (except job commands and job logs).

Note: If you configured your 4.x installation to use an externally managed PostgreSQL database, this step is skipped.

- The object store, which contains all of your 3.x installation's job commands and job logs.

Note: The 4.x series replaces external Artifactory and Amazon S3 object storage with a built-in highly available object storage system. Your 3.x object store is migrated to the built-in 4x object store.

- Any of these configuration settings, if in use on 3.x:

```
CD4PE_REPO_CACHE_RETRIEVAL_TIMEOUT_MINUTES
CD4PE_LDAP_GROUP_SEARCH_SIZE_LIMIT
CD4PE_ENABLE_REPO_CACHING
CD4PE_HTTP_CONNECTION_TIMEOUT_SEC
CD4PE_HTTP_READ_TIMEOUT_SEC
CD4PE_HTTP_WRITE_TIMEOUT_SEC
CD4PE_HTTP_REQUEST_TIMEOUT_SEC
CD4PE_INCLUDE_GIT_HISTORY_FOR_CD4PE_JOBS
CD4PE_JOB_GLOBAL_TIMEOUT_MINUTES
CD4PE_JOB_HTTP_READ_TIMEOUT_MINUTES
```

The migration task **updates** in the 4.x installation:

- The webhooks between your source control system and the Continuous Delivery for PE backend endpoint URL.

Note: This step is optional when running the migration task. The migration task is designed safely run multiple times, so you can make sure your 4.x installation is ready for full-time use before you enable webhook updates.

The migration task **does not copy** or move to 4.x:

- The 3.x root user, because you create a new root user when installing 4.x.
- The JVM_ARGS configuration setting, if in use on 3.x.
- These backup tables created for you by Continuous Delivery for PE during the 3.x series:
 - `app-pipelines-3-0-backup.pfi`, which was created during the upgrade from 2.x to 3.x
 - `pe-ia-node-results-cve-2020-7944-backup.pfi` and `pe-ia-resource-results-cve-2020-7944-backup.pfi`, which were created as part of the [CVE 2020-7944](#) remediation in version 3.4.0

Migration cautions

A super user or the root user must perform this process.

Migrating from a local database in a 3.x installation to an external database in a 4.x installation is not supported.

Important: Once you initiate the migration task, any new data created in the 3.x installation may not be migrated to the 4.x installation. To avoid losing data, restrict access to Continuous Delivery for PE during migration.

Run the migration task

The migration task uses temporary credentials to establish a connection between your 3.x and 4.x installations, then migrates your 3.x installation data to your new 4.x installation.

Before you begin

1. **Upgrade your 3.x installation to version 3.13.8.** The migration task fails if you run it on any other 3.x version.

2. Create a new 4.x installation:

- a. Review the [Puppet Application Manager \(PAM\) system requirements](#).
- b. Follow the instructions for your environment to [Install PAM](#) on page 80.
- c. For hosts with internet access, [Deploy Continuous Delivery for PE](#) on page 130. For airgapped hosts, [Deploy Continuous Delivery for PE offline](#) on page 131.



CAUTION: Do not configure integrations or create new data in the 4.x installation until you complete migration.

3. Make sure your 3.x installation can reach the Kubernetes API running on your 4.x installation.
4. If your 4.x installation is behind a proxy, set `HTTP_PROXY` and `HTTPS_PROXY` as values for the `cd4pe_docker_extra_params` parameter on the `cd4pe` class for your 3.x installation, passing in the URL of your proxy.

Note: To pass additional arguments to the Docker process running the Continuous Delivery for PE container, you must specify them as an array. For example: `"-e HTTPS_PROXY=https://proxy.example.com", "-e HTTP_PROXY=https://proxy.example.com"`

Once you've completed the above preparations, you're ready to run the migration task:

1. In PAM, click **Config**.
2. In the **Migration** area, select the **We're migrating an existing Continuous Delivery for PE 3.x instance** option. Scroll to the bottom of the page and click **Save config**.
Enabling this setting allows data to move between your 3.x and 4.x installations.
3. When the success message appears, click **Go to new version**. On the **Version history** page, click **Deploy**.

- To generate temporary credentials to connect your two installations, run the following script on the server where you installed 4.x. If necessary, change the value of the `CD4PE_NAMESPACE` variable to your installation's namespace. Most installations use the default namespace.

```
#!/bin/bash
C_DEFAULT="\e[0m"
C_GREEN="\e[0;32m"

CD4PE_NAMESPACE="default"

K8S_SERVER_URL="$(kubectl config view --minify -o
  jsonpath='{.clusters[*].cluster.server}')"
K8S_TOKEN="$(kubectl -n $CD4PE_NAMESPACE get secret $(kubectl -n
  $CD4PE_NAMESPACE get secrets | grep ^cd4pe-migration | cut -f1 -d ' ') -o
  jsonpath='{[\"data\"][\"token\"]}')"
if [ -z "${K8S_TOKEN}" ]; then
  printf "Kubernetes token not found. Check to ensure 'Enable
  migration of an existing CD4PE instance' is checked in the application
  configuration\n"
  exit -1
fi

K8S_CA_CERT="$(kubectl config view --raw --minify -o
  jsonpath='{.clusters[*].cluster.certificate-authority-data}')"

printf "${C_GREEN}Kubernetes API server URL:${C_DEFAULT}\n\n%s\n\n" \
  "${K8S_SERVER_URL}"

printf "${C_GREEN}Kubernetes service account token:${C_DEFAULT}\n\n%s\n\n" \
  "${K8S_TOKEN}"

printf "${C_GREEN}Kubernetes API CA certificate:${C_DEFAULT}\n\n%s\n\n" \
  "${K8S_CA_CERT}"
```

- In the Continuous Delivery for PE 3.x root console, go to **Settings > Migration**. Copy the temporary credentials you generated in the previous step into the appropriate fields.
- Click **Start migration** and confirm your settings.
- Monitor the status of the migration task in the 3.x Docker logs. Run `docker logs cd4pe` to access the logs. The migration task first migrates your database tables, then migrates your object store, and lastly updates your webhooks, if applicable. When complete, `ran migration task in <elapsed time>` prints to the log.
- Redeploy the 4.x installation. In PAM, click **Config**.
- In the **Migration** area, deselect the **We're migrating an existing Continuous Delivery for PE 3.x instance** option. Click **Save config**.
- When the success message appears, click **Go to new version**. On the **Version history** page, locate the new version and click **Deploy**.
- Go to your 4.x installation, which now is populated with migrated data, and log in with your 3.x user credentials (not the root account).

Test the new 4.x installation

You must thoroughly test the 4.x installation to make sure the data was properly migrated and all functionality is working as expected.

- Test the data migration to make sure your 3.x installation's data is present in your 4.x installation. For each workspace in your 4.x installation:
 - Review event logs for all control repos and modules.
 - Check that all jobs and hardware connections are present.
 - Check that all users and user groups are present.

2. If any data is missing or looks incorrect, [rerun the migration task](#).
3. Test manual pipeline runs to make sure your 4.x installation is functioning properly. For each workspace in your 4.x installation, trigger a manual pipeline run for all control repos and modules. To do this, select **Trigger pipeline** from the **Manual actions** menu above each pipeline.

Once you've confirmed your 3.x data is present in your 4.x installation and all manual pipeline runs succeed, [Update webhooks](#) on page 140.

Update webhooks

As the final step in the 3.x to 4.x data migration process, update the webhooks that connect Continuous Delivery for Puppet Enterprise (PE) to your source control system.

Follow these steps if you're updating webhooks after migrating data from 3.x to 4.x. There are separate instructions to [Update webhooks](#) on page 153 in other scenarios.

1. In the Continuous Delivery for PE 3.x root console, navigate to **Settings > Endpoints** and copy the backend service endpoint.
2. In the Continuous Delivery for PE 4.x root console, click **Settings > Webhooks**, and enter the backend service endpoint you copied from the 3.x root console.
3. Click **Update webhooks**.

Continuous Delivery for PE updates all webhooks to point to the current installation.



CAUTION: After migrating data from 3.x to 4.x, once you update webhooks to point to your 4.x installation, do not create new data in the 3.x installation, and **do not run the migration task again**.

Post-migration guidance and troubleshooting

After migrating Continuous Delivery for Puppet Enterprise (PE) data from 3.x to 4.x, thoroughly testing your 4.x installation, and updating webhooks, you're ready to begin using 4.x with your team. However, do not immediately decommission your 3.x installation. You may need to roll back to 3.x if issues arise.

Important: Keep your 3.x installation intact until you are fully confident that all features of the new 4.x installation are working as expected. **As a precaution, we strongly advise maintaining your inactive 3.x installation for the first two months of using 4.x.**

Maintaining your dormant 3.x installation for an extended test period allows you to roll back to 3.x and prevent production environment disruptions if an issue arises with your 4.x migration.

Rolling back to 3.x

If necessary, you can temporarily roll back to your 3.x installation by switching the active webhooks from the 4.x installation to the 3.x installation and generating new access tokens for all PE instances connected with the 3.x installation.

1. In your Continuous Delivery for PE 3.x installation's web UI, navigate to a control repo's pipeline and click **Manage pipelines**.
2. Select **Manage in the web UI** and locate the **Automation webhook** section where the webhook (including token) for your control repo is shown.
3. Copy the webhook and navigate to the corresponding repository in your source control system. Add the 3.x webhook to the repository and remove the 4.x webhook.
4. Repeat this process for each control repo and module repo in your 3.x installation.
5. Generate a new access token for each PE instance connected to the 3.x installation. In the Continuous Delivery for PE web UI, navigate to **Settings > Puppet Enterprise** and click **More actions > Edit integration**.

6. Select **Basic authorization** or **API token** and enter the required information:

- For **Basic authorization**, enter the username and password for your "Continuous Delivery" user. Continuous Delivery for PE uses this information to generate an API token for you. The username and password are not saved.
- For **API token**, generate a PE access token for your "Continuous Delivery" user using `puppet-access` or the RBAC API v1 service, and paste it into the **API token** field. For information about generating PE access tokens and the RBAC API v1 service, see [Token-based authentication](#) in the PE documentation.

Note: To avoid unintended service interruptions, create access tokens with a multi-year lifespans.

Once the 3.x webhooks and new PE access tokens are in place, you can resume using the 3.x installation.

When you're ready to return to 4.x, repeat the migration process, starting with [Run the migration task](#) on page 137.

Configure and integrate

Configure your Continuous Delivery for Puppet Enterprise (PE) instance so that it communicates with your source control system, Puppet Enterprise, and the job hardware you use to run tests on your Puppet code.

- [Analytics data collection](#) on page 142

Continuous Delivery for Puppet Enterprise (PE) automatically collects data about how you use the software. You can opt out of providing this data when installing Continuous Delivery for PE.

- [Integrate with Puppet Enterprise](#) on page 143

To set up an integration between your Puppet Enterprise (PE) instance and Continuous Delivery for PE, you must first set up a dedicated PE user with appropriate permissions, then add your PE instance's credentials to Continuous Delivery for PE.

- [Integrate with source control](#) on page 146

Integrate your source control system with Continuous Delivery for Puppet Enterprise (PE) by following the instructions for your source control provider.

- [Configure job hardware](#) on page 153

The servers Continuous Delivery for Puppet Enterprise (PE) uses to run tests on your Puppet code are known as job hardware. You must configure job hardware before you can run tests or use pipelines.

- [Add repositories](#) on page 156

You must tell Continuous Delivery for Puppet Enterprise (PE) which repositories in your source control system are your control repos and module repos. This tells Continuous Delivery for PE where to listen for code changes.

- [Configure SAML](#) on page 157

Continuous Delivery for Puppet Enterprise (PE) supports the use of Security Assertion Markup Language (SAML) authentication from a SAML identity provider (IDP). Once you configure your SAML IDP to integrate with Continuous Delivery for PE, you can use your chosen single sign-on tool to authenticate users to Continuous Delivery for PE.

- [Configure LDAP](#) on page 158

Continuous Delivery for Puppet Enterprise (PE) supports use of the Lightweight Directory Access Protocol (LDAP) for managing user authentication. After configuring LDAP, use group mapping to associate your existing LDAP groups with role-based access control (RBAC) groups in Continuous Delivery for PE.

- [Configure SMTP](#) on page 161

Configure SMTP for your Continuous Delivery for Puppet Enterprise (PE) installation so that users can receive email notifications from the software.

- [Set up external PostgreSQL](#) on page 162

To use an external PostgreSQL database with Continuous Delivery for Puppet Enterprise (PE), set up the connection on the **Config** page in Puppet Application Manager (PAM).

- [Load balancing](#) on page 162

The following load balancer requirements are needed for an HA install.

- [Advanced configuration](#) on page 163

Advanced configuration settings for Continuous Delivery for PE help you fine-tune aspects of the software that can impact runtime and operation speed.

Related information

[Configure impact analysis](#) on page 192

Impact analysis is a Continuous Delivery for Puppet Enterprise (PE) tool that shows you the potential impact that new Puppet code can have on your PE-managed infrastructure, without actually merging the new code. When you add impact analysis to a control/module repo's pipeline, Continuous Delivery for PE automatically generates a report on every proposed code change to that repo.

Analytics data collection

Continuous Delivery for Puppet Enterprise (PE) automatically collects data about how you use the software. You can opt out of providing this data when installing Continuous Delivery for PE.

Continuous Delivery for PE collects analytics data in order to better understand how our customers use the software. For example, knowing how many control repos you manage helps us develop more realistic product testing, and learning which source control systems are most and least popular helps us decide where to prioritize new functionality.

What data does Continuous Delivery for PE collect?

Continuous Delivery for Puppet Enterprise (PE) collects analytics data when you use the software. It does not collect any personally identifiable information (PII). The data we collect is never used or shared outside Puppet by Perforce.

Continuous Delivery for PE collects the following data when you start Continuous Delivery for PE for the first time, when Continuous Delivery for PE restarts after an upgrade, and weekly thereafter:

- License UUID
- For each active user account:
 - Number of control repos
 - Number of control repo pipelines
 - Number of control repo pipelines with impact analysis enabled
 - Number of modules
 - The deployment policy selected for each pipeline deployment
 - Configured integrations (PE and source control systems)

Continuous Delivery for PE collects the following data while you use Continuous Delivery for PE:

- Pageviews
- Progress through the initial installation and setup process
- Progress through the integration configuration process (PE and source control systems)

Continuous Delivery for PE **does not** collect:

- Any PII about a user, such as name, email address, password, or company name.
- User inputs such as usernames, control repo names, module names, job names, or job hardware information.

Opt out of analytics data collection

You can opt out of Continuous Delivery for Puppet Enterprise (PE) analytics data collection on the Puppet Application Manager (PAM) configuration settings page.

1. In PAM, click **Config**.
2. In the **Required setup** section of the page, deselect **Report analytics data to Puppet**.
3. Scroll to the bottom of the page and click **Save config**.
4. When the success message appears, click **Go to new version**.

5. On the **Version history** page, click **Deploy**.

You have opted out of data collection, and your instance of Continuous Delivery for PE no longer sends analytics data to Puppet.

Integrate with Puppet Enterprise

To set up an integration between your Puppet Enterprise (PE) instance and Continuous Delivery for PE, you must first set up a dedicated PE user with appropriate permissions, then add your PE instance's credentials to Continuous Delivery for PE.

Before you begin

You must enable Code Manager on the PE instance before integrating with Continuous Delivery for PE. For instructions, see [Configure Code Manager](#).

Do not enable invalid branches in PE. Ensure that the Hiera key `puppet_enterprise::master::code_manager::invalid_branches` is not set to `correct` in PE. This is set in the PE console using **Node Groups > PE Infrastructure > PE Master > Configuration Data**.

The Puppet primary server certificate must have `dns_alt_names` configured. To confirm your certificate configuration, run:

```
openssl x509 -in $(puppet config print hostcert) -text |grep -A 1 "Subject Alternative Name"
```

If this returns no output, you must regenerate the primary server certificate before integrating PE with Continuous Delivery for PE. For instructions, go to [Regenerate primary server certificates](#) in the PE documentation.

Important: If your PE instance has a replica configured for disaster recovery, **Continuous Delivery for PE is not available if a partial failover occurs**. Learn more at [What happens during failovers](#) in the PE documentation. To restore Continuous Delivery for PE functionality, you must promote the replica to primary server.

Create a Continuous Delivery user and user role in PE

Create a "Continuous Delivery" user and user role in PE. You'll use this to generate the PE authentication token required during the setup process and to view a centralized log of the activities Continuous Delivery for PE performs on your behalf.

1. In the PE console, click **Access control > Users**.
2. Enter a full name (such as `Continuous Delivery User`) and a login name (such as `cdpe_user`) and click **Add local user**.
3. Next, create a user role containing the permissions the Continuous Delivery user needs when operating Continuous Delivery for PE. In the PE console, click **Access control > User roles**.
4. Enter a name and an optional description for new role, such as `CDPE User Role`. Then, click **Add role**.
5. Select the new user role from the list on the **User roles** page.

- Click **Permissions**, and assign these permissions to the user role:

Type	Permission	Object
Job orchestrator	Start, stop, and view jobs	-
Node groups	Create, edit, and delete child groups	All
Node groups	View	All
Node groups	Edit configuration data	All
Node groups	Set environment	All
Nodes	View node data from PuppetDB	-
Puppet agent	Run Puppet on agent nodes	-
Puppet environment	Deploy code	All
Puppet Server	Compile catalogs for remote nodes	-
Tasks	Run tasks	At minimum, <code>cd4pe_jobs::run_cd4pe_job</code> , which is required when running job hardware using a Puppet agent.

- Once all permissions are added, click **Commit changes**.
- Add your Continuous Delivery user to the user role. Click **Member users**, select the name of the user you created earlier, click **Add user**, and then commit your change.
- Create a password for the Continuous Delivery user.
 - Return to the **Users** page.
 - Find and click the full name of your Continuous Delivery user, and then click **Generate password reset**.
 - Follow the password reset link and create a password for the user. You'll use this password when adding your PE credentials to Continuous Delivery for PE.

Your Continuous Delivery user is now configured, has a password, and has the permissions needed to operate Continuous Delivery for PE.

Add your Puppet Enterprise credentials

Establishing an integration with your Puppet Enterprise (PE) instance allows Continuous Delivery for PE to work with PE tools, such as Code Manager and the orchestrator service, to deploy Puppet code changes to your nodes.

If necessary, you can add multiple PE instances to your Continuous Delivery for PE installation.

- Make sure you are signed in to your individual Continuous Delivery for PE user account.

Important: Do not perform these steps as the root user. Sign into your individual Continuous Delivery for PE user account before proceeding.

- In the Continuous Delivery for PE web UI, click **Settings > Puppet Enterprise**.
- Click **Add new integration**.
- On the **Integrate with Puppet Enterprise** page, enter a unique, friendly name for your Puppet Enterprise instance.

If you need to work with multiple PE instances within Continuous Delivery for PE, these friendly names help you to differentiate which instance's resources you're managing, so choose them carefully.

- Enter the fully qualified domain name (FQDN) you use to access the PE console, such as `sample.pe.instance`. The FQDN must match the certname of your PE primary server or an alias included in the `dns_alt_names` entry in your `puppet.conf` file.

6. Select **Basic authorization** or **API token** and enter the required information.

Important: If SAML is enabled for your Continuous Delivery for PE installation, you must select **API token**.

- **Basic authorization:** Enter the username and password for your "Continuous Delivery" user. Continuous Delivery for PE uses this information to generate an API token for you. The username and password are not saved. Optionally, you can change the token's lifetime.
- **API token:** Use `puppet-access` or the RBAC v1 API to generate a PE access token for your "Continuous Delivery" user, as explained in [Token-based authentication](#). Paste the token into the **API token** field.

Tip: To avoid unintended service interruptions, create an access token with a multi-year lifespan.

7. Click **Add integration**.

Continuous Delivery for PE uses the information you provide to access the primary SSL certificate generated during PE installation and to look up the endpoints for PuppetDB, Code Manager, orchestrator, and node classifier.

Note: If Continuous Delivery for PE cannot automatically locate the endpoints for your PE services, you are directed to a new page where you can manually enter these credentials. If this happens, go to [Manually configure a Puppet Enterprise integration](#).

Your PE instance is now integrated with Continuous Delivery for PE. You can see integration details on the **Puppet Enterprise integrations** page. To edit an integration, click **More actions** > **Edit integration**.

To enable impact analysis for this instance, see [Configure impact analysis](#).

If you want code deployments to skip unavailable compilers, go to [Enable compiler maintenance mode](#) on page 166.

To delete an integration, click **More actions** > **Remove integration**. You must remove all hardware and pipelines associated with the integration before you can delete the integration itself.

Create environment node groups

For code deployments managed by Continuous Delivery for PE to work correctly, your environment node groups must be set up in a specific hierarchy.

Continuous Delivery for PE deploys changes to environment node groups. Setting up environment node groups allows you to define groups of nodes that you can choose to deploy changes to.

1. In the PE console, click **Node groups**.
2. If there is an **Agent-specified environment** node group, make sure the **All Environments** group is that group's parent. The **Agent-specified environment** node group doesn't have rules and doesn't match any nodes.
3. Make sure the **All Environments** group is the parent of the **Production** environment node group. Make sure the **Production** environment node group's rules only match production nodes.

4. For each of your environments (such as development, testing, and staging), deploy an environment branch and create an environment node group:
 - a) In your control repo, create a Git branch to represent the environment.
 - b) On your PE primary server, run:

```
puppet-code deploy <ENVIRONMENT_NAME>
```

Tip: For more information about the `puppet-code` command, read about [Triggering Code Manager on the command line](#) in the PE documentation. You can also use webhooks and scripts to trigger deployments.

- c) Click **Add group** and create a new node group with these specifications:
 - **Parent name:** Select **All Environments**.
 - **Group name:** Enter the environment name. Match the corresponding Git branch name.
 - **Environment:** Select the name of the environment, which is derived from the corresponding Git branch that you just deployed.
 - **Environment group:** Select this option.
 - **Description:** Enter any relevant description.

Tip: You can learn more [About environments](#) in the Puppet documentation and learn about [Managing environments with a control repository](#) in the PE documentation.

- d) Associate the relevant nodes with the environment group by creating rules or pinning nodes. For more information and instructions, refer to [Add nodes to a node group](#) in the PE documentation.

Follow these best practices for associating nodes with environment node groups:

- Use the `pp_environment` trusted fact, or a similar custom fact, to define which environment each node belongs to. Write a rule in each environment group that uses `pp_environment` or your custom fact to match nodes.
 - See if you can use other facts or trusted facts to create rules that match nodes to one, and only one, environment group.
 - If you can't use trusted facts, custom facts, or other facts to determine node environments, use pinning. Pin each node to only one environment group.
- e) Specify the Git branch corresponding to the environment.

5. Optional: Usually environment node groups do not have children; however there are two scenarios where your environment node groups can have children:

- You want to perform canary testing. Refer to [Environment-based testing](#) in the PE documentation for information and instructions.
- You want to [Run impact analysis on fewer nodes](#) on page 197.

Once you've created an environment node group for each of your environments (represented by Git branches in source control), you'll have a hierarchy of groups starting with **All Environments** as the parent, a number of direct children equal to the number of environments you have (including **Production**) plus (if present) an **Agent-specified environment**. Your environment node groups might have additional children if you configured canary testing or impact analysis groups.

After configuring environment node groups, you can deploy new code to your nodes.

Integrate with source control

Integrate your source control system with Continuous Delivery for Puppet Enterprise (PE) by following the instructions for your source control provider.

Temporary branches

For Impact Analysis and Temporary Branch deployments, Continuous Delivery for Puppet Enterprise creates temporary branches on your repository.

The temporary branches are formatted as follows:

Impact Analysis:

Format: <TARGET_BRANCH_NAME>_cdpe_ia_<UNIX_TIMESTAMP>

Example: production_cdpe_ia_1719840815292

Temporary Branch deployment:

Format: rolling_deployment_<DEPLOYMENT_ID_NUMBER>

Example: rolling_deployment_236

Note: If your repositories have branch naming policies, adjust or remove them.

Status notification prefixes for source control

Once you've integrated your Continuous Delivery for PE installation with your source control provider, Continuous Delivery for PE sends information about the outcome of each stage of each pipeline run to your source control provider.

When reporting to your source control provider, Continuous Delivery for PE uses the following default format to label each pipeline stage: cd-pe/stage-<pipeline stage number>

This labeling system is adequate if you connect a control repo or module repo to one (and only one) workspace. But if you connect multiple workspaces to a certain control repo or module repo, your source control system might receive identical notifications from multiple workspaces about multiple pipelines. The source control system can't differentiate between these identical notifications when performing automated testing.

To prevent this issue, you can add a status notification prefix to all communications Continuous Delivery for PE sends from your workspace to your source control provider. With the status notification prefix, your source control system can differentiate between, and accurately act on, pipeline status notifications coming from multiple workspaces to the same control repo or module repo.

To add a status notification prefix:

1. In the Continuous Delivery for PE web UI, click **Settings > Source control**.
2. In the **Status notification prefix** section, click **Edit prefix** .
3. Enter a prefix, such as your workspace's name. Click **Save**.

After saving your prefix, the example code updates to show the prefixed pipeline status labels that this workspace sends to your source control provider.

Integrate with Azure DevOps Cloud Services

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. You must create an Azure DevOps Services OAuth application to integrate your Azure DevOps Services instance with Continuous Delivery for PE and start using these tools.

Before you begin



CAUTION: As of April 2025, Microsoft has moved to using Entra ID to create Azure DevOps OAuth applications, which is not supported by Continuous Delivery for PE. OAuth applications created prior to April 2025 continue to function.

Important: In order for a control repo in Continuous Delivery for Puppet Enterprise to deploy Puppet code from a Puppet Enterprise (PE) instance, you need to configure Code Manager/r10k in PE to communicate to that control repo. For Azure DevOps Services, the SSH URL configured in Code Manager/r10k needs to include the Git SSH username. For example: `ssh://git@<YOUR.ADO.SERVER.COM>:puppet/control.git`.

An administrator on your team must create an Azure DevOps Services OAuth application for Continuous Delivery for PE.

1. Sign into Continuous Delivery for PE as the root user, and click **Settings > Integrations**. The authorization callback URL required to create your OAuth app is shown in the root console.
2. Go to <https://app.vsaex.visualstudio.com/app/register>.
3. Enter your company name.
4. In the **Application Information** section, enter a name for your OAuth application, such as CD for PE.
5. In the **Application website** field, enter the base URL for your Continuous Delivery for PE instance.
6. In the **Authorization callback URL** field, enter the authorization callback URL printed in the Continuous Delivery for PE root console.
7. In the **Authorized scopes** section, select **Code (read and write)**.
8. Click **Create Application**. When you are redirected to the page showing the new application's setting, stay on this page.

Important: Don't close this page. You need the application settings information in the next step.

9. Return to the Continuous Delivery for PE root console. On the **Integrations** page, enter the application ID and client secret for your Azure DevOps Services OAuth application, and click **Add**.

Once you have an Azure DevOps Services OAuth application for your organization, each workspace must be authenticated with the application in order to integrate your Continuous Delivery for PE instance with Azure DevOps Services. To do this, you must grant code read and write permissions and add a public SSH key, which allows cloning of modules and control repos during automated tasks.

Important:

If your organization uses Azure DevOps Services branch permissions to limit user access to Git branches, review the permissions granted to Continuous Delivery for PE users. Make sure these users can force push to the relevant control repos and module repos.

Azure DevOps Services only supports cloning over SSH. HTTP(S) cloning is not supported. You must enable SSL on Continuous Delivery for PE to use Azure DevOps Services.

1. In the Continuous Delivery for PE web UI, click **Add new source**.
2. Select **Azure DevOps** from the **Source** menu.
3. Click **Add credentials** to give Continuous Delivery for PE permission to read and write code to your Azure DevOps Services account.
You are redirected to a Microsoft page.
4. Click **Accept**.
You are redirected back to the **Source control** page.
5. In the Continuous Delivery for PE web UI, click **SSH key**.
6. Click **Show** to display your public SSH key, and click **Copy**.
7. In the Azure DevOps Services web UI, open the user menu, click **Security**, and then click **SSH public keys**.
8. Click **Add** and paste your public SSH key into the **Key Data** field.
9. Add a description and click **Save**.

After integrating with your source control provider, [Add repositories](#) on page 156.

Integrate with Azure DevOps Server on prem

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. Integrate your Azure DevOps Server instance with Continuous Delivery for PE to start using these tools.

Important:

If your organization uses Azure DevOps Services branch permissions to limit user access to Git branches, review the permissions granted to Continuous Delivery for PE users. Make sure these users can force push to the relevant control repos and module repos.

Azure DevOps Services only supports cloning over SSH. HTTP(S) cloning is not supported.

1. In the Continuous Delivery for PE web UI, click **Add new source**.
2. Select **Azure DevOps** from the **Source** menu.
3. Select **Connect to Azure DevOps on prem**.
4. In the **Host** field, enter the public IP or DNS for your Azure DevOps instance.
5. Create a token allowing Continuous Delivery for PE to access your Azure DevOps instance. In the Azure DevOps Server web UI:
 - a) From the **View your profile** menu, click **Security**.
 - b) From **Personal access tokens**, click **New Token**
 - c) Enter a token name.
 - d) Set **Organization** to **All accessible organizations**.
 - e) Select as long of a lifetime as your security policy allows.
 - f) Select **Full Scope**.
 - g) Click **Generate token**.
 - h) Copy the personal access token created by Azure DevOps.
6. In the Continuous Delivery for PE web UI, enter the Azure DevOps Server token you just created in the **Token** field.
7. Based on your Azure DevOps Server configuration, select either **This instance uses a standard CA certificate**, or **This instance uses a custom CA certificate**. If you are using a custom certificate, paste the full certificate in the **Custom CA certificate** field.
8. Click **Add credentials**.
9. In the Continuous Delivery for PE web UI, click **SSH key**.
10. Click **Show** to display your public SSH key and click **Copy**.
11. In the Azure DevOps Services web UI, open the user menu, click **Security**, and then click **SSH public keys**.
12. Click **Add** and paste your public SSH key into the **Key Data** field.
13. Add a description and click **Save**.

After integrating with your source control provider, [Add repositories](#) on page 156.

Integrate with Bitbucket Cloud

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. You must create a Bitbucket Cloud OAuth application to integrate your Bitbucket Cloud instance with Continuous Delivery for PE and start using these tools.

Before you begin

An administrator on your team must create a Bitbucket Cloud OAuth consumer for Continuous Delivery for PE.

1. Sign into Continuous Delivery for PE as the root user, and click **Settings > Integrations**.

- In your organization's Bitbucket Cloud account, create an OAuth consumer. Instructions to [Create a consumer](#) are in the Bitbucket Cloud documentation. The authorization callback URL required to create your OAuth consumer is shown in the root console.
- Give the OAuth consumer these permissions:

Category	Permissions
Account	Email, Read
Workspace membership	Read
Repositories	Read, Write
Pull requests	Read, Write
Webhooks	Read and write

- After creating the OAuth application, note the key and secret shown on the OAuth settings page in the Bitbucket Cloud web UI.
- Return to the Continuous Delivery for PE root console. On the **Integrations** page, enter the client ID (key) and client secret for your Bitbucket Cloud OAuth consumer, and click **Add**.

Once you have a Bitbucket Cloud OAuth application for your organization, each workspace must be authenticated with the application in order to integrate your Continuous Delivery for PE instance with Bitbucket Cloud.

Important:

If your organization uses Bitbucket Cloud branch permissions to limit user access to Git branches, review the permissions granted to Continuous Delivery for PE users. Make sure these users have write access and the ability to rewrite history on the relevant control repos and module repos.

Bitbucket Cloud only supports cloning over HTTP(S). Bitbucket Cloud does not support SSH cloning, and it does not support pull requests from forks. Jobs run on pull requests from forks will fail.

- In the Continuous Delivery for PE web UI, click **Add new source**.
- Select **Bitbucket Cloud** from the **Source** menu.
- Click **Add credentials** to give Continuous Delivery for PE permission to read and write code to your Bitbucket Cloud account.
- Click **Add credentials**.
You're redirected to Bitbucket Cloud to authorize the OAuth application set up by your workplace administrator.
- Click **Grant access** to allow Continuous Delivery for PE to access your Bitbucket Cloud account.

After integrating with your source control provider, [Add repositories](#) on page 156.

Integrate with Bitbucket Server

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. Integrate your Bitbucket Server instance with Continuous Delivery for PE to start using these tools.

Important:

If your organization uses Bitbucket Server branch permissions to limit user access to Git branches, review the permissions granted to Continuous Delivery for PE users. Create an exemption rule to ensure these users can force push to the relevant control repos and module repos.

Bitbucket Server only supports cloning over SSH. Bitbucket Server does not support HTTP(S) cloning, and it does not support pull requests from forks. Jobs run on pull requests from forks will fail.

Continuous Delivery for PE supports Bitbucket Server versions 5.0 and newer.

1. In the Continuous Delivery for PE web UI, click **Add new source**.
2. Select **Bitbucket Server** from the **Source** menu.
3. In the **Bitbucket Server host** field, enter the public IP or DNS for your Bitbucket Server instance.
4. In the **Username** and **Password** fields, enter the credentials associated with the account you want to connect to Continuous Delivery for PE.
5. In the **SSH port** field, enter the port number on which your Bitbucket Server listens for SSH requests. To locate this port number:
 - a) In the Bitbucket Server web UI, click **Administration** (the gear icon) and then click **Server settings**.
 - b) Locate the SSH port in the **SSH access** section of the **Server settings** page.
6. If your Bitbucket Server's SSH base URL is different from the host URL, enter the SSH base URL. To view your SSH base URL:
 - a) In the Bitbucket Server web UI, click **Administration** (the gear icon) and then click **Server settings**.
 - b) Locate the SSH base URL in the **SSH access** section of the **Server settings** page.
7. Enter the SSH user for clones if it is something other than `git`.
8. Click **Add credentials**.

After integrating with your source control provider, [Add repositories](#) on page 156.

Integrate with GitHub

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. You must create a GitHub OAuth application to integrate your GitHub instance with Continuous Delivery for PE and start using these tools.

Before you begin

An administrator on your team must create a GitHub OAuth application for Continuous Delivery for PE.

1. Sign into Continuous Delivery for PE as the root user, and click **Settings > Integrations**.
2. In your organization's GitHub account, create an OAuth application. Instructions for [Creating an OAuth App](#) are in the GitHub documentation. In the **Homepage URL** field, enter the base URL for your Continuous Delivery for PE instance (`http://<CD4PE-HOST-SERVER>`). The **Authorization callback URL** is shown in the Continuous Delivery for PE root console.
3. After creating your OAuth application, note the Client ID and Client Secret shown on the application's page in the GitHub UI.
4. Return to the Continuous Delivery for PE root console. On the **Integrations** page, enter the client ID and secret for your GitHub OAuth application, and click **Add**.

Once you have a GitHub OAuth application for your organization, each workspace must be authenticated with the application in order to integrate your Continuous Delivery for PE instance with GitHub.

Important:

If your organization uses protected branches on GitHub, make sure you allow force pushing to protected branches, or that you use the GitHub Administrator user to connect control repos or module repos to Continuous Delivery for PE.

GitHub only supports cloning over HTTP(S). SSH cloning is not supported.

1. In the Continuous Delivery for PE web UI, click **Add new source**.
2. Select **GitHub** from the **Source** menu.
3. Click **Add credentials**.
You're redirected to GitHub to authorize the OAuth application set up by your team's administrator.
4. Click **Grant access** to allow Continuous Delivery for PE to access your GitHub account.

After integrating with your source control provider, [Add repositories](#) on page 156.

Integrate with GitHub Enterprise

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. Integrate your GitHub Enterprise instance with Continuous Delivery for PE to start using these tools.

Important:

If your organization uses protected branches on GitHub Enterprise, make sure you allow force pushing to protected branches, or that you use the GitHub Enterprise Administrator user to connect control repos or module repos to Continuous Delivery for PE.

GitHub Enterprise only supports cloning over HTTP(S). SSH cloning is not supported.

1. In the Continuous Delivery for PE web UI, click **Add new source**.
2. Select **GitHub Enterprise** from the **Source** menu.
3. In the **Host** field, enter the public IP or DNS for your GitHub Enterprise instance.
4. Create a token allowing Continuous Delivery for PE to access your GitHub Enterprise instance.
 - a) In the GitHub Enterprise web UI, click **View your profile and more** (profile photo icon) and then click **Settings**.
 - b) Click **Developer settings** > **Personal access tokens** > **Generate new token**.
 - c) Enter a token description, such as `CD for PE`.
 - d) Select the **repo**, **read:org**, and **user:email** scopes.
 - e) Click **Generate token**.
 - f) Copy the personal access token created by GitHub Enterprise.
5. In the Continuous Delivery for PE web UI, enter the GitHub Enterprise token in the **Token** field.
6. Based on your GitHub Enterprise configuration, select either **This instance uses a standard CA certificate** or **This instance uses a custom CA certificate**. If you're using a custom certificate, paste the full certificate in the **Custom CA certificate** field.
7. Click **Add credentials**.

After integrating with your source control provider, [Add repositories](#) on page 156.

Integrate with GitLab

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. Integrate your GitLab instance with Continuous Delivery for PE to start using these tools.

Important: If your organization uses protected branches on GitLab, make sure the GitLab user account connected to Continuous Delivery for PE is assigned to a GitLab role with “allow” rules that enable the user to push to the protected branch.

GitLab supports cloning over both SSH and HTTP(S). You set the cloning protocol for each Continuous Delivery for PE workspace.

1. In the Continuous Delivery for PE web UI, click **Add new source**.
2. Select **GitLab** from the **Source** menu.
3. In the **Host** field, enter the resolvable HTTP or HTTPS URL for your GitLab instance, such as `http://gitlab.example.com`.
The URL must be formatted as a resolvable HTTP or HTTPS address even if you are using an SSH connection.

4. Create a token allowing Continuous Delivery for PE to access your GitLab instance.
 - a) In the GitLab web UI, navigate to your user settings and click **Access tokens**.
 - b) Enter a name for the application, such as `CD for PE`, and set an expiration date for the token. This token has a maximum expiry of 1 year. If you need to create a token that disables the maximum expiry setting, [use a service account token](#) instead.

Note: To create a service account access token, you need the following:

 - GitLab must be at least version 16.1
 - Premium or ultimate tiered account
 - You must [disable the service account max expiry date setting](#)
 - c) Select the **api** and **read_user** scopes.
 - d) Click **Create personal access token**.
 - e) Copy the personal access token created by GitLab.
5. In the Continuous Delivery for PE web UI, enter the GitLab token in the **Token** field.
6. Select whether your workspace clones GitLab repositories via SSH or HTTP(S).
 - a) For SSH:
 - **Optional:** Add the SSH user's credentials in the **SSH user** field.
 - **Optional:** In the **SSH port** field, specify the port on which your GitLab server listens for SSH requests. The default port number is 22.
 - b) For HTTP(S):
 - If you're using a custom certificate, paste the full certificate into the **Custom CA certificate** field.
7. Click **Add credentials**.

After integrating with your source control provider, [Add repositories](#) on page 156.

Update webhooks

You must update the webhooks that connect Continuous Delivery for PE to your source control system if you change the location of your Continuous Delivery for Puppet Enterprise (PE) installation, change the hostname when migrating your Puppet Application Manager (PAM) installation, or change the backend service endpoint.

1. In the Continuous Delivery for PE root console, click **Settings > Webhooks**.
2. Enter the backend service endpoint:
 - If you changed your Continuous Delivery for PE installation location, enter the backend service endpoint for your previous Continuous Delivery for PE installation.
 - If you changed the hostname when migrating your PAM installation, run `kubectl describe pod <CD4PE_POD> | grep BACKEND` to get the backend service endpoint.
3. Click **Update webhooks**.

Continuous Delivery for PE updates all webhooks according to your configuration.

Configure job hardware

The servers Continuous Delivery for Puppet Enterprise (PE) uses to run tests on your Puppet code are known as job hardware. You must configure job hardware before you can run tests or use pipelines.

- [Job hardware requirements](#) on page 154

System requirements for Continuous Delivery for PE job hardware depend on the size of your Continuous Delivery for PE installation, the type of jobs you run, and how frequently you run them.

- [Configure job hardware](#) on page 154

The servers Continuous Delivery for Puppet Enterprise (PE) uses to run tests on your Puppet code are known as job hardware. You must configure job hardware before you can run tests or use pipelines. Any node with a Puppet agent installed on it can be job hardware.

- [Add job hardware capabilities](#) on page 155

Job hardware servers are organized by capabilities in Continuous Delivery for PE. A capability is a tag that indicates what type of jobs can run on that job hardware server. If you're managing a lot of job hardware servers, use capabilities to distribute the testing load.

- [Configure global shared job hardware](#) on page 156

All workspaces in your Continuous Delivery for Puppet Enterprise (PE) installation can use global shared job hardware. The root user or a super user must set up these special job hardware servers in the root console.

Job hardware requirements

System requirements for Continuous Delivery for PE job hardware depend on the size of your Continuous Delivery for PE installation, the type of jobs you run, and how frequently you run them.

The size of the load placed on a job hardware server determines how robust the server's resources must be. Due to the number of variables involved in estimating this (such as the number of jobs running concurrently and the languages the jobs are written in), it's impossible to provide universal system requirements for job hardware.

Instead, we've created a sizing guide based on the estimated number of concurrent jobs you expect your job hardware server to handle regularly. While this guide represents our best estimates and understanding, it's only a starting point. Through testing and experience, you can fine-tune your job hardware and determine the optimum resource configuration for your Continuous Delivery for PE installation.

Estimated concurrent job load	Memory	Disk storage	CPUs
2 - 4 concurrent spec tests	4 GB	100 GB	2
4 - 8 concurrent spec tests	8 GB	100 GB	4
6 - 12 concurrent spec tests	8 GB	100 GB	6

When setting up your job hardware, keep in mind:

- Disk storage requirements are minimal and don't increase with added load. After a job run is complete, the job's log is passed to the object storage, and all data related to the job run is erased from the job hardware.
- You can run more jobs concurrently without increasing CPUs, but the jobs run slower.

Job hardware requirements for Docker-based jobs

To run Docker-based jobs, your job hardware must have a modern version of Docker CE or Docker EE installed. The [puppetlabs/docker](#) module (from the Forge) is our recommended way to install Docker and keep it updated.

Job hardware used for Docker-based jobs also requires internet access. If you're working in an air-gapped environment, set up an internal Docker registry, as explained in the [Docker documentation](#).

Global shared job hardware uses a shared Docker image set in the root console. The default image used for Docker-based jobs is [puppet/puppet-dev-tools:4.x](#). You can find details on the available commands in the image documentation.

Configure job hardware

The servers Continuous Delivery for Puppet Enterprise (PE) uses to run tests on your Puppet code are known as job hardware. You must configure job hardware before you can run tests or use pipelines. Any node with a Puppet agent installed on it can be job hardware.

Before you begin

Review the [Job hardware requirements](#) sizing guide.

Important: Only the administrator of a workspace or a super user can configure job hardware for a workspace.

1. Install a Puppet agent on each node you want to use as job hardware. The PE documentation provides instructions for [Installing agents](#).
2. Make sure your [Continuous Delivery user role in PE](#) has permission to run the `cd4pe_jobs::run_cd4pe_job` task.
3. Install and configure Docker on each job hardware node. Docker is required for Docker-based [pre-built jobs](#) included in Continuous Delivery for PE. Instructions are available in the [Docker documentation](#).
4. Assign nodes to the **Docker** hardware capability (which Continuous Delivery for PE automatically creates for you) to tell Continuous Delivery for PE the nodes can be used as job hardware for Docker-based jobs.
 - a) In the Continuous Delivery for PE web UI, click **Hardware**.
 - b) Locate the **Docker** capability and click **Edit**.
 - c) Select the PE instance that manages the nodes you've selected as job hardware, and then select the job hardware nodes. Selected nodes appear in the **Hardware with this capability** list.
 - d) After selecting all the job hardware nodes, click **Save**.
5. Optional: Create additional capabilities or add job hardware nodes to other capabilities, as explained in [Add job hardware capabilities](#). If you're managing a lot of job hardware nodes, additional capabilities help distribute the testing load and ensure that each job runs on a hardware with the correct characteristics.

Your job hardware is configured and ready for use.

Add job hardware capabilities

Job hardware servers are organized by capabilities in Continuous Delivery for PE. A capability is a tag that indicates what type of jobs can run on that job hardware server. If you're managing a lot of job hardware servers, use capabilities to distribute the testing load.

Capabilities organize your job hardware servers and ensure that jobs run on hardware with the right characteristics. Continuous Delivery for PE automatically creates a **Docker** capability for you, and you can create additional capabilities according to your needs.

For example, assume you want to create a new job that requires Python 3. To ensure Continuous Delivery for PE can run this job, you need to create a **python-3** capability in Continuous Delivery for PE and assign to that capability all job hardware servers that have Python 3 installed and configured. Then, when you create the job that uses Python 3, specify that the job requires job hardware with the **python-3** capability. When you run this job, Continuous Delivery for PE automatically locates a job hardware server with the **python-3** capability and uses that job hardware server to run the job.

In addition to software requirements, useful capabilities include:

- Operating systems
- Organizational designations, such as `dev-team-philly-testing`
- Puppet testing resources, such as `onceover` or Puppet Development Kit

Important: Only the administrator of a workspace, the root user, and super users can add hardware capabilities to a workspace.

1. In the Continuous Delivery for PE web UI, click **Hardware**.
2. Click **Add capability** and name your new capability.
3. To assign job hardware servers to the capability, select the PE instance that manages the node you've selected as job hardware, and then select the nodes you want to assign to the capability. Selected nodes appear in the **Hardware with this capability** list.
4. After selecting all the nodes you want to assign to this capability, click **Save**.

Your job hardware is now assigned to a capability. To add capabilities to jobs in your workspace, click **Jobs > Edit job**, and select options from the list of capabilities.

Configure global shared job hardware

All workspaces in your Continuous Delivery for Puppet Enterprise (PE) installation can use global shared job hardware. The root user or a super user must set up these special job hardware servers in the root console.

Before you begin

Review the [Job hardware requirements](#) sizing guide.

1. Install a Puppet agent on each node you want to use as global shared job hardware. The PE documentation provides instructions for [Installing agents](#).
2. Make sure your [Continuous Delivery user role in PE](#) has permission to run the `cd4pe_jobs::run_cd4pe_job` task.
3. Install the `puppetlabs-cd4pe_jobs` module from the Forge. This module is required to run Continuous Delivery for PE jobs on your nodes.
 - a) Add the `puppetlabs-cd4pe_jobs` module to:
 - The Puppetfile for the production environment on the PE primary server that manages the agent nodes you've selected as job hardware.
 - The Puppetfile on the main branch of the control repo associated with the PE primary server managing the agent nodes you've selected as job hardware.

A sample Puppetfile entry:

```
mod 'puppetlabs-cd4pe_jobs', '1.6.0'
```

- b) Run `puppet code deploy production --wait` to deploy the updated code to the production environment.
4. Install and configure Docker on your selected global shared job hardware nodes. Instructions are available in the [Docker documentation](#).
5. In the Continuous Delivery for PE [root console](#), click **Hardware**.
6. You must assign all global shared job hardware servers to the **Docker** capability, which Continuous Delivery for PE automatically creates for you. Locate the **Docker** capability and click **Edit**.
7. Assign all your global shared job hardware servers to the **Docker** capability. Select the PE instance that manages the nodes you've selected as job hardware, and then select the global shared job hardware nodes. Selected nodes appear in the **Hardware with this capability** list.
8. After selecting all the global shared job hardware nodes, click **Save**.
9. Optional: Create additional capabilities for your global job hardware servers, as explained in [Add job hardware capabilities](#).

Your global shared job hardware is configured. Users in all workspaces now see a **Run on shared hardware** option when creating or editing a job. Jobs configured to run on your global shared job hardware have the **Docker** capability automatically selected, and these jobs run on the global shared job hardware assigned to the **Docker** capability.



CAUTION: If your installation specifies a Docker image in Puppet Application Manager (PAM) (this is common in offline environments), this image takes precedence over the image set in the Continuous Delivery for PE root console (which is the one shown when creating or editing a job).

Add repositories

You must tell Continuous Delivery for Puppet Enterprise (PE) which repositories in your source control system are your control repos and module repos. This tells Continuous Delivery for PE where to listen for code changes.

Before you begin

[Integrate with source control](#) on page 146 and learn about [Working with Git branches in Continuous Delivery for PE](#) on page 6 and [Understanding the Continuous Delivery for PE workflow](#) on page 7.

Control Repos in Continuous Delivery for PE track changes made on active development branches of your source control system. Module repos, referred to as **Modules** in the Continuous Delivery for PE web UI, contain information about Puppet modules. When adding repos to Continuous Delivery for PE, it's important to connect the repo's `main` Git branch.

Adding a repo in Continuous Delivery for PE configures a webhook to the repository in your source control system. The webhook reports new commit activity on the repository to Continuous Delivery for PE, enabling you to track code changes and take action (such as triggering pipelines or running individual tests on the new code).

1. In the Continuous Delivery for PE web UI, click **Control repos > Add control repo**.

To add a module repo, click **Modules > Add module**.

2. Select the source control provider and repository.

The **Repository** dropdown menu lists a maximum of 10 repositories. Begin typing a repository name to refresh and refine the list. If there are still more than 10 results, you'll need to provide a more specific repository name until the desired repository appears.

3. The repo's `main` branch is automatically selected, if one exists. If the repo does not have a `main` branch, select **Create a main branch from an existing branch** and select your active development branch as the basis for the new `main` branch. Continuous Delivery for PE creates a `main` branch for you based on the existing branch you choose.



CAUTION: When working with Continuous Delivery for PE, only commit to the `main` branch and any feature branches (which are eventually merged back into the `main` branch). Do not push code changes to any other Git branches; this can create conflicts with Continuous Delivery for PE workflows.

4. Optional: Edit the repo's **Display name** as it appears in the Continuous Delivery for PE web UI. The name can contain only numbers, letters, dashes, and underscores.
5. Click **Add control repo** (or **Add module**).
6. If you added a control repo, make sure you add the `puppetlabs-cd4pe_jobs` module to the Puppetfile on the control repo's `main` branch, as explained in [Configure job hardware](#) on page 154.
7. Repeat to add more repos to Continuous Delivery for PE.

[Construct pipelines](#) to test new code committed to your repos.

Configure SAML

Continuous Delivery for Puppet Enterprise (PE) supports the use of Security Assertion Markup Language (SAML) authentication from a SAML identity provider (IDP). Once you configure your SAML IDP to integrate with Continuous Delivery for PE, you can use your chosen single sign-on tool to authenticate users to Continuous Delivery for PE.

Before you begin

Your enterprise SAML team must configure your organization's SAML IDP to communicate with Continuous Delivery for PE. Provide the SAML team with the Continuous Delivery for PE SAML redirect URL for your installation: `<YOUR_CD4PE_WEB_UI_ENDPOINT>/cd4pe/saml-auth`. The SAML team uses this to register Continuous Delivery for PE as an application with permissions to interact with the IDP.

Get this information from your enterprise SAML team:

- The IDP-initiated SSO URL that Continuous Delivery for PE needs to direct user authentication requests.
- The IDP public signing certificate for Continuous Delivery for PE.
- The SAML attribute names that come back in the SAML assertion for these fields: first name, last name, email address, and username. (Attribute mapping is explained in step 4, below.)

If your IDP cannot use the `saml-auth` endpoint for configuration, the following may be helpful:

- **Entity ID:** CD4PE

- **Signing certificate:** This is generated by the IDP or the team managing the IDP. The public cert is then saved in Continuous Delivery for PE via a settings page in the Continuous Delivery for PE root console.
- **ACS (Assertion Consumer Service) URL:** <CD4PE WEB UI ENDPOINT>/cd4pe/saml-auth
- **Logout URL:** We do not support a logout URL for SAML integrations, this can only be performed through the Continuous Delivery for PE UI. If this is an optional config option in PingID, it can be left blank.

A super user or the root user must perform this task.

1. Log into the root console by signing in as the root user or by selecting **Root console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar.
2. Click **Settings** and click the **Single sign on** tab if you are not already on it.
3. Select **SAML**.
4. Enter the required configuration information as per the instructions below.

IdP Initiated SSO URL

The unique URL created by the SAML IDP used by your organization that acts as a single sign-on (SSO) gateway for Continuous Delivery for PE. Your enterprise SAML team provides this URL.

Public Signing Certificate

The SAML IDP public signing certificate verifies SAML assertions from the IDP. Your enterprise SAML team provides this certificate. Paste the entirety of the certificate, including the header and footer, into this field.

Note: This field is for a SAML certificate only. To use a custom certificate for Continuous Delivery for PE overall, refer to [Use custom TLS certificates](#) on page 165.

Attribute Mapping

The SAML assertion sends four attributes to Continuous Delivery for PE. The **Attribute Mapping** matches attribute keys from the SAML IDP assertion to user accounts created by Continuous Delivery for PE.

- **First Name:** The SAML attribute key for the user's first name.
 - **Last Name:** The SAML attribute key for the user's last name.
 - **Email:** The SAML attribute key for the user's email address. This is the unique user identifier, so each user's email address must be unique.
 - **Username:** The SAML attribute key for the user's username in Continuous Delivery for PE. Each username must be unique.
5. Click **Run Configuration Test** to send a sample authentication query to your SAML IDP.
 6. If the configuration test is successful, you're ready to enable SAML authentication for your Continuous Delivery for PE instance, enable the SAML configuration switch and click **Save Configuration**.



CAUTION: If the SAML IDP or the SAML information saved in Continuous Delivery for PE is misconfigured, you might be locked out of Continuous Delivery for PE. If this happens, navigate to <YOUR CD4PE WEB UI ENDPOINT>/root/login, sign in as the root user, disable the SAML configuration switch, and click **Save Configuration**.

Configure LDAP

Continuous Delivery for Puppet Enterprise (PE) supports use of the Lightweight Directory Access Protocol (LDAP) for managing user authentication. After configuring LDAP, use group mapping to associate your existing LDAP groups with role-based access control (RBAC) groups in Continuous Delivery for PE.

For organizational or failover protection purposes, you can add multiple LDAP configurations, each specifying a separate LDAP server, to your Continuous Delivery for PE instance. Continuous Delivery for PE uses the LDAP configurations you set up to search LDAP users in a specified order. Once a user is found, the search ends and that LDAP configuration is used to perform the login operation.

Note: If your LDAP server has a search result limitation below 500, you'll need to [Configure LDAP server search result limits](#) on page 164.

Create a new LDAP configuration

Add an LDAP configuration to Continuous Delivery for Puppet Enterprise (PE) by providing key information on the mapping of user and group attributes in your LDAP server implementation.

A super user or the root user must perform this task.

1. Log into the root console by signing in as the root user or by selecting **Root console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar.
2. Click **Settings** and click the **Single sign on** tab if you are not already on it.
3. Select **LDAP** and click **Add LDAP configuration**.

4. In the **New LDAP configuration** window, enter the configuration information as per the instructions below.

Name

A friendly identifier for this LDAP configuration. You can't change this, so choose carefully.

Endpoint URL

The LDAP server's endpoint URL, including the LDAP scheme, hostname, and port. If these aren't included, the default scheme is `ldaps` and the default port is `636`.

Bind DN

The distinguished name of the LDAP account that Continuous Delivery for PE binds as when performing LDAP operations. Usually, this is an admin account or a service account created specifically for Continuous Delivery for PE. If you create a new account, make sure it has permission to search for users and groups.

Bind DN password

The password associated with the bind DN account.

Note: You must enter this password each time you update the LDAP configuration. You don't have to enter the password when you disable the LDAP configuration.

User base DN

The LDAP base DN that informs Continuous Delivery for PE where users are located in the directory. Using a more specific DN results in better LDAP search performance.

User attribute

The LDAP user attribute that maps LDAP users to Continuous Delivery for PE usernames. Usually, this is the `mail` attribute, but you can use any attribute as long as the LDAP database doesn't contain any duplicate values for that attribute.

Optional: User base filter

A filter that Continuous Delivery for PE can use to restrict user search results. This is useful in edge cases where you want to include or exclude certain users.

Group base DN

The LDAP base DN that informs Continuous Delivery for PE where groups are located in the directory. Using a more specific DN results in better LDAP search performance.

Note: If users and groups are stored in the same location, the **Group base DN** is the same as the **User base DN**.

Group user attribute

The user attribute that group entries use to identify users. Usually, this is `dn`.

Group member attribute

The group attribute that maps to a group member. Usually, this is either `member` or `uniqueMember`.

Group name attribute

The group attribute that identifies the group name. Usually, this is `cn`.

Optional: Group base filter

A filter that Continuous Delivery for PE can use to restrict group search results. This is useful in edge cases where you want to include or exclude certain groups.

User object class

Specifies the value of the `objectClass` attribute that allows Continuous Delivery for PE to query user entries. Usually, this is either `user` or `person`.

Group object class

Specifies the value of the `objectClass` attribute that allows Continuous Delivery for PE to query group entries. Usually, this is either `group` or `groupOfUniqueNames`.

Optional: Mail attribute

The LDAP user attribute used to identify each member's email address. Defaults to `mail` if unset.

Optional: User member attribute

Specifies the user attribute that can be used to identify the membership of a group. If present, this is usually

5. Optional: Enter the trusted server's CA certificate. If the LDAP server uses a certificate signed by a trusted CA, you do not need to enter a CA certificate here.
6. Select the **Priority** number for this LDAP configuration. If you have multiple LDAP configurations, this number indicates the order in which Continuous Delivery for PE searches for users in your LDAP configurations during login and when synchronizing groups.
7. Optional: Set the toggle to enable recursive LDAP queries for nested groups. This option is available only if your LDAP server's implementation supports the [ExtensibleMatch search filter](#).
8. Optional: Set the toggle to enable use of login filtering. When enabled, users who are not part of mapped LDAP groups are not allowed to log in to Continuous Delivery for PE.
9. Activate the **Enable LDAP** switch and click **Run configuration test** to check the connection between Continuous Delivery for PE and the LDAP server.

Note: This configuration test checks the LDAP server connection; it does not test the other configuration options.

10. Click **Save configuration**. Your new LDAP configuration is shown on the **Single sign on** settings page, where you can edit or delete the configuration.

Important: Once you enable an LDAP configuration, Continuous Delivery for PE automatically disables all local Continuous Delivery for PE accounts other than the root account, and it attempts to use LDAP authentication. If LDAP authentication fails, navigate to `<YOUR CD4PE WEB UI ENDPOINT>/root/login`, sign in as the root user, and adjust the LDAP settings.

Create an LDAP group map

After adding an LDAP configuration to Continuous Delivery for Puppet Enterprise (PE), use group maps to map your existing LDAP groups to Continuous Delivery for PE RBAC groups. This makes it possible to mirror LDAP group membership in Continuous Delivery for PE groups.

Before you begin

You must add at least one LDAP configuration to your Continuous Delivery for PE instance before you can create an LDAP group map.

1. Log in to the root console by navigating to `<YOUR CD4PE WEB UI ENDPOINT>/root/login` and signing in as the root user.
2. Click **Settings** and click the **Single sign on** tab if you are not already on it.
3. Click **LDAP > Manage groups > Add LDAP group mapping**.
4. From the **LDAP configuration name** list, select the LDAP configuration you want to create group mapping for.
5. From the **LDAP group name** list, select the group to use to perform the mapping.

Tip: You can search for groups by name (partial matches and case-insensitive matches allowed) or distinguished name (case-insensitive matches allowed).

6. Select a Continuous Delivery for PE account associated with the RBAC group you want to map to the selected LDAP group.
7. From the list of available Continuous Delivery for PE RBAC groups, select the RBAC group you want to map to the selected LDAP group.
8. Click **Add group mapping**.

After setting up a group map, Continuous Delivery for PE synchronizes with your LDAP groups based on the mapping you created.

Configure SMTP

Configure SMTP for your Continuous Delivery for Puppet Enterprise (PE) installation so that users can receive email notifications from the software.

The root user or a super user must do this task.

1. Log into the root console by signing in as the root user or by selecting **Root console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar.
2. Click **Settings > SMTP**.
3. If your SMTP server requires authentication, enter the **SMTP username** and **SMTP password** in the relevant fields.
4. In the **Send emails from this address** field, enter the sender address you want to show on Continuous Delivery for PE emails.

Important: If this field is not specified, Continuous Delivery for PE uses the root user's email address as the sender/from address when it sends an email. In this case, the specified SMTP account must have permission to send emails from the root user's email address.

5. Enter your **SMTP host** name and the **SMTP port** number in the relevant fields. Contact your email server administrator if you need help determining the correct SMTP port.
6. If your server requires TLS authentication, toggle the **Enable TLS** switch. Only implicit TLS is supported, typically on port 465 or 25.
7. Click **Save settings**.
8. To test your new SMTP configuration, generate a password reset email:
 - a) Log out of Continuous Delivery for PE.
 - b) On the sign-in screen, click **Forgot your password?**
 - c) Enter the email address associated with your Continuous Delivery for PE account and click **Send reset instructions**.
 - d) Check your inbox for the password reset email. You can ignore the password reset instructions and keep your current password; this test just ensures Continuous Delivery for PE can send emails.

After configuring SMTP, Continuous Delivery for PE sends these emails:

- Password reset instructions when requested by a user.
- Password reset confirmation when a password is updated.
- Deployment approval request notifications when a [deployment to a protected environment](#) is proposed.

To remove an unwanted SMTP configuration, you can delete it from the **SMTP** settings page in the root console.

Set up external PostgreSQL

To use an external PostgreSQL database with Continuous Delivery for Puppet Enterprise (PE), set up the connection on the **Config** page in Puppet Application Manager (PAM).

On the **Config** page in PAM, click **Set up external PostgreSQL** and complete these fields:

- **PostgreSQL endpoint:** The hostname and port (in `hostname:port` format) used to connect to PostgreSQL.
- **CD4PE PostgreSQL username:** The username for a user authorized to connect to the CD4PE PostgreSQL database.
- **CD4PE PostgreSQL password:** The password for the CD4PE PostgreSQL username.
- **CD4PE PostgreSQL database:** The database to use with CD4PE. Defaults to `cd4pe` if unspecified.

Select the **PostgreSQL SSL mode** to use when connecting to the databases during preflight checks.

Click **Save config**.

Load balancing

The following load balancer requirements are needed for an HA install.

- A network (L4, TCP) load balancer for port 6443 across primary nodes. This is required for Kubernetes components to continue operating in the event that a node fails. The port is only accessed by the Kubernetes nodes and any admins using `kubectl`.
- A network (L4, TCP) or application (L7, HTTP/S) load balancer for ports 80, and 443 across all primaries and secondaries. This maintains access to applications in event of a node failure. Include 8800 if you want external access to the Puppet Application Manager UI.

Note: Include port 8000 for webhook callbacks if you are installing Continuous Delivery for PE.

Important: If you are using application load balancing, be aware that Ingress items use Server Name Indication (SNI) to route requests, which may require additional configuration with your load balancer. If your load balancer does not support SNI for health checks, enable **Enable load balancer HTTP health check** in the Puppet Application Manager UI **Config** page.

Advanced configuration

Advanced configuration settings for Continuous Delivery for PE help you fine-tune aspects of the software that can impact runtime and operation speed.

Adjust available memory

Adjust the amount of memory available to Continuous Delivery for Puppet Enterprise (PE) based on repository size and workflow complexity.

In some cases, the amount of memory allocated to the Continuous Delivery for PE application by default isn't sufficient. Memory consumption scales with repository size and the number of jobs running concurrently.

To adjust the amount of memory allocated to Continuous Delivery for PE:

1. Navigate to the **Config** page in Puppet Application Manager (PAM).
2. In the **Advanced configuration and tuning** section, adjust the **Memory available for CD4PE (in MiB)** setting.

Adjust the timeout period for snapshots

When creating a snapshot, Puppet Application Manager (PAM) makes backup copies of various Continuous Delivery for Puppet Enterprise (PE) components. You can change the amount of time PAM spends making each component's backup copy.

You can learn more about creating and using snapshots in the [PAM](#) documentation.

To configure timeout periods, navigate to the **Config** page in PAM, and edit these settings in the **Advanced configuration and tuning** section:

- **Object store backup timeout:** The amount of time a running snapshot spends attempting to backup the Continuous Delivery for PE object store. The default is 30 minutes.
- **Continuous Delivery for PE database backup timeout:** The amount of time a running snapshot spends attempting to backup the Continuous Delivery for PE database. The default is 10 minutes.
- **Nodes database backup timeout:** The amount of time a running snapshot spends attempting to backup the database for the **Nodes** page. The default is 10 minutes.

Improve job performance by caching Git repositories

If you have large Git repositories, you can enable Git repository caching to improve job performance. By default, repository caching is disabled.

To enable repository caching:

1. In Puppet Application Manager (PAM), navigate to the **Config** page.
2. In the **Advanced configuration and tuning** section, enable Git **Repo caching**.

The cached repository's files and data are stored on the container running Continuous Delivery for PE at /<DEFAULT_ROOT_STORAGE_DIRECTORY>/repos. The entire repository is cloned from source control, including branches; therefore, caching requires space equivalent to the size of the uncompressed repository.

Cached repositories are not automatically deleted. When attempting to read from the cached repository, if the cached version is missing object ID references, or if the previous cache attempt failed, then the cached version is deleted and re-cloned.

Including the `.git` directory in cached repositories

The `.git` directory is automatically omitted when copying cached Git repositories to job hardware. This means that the job cannot perform Git actions on the code. If needed, you can adjust this setting so that the `.git` directory is included in the cached repository.

To include the `.git` directory in copies of cached Git repositories sent to job hardware:

1. In PAM, navigate to the **Config** page.
2. In the **Advanced configuration and tuning** section, enable **Include Git history for jobs**.

Adjust the timeout period for jobs

In some circumstances, such as when working with large Git repositories, you may need to adjust the length of the job timeout period. Use the settings in this section to customize the job timeout period.

To configure timeout periods, navigate to the **Config** page in PAM, and edit these settings in the **Advanced configuration and tuning** section:

- **Repo cache retrieval timeout (minutes)**: Sets the timeout period for a thread attempting to access a cached Git repository. The default is 28 minutes.

Note: This is only available if you enabled [Git repository caching](#).

- **Job HTTP read timeout (minutes)**: Sets the timeout period for a job connecting to an endpoint. The default is 29 minutes.
- **Job global timeout (minutes)**: Sets the global default timeout period for jobs. Once this time elapses, the job fails and a timeout message is printed to the log. The default is 30 minutes.
- **Bolt PCP read timeout (seconds)**: Sets the Bolt PCP read timeout period. The default is 60 seconds.

Note: Jobs cannot proceed while file sync is running. If a file sync operation is not completed before the Bolt PCP read timeout period elapses, the job fails. Increase the Bolt PCP read timeout period to prevent these job failures.

Important: To ensure useful error messages are printed to the logs, make sure that the value of **Job global timeout** is larger than the value of **Job HTTP read timeout**, and that both are larger than the value of **Repo cache retrieval timeout**.

Configure LDAP server search result limits

By default, Continuous Delivery for Puppet Enterprise (PE) requests 500 search results at a time from a connected LDAP server. If your LDAP server's search result limitation is below 500, you can configure Continuous Delivery for PE to match the LDAP server's search result threshold.

Navigate to the **Config** page in Puppet Application Manager (PAM) and set the **LDAP group search size limit** in the **Advanced configuration and tuning** section.

Adjust HTTP timeout periods

If necessary, you can adjust the timeout periods for various HTTP operations.

To configure HTTP timeout periods, navigate to the **Config** page in PAM, and edit these settings in the **Advanced configuration and tuning** section:

- **Global HTTP connection timeout (seconds):** Sets the timeout period for all external HTTP connections. The default is 120 seconds.
- **Global HTTP read timeout (seconds):** Sets the timeout period for all external HTTP read actions. The default is 120 seconds.

Note: This value is also used as the value for the `CD4PE_MODULE_DEPLOY_READ_TIMEOUT` environment variable in deployment tasks.

- **Global HTTP write timeout (seconds):** Sets the timeout period for all external HTTP write actions. The default is 120 seconds.
- **Global HTTP request timeout (seconds):** Sets the total amount of time an external HTTP request remains open. The default is 300 seconds.

Configure login attempt limits

By default, Continuous Delivery for PE limits the number of unsuccessful login attempts users can make within a certain timeframe. If the user exceeds the allowed number of unsuccessful login attempts within that timeframe, their account is temporarily locked. You can customize login attempt limits for your installation.

To configure login attempt settings, navigate to the **Config** page in PAM, and edit these settings in the **Advanced configuration and tuning** section:

- **Max login attempts before lockout:** The number of unsuccessful login attempts a user can make before the account is locked. The default is 10 attempts.
- **Time period (minutes) to look at for failed logins:** The amount of time that must elapse before the failed login count resets. The default is 15 minutes.
- **Time period (minutes) to lock an account:** How long user accounts are locked after exceeding the number of unsuccessful login attempts within the failed login timeframe. The default is 120 minutes.

Configure data retention settings

Configures Continuous Delivery for PE to keep pipeline or value reporting data for a period of time. By default, data is retained indefinitely. When these settings are set to values other than 0, Continuous Delivery for PE runs a task on startup and every 24 hours which deletes data older than the configured retention period.

To configure data retention settings, navigate to the **Config** page in PAM, and edit these settings in the **Advanced configuration and tuning** section:

- **Number of days to retain historical pipelines data:** The number of days to retain historical pipelines data, which includes pipeline runs, deployments, jobs, job logs, and impact analysis reports. By default this is set to 0, which retains historical pipelines data indefinitely.
- **Number of days to retain historical value reporting data:** The number of days to retain historical value reporting data, which only includes the daily activity data on the **Value reporting** page. By default this is set to 0, which retains historical value reporting data indefinitely.

Use custom TLS certificates

By default, Continuous Delivery for Puppet Enterprise (PE) uses automatically generated certificates. Your organization's security policies might require using custom certificates or adding additional certificates. Use these steps to configure custom TLS certificates for the Continuous Delivery for PE web UI connection and, optionally, the Continuous Delivery for PE webhooks connections.

Before you begin

It is recommended to do this after you [Deploy Continuous Delivery for PE](#) on page 130, but you could do this during initial setup.

1. Obtain a custom certificate and accompanying key pair. You need the entire certificate, including the header and footer, and the private key. Most configurations also need a CA certificate chain.
Make sure you have configured the DNS names you want to use for Puppet Application Manager (PAM) and Continuous Delivery for PE. When you generate and sign the CSR, make sure it includes enough subject alternative names to capture both applications.
2. In PAM, go to the **Config** page for your Continuous Delivery for PE installation.
3. In the **Optional configuration** section, select **View options for certificates**, and select **Provide my own certs**.
4. In the **TLS certificate** field, paste the entire certificate, including the header and footer.
5. In the **TLS private key** field, paste the entirety of the private key corresponding with your custom certificate.
6. Enter a CA certificate chain in the **CA certificate** field so Puppet agents trust your certificate.
There are some situations where you might not need to provide the CA certificate chain. This depends on your certificate configuration.
7. To use your custom certificate for Continuous Delivery for PE webhooks, select **View options for using a proxy or external load balancer**, and then select **Use TLS for Webhook Service**. This uses the same certificate you provided above.
This option is not available if you chose **Use an Ingress with a hostname** when you [deployed Continuous Delivery for PE](#).
8. Click **Save config** > **Go to updated version**. Once preflight checks are done, deploy the updated version.
For details about preflight checks and deploying a new version, refer to [Deploy Continuous Delivery for PE](#) on page 130.
9. Optional: Use OpenSSL or curl commands to verify your certificates.

If you want to go back to using the automatically generated certificates, under **Optional configuration**, switch to **Use generated certs**.

To use custom TLS certificates for PAM, refer to [Update the PAM TLS certificate](#) on page 123.

To use a custom certificate for your Continuous Delivery for PE SAML SSO configuration, refer to [Configure SAML](#) on page 157.

Enable compiler maintenance mode

You can tell Continuous Delivery for Puppet Enterprise (PE) to skip offline or unavailable compilers and replicas when deploying code.



CAUTION: With this setting enabled, Code Manager deployments are reported as successful as long as the code was deployed to the primary server. Continuous Delivery for PE does not tell you if it skipped a compiler or replica. You are only notified if the deployment to the primary server failed.

You must manually monitor the status of your compilers and replicas to ensure they're in sync with the primary server. If a compiler or replica is out of sync, you'll need to manually deploy code to that compiler or replica.

To enable this setting:

1. In the Continuous Delivery for PE web UI, navigate to **Settings** > **Puppet Enterprise**.
2. Locate the PE instance you want to configure and click **More actions** > **Edit integration**.
3. In the **Compiler maintenance mode** section, enable **Ignore unavailable compilers or replicas when deploying code**.
4. Click **Save changes**.

Important: This setting allows your [pipelines](#) to continue while compilers are undergoing maintenance or experiencing transient issues. When enabled, Code Manager deployments report as successful as long as the code was deployed to the primary server. Therefore, you must manually track that the code on your compilers and replicas is in sync.

Use the [Code Manager API GET /v1/deloys/status endpoint](#) to make sure your compilers and replicas are in sync with the primary server. The `file-sync-client-status` portion of the response contains all servers with code synced. In the `deployed` array for each server, compare the `deploy-signature` and date for each deployment. The `deploy-signature` is the hash of the git commit that was last synced to the server. If a compiler or replica has a different hash than the primary, you must [Deploy code manually](#) on page 222 to the desynchronized compiler or replica.

Disable Vault

You can disable Vault to improve job performance. By default, Vault is enabled to allow for migrations on older Continuous Delivery for PE releases (4.19.0 and older).

Note: Continuous Delivery for PE no longer uses Vault. It is enabled by default to support older releases (4.19.0 and older). If you do not need to support an older release, please disable Vault to free the resources it uses.

To disable Vault:

1. In Puppet Application Manager (PAM), navigate to the **Config** page.
2. In the **Advanced configuration and tuning** section, disable **Enable Vault**.

Manage workspaces and access

- [Manage workspaces](#) on page 167

Use workspaces to share access to key Continuous Delivery for Puppet Enterprise (PE) resources, such as control repos, modules, and jobs, with the other members of your team. After setting up a workspace, add your team members to that workspace and give them the permissions needed to do their work.

- [Manage access](#) on page 169

The user access and management tools in Continuous Delivery for Puppet Enterprise (PE) ensure that all users, from read-only to super users, have the access and permissions appropriate to their role on the team.

Manage workspaces

Use workspaces to share access to key Continuous Delivery for Puppet Enterprise (PE) resources, such as control repos, modules, and jobs, with the other members of your team. After setting up a workspace, add your team members to that workspace and give them the permissions needed to do their work.

Best practices for creating workspaces

These suggestions and best practices are intended to help you and your organization understand Continuous Delivery for Puppet Enterprise (PE) workspaces and use them effectively.

Workspaces support teams who write and deploy Puppet code to nodes managed by PE. Whether your organization has one team writing and deploying all Puppet code, or multiple teams writing and testing Puppet code with a central deployment team pushing changes to production, workspaces help ensure each team member has the Continuous Delivery for PE resources they need.

Single team/repository recommendations

If your organization has one team writing, testing, and deploying all Puppet code or stores Puppet code in one source control repository, we recommend:

- Using one workspace for the whole team.
- Applying permissions carefully so each team member only has access to what they need.

Multiple teams/repositories recommendations

If your organization has multiple teams writing and testing Puppet code, multiple source control repositories for Puppet code, and/or a deployment team responsible for pushing Puppet code changes to production, we recommend:

- Having separate workspaces for each writing and testing team; ideally, one workspace for each control repository.
- Creating a separate workspace for the deployment team, and allow these team members to access all other workspaces.
- Applying permissions carefully so each team member only has access to what they need.

Set up a workspace

These steps explain how to create and add users to a workspace. If you need multiple workspaces, repeat this process as many times as you need.

Before you begin

Make sure all team members have created their individual Continuous Delivery for Puppet Enterprise (PE) accounts.

Note: Tell your team members to log out when they reach the **Choose a workspace** screen after creating their account.

1. Log in to the Continuous Delivery for PE web UI, and select **Manage workspaces** from the workspaces drop down menu at top of the navigation bar.
2. Click **Add new workspace** and give the workspace a descriptive name (If necessary, you can change the name later on the **Settings** page).

After creating a workspace, you are automatically set as the workspace owner. Other Continuous Delivery for PE super users can access your workspace by navigating to the root console and clicking the name of your workspace in the **Workspaces** tab.

3. Switch to your new workspace by clicking its name in the **My workspaces** list or by selecting it from the workspaces drop down menu.
4. Follow the new workspace prompts at the top of the page to set up the workspace's required resources:
 - a) If necessary, [Integrate with Puppet Enterprise](#) on page 143.
 - b) [Integrate with source control](#) on page 146.
 - c) [Configure job hardware](#) on page 154 specifically for this workspace or use global shared job hardware (If you haven't already done so, [Configure global shared job hardware](#) on page 156).
5. Click **Control repos** > **Add control repo**, and add the control repo where the team stores Puppet code.
6. Click **Settings** > **Users**, and add each team member as a user of this workspace.

Note: If a team member is missing, they have not created their Continuous Delivery for PE account yet.

7. Click **Groups** and create one or more groups. Assigning appropriate permissions and users to each group. Make sure every team member is assigned to at least one group.
If you add a user to a workspace without adding them to a group, they get a 403 error when signing in to Continuous Delivery for PE.
8. Invite the team members to sign in to Continuous Delivery for PE. They'll now see the new team workspace in the workspaces drop down menu on the Continuous Delivery for PE web UI.

According to their group permissions, team members can view and interact with the resources you added to the workspace.

To delete workspaces, navigate to **Settings** > **Workspace** > **Delete workspace**. Only super users or the workspace owner can do this.

Transfer workspace ownership

When a workspace owner changes teams, leaves your organization, or is otherwise no longer the right person to manage a workspace, you can reassign workspace ownership to a different Continuous Delivery for PE user.

To transfer workspace ownership, you must have root or super user permissions allowing you to access the root console and perform this action.

1. Log into the root console by signing in as the root user or by selecting **Root console** from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.
2. Click **Workspaces**, locate the workspace you need to reassign, and click **Reassign workspace** .
3. Select the new owner's username and click **Save changes**.

After transferring ownership, the original workspace owner is automatically removed from the workspace. The new owner can add the former owner as a workspace user by navigating to **Settings > Users**.

Manage access

The user access and management tools in Continuous Delivery for Puppet Enterprise (PE) ensure that all users, from read-only to super users, have the access and permissions appropriate to their role on the team.

Note: [Configure login attempt limits](#) on page 165 explains how to customize failed login controls for your installation.

- [Create a user account](#) on page 169

After an administrator installs Continuous Delivery for Puppet Enterprise (PE), each person who needs to use the software must create an individual user account.

- [Add users and groups](#) on page 170

Continuous Delivery for Puppet Enterprise (PE) facilitates collaboration between team members. You can share resources by adding users to your workspace, and you can set up groups to manage permissions granted to each team member.

- [Permissions reference](#) on page 171

This table lists Continuous Delivery for Puppet Enterprise (PE) group permissions and a short explanation of each permission's scope.

- [Use the root console](#) on page 173

Super users and the root user can access the Continuous Delivery for Puppet Enterprise (PE) root console to manage users' account credentials, change super user permissions, configure single sign-on, access all workspaces associated with the installation, transfer workspace ownership, and update the installation's settings.

- [Manage personal access tokens](#) on page 175

Authentication tokens allow a user to enter their credentials once, then receive an alphanumeric token to use to access different services or parts of the system infrastructure.

Create a user account

After an administrator installs Continuous Delivery for Puppet Enterprise (PE), each person who needs to use the software must create an individual user account.

Before you begin

Obtain the Continuous Delivery for PE web UI endpoint from the administrator who installed the software.

1. Point your browser to the Continuous Delivery for PE web UI endpoint.
2. On the login page, click **Create an account**.
3. Fill in the registration form and create a username and password.
4. Click **Create account**.

You now have a Continuous Delivery for PE account. You can view your account credentials by clicking your username at the bottom of the navigation bar in the Continuous Delivery for PE web UI.

Add users and groups

Continuous Delivery for Puppet Enterprise (PE) facilitates collaboration between team members. You can share resources by adding users to your workspace, and you can set up groups to manage permissions granted to each team member.

Add a user to a workspace

Adding users to your workspace allows you to collaborate on projects and share account resources such as jobs and pipelines. You can add users in the **Settings** area of the Continuous Delivery for PE web UI.

Before you begin

Make sure each user you wish to add has signed up for a Continuous Delivery for PE account. You'll need their selected username or the email address they used when signing up.

1. In the Continuous Delivery for PE web UI, click **Settings** then click **Users**.
2. Click **Add a user**.
3. On the **Add users to workspace** page, search for the username or email address of the user you wish to add to your workspace.
4. Select the user you wish to add to the workspace.
5. **For version 4.4.0 and newer versions:** Assign the user to one or more permission groups.

Note: Beginning in Continuous Delivery for PE version 4.4.0, newly created workspaces include three built-in user groups: Administrators, Operators, and Viewers. For details on the permissions included in each built-in group, see the [Permissions reference](#).

6. Click **Add users to workspace**. Repeat these steps to add additional users.
7. When you're finished adding users, click **Done** to return to the **Users** screen, where you'll see the full list of users with access to your Continuous Delivery for PE workspace.

Create a group and set group permissions

Groups to organize users by permission level, job focus, geography, or any other criteria you choose. Permissions define the tasks that group members can perform in Continuous Delivery for Puppet Enterprise (PE), such as adding control repos, editing jobs, and deploying code changes. Assign permissions to groups based on each group's functions and needs.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. On the **Groups** tab, click **Create new group**.
3. Enter a name and description for the new group.

Tip: Choose a name that describes the group's purpose, role, or permission level, such as `Read-only users` or `Module developers`.



CAUTION: You can't edit group names or descriptions after this point. If you need to change the name or description, you must delete the group and recreate it.

4. Select permissions to assign to this group. For explanations of each permission's scope, see the [Permissions reference](#) on page 171. Group permissions are additive; therefore, if a user belongs to multiple groups, that user can perform all actions allowed by all permissions from all of their assigned groups.

To assign permissions on a certain subset of the control repos or modules in your workspace:

a)

In the permissions section, click **Edit permissions**  next to **Control Repos subset** or **Modules subset**.

b) Select the repos or modules to include in the subset and click **Save selection**.

c) Select permissions to give the user group. You can allow list (view), edit, and delete permissions for all control repos/modules in the workspace or only on the selected subset.

If you create a subset and do not assign any permissions on the subset, the subset is discarded when you save your permission selections.

5. Click **Save and add users**.
6. On the **Add users** page, select workspace members to add to this group, and click **Add users to group**.
7. Click **Done**. Continuous Delivery for PE creates your user group and redirects you to the group page.

Review the group members and permissions on the group page, and make any changes, if needed.

To delete a group, first remove all members from the group, and then click **Delete group** .

Remove users from workspaces

You can remove users from Continuous Delivery for PE workspaces in the **Settings** area of the Continuous Delivery for PE web UI.

Note: Removing a user from a Continuous Delivery for PE workspace automatically removes the user from all groups in that workspace.

1. Log in to the Continuous Delivery for PE web UI and make sure you are viewing the workspace you want to remove the user from.
2. Click **Settings > Users**.
3. Locate the user you want to remove from the workspace.
4. Click **Delete user**  and confirm your decision.
5. Repeat these steps to remove additional users from the workspace.

Permissions reference

This table lists Continuous Delivery for Puppet Enterprise (PE) group permissions and a short explanation of each permission's scope.

Type	Permission	Definition
Control repos	Create	The ability to create new control repos in Continuous Delivery for PE.
Control repos	Delete	The ability to delete existing control repos.
Control repos	Edit	The ability to edit existing control repos and deploy code.
Control repos	List	The ability to view existing control repos.
Control repos	Promote	The ability to promote changes in control repo pipelines.

Type	Permission	Definition
Modules	Create	The ability to create new module repos in Continuous Delivery for PE.
Modules	Delete	The ability to delete existing module repos.
Modules	Edit	The ability to edit existing module repos.
Modules	List	The ability to view existing module repos.
Modules	Promote	The ability to promote changes in module pipelines.
Jobs	Create	The ability to create new jobs in Continuous Delivery for PE.
Jobs	Delete	The ability to delete existing jobs.
Jobs	Edit	The ability to edit existing jobs.
Jobs	List	The ability to view all existing jobs and job hardware capabilities.
Jobs	Run	The ability to run all existing jobs.
Users	Edit	The ability to add and remove users from Continuous Delivery for PE workspaces.
Groups	Create	The ability to create new user groups.
Groups	Delete	The ability to delete existing user groups.
Groups	Edit	The ability to edit existing user groups by adding or removing members and user permissions.
Groups	List	The ability to view all existing user groups and their permissions.
Integrations	Connect	The ability to configure new source control or PE integrations.
Integrations	Disconnect	The ability to remove existing source control or PE integrations.
Inventory	List	The ability to view all node inventory information.

Built-in user group permissions reference

New workspaces in Continuous Delivery for PE versions 4.4.0 and newer include three built-in user groups. These tables list the permissions granted to each built-in group.

Administrators group permissions

Type	Permissions granted
Control repos	Create, list, edit, promote, delete
Modules	Create, list, edit, promote, delete

Type	Permissions granted
Jobs	Create, list , edit, run, delete
Users	None
Groups	Create, list, edit, delete
Integrations	Connect, disconnect
Inventory	List

Operators group permissions

Type	Permissions granted
Control repos	Create, list, promote
Modules	Create, list, promote
Jobs	Create, list
Users	None
Groups	None
Integrations	None
Inventory	None

Viewers group permissions

Type	Permissions granted
Control repos	List
Modules	List
Jobs	List
Users	None
Groups	None
Integrations	None
Inventory	None

Use the root console

Super users and the root user can access the Continuous Delivery for Puppet Enterprise (PE) root console to manage users' account credentials, change super user permissions, configure single sign-on, access all workspaces associated with the installation, transfer workspace ownership, and update the installation's settings.

To access the root console as the root user, sign into Continuous Delivery for PE with the root account credentials established during installation. As a super user, select **Root console** from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.

To exit the root console, click any workspace name.

Designate super users

Any Continuous Delivery for Puppet Enterprise (PE) user except the root user can be designated as a super user. Super users can access the root console to reset other users' credentials, delete users, manage global shared job hardware, and designate the users allowed to manually approve deployments to protected Puppet environments.

To designate super users, you must have root or super user permissions to access the root console.

1. Log into the root console by signing in as the root user or by selecting **Root console** from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.
2. Click **Accounts**.
3. Locate the user's name in the list, enable the **Super User** switch, and confirm the change.
4. Repeat to designate additional super users.

Change a user's password

Users can reset their own passwords at any time by clicking **Forgot your password?** on the Continuous Delivery for Puppet Enterprise (PE) login screen. If you need to update the root user's password, change a user's password on their behalf, or revoke a user's access to Continuous Delivery for PE, use the root console.

To perform these actions, you must have root or super user permissions to access the root console.



CAUTION: Resetting a user's password does not automatically log the user out of Continuous Delivery for PE. If you reset a user's password to revoke their access to Continuous Delivery for PE, the user can still access their account until their session expires or they log out. You also need to disable the user's corporate email account or [Reset the user's email address](#) to prevent the user from manually resetting their password through the **Forgot your password?** link.

1. Log into the root console by signing in as the root user or by selecting **Root console** from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.
2. Click **Accounts**.
3. Locate the user whose password you want to reset and click **User settings** .
4. Click **Change password**, enter and confirm the new password, and click **Change password**.

Reset a user's email address

Users can change their own email address at any time by clicking **Change email** on the **Profile** page. If you need to change a user's email address on their behalf, use the root console.

You must have root or super user permissions to access the root console and perform this action.

1. Log into the root console by signing in as the root user or by selecting **Root console** from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.
2. Click **Accounts**.
3. Locate the user whose email address you want to change and click **User settings** .
4. Enter the user's new email address and click **Change email**.

Delete a user

Use the root console to permanently delete a user from Continuous Delivery for Puppet Enterprise (PE).

You must have root or super user permissions to access the root console and perform this action.

1. Log into the root console signing in as the root user or by selecting **Root console** from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.
2. Click **Accounts**.
3. Locate the user you want to delete and click **Delete user** .
4. Confirm that you want to permanently delete the user by entering the user's user name and clicking **Yes, delete user**.

The user is permanently deleted from your Continuous Delivery for PE installation.

Manage personal access tokens

Authentication tokens allow a user to enter their credentials once, then receive an alphanumeric token to use to access different services or parts of the system infrastructure.

Authentication tokens are tied to the permissions granted to the user through role-based access control (RBAC), and they provide the user with the appropriate access to application programming interfaces (APIs). You can use tokens in API endpoint requests.

You can manage authentication tokens using the **Profile** page.

Add a personal access token

Add a personal access token from the **Profile** page.

This task walks you through creating a personal access token that is required to access the Continuous Delivery for Puppet Enterprise (PE) API functionality. Be sure to copy the token and save it to a secure location. The token is required to [authenticate the Continuous Delivery for PE public API](#).

1. Go to the **Profile** page.
2. Enter the name of the token you would like to add in the **NAME** field.
3. Specify an expiration date for the token in the **EXPIRES AT** field by typing the date or selecting it from the calendar. You can set a maximum time for the token expiration of 1 year. By default the token expires in 90 days.
4. Click **Create personal access token** to create the token.



CAUTION: After you create a token a window with the personal access token appears. Copy and save it to a secure location as **this is the only time it is accessible**.

5. When you are done copying and saving the token, click **Close**.

The new token appears in the **Active personal access tokens** table.

Revoke a personal access token

Revoke a personal access token from the **Profile** page.

There may be times when you need to remove a personal access token before it expires. Use this process to revoke a personal access token.

1. Go to the **Profile** page. The list of tokens appears in the **Active personal access tokens** table.
2. Click the **Revoke** button next to the token you would like to remove.



CAUTION: Revoking a personal access token is permanent, you cannot undo this action.

3. Select the **Yes** check box from the window that appears to confirm you wish to revoke the token.
4. Click **Revoke** to revoke the token.

REST API

Fetch data and automate your workflows with the Continuous Delivery for Puppet Enterprise (PE) REST API. Use the API to promote pipelines based on your own criteria and automate workspace setup to make onboarding new users more streamlined.

The Continuous Delivery for PE REST API uses the OpenAPI standard to define and document the API via a specification. The OpenAPI Initiative has extensive documentation of the standard which can be followed. Continuous Delivery for PE uses [version 3.0.1](#) of the standard.

If you are familiar with REST and OpenAPI, skip to the [OpenAPI specification](#) section for more specific details on how Continuous Delivery for PE implements OpenAPI.

- [Authenticate public APIs](#) on page 176

The Continuous Delivery for Puppet Enterprise (PE) API supports authentication via personal access tokens associated with Continuous Delivery for PE accounts. Authentication tokens are tied to the permissions granted to the user through role-based access control (RBAC), and they provide the user with the appropriate access to application programming interfaces (APIs).

- [REST API tutorial](#) on page 176

Learn how to navigate a Continuous Delivery for Puppet Enterprise (PE) workflow entirely through the API using curl and bash.

- [OpenAPI specification](#) on page 177

Continuous Delivery for Puppet Enterprise (PE) includes the OpenAPI specification and it can be fetched from the web server from the following endpoint: `curl --request GET "https://<hostname>/cd4pe/api/openapi.yaml"`

- [Common API workflows](#) on page 178

Continuous Delivery for Puppet Enterprise's API enables users to automate many common workflows.

Authenticate public APIs

The Continuous Delivery for Puppet Enterprise (PE) API supports authentication via personal access tokens associated with Continuous Delivery for PE accounts. Authentication tokens are tied to the permissions granted to the user through role-based access control (RBAC), and they provide the user with the appropriate access to application programming interfaces (APIs).

Before you begin

You need the personal access token you created in [Add a personal access token](#) on page 175. It is required for authentication.

Using personal access token to authenticate the Continuous Delivery for PE public API.

Add the personal access token you copied when creating the token and add this to the Authorization header of the API call. For example:

```
curl --insecure --header \
  "Authorization: <token value>" \
  "https://<hostname>/cd4pe/api/v1/user"
```

REST API tutorial

Learn how to navigate a Continuous Delivery for Puppet Enterprise (PE) workflow entirely through the API using curl and bash.

Before you begin

Make sure that you have read the [Authentication](#) section and have a personal access token. You need the personal access token you created in [Add a personal access token](#) on page 175. It is required for authentication.

This tutorial guides you through a Continuous Delivery for PE workflow using the API.

1. Create a workspace. From the command line run:

```
curl --insecure --header "Authorization: <token value>" --request
  POST "https://<hostname>/cd4pe/api/v1/workspaces" --data '{ "name":
  "my_workspace" }'
```

This returns: { "domain": "d3", "name": "my_workspace" }

The application returns a 200 HTTP status code along with a JSON response body which contains the created workspace information. To perform subsequent API calls you need to extract the workspace ID from the response (referred to as the *domain* in the v1 create workspaces endpoint).

Any authenticated user can create workspaces in Continuous Delivery for PE, which means that no RBAC privileges are required for this step.

2. Use the API to integrate a VCS provider in the workspace. This allows you to create code projects from the VCS provider. For example, connect a GitLab instance to the workspace:

```
curl --insecure --header "Authorization: <token value>" --request POST
  "https://<hostname>/cd4pe/api/v1/vcs/gitlab?workspaceId=d3" --data
  '{ "token": "<gitlab_token>", "host": "https://mygitlab.example.net",
  "caCertificate": "<ca_cert_data>" , httpClone: true }'
```

This returns: { "id": "<uuid>", "host": "https://mygitlab.example.net",
 httpClone: true }

Note: The workspace ID is used in the query parameter of the request. Many entities in the application exist within a workspace, the workspace ID identifies which workspace to create the Gitlab integration in.

Once the VCS provider is integrated, you can use the API to create code projects and pipelines within those projects. See the next section on the [OpenAPI specification](#) on page 177 for more information about the OpenAPI specification file as much of the API documentation is within this file.

OpenAPI specification

Continuous Delivery for Puppet Enterprise (PE) includes the OpenAPI specification and it can be fetched from the web server from the following endpoint: `curl --request GET "https://<hostname>/cd4pe/api/openapi.yaml"`

Continuous Delivery for PE-specific metadata

While the OpenAPI specification adheres to a specific standard schema, it is very extensible and allows API authors to define metadata on any object in the API.

Note: The `x-internal` parameter is set to `true` on API operations to indicate that the endpoint is an internal API meant to be used by the UI. This tag is not intended for customer use. The parameter is set either on a path or specific HTTP verb, for endpoints where some operations are allowed but not others. For more information on this tag and others, please refer to the [tags array documentation](#).

Please refer to the OpenAPI specification when learning how to use the API. The file is quite large and so it may be helpful to use one of the many [available tools](#) to visualize the specification.

Version and compatibility

All endpoints in the Continuous Delivery for PE are versioned so that updates do not break existing workflows. When an endpoint is removed from the Continuous Delivery for PE API, a deprecation warning is issued at least 2 minor releases before the endpoint is removed. The [deprecated field in the OpenAPI specification](#) indicates what operation or component is deprecated.

An API breaking change is any change to the API that would cause a caller to begin failing, such as:

- Removing an endpoint without giving a deprecation warning.
- Adding a new required field in a request body, query parameter, or response body.
- Removing a field from a request body, query parameter, or response body.
- Changing the type of a field in a request body, query parameter, or response body.

Changes that are not considered to be API breaking:

- Adding a new optional field in a request body, query parameter, or response body.
- Making a required field optional.

Common API workflows

Continuous Delivery for Puppet Enterprise's API enables users to automate many common workflows.

Get your authorization token

You need a personal access token to automate common workflows using the API.

Make sure that you have read the [Authentication](#) section and have a personal access token. You need the personal access token you created in [Add a personal access token](#) on page 175. It is required for authentication.

Find a workspace ID

When using most API endpoints, you need to specify a workspace ID.

Find the workspaces you have access to by using the GET `/v1/user` endpoint. The *domain* of each workspace is its ID.

```
auth_header="Authorization: <API Token>"
uri="https://<CD4PE hostname>/cd4pe/api/v1/user"
curl --insecure --header "$auth_header" --request GET "$uri"
```

This returns information about your workspaces. For example, it might look similar to the following:

```
{
  "user" : { ... },
  "workspaces" : [
    {
      "domain" : "d6",
      "name" : "example_workspace_1"
    },
    {
      "domain" : "d7",
      "name" : "example_workspace_2"
    }
  ]
}
```

Use the workspace ID as needed for API endpoints, such as finding your Puppet Enterprise environment node group ID, triggering an impact analysis, and triggering a deployment.

Find a Puppet Enterprise environment node group ID

Find the ID of the environment node group in Puppet Enterprise using the ID of your Puppet Enterprise integration.

Before you begin

You need to have your [workspace ID](#) obtained in the previous task.

1. Find the ID of your Puppet Enterprise integration.

```
auth_header="Authorization: <API Token>"
uri="https://<CD4PE host>/cd4pe/api/v2/pe-integrations?
workspaceId=<workspace id>"
curl --insecure --header "$auth_header" --request GET "$uri"
```

This returns information about your Puppet Enterprise integrations. For example, it might look similar to the following:

```
{
  "peIntegrations" : [
    {
      "id" : "e274e692-8423-48b6-a1f3-6e09138624ec",
      "workspaceId" : "d5",
      "name" : "example-pe",
      "nodeClassifierUrl" : "puppet.example.com:4433",
      ...
    }
  ]
}
```

2. Use the ID from your Puppet Enterprise integration to get information about the environments:

```
auth_header="Authorization: <API Token>"
uri="https://<CD4PE host>/cd4pe/api/v2/pe-integrations/<PE integration
ID>/environments"
curl --insecure --header "$auth_header" --request GET "$uri"
```

This returns information about the environments associated with your Puppet Enterprise integration. For example, it might look similar to the following:

```
"environments" : [
  {
    "name" : "All Environments",
    "id" : "00e0c92e-30c9-4569-bcc2-0a4ebefeffe3a3",
    "description" : "Environment group parent and default",
    "environment" : "production"
  },
  {
    "name" : "Development environment",
    "id" : "c2184ce4-cb96-49eb-9724-4618db267769",
    "description" : "Development nodes",
    "environment" : "development"
  }
]
```

Use the environment node group ID as needed for API endpoints, such as triggering a deployment.

Get the SHA of a repo branch

Find the SHA of the HEAD of a given repo branch using Continuous Delivery for PE's `/v1/vcs/branches` API endpoint.

To find the SHA of the HEAD of a given repo branch, you can either find the commit in your VCS provider or you can use Continuous Delivery for PE's `/v1/vcs/branches` API endpoint. Here, *project* is used to refer to the owner/org of the repo for all providers except GitHub. If your repo comes from GitHub, you must supply either *organization* or *owner* instead, depending on whether the repo belongs to an individual or an organization.

Use your [workspace ID](#) to find the SHA.

```
auth_header="Authorization: <API Token>"
```

```
uri="https://<CD4PE host>/cd4pe/api/v1/vcs/branch?workspaceId=<workspace ID>&provider=BITBUCKET&project=puppet&name=my_control_repo&branch=main
curl --insecure --header "$auth_header" --request GET "$uri"
```

This returns information about the repo branch. For example, it might look similar to the following:

```
{
  "name" : "main",
  "headSha" : "4c2f400a77cd73ebf67271361e1dd9b1102b43b4"
}
```

Set up HTTPS access for ADO via OAuth

Configure the existing ADO Cloud OAuth integration to use HTTPS cloning instead of SSH via the REST API.

Before you begin

To configure the existing ADO Cloud OAuth integration to use HTTPS cloning instead of SSH, you must be on at least Continuous Delivery for PE 4.28.0 or later. This is only available using the API, which uses the OAuth token already stored for auth. After upgrading to version 4.28.0 or later, to switch to HTTPS cloning use curl from a host that can resolve the Continuous Delivery for PE console:

```
curl -v -X PATCH https://<CD4PE host>/cd4pe/api/v1/vcs/oauth/azure-devops
-H "Authorization: <token from super user>" --cacert <console cert> -H
"Content-Type: application/json" -d '{"httpClone": true}'
```

See [Add a personal access token](#) for instructions on how to get the Authorization token. This token must be from root or a superuser, so you must be logged in as a user with superuser privileges (this can be root or some other user) to create a token to use with this request.

Note that no app restart is required and there is no need to recreate control repos or modules. It is also worth noting that this setting is persistent, so once the HTTPS cloning value is set to true, it remains until changed.

Trigger an Impact Analysis

Use the Continuous Delivery for PE API to trigger an impact analysis on the module of a control repo.

Before you begin

In order to trigger an Impact Analysis run of your code changes, you need to collect the following:

- The [ID of the workspace](#) containing the module of control repo you want to test
- The name of the module or control repo being analyzed
- The [ID of the Puppet Enterprise node group for each environment](#) you want to see the impact on
- The branch and [SHA](#) of the code you want to test the impact of

```
data="$(cat <<EOF
{
  "workspaceId": "<workspace ID>",
  "projectName": "control-repo",
  "projectType": "CONTROL_REPO",
  "settings": {
    "environments": [
      {
        "nodeGroupId": "<environment node group ID>",
        "peCredentialsId": {
          "name": "example-pe",
          "domain": "<workspace ID>"
        }
      }
    ]
  }
},
{
```

```

        "nodeGroupId": "defcea94-e7e5-4fe3-b971-1acd44785695",
        "peCredentialsId": {
          "name": "example-pe",
          "domain": "<workspace ID>"
        }
      },
      ],
      "sourceBranch": "<branch to test>",
      "sourceSha": "<SHA to test>",
      "analyzeAllDeployments": false,
      "compileBatchSize": 10,
      "controlRepoId": {
        "name": "control-repo",
        "domain": "<workspace ID>"
      }
    }
  }
}
EOF
)"
type_header='Content-Type: application/json'
auth_header="Authorization: <authorization token>"
uri="https://<cd4pe host>/cd4pe/api/v1/impact-analysis/trigger"
curl --insecure --header "$type_header" --header "$auth_header" --request
  POST "$uri" --data "$data"

```

This returns something similar to the following:

```

{
  "appEventId" : 18900,
  "baseTaskUrl" : "https://<cd4pe host>/cd4pe/root/deployments",
  "domain" : "d5",
  "id" : 110,
  ..data repeated from request...
  "state" : "QUEUED"
}

```

Once the impact analysis has finished, you might want to check the state of the run and fetch the analysis report, to do so use the ID of the response that was returned like so:

```

type_header='Content-Type: application/json'
auth_header="Authorization: <API token>"
uri="https://<cd4pe host>/cd4pe/api/v1/impact-analysis/<IA run ID>/csv?
workspaceId=<workspace ID>"
curl --insecure --header "$type_header" --header "$auth_header" --request
  GET "$uri"

```

Trigger a deployment

Use the Continuous Delivery for PE API to trigger a deployment from one of your branches to a Puppet Enterprise environment.

Before you begin

In order to trigger a deployment, you need to collect the following:

- The [ID of the workspace](#) containing the module of control repo you want to deploy from
- The [ID of the Puppet Enterprise node group for each environment](#) you want to deploy to
- The source and target branch of the repo and the [SHA](#) of the source branch that you want to deploy
- The details of the deployment policy you want to use

Note: If the deployment targets a protected environment, it does not run without approval. An approval request is created in the Continuous Delivery for PE console that must be approved to trigger the code to be deployed to the environment.

1. Get the deployment policy details. A deployment policy describes how you want the code from your source branch to be deployed to your Puppet Enterprise instance. You can list all the policies available in your workspace using:

```
auth_header="Authorization: <API token>"
uri="https://<CD4PE host>/cd4pe/api/v1/deployments/policies?
workspaceId=<workspace
ID>&projectName=my_control_repo&projectType=CONTROL_REPO"
curl --insecure --header "$type_header" --header "$auth_header" --request
GET "$uri"
```

This returns information about the policies in your workspace. For example, it might look similar to the following:

```
{
  "deploymentPolicies" : [
    {
      "custom" : false,
      "displayName" : "Direct deployment policy",
      "name" : "cd4pe_deployments::direct",
      "parameters" : [
        {
          "name" : "max_node_failure",
          "type" : "Optional[Integer]",
          "value" : null
        },
        {
          "name" : "noop",
          "type" : "Boolean",
          "value" : false
        },
        {
          "name" : "fail_if_no_nodes",
          "type" : "Boolean",
          "value" : true
        }
      ]
    },
    ....
  ]
}
```

From this you can choose which policy you want to use for this deployment and supply it to the deployment request, customizing the values of any parameters that you want to change.

2. Trigger the deployment. Once you have collected all the information you need, you can trigger the deployment.

```

data="$(cat <<EOF
{
  "workspaceId": "<workspace ID>",
  "projectName": "<name of control repo or module to deploy from>",
  "projectType": "CONTROL_REPO",
  "displayName": "Deployment to production on example-pe",
  "description": "Run Production Deployment from main",
  "maxRuntime": 3600000,
  "deploymentPolicy": {
    "custom": false,
    "displayName": "Direct deployment policy",
    "name": "cd4pe_deployments::direct",
    "parameters": [
      {
        "name" : "max_node_failure",
        "type" : "Optional[Integer]",
        "value" : 1
      },
      {
        "name" : "noop",
        "type" : "Boolean",
        "value" : false
      },
      {
        "name" : "fail_if_no_nodes",
        "type" : "Boolean",
        "value" : true
      }
    ],
    "controlRepoId": {
      "domain": "<workspace ID>",
      "name": "<project name>"
    }
  },
  "deploymentTarget": {
    "type": "NODE_GROUP",
    "nodeGroupId": "<ID of the enviornment node group to deploy to>",
    "environmentPrefix": ""
  },
  "puppetEnterpriseServer": "example-pe",
  "repoTargetBranch": "<branch to deploy to>",
  "repoSourceBranch": "<branch to deploy from>",
  "repoSourceSha": "<SHA of HEAD of branch>"
}
EOF
)"
type_header='Content-Type: application/json'
auth_header="Authorization: <API token>"
uri="https://<CD4PE host>/cd4pe/api/v1/deployments/trigger"
curl --insecure --header "$type_header" --header "$auth_header" --request
  POST "$uri" --data "$data"

```

Trigger a pipeline

Use the Continuous Delivery for PE API to trigger a pipeline from one of your branches to a Puppet Enterprise environment.

Before you begin

In order to trigger a pipeline, you need to collect the following:

- The [ID of the workspace](#) containing the pipeline you want to trigger
- The name of the control repo or module whose pipeline you want to trigger
- The ID of the pipeline belonging to that code project
- The [SHA](#) with which you want to trigger the pipeline

1. Get the pipeline ID, which are returned in the details for control repos and modules. You can find this using:

```
auth_header="Authorization: <API token>"
uri="https://<CD4PE host>/cd4pe/api/v1/<control-repos OR modules>/<repo name>?workspaceId=d5"
curl --insecure --header "$auth_header" --request GET "$uri"
```

This returns information about the control repo. For example, it might look similar to the following:

```
{
  "name": "my_control_repo",
  "srcRepoProvider": "BITBUCKET",
  "srcRepoOwner": "{a8b34a31-8814-4c5d-aca1-580ee518c290}",
  "srcRepoName": "my_control_repo",
  "srcRepoId": "{a8b34a31-8814-4c5d-aca1-580ee518c290}",
  "srcRepoDisplayName": "my_control_repo",
  "pipelines": [
    {
      "name": "regex",
      "pipelineId": "e64kq33s04kh05pbxfr6w8e9y"
    },
    {
      "name": "main",
      "pipelineId": "1jqloxhyphh0w03r8h82i996tr"
    }
  ],
  ...
}
```

From this you can choose which pipeline you want to trigger.

2. Trigger the pipeline. Once you have collected all the information you need, you can trigger the pipeline.

```
data="$(cat <<EOF
{
  "workspaceId": "<workspace ID>",
  "projectName": "<name of control repo or module>",
  "sha": "<SHA to run against>"
}
EOF
)"
type_header='Content-Type: application/json'
auth_header="Authorization: <API token>"
uri="https://<cd4pe host>/cd4pe/api/v1/pipelines/<pipeline ID>/trigger"
curl --insecure --header "$type_header" --header "$auth_header" --request
POST "$uri" --data "$data"
```

Test and deploy Puppet code

- [Test Puppet code with jobs](#) on page 185

Jobs are fully customizable tests for your Puppet code. You can create a job that runs any sort of test you want, including module validation, linting, and more.

- [Analyze the impact of code changes](#) on page 192

Impact analysis is a Continuous Delivery for Puppet Enterprise (PE) tool that shows you the potential impact that new Puppet code can have on your PE-managed infrastructure, without actually merging the new code. You can generate impact analysis reports on demand for commits in your control/module repos, and, if you add impact analysis to a repo's pipeline, Continuous Delivery for PE automatically generates a report on every proposed code change to that repo.

- [Construct pipelines](#) on page 199

Pipelines drive the continuous aspect of Continuous Delivery for Puppet Enterprise (PE). Constructing a pipeline involves defining work that must happen to ensure every new line of Puppet code is ready for deployment. Once your pipeline is set up, this work happens automatically each time the pipeline is triggered.

- [Deploy Puppet code](#) on page 215

Use deployment policies to control how and when code changes are deployed. You can use built-in deployment policies or create your own custom policies. To apply a deployment policy and deploy code, you can either manually deploy code or use pipelines to automate deployments.

Test Puppet code with jobs

Jobs are fully customizable tests for your Puppet code. You can create a job that runs any sort of test you want, including module validation, linting, and more.

Important: Before adding jobs to pipelines, make sure you [Configure job hardware](#) on page 153 and install any additional software, such as Docker or Puppet Development Kit (PDK).

What is a job?

In Continuous Delivery for Puppet Enterprise (PE), jobs test your Puppet code. You create and store jobs in the web UI, where you can run them on demand and add them to automated pipelines to run tests every time new code is committed to a repository.

Jobs are fully customizable and can be written to test any aspect of your code. Jobs can leverage Puppet Development Kit (PDK) and other testing tools created and maintained by the Puppet community, such as:

- [rspec-puppet](#)
- [onceover](#)
- [puppet-lint](#)
- [rspec-puppet-facts](#)

Where do jobs run?

Jobs run on designated nodes called job hardware. There are two job hardware types:

- **Global shared job hardware:** These are job hardware servers configured in the root console and available to all workspaces in your Continuous Delivery for PE installation.
- **Workspace hardware:** These are job hardware servers configured in a specific workspace and available to members of that workspace, but not available to other workspaces.

For more information about setting up job hardware, go to [Configure job hardware](#).

Important: Be aware of any prerequisites or dependencies needed to run a job, and ensure your job hardware has the necessary software, operating systems, and other resources installed before attempting to run the job.

Global shared job hardware uses a shared Docker image set in the root console. The default image used for Docker-based jobs is [puppet/puppet-dev-tools:4.x](#). You can find details on the available commands in the image documentation.

When configuring a job to run on workspace hardware, you can use the shared Docker image or environment variables in the job to overwrite the default image. However:

Important: If your installation specifies a Docker image in Puppet Application Manager (PAM) (this is common in offline environments), this image takes precedence over the image set in the root console (which is the one shown when configuring jobs).

Related information

[Improve job performance by caching Git repositories](#) on page 163

If you have large Git repositories, you can enable Git repository caching to improve job performance. By default, repository caching is disabled.

[Adjust the timeout period for jobs](#) on page 164

In some circumstances, such as when working with large Git repositories, you may need to adjust the length of the job timeout period. Use the settings in this section to customize the job timeout period.

Add secrets to jobs

You can add secrets, such as tokens and certificates, to Continuous Delivery for Puppet Enterprise (PE) jobs. Jobs can use secrets when they run.

Before you begin

You must create a job before you can add a secret. This could be one of the [Pre-built jobs](#) or a custom job, as demonstrated in the [Sample non-Docker-based module jobs](#) on page 189 and [Sample non-Docker-based control repo jobs](#) on page 190.

Important: Jobs with secrets require `puppetlabs-cd4pe_jobs` module version 1.6.0 or higher. You install this module when you [Configure job hardware](#) on page 153.

Secrets are limited to the job where you create them. If a secret is applicable to multiple jobs, you must add that secret to each relevant job.

When you run jobs that use secrets, you can see where jobs use secrets in the jobs' logs. Sensitive values are redacted.

1. In the Continuous Delivery for PE web UI, click **Jobs**.

2.

Locate the job you want to add a secret to and click **Edit** .

3. Click **Add secret**.

4. Enter a **Name** and (optional) **Description** for the secret.

Secret names must be unique within each workspace and can only contain letters, numbers, and underscores. If you use a dash or space in a secret name, it is automatically converted to an underscore.

Important: Once you save a secret, you can't change its name. If you need to change a secret's name, you must delete and recreate it.

5. Select the **Secret Type**, complete the remaining fields according to the secret type, and click **Save**.

Once you save the secret, you can't see the information you put in these sensitive fields when you edit the secret.

6. Continuous Delivery for PE stores secrets at environment variables. After saving a secret, Continuous Delivery for PE shows you the environment variable you can use to reference the secret in the code in the **Job commands** section.

Example of partial job command code without a secret variable:

```
curl -H "Authorization: Bearer <TOKEN>" -k -s
```

Example of partial job command code with a secret variable:

```
curl -H "Authorization: Bearer $CD4PE_SECRET_<SECRET_NAME>" -k -s
```

Secrets with multiple values, such as username and password combinations, generate multiple environment variables. Make sure you use the correct environment variables at the correct locations in your code. You can click the secret's name on the **Edit job** page to see the secret's environment variable(s) at any time.

7. Click **Save changes** when you are done adding secrets to the job and customizing the **Job commands** code.

Using a custom image to test Puppet 8

The `puppet/puppet-dev-tools:4.x` image that jobs use by default does not support testing modules with Puppet 8. To test with Puppet 8, create a custom job that uses the `puppet/puppet-dev-tools:puppet8` image. This image contains PDK 3, Puppet 8, and Onceover 3.22. See the [puppet-dev-tools readme](#) for the full list of commands provided by the image.

To create a custom job using the Puppet 8 image:

1. In the Continuous Delivery for Puppet Enterprise (PE) web UI, click **Jobs**.
2. Click **New Job**.
3. Enter your job's info, any commands you want to run, and any secrets you need.
4. Select **Run on <workspace> hardware** and the **Docker** hardware capability.
5. Toggle the switch to run this job in a Docker container and select **Custom image**.
6. Specify `puppet/puppet-dev-tools:puppet8` as the custom image name.
7. Click **Save Job**.

This job is now available to add to the pipelines where you need to test code against Puppet 8.

Pre-built job reference

Use these job templates to set up Docker-based jobs for testing control repos and modules. These jobs run inside a Docker container and require job hardware that has a Docker capability.

Validate the syntax of a control repo's Puppetfile

To add this job to your Continuous Delivery for Puppet Enterprise (PE) instance:

1. In the Continuous Delivery for PE web UI, click **Jobs > New job**.
2. Complete the fields:
 - a. **Name:** `control-repo-puppetfile-syntax-validate`
 - b. **Description:** Validate that a control repo's Puppetfile is syntactically correct
 - c. **Job commands:** `rake -f /Rakefile r10k:syntax`
 - d. **Where can this job run?:** Select **Run on <WORKSPACE_NAME> hardware**
 - e. **Hardware capabilities:** Select **Docker** and click **Apply**
 - f. **Docker configuration:** Enable the **Run this job in a Docker container** switch and select **Default image** (`puppet/puppet-dev-tools`)
3. Click **Save job**.

[Configure job hardware](#) on page 154 and ensure Docker is installed before running this job.

Validate the syntax of a control repo's Puppet templates

To add this job to your Continuous Delivery for PE instance:

1. In the Continuous Delivery for PE web UI, click **Jobs > New job**.
2. Complete the fields:
 - a. **Name:** control-repo-template-syntax-validate
 - b. **Description:** Validate that a control repo's Puppet templates are syntactically correct.
 - c. **Job commands:** `rake -f /Rakefile syntax:templates`
 - d. **Where can this job run?:** Select **Run on <WORKSPACE_NAME> hardware**
 - e. **Hardware capabilities:** Select **Docker** and click **Apply**
 - f. **Docker configuration:** Enable the **Run this job in a Docker container** switch and select **Default image** (`puppet/puppet-dev-tools`)
3. Click **Save job**.

[Configure job hardware](#) on page 154 and ensure Docker is installed before running this job.

Validate the syntax of a control repo's Hiera data

To add this job to your Continuous Delivery for PE instance:

1. In the Continuous Delivery for PE web UI, click **Jobs > New job**.
2. Complete the fields:
 - a. **Name:** control-repo-hiera-syntax-validate
 - b. **Description:** Validate that a control repo's Hiera data is syntactically correct.
 - c. **Job commands:** `rake -f /Rakefile syntax:hiera`
 - d. **Where can this job run?:** Select **Run on <WORKSPACE_NAME> hardware**
 - e. **Hardware capabilities:** Select **Docker** and click **Apply**
 - f. **Docker configuration:** Enable the **Run this job in a Docker container** switch and select **Default image** (`puppet/puppet-dev-tools`)
3. Click **Save job**.

[Configure job hardware](#) on page 154 and ensure Docker is installed before running this job.

Validate the syntax of a control repo's Puppet manifest code

To add this job to your Continuous Delivery for PE instance:

1. In the Continuous Delivery for PE web UI, click **Jobs > New job**.
2. Complete the fields:
 - a. **Name:** control-repo-manifest-validate
 - b. **Description:** Validate that a control repo's Puppet manifest code is syntactically correct
 - c. **Job commands:** `rake -f /Rakefile syntax:manifests`
 - d. **Where can this job run?:** Select **Run on <WORKSPACE_NAME> hardware**
 - e. **Hardware capabilities:** Select **Docker** and click **Apply**
 - f. **Docker configuration:** Enable the **Run this job in a Docker container** switch and select **Default image** (`puppet/puppet-dev-tools`)
3. Click **Save job**.

[Configure job hardware](#) on page 154 and ensure Docker is installed before running this job.

Validate the syntax of a module's Puppet manifest code

To add this job to your Continuous Delivery for PE instance:

1. In the Continuous Delivery for PE web UI, click **Jobs > New job**.

2. Complete the fields:

- a. **Name:** module-pdk-validate
- b. **Description:** Validate that a module's Puppet manifest code is syntactically correct
- c. **Job commands:** `pdk validate --parallel`
- d. **Where can this job run?:** Select **Run on <WORKSPACE_NAME> hardware**
- e. **Hardware capabilities:** Select **Docker** and click **Apply**
- f. **Docker configuration:** Enable the **Run this job in a Docker container** switch and select **Default image** (`puppet/puppet-dev-tools`)

3. Click **Save job**.

[Configure job hardware](#) on page 154 and ensure Docker is installed before running this job.

Run rspec-puppet unit tests on a module

To add this job to your Continuous Delivery for PE instance:

1. In the Continuous Delivery for PE web UI, click **Jobs > New job**.

2. Complete the fields:

- a. **Name:** module-rspec-puppet
- b. **Description:** Run rspec-puppet unit tests on a module
- c. **Job commands:** `pdk test unit`
- d. **Where can this job run?:** Select **Run on <WORKSPACE_NAME> hardware**
- e. **Hardware capabilities:** Select **Docker** and click **Apply**
- f. **Docker configuration:** Enable the **Run this job in a Docker container** switch and select **Default image** (`puppet/puppet-dev-tools`)

3. Click **Save job**.

[Configure job hardware](#) on page 154 and ensure Docker is installed before running this job.

Sample non-Docker-based module jobs

You can use these Continuous Delivery for Puppet Enterprise (PE) job templates as-is or customize them to your deployment's needs. To add jobs, go to **Jobs > New job** in the Continuous Delivery for PE web UI.

Note: When configuring jobs that are not Docker-based, you can use the `$REPO_DIR` environment variable to reference the directory containing the relevant module repo.

Note: You can [Add secrets to jobs](#) on page 186 after you create them.

Puppet Development Kit validation tests

These jobs validate module code against Puppet Development Kit (PDK). Use the configuration for your operating system.

For Linux:

- **Job name:** module-pdk-validate-linux
- **Description:** Validate via PDK
- **Job commands:** `pdk validate`
- **Hardware capabilities:** Linux

For Windows:

- **Job name:** module-pdk-validate-windows
- **Description:** Validate via PDK
- **Job commands:** `powershell.exe -c "pdk validate"`
- **Hardware capabilities:** Windows

To use these jobs you must [Install PDK](#) and install `puppet-agent` on the job hardware.

Puppet Development Kit `rspec-puppet` tests

These jobs run unit tests on your module code with `rspec-puppet`. Use the configuration for your operating system.

For Linux:

- **Job name:** `module-pdk-test-unit-linux`
- **Description:** Run unit tests via PDK
- **Job commands:** `pdk test unit`
- **Hardware capabilities:** Linux

For Windows:

- **Job name:** `module-pdk-test-unit-windows`
- **Description:** Run unit tests via PDK
- **Job commands:** `powershell.exe -c "pdk test unit"`
- **Hardware capabilities:** Windows

To use these jobs you must [Install PDK](#) and install `puppet-agent` on the job hardware.

Sample non-Docker-based control repo jobs

You can use these Continuous Delivery for Puppet Enterprise (PE) job templates as-is or customize them to your deployment's needs. To add jobs, go to **Jobs > New job** in the Continuous Delivery for PE web UI.

Note: When configuring jobs that are not Docker-based, you can use the `$REPO_DIR` environment variable to reference the directory containing the relevant module repo.

Note: You can [Add secrets to jobs](#) on page 186 after you create them.

Syntax validation

This job validates the syntax of everything in your control repo.

To use this job you must use a *nix host and install `puppet-agent` on the job hardware.

- **Job name:** `control-repo-validate-linux`
- **Description:** Validate syntax
- **Hardware capabilities:** Linux
- **Job commands:**

```
#!/bin/bash

shopt -s globstar nullglob
green="$(tput setaf 2)"
red="$(tput setaf 1)"
reset="$(tput sgr0)"

for f in **/**.pp; do
  [[ $f =~ plans/ ]] && continue

  if puppet parser validate "$f"; then
    echo "${green}SUCCESS: $f${reset}"
  else
    echo "${red}FAILED: $f${reset}"
    failures+=("$f")
  fi
done
```

```

if (( ${#failures[@]} > 0 )); then
    echo "${red}Syntax validation on the Control Repo has failed in the
    following manifests:"
    echo -e "\t ${failures[@]}${reset}"
    exit 1
else
    echo "${green}Syntax validation on the Control Repo has succeeded.
    ${reset}"
fi

```

Puppet linter

This job checks the Puppet code in your control repo for programming and stylistic errors.

To use this job you must use a *nix host and install `puppet-agent` on the job hardware.

- **Job name:** control-repo-lint-linux
- **Description:** Lint Puppet code
- **Hardware capabilities:** Linux
- **Job commands:**

```

#!/bin/bash

shopt -s globstar nullglob
green="$(tput setaf 2)"
red="$(tput setaf 1)"
reset="$(tput sgr0)"

sudo /opt/puppetlabs/puppet/bin/gem install puppet-lint || {
    echo "${red}Failed to install puppet-lint gem"
    exit 2
}

LINT_OPTS=( "--fail-on-warnings" "--no-documentation-check"
  "--no-140chars-check" "--no-autoloader_layout-check" "--no-
  class_inherits_from_params_class-check")

for f in **/**.pp; do
    [[ $f =~ plans/ ]] && continue

    if /opt/puppetlabs/puppet/bin/puppet-lint "${LINT_OPTS[@]}" "$f"; then
        echo "${green}SUCCESS: $f${reset}"
    else
        echo "${red}FAILED: $f${reset}"
        failures+=("$f")
    fi
done

if (( ${#failures[@]} > 0 )); then
    echo "${red}Puppet-lint validation on the Control Repo has failed in
    the following manifests:"
    echo -e "\t ${failures[@]}${reset}"
    exit 1
else
    echo "${green}Puppet-lint validation on the Control Repo has succeeded.
    ${reset}"
fi

```

Attaching known_hosts to containerized jobs

Mounting your `known_hosts` file to a volume in your container.

If you need to `git clone`, or run `r10k` or `rake spec_prep` to download artifacts in your containerized job and are using strict host-key verification, you need to mount a volume in your container to support this. If you are using `puppet/puppet-dev-tools:4.x` (the default container), you can accomplish this by mounting your good `known_hosts` file into the root `.ssh` directory:

```
--volume=/path/to/your/known_hosts:/root/.ssh/known_hosts
```

Add the above line to your job configuration under **Container runtime arguments** to tell Docker or Podman to mount your `known_hosts` file into the `puppet/puppet-dev-tools` container. This is then automatically referenced when reaching out to your internal server.

Analyze the impact of code changes

Impact analysis is a Continuous Delivery for Puppet Enterprise (PE) tool that shows you the potential impact that new Puppet code can have on your PE-managed infrastructure, without actually merging the new code. You can generate impact analysis reports on demand for commits in your control/module repos, and, if you add impact analysis to a repo's pipeline, Continuous Delivery for PE automatically generates a report on every proposed code change to that repo.

- [Configure impact analysis](#) on page 192

Impact analysis is a Continuous Delivery for Puppet Enterprise (PE) tool that shows you the potential impact that new Puppet code can have on your PE-managed infrastructure, without actually merging the new code. When you add impact analysis to a control/module repo's pipeline, Continuous Delivery for PE automatically generates a report on every proposed code change to that repo.

- [Generate impact analysis reports](#) on page 194

Impact analysis reports show you the potential impact and risk of proposed code changes, so you can decide how, and if, you want to handle those potential changes. You can generate reports on demand or add impact analysis tasks to your pipelines to generate impact analysis reports for every change submitted to your repos. Impact analysis reports are generated by diffing the deployment's current catalog against a newly-generated catalog for your specified deployment conditions.

- [Run impact analysis on fewer nodes](#) on page 197

If an environment has a lot of nodes, it might take a long time for impact analysis to run. It is possible to only analyze a subset of your total nodes, but there are tradeoffs.

- [Impact analysis limitations](#) on page 198

Impact analysis has some technical limitations, and code changes can impact your infrastructure beyond what the impact analysis report displays.

Configure impact analysis

Impact analysis is a Continuous Delivery for Puppet Enterprise (PE) tool that shows you the potential impact that new Puppet code can have on your PE-managed infrastructure, without actually merging the new code. When you add impact analysis to a control/module repo's pipeline, Continuous Delivery for PE automatically generates a report on every proposed code change to that repo.

Configuration process and preparation

To configure impact analysis, you must set the destination server for impact analysis tasks, install the `puppetlabs-cd4pe` module, and make a classification update.

Before configuring impact analysis, you must:

- [Install Continuous Delivery for PE](#)
- [Integrate with your source control system](#)
- [Integrate with Puppet Enterprise](#)

Set the impact analysis destination

When you integrate a new Puppet Enterprise (PE) instance with Continuous Delivery for PE, impact analysis tasks are automatically configured to run on the primary server. In many cases, it's better to change the impact analysis task destination to a different server, such as a compiler or load balancer, to preserve the primary server's compile capacity.

Before you begin

Determine a place in your PE infrastructure to direct impact analysis tasks:

- **Best:** A load balancer managing a pool of compilers.
- **Good:** An individual compiler dedicated to impact analysis tasks.
- **For small installations only:** The PE primary server.

Note: The primary server is the default configuration for running impact analysis tasks, but it is best suited to PE installations with limited node counts using the [standard installation architecture](#). Impact analysis tasks can easily overrun the primary server's limited compile capacity in a larger distributed architecture.

If you choose to run impact analysis tasks on the primary server, skip to [Install modules](#) on page 193.

To run impact analysis tasks on a load balancer or compiler:

1. In the Continuous Delivery for PE web UI, navigate to **Settings > Puppet Enterprise** and locate the PE instance you are configuring.
2. Click **More actions > Edit integration**.
3. If you want to run impact analysis tasks through a dedicated load balancer or compiler, in the **Puppet Server service** field, enter the hostname of the primary server or a specific compiler or load balancer at `:8140` (for example, `loadbalancer.example.com:8140`).

Tip: The Puppet Server service is used for impact analysis, among other processes. You can run impact analysis tasks on a compiler or load balancer instead of the primary server. **This is strongly recommended for PE installations that use compilers or load balancers as part of their architecture.**

4. Optional: You can expand the **Impact Analysis Settings** section to adjust the maximum number of catalogs that Continuous Delivery for PE is able to simultaneously compile, as explained in [Optional impact analysis settings](#).
5. Click **Save Changes**.

Next, [Install modules](#) on page 193.

Optional impact analysis settings

To adjust these optional settings, edit your Puppet Enterprise (PE) integration settings and expand the **Impact Analysis Settings** section.

Concurrent catalog compilations

This setting sets the maximum number of catalog compilations Continuous Delivery for PE is permitted to perform simultaneously. By default, Continuous Delivery for PE performs up to 10 concurrent catalog compilations.

Tune this number to fit the needs and limitations of your installation. A lower number preserves processing capacity, while a higher number reduces the time Continuous Delivery for PE spends completing each impact analysis task.

Install modules

Impact analysis requires you to install the `puppet-labs-cd4pe` module and its dependent modules.

1. Add the `puppetlabs-cd4pe` module, its [dependencies](#), and the `puppetlabs-cd4pe_jobs` module to the Puppetfile for each environment against which your compilers compile catalogs.

There are multiple ways to [Declare Forge modules in the Puppetfile](#). You can specify a specific version or specify `:latest` to automatically check for new versions. For example:

```
mod 'puppetlabs-cd4pe', :latest
# Requirements for cd4pe
mod '<DEPENDENT_MODULE_NAME>', :latest
mod 'puppetlabs-cd4pe_jobs', :latest
```

Important: Make sure to declare all required modules, which includes the `puppetlabs-cd4pe` module, all of its dependent modules, and the `puppetlabs-cd4pe_jobs` module.

If you specify specific module versions in your Puppetfile, keep in mind that you must install version 1.6.0 or later of the `cd4pe_jobs` module to [Add secrets to jobs](#) on page 186.

2. If you configured Continuous Delivery for PE to automatically deploy code changes to the relevant environments, Continuous Delivery for PE deploys the updated code for you. If you have not configured automatic code deployments, run `puppet code deploy <ENVIRONMENT>` to deploy the updated code to the relevant environments.

Next, [Update classification](#) on page 194.

Update classification

Once `puppetlabs-cdpe` and its dependencies are deployed, you must update classification of your nodes.

1. In the PE console, click **Node groups** (or **Classification** in PE versions prior to 2019.8.1) and open the **PE Infrastructure** group.
2. Select the **PE Master** group and click **Classes** (or **Configuration** in PE versions prior to 2019.8.1).
3. In the **Add new class** field, select `cd4pe::impact_analysis` and click **Add class**, then commit your change. If you don't find `cd4pe::impact_analysis` in the class list, click **Refresh** to update class definitions.
4. Run Puppet on the nodes in the **PE Master** group.

Important: This Puppet run restarts the `pe-puppetserver` service.

5. Optional: You can adjust the maximum number of catalogs that Continuous Delivery for PE is able to simultaneously compile, as explained in [Optional impact analysis settings](#).

Now you're ready to [Generate impact analysis reports](#) on page 194.

Generate impact analysis reports

Impact analysis reports show you the potential impact and risk of proposed code changes, so you can decide how, and if, you want to handle those potential changes. You can generate reports on demand or add impact analysis tasks to your pipelines to generate impact analysis reports for every change submitted to your repos. Impact analysis reports are generated by diffing the deployment's current catalog against a newly-generated catalog for your specified deployment conditions.

Important: With impact analysis enabled, catalog compiles are generated every time impact analysis detects that a node could be impacted by a Puppet code change. These additional compiles increase the performance load placed on your PE primary server.

Restriction: Impact analysis fails if you include the `$environment` variable in your Puppet manifest. Instead, use `Hiera` and class parameters.

Impact analysis workflow

Impact analysis targets a specific environment (such as `development`) and a specific version of code, identified by a Git commit SHA. Using the designated environment and SHA, Continuous Delivery for PE does the following during an impact analysis run:

1. Create a temporary branch in the control repo using the naming scheme `<BRANCH>_cdpe_ia_<TIMESTAMP>`. If analyzing a module repo, temporary branches are created in the module repo *and* in the control repo that is associated with the module repo. The temporary branch serves as temporary environment in which to deploy your new code and detect possible changes.
2. Deploy code associated with the targeted commit SHA in the temporary branch.
3. Calculate module version changes by comparing module deployment data from the target environment and the temporary environment.
4. Calculate changes to Hieradata parameters. If Hieradata regex is configured, Continuous Delivery for PE uses the regex to locate Hieradata files. Otherwise, it looks for changes in Hieradata files ending with `.yaml` in the `data` or `hieradata` directories.
5. Calculate impacted nodes:
 - a. Query PuppetDB for nodes that contain resources where the file parameter matches one of the modified module versions or Hieradata files. Module versions are queried as a file path to the module.
 - b. Query PuppetDB for catalog input data matching modified Hieradata parameters.
6. Calculate impacted node resources:
 - a. Query PuppetDB to get the current catalog for each impacted node.
 - b. Use the code deployed to the temporary environment to compile a new catalog for each impacted node.
 - c. For each impacted node, compare the new (temporary environment) catalog to the current (live environment) catalog.
 - d. Build the impact analysis report based on the discovered differenced between the catalogs.
7. Report the impacted nodes and resources.

Add impact analysis to a control repo pipeline

If you add an impact analysis task to your control repo pipeline, an impact analysis report is automatically generated each time the pipeline runs and your specified conditions are met.

Before you begin

1. [Configure impact analysis](#).
2. [Construct a pipeline](#) for your module that includes at least one deployment.

Each pipeline can have unlimited impact analysis tasks, but each stage in a pipeline can have only one impact analysis task. Additionally, an impact analysis task cannot be in the same stage as a deployment task.

This section explains how to add an impact analysis step to a [pipeline constructed in the web UI](#). If you manage your pipelines with code, go to [.cd4pe.yaml file structure](#) on page 206 for guidance on adding impact analysis steps to your pipelines.

1. In the Continuous Delivery for PE web UI, click **Control repos**, and click the name of the control repo you want to add impact analysis to.
2. Select the pipeline you want to add the impact analysis step to. Make sure the pipeline has at least one deployment step, because impact analysis is calculated based on the pipeline's deployment conditions.
3. Identify the stage you want to add the impact analysis step to. The stage must not contain a deployment step, and we recommend choosing a stage before any deployment stages or PR gates.

4.

In your chosen stage, click **More actions** , select **Add item to stage**, and select **Impact analysis**.

- Optional: Set the catalog compilation batch size. By default, Continuous Delivery for PE compiles 10 catalogs at a time when performing an impact analysis task.

Tip: If your compilers are hitting capacity when performing an impact analysis, lower this number. However, lowering this number increases the impact analysis run time.

- Determine the environments you want to generate an impact analysis report for.
 - Run for all environments in the pipeline** runs impact analysis on all environments used by all deployments in the pipeline.
 - Run for selected environments** runs impact analysis on specific environments. Select the relevant environments from among those used by the pipeline's deployments.
- Click **Add impact analysis**.

Impact analysis is now enabled for this control repo pipeline. An impact analysis report is generated each time the pipeline runs.

Add impact analysis to a module pipeline

If you add an impact analysis task to your module pipeline, an impact analysis report is automatically generated each time the pipeline runs and your specified conditions are met.

Before you begin

- [Configure impact analysis](#).
- [Construct a pipeline](#) for your module that includes at least one deployment.

Each pipeline can have unlimited impact analysis tasks, but each stage in a pipeline can have only one impact analysis task. Additionally, an impact analysis task cannot be in the same stage as a deployment task.

This section explains how to add an impact analysis step to a [pipeline constructed in the web UI](#). If you manage your pipelines with code, go to [.cd4pe.yaml file structure](#) on page 206 for guidance on adding impact analysis steps to your pipelines.

- In the Continuous Delivery for PE web UI, click **Modules**, and click the name of the module you want to add impact analysis to.
- Select the pipeline you want to add the impact analysis step to. Make sure the pipeline has at least one deployment step, because impact analysis is calculated based on the pipeline's deployment conditions.
- Identify the stage you want to add the impact analysis step to. The stage must not contain a deployment step, and we recommend choosing a stage before any deployment stages or PR gates.
- In your chosen stage, click **More actions** , select **Add item to stage**, and select **Impact analysis**.
- Optional: Set the catalog compilation batch size. By default, Continuous Delivery for PE compiles 10 catalogs at a time when performing an impact analysis task.

Tip: If your compilers are hitting capacity when performing an impact analysis, lower this number. However, lowering this number increases the impact analysis run time.

- Select the environments you want to generate an impact analysis report for.
- For each selected environment, choose the control repo where the code associated with that environment is stored.
- Click **Add impact analysis**.

Impact analysis is now enabled for this module pipeline. An impact analysis report is generated each time the pipeline runs.

Generate an impact analysis report on demand

Without triggering a pipeline, you can generate an impact analysis report for any commit made to a control repo or module that Continuous Delivery for Puppet Enterprise (PE) monitors.

Before you begin

Configure impact analysis.

1. In the Continuous Delivery for PE web UI, navigate to the control repo or module you want to generate an impact analysis report for.
2. Click **Manual actions** and select **New impact analysis**.
3. Select the branch where the code you want to analyze is located, and then select the commit containing the changes you want to analyze.
4. Select the PE instance that manages the nodes you want to analyze, and select the specific node group.
 - For control repo impact analysis, select an environment prefix, if applicable.
 - For module impact analysis, select the control repo where the module's environment code is deployed.
5. Set the number of node catalogs to compile concurrently. Lower numbers can have a lesser impact on performance but take longer to complete the impact analysis.
6. Click **Analyze**, and wait while the report generates. It may take several minutes for large node groups.
7. Once the analysis is complete, click **View impact analysis** to access the report.

Tip: Click **Export** on the report page to get a CSV file containing the impact analysis report's data.

Run impact analysis on fewer nodes

If an environment has a lot of nodes, it might take a long time for impact analysis to run. It is possible to only analyze a subset of your total nodes, but there are tradeoffs.

Important: If you're experiencing degraded performance when running impact analysis, we initially recommend [changing the impact analysis destination](#) to use a dedicated compiler or a pool of compilers. If you're already using compilers for impact analysis and the runtime or performance burden isn't acceptable, you might consider running impact analysis on fewer nodes. However, keep in mind the following tradeoffs:

- **Not all nodes are analyzed.** By definition, running impact analysis on fewer nodes means that some nodes don't get analyzed. For example, if you're analyzing only 10% of your nodes, the remaining 90% are not analyzed. When your code is deployed, excluded nodes might have unexpected changes that weren't detected since those nodes weren't analyzed.
- **Additional heap space is consumed.** To run impact analysis on fewer nodes, you must create one or more dedicated impact analysis environments. Each impact analysis environment has the same code as its corresponding primary environment (for example, `production` and `production-ia`). Because environments consume heap space in Puppet Server, adding these additional environments consumes additional heap space deploying the same code to multiple environments.

Impact analysis runs on nodes in a designated environment. Therefore, if your control repo pipeline runs impact analysis on your `production` environment, it analyzes all nodes in the `production` environment node group. If you have a lot of nodes, this can take a long time to run and might be taxing on system resources. If your nodes are mostly similar, it might make sense to run impact analysis on a subset of your total nodes, rather than always analyzing every node. However, this requires changing your environment structure to accommodate one or more `impact-analysis` environments.

For example, if you want to run impact analysis on a few nodes before deploying code to all `production` nodes, you'll need to set up a `production impact analysis` environment. First, create a `production-ia` branch in your control repo and [deploy the new environment](#). Next, create a `production-ia environment node group` as a child of your `production` environment node group. Then, [add nodes](#) to the `production-ia` group representing a subset of your total production nodes.

Tip: You'll only analyze the nodes in the `production-ia` group, so make sure the nodes in this group are a good representation of your total production nodes. For example, make sure to include different operating systems or geographic locations, as well as any outliers and known problematic nodes.

You now have two environments where your production code is deployed: `production-ia`, which contains some production nodes, and `production`, which contains all production nodes. To run impact analysis on the smaller `production-ia` group, you need to add the new `production-ia` environment to your control repo pipeline:

- [Add a deployment](#) for the `production-ia` environment, in addition to the `production` environment's deployment.

Tip: Since you're deploying the same code to `production-ia` and `production`, you can configure your pipeline to auto-promote to the `production` deployment stage after completing the `production-ia` deployment.

- [Edit the impact analysis task](#) so that it only runs on nodes in the `production-ia` environment. Make sure the impact analysis task is set to **Run for selected environments** and includes only the `production-ia` environment. Since your goal is to analyze only a subset of nodes, you don't want to run impact analysis on the `production` environment anymore.
- Check the promotion settings between the impact analysis stage and the `production-ia` deployment stage. If you want to review the impact analysis report before deploying your code, make sure your pipeline doesn't auto-promote to the deployment stage.

While the above example used the `production` environment, you could set up similar structures for any environments you wanted to partially analyze, such as UAT or preproduction.

You can help us improve this feature: We invite you to tell us why you run (or would run) impact analysis on fewer nodes, whether the above approach works in your infrastructure, and what changes you'd like us to make. Please visit our [product board](#) to learn more, vote for features you like, and tell us your thoughts.

Related information

[Create environment node groups](#) on page 145

For code deployments managed by Continuous Delivery for PE to work correctly, your environment node groups must be set up in a specific hierarchy.

Impact analysis limitations

Impact analysis has some technical limitations, and code changes can impact your infrastructure beyond what the impact analysis report displays.

Important: We've designed impact analysis to give you helpful information about the potential results of code changes to your infrastructure. However, impact analysis is not intended to be an exhaustive report — do not use it as a substitute for more exhaustive testing.

There are certain circumstances in which we can't reliably calculate the full impact of a code change. As an impact analysis user, it's critical that you understand these limitations exist, and you are aware of the possibility that code changes can have consequences for your infrastructure beyond what impact analysis shows.

This is a non-exhaustive list of some of the limitations of impact analysis. While this list might be updated periodically, it is never complete.

- **Changes to an environment's `environment.conf` file.** Impact analysis can't determine the downstream impact of changes to an `environment.conf` file.
- **Brand-new code.** Impact analysis can't determine the impact of new Puppet classes, facts, or functions that have never been applied to any of your nodes.
- **Changes to functions.** The impact of changes to a function cannot be analyzed.
- **Changes to facts.** The impact of changes to a fact cannot be analyzed.
- **Changes to class names used in classification.** Changes to class names that were previously classified in the classifier cannot be analyzed. A classification update in the classifier is required before Puppet can locate the newly renamed class.
- **Imported resources.** While impact analysis can provide information about nodes that export resources, impact analysis can't determine the impact to nodes that receive those exported resources.
- **Sensitive data types.** Any changes to data types marked as sensitive are not be analyzed.

- **Alias metaparameter.** Any resources using the alias metaparameter cannot be analyzed.
- **Changes to Embedded Puppet templates.** The impact of changes made to Embedded Puppet (`.epp`) or Embedded Ruby (`.erb`) template files alone cannot be analyzed.
- **Patch fact generation script files.** The `pe_patch_fact_generation.ps1` and `pe_patch_fact_generation.sh` files are excluded from impact analysis reports because these files always report a change of `n/a` to `n/a`.

Remember: Impact analysis reports are provided *as-is*.

Construct pipelines

Pipelines drive the continuous aspect of Continuous Delivery for Puppet Enterprise (PE). Constructing a pipeline involves defining work that must happen to ensure every new line of Puppet code is ready for deployment. Once your pipeline is set up, this work happens automatically each time the pipeline is triggered.

Stages and tasks

Pipelines in Continuous Delivery for PE are made up of *stages* and *tasks*. Tasks include deployments, impact analyses, and jobs to test code. Stages are groups of tasks. Stages allow you to break pipelines into a series of sequential task sets. This allows you to have logical control in your pipeline, such as:

- Start task B only if all tasks in stage A succeed.
- If a task in stage A fails, stop the pipeline and report the error.

You can configure pipelines to require manual approval to proceed to the next stage or advance automatically based on your defined conditions.

You can also [Enable compiler maintenance mode](#) on page 166 if you want code deployments in your pipelines to skip unavailable (or offline) compilers and replicas.

Use the web UI or code to build and manage pipelines

You can build and manage pipelines in the Continuous Delivery for PE web UI or through a `.cd4pe.yaml` file. The web UI is simpler, and it is easier to make iterative changes with the web UI controls. Using a YAML file to build pipelines as code is more complex, but it is preferred if you:

- Need a record of changes to your pipeline over time
- Want to avoid manually creating similar pipelines for many control repos or modules.
- Make changes to your Puppet code that require new pipeline definitions and you want to commit changes to your pipelines-as-code at the same time.

Important: You can't mix these methods within a single control repo or module. Web UI controls are disabled when you choose to manage pipelines with code. The following image compares pipelines in the web UI. The pipeline managed in the web UI has controls available (such as deleting the

pipeline, deleting tasks, and adding stages), whereas the pipeline managed as code does not have these

The image shows two side-by-side screenshots of the Puppet Enterprise web UI for pipeline management. The left screenshot is titled "Managed in the web UI" and shows the "main" pipeline. It features a "Code Validation stage" with four jobs: "control-repo-manifest-validate", "control-repo-hiera-syntax-validate", "control-repo-template-syntax-validate", and "control-repo-puppetfile-syntax-validate". Below this is an "Impact Analysis stage" with jobs "Impact Analysis: All Environments" and "Pull request gate". The pipeline status is "All succeeded" and "Auto promote" is checked. The right screenshot is partially visible and titled "Managed as code", showing a "pac_test" pipeline with a "Code Validation" stage and an "Impact Analysis" stage.

controls.

- [Construct pipelines in the web UI](#) on page 201

Build and manage pipelines for your control repo or module in the Continuous Delivery for Puppet Enterprise (PE) web UI using the tools built into the interface. The steps on this page walk through the process of setting up and using a basic pipeline.

- [Construct pipelines from code](#) on page 204

If you manage your pipelines with code, rather than in the web UI, you can maintain a record of pipeline changes over time. When you choose to manage pipelines with code, a `.cd4pe.yaml` file, containing the pipelines' definitions

for a control repo or module, is stored in your source control system alongside the Puppet code for that control repo or module.

Construct pipelines in the web UI

Build and manage pipelines for your control repo or module in the Continuous Delivery for Puppet Enterprise (PE) web UI using the tools built into the interface. The steps on this page walk through the process of setting up and using a basic pipeline.

Create a pipeline

Set up a pipeline to enable automatic testing of new code by adding stages and jobs in the Continuous Delivery for Puppet Enterprise (PE) web UI.

Every pipeline must have at least one stage.

1. In the Continuous Delivery for PE web UI, click **Control repos**, and click the name of the control repo you want to create a pipeline for.

Continuous Delivery for PE automatically creates a pipeline for the branch you selected when you set up the control repo (the main branch). The branch name is at the top of the **Pipelines** section. You can set up pipelines for other branches in the repo if you want to automatically test code changes that Continuous Delivery for PE makes on your behalf. To create a pipeline for a different branch, click **Add pipeline**. If you don't want a pipeline,

click **Delete pipeline** .

You can also go to **Modules** to create pipelines for module repos.

2. Click **Add stage** and enter a **Stage name**.
3. Select **Jobs** from the **Select item** dropdown menu, select a job from the **Select jobs** list, click **Add stage** to add the job to the pipeline, and click **Done**.
4. If you only need to run a single test on new code, the pipeline can be as simple as one stage with one job. If you want to add additional tests, you can either add them to the same stage or create another stage to contain other jobs.
 - a) To add another job to the same stage, click **More actions** , select **Add item to stage**, and repeat step 3.
 - b) To add another stage to your pipeline, repeat steps 2 and 3.

Note: All steps (jobs, deployments, and so on) in a stage run concurrently. If you need certain steps to occur after others, use stages to separate them.

To reorganize, delete, or rename stages, click **More actions** .

5. Once you have a pipeline with two stages, an promotion step is automatically added between the stages. Select **Auto promote** and choose a promotion condition from the dropdown menu. The promotion condition tells the pipeline how to decide if it can continue to the next stage. The promotion conditions, from most to least restrictive, are:
 - **All succeeded:** The pipeline continues to the next stage only if all tasks in the current stage finish with "succeeded" or "done" status.
 - **All completed:** The pipeline continues to the next stage if all tasks in the current stage finish with "succeeded," "done," or "failed" status. The pipeline stops if any task in the current stage was canceled.
 - **Any succeeded:** The pipeline continues to the next stage if at least one task in the current stage finishes with a "succeeded" or "done" status.
 - **Any completed:** The pipeline continues to the next stage if at least one task in the current stage finishes with a "succeeded," "done," or "failed" status. The pipeline stops if all tasks in the current stage were canceled.
6. Optional: If you want the pipeline to run on both commits and pull requests, click **Manage pipelines** to enable pull requests as a pipeline trigger.

This pipeline runs on one branch in one control/module repo.

Related information

[Test Puppet code with jobs](#) on page 185

Jobs are fully customizable tests for your Puppet code. You can create a job that runs any sort of test you want, including module validation, linting, and more.

Create a regex branch pipeline

Most pipelines are linked to a single branch in your control repo or module repo. A regex branch pipeline can recognize and act on changes to any branch in your control/module repo that has a name matching the regular expression you set.

A regex branch pipeline lets you use a single pipeline for all feature branches you and your team create (and destroy) in the process of developing new code. Regex branch pipelines use naming conventions to detect branches to run the pipeline against. You define the naming convention when you create the pipeline. When you commit to (or create a pull request for) a feature branch that has a name matching the naming convention, Continuous Delivery for Puppet Enterprise (PE) automatically runs the regex pipeline against that branch.

1. In the Continuous Delivery for PE web UI, click **Control Repos** or **Modules** and select the control repo or module repo where you want to create the regex pipeline.
2. Click **Add pipeline**.
3. In the **Add pipeline** window, select **Regex branch**.

Each control repo or module repo can have only one regex branch pipeline. If the **Regex branch** option is not available, then a regex branch pipeline already exists for this repo.

4. In the **Configure regex** field, enter the naming convention, as a regular expression, that you want the regex branch pipeline to use to detect feature branches (and initiate pipeline runs against those branches).

The default regular expression is `feature_.*`. If you use the default regex, your feature branch names must follow the `feature_<BRANCHNAME>` naming convention. For example, a branch named `feature_1234` triggers the pipeline, but a branch named `1234_feature` does not.

Tip: All team members must follow this naming convention when creating feature branches if they want their changes to trigger the regex branch pipeline. We strongly recommend using the default regular expression or one that's similarly simple and memorable.

5. Click **Add pipeline**. The regex branch pipeline is created.
6. Click **Manage pipelines** to select whether the regex branch pipeline is triggered by commits, pull requests, or both.
7. Build out your pipeline by adding stages, jobs, impact analysis tasks, deployments, and promotion conditions.

This pipeline runs on any branch that matches the defined naming convention in the associated control/module repo.

Test code automatically with a pipeline

After creating a pipeline for your control repo's main branch, when you commit new code to the main branch, the pipeline tests the code (by running the jobs you selected) and reports any errors or failures.

Before you begin

[Create a pipeline](#) on page 201

1. To trigger the pipeline, make a trivial change, such as adding a line to a README file, in your control repo's main branch, and commit your change.
2. In the Continuous Delivery for PE web UI, click **Control repos** and click the name of the control repo you just committed to.
3. The **Events** area shows the push event initiated by your commit and an entry for each job in the pipeline. Job statuses appear here and update to report success or failure when a job run is complete.

Each event summary includes a link to the commit in the control repo, and each job event includes a job number. Click the job number to go to the **Job details** page, where you can see the job's log for its run and review error messages.

Test pull requests automatically

If you enable pull requests as a pipeline trigger, your pipeline automatically tests new code when a pull request (PR) is opened against the relevant branch. Add PR gates to pipelines to test new code but stops the pipeline before it can advance to further tests or deployment.

Before you begin

Create a [pipeline](#) on page 201 that has pull requests as a pipeline trigger.

A PR gate stops the pipeline at a certain point if the pipeline was triggered by a pull request. With a PR gate, you can automatically run acceptance tests on new pull requests while manually managing additional testing or deployment based on the results of the initial automated tests.

GitLab users: *Pull request* is another term for *merge request*. For the sake of simplicity and consistency, the term *pull request (PR)* is used in the Continuous Delivery for PE web UI and documentation.

1. To add a PR gate to your pipeline, navigate to the pipeline in the web UI.
2. In your pipeline's first stage, click **More actions** , select **Add item to stage**, and select **Pull request gate**.
3. Click **Add PR gate**, and click **Done** if the PR gate was added successfully.

Important: A pipeline can only have one PR gate. To move a PR gate, delete the existing gate and create a new one in the desired stage.

4. To test PR gate, switch to a branch in your control repo other than the main branch. Make a trivial change (such as adding a new line to the README file) in your control repo, commit your change, and open a pull request against the main branch for this commit.

Remember: You must make the pull request against the branch associated with the pipeline.

5. In the Continuous Delivery for PE web UI, click **Control repos**, then click the name of the control repo you just created the pull request against.
6. The **Events** area shows the pull request event initiated by your commit and entries for jobs in your pipeline before the PR gate. Notice that jobs in pipeline stages beyond the PR gate have not been triggered.

Deploy code automatically with a pipeline

You can use a control repo pipeline to automatically deploy new code to a specified set of nodes every time a commit is made.

Before you begin

Create a [pipeline](#) on page 201

1. In the Continuous Delivery for PE web UI, click **Control repos**, and click the name of the control repo you've created a pipeline for.
2. Click **Add stage**.
3. In the **Add new stage** window, select **Deployment** from the **Select item** dropdown menu.
4. Select your Puppet Enterprise instance and the node group you want to deploy changes to when this pipeline runs.
5. Select a deployment policy.

You can use one of the [Built-in deployment policies](#) on page 216 or create [Custom deployment policies](#) on page 220.

Tip: If this is your first pipeline and you're just testing the process, select **Direct deployment policy**.

6. Optional: Set termination conditions for this pipeline's deployments, and choose the number of nodes that can fail before the deployment is stopped.
7. Click **Add Stage** and click **Done**. Your new stage, containing the deployment details, is added to the pipeline.

8. To test the deployment, make a trivial change (such as adding a new line to the README file) on the main branch of your control repo and commit your change.
9. In the Continuous Delivery for PE web UI, click **Control repos** and click the name of the control repo you just committed to.
10. The **Events** area shows the push event initiated by your commit and a deployment event. The deployment status appears here and updates to report success or failure when the deployment is complete.
Each event summary includes a link to the commit in the control repo, and deployment events include a deployment number. Click the deployment number to go to the deployment details page.
11. You can run the same deployment again from the web UI by retriggering the webhook. In the **Events** area, locate the push event you want to rerun and click **View webhook**.
The webhook's request and response data is shown in the **Webhook data** pane. This can be useful for troubleshooting.
12. Click **Redeliver webhook** and confirm your action. The **Events** area shows the new event triggered by the redelivery of the webhook.
If you made changes to your pipeline since the last time this webhook was delivered, the redelivered webhook follows the current pipeline's sequence and rules.

Construct pipelines from code

If you manage your pipelines with code, rather than in the web UI, you can maintain a record of pipeline changes over time. When you choose to manage pipelines with code, a `.cd4pe.yaml` file, containing the pipelines' definitions for a control repo or module, is stored in your source control system alongside the Puppet code for that control repo or module.

- [Configure your pipelines for management with code](#) on page 204

To manage pipelines-as-code, you need a `.cd4pe.yaml` file containing the pipeline definitions. Use these steps to convert existing pipelines to code, update pipelines-as-code, or create new pipelines for new repos.

- [.cd4pe.yaml file structure](#) on page 206

When managing your pipelines with code, pipeline definitions are expressed in a structured format and stored in a `.cd4pe.yaml` file that is stored in your control repo or module repo.

Configure your pipelines for management with code

To manage pipelines-as-code, you need a `.cd4pe.yaml` file containing the pipeline definitions. Use these steps to convert existing pipelines to code, update pipelines-as-code, or create new pipelines for new repos.

Choose the procedure appropriate for your circumstances:

- If you created pipelines for a control repo or module in the Continuous Delivery for Puppet Enterprise (PE) web UI and you now want to manage those pipelines with code, you can [Convert your existing pipelines to code](#) on page 205. Continuous Delivery for PE generates a `.cd4pe.yaml` file based on the repo's existing pipelines. You can then [Update your pipelines with a new .cd4pe.yaml file](#) on page 205 when you need to make changes to the pipelines.
- If you've created a `.cd4pe.yaml` file you want to use to replace existing pipelines (that you created in the web UI or with another `.cd4pe.yaml` file), you need to [Update your pipelines with a new .cd4pe.yaml file](#) on page 205.
- If you have created your own `.cd4pe.yaml` file for a new repo that you haven't yet added to Continuous Delivery for PE, you can [Create new pipelines using a .cd4pe.yaml file](#).

Note: This process only applies to new repos you haven't yet added to Continuous Delivery for PE. If you already added a repo to Continuous Delivery for PE, you need to [Convert your existing pipelines to code](#) on page 205 or [Update your pipelines with a new .cd4pe.yaml file](#) on page 205.

- If you want to re-enable the pipeline controls in the web UI, you can [Stop managing your pipelines with code](#) on page 206.

Convert your existing pipelines to code

If you've created pipelines using the Continuous Delivery for Puppet Enterprise (PE) web UI, and now want to manage those pipelines with code, Continuous Delivery for PE can render your existing pipelines in YAML format.

Before you begin

Add a control repo or module to Continuous Delivery for PE, configure job hardware, and set up a pipeline in the web UI:

- [Add repositories](#) on page 156
- [Configure job hardware](#) on page 153
- [Construct pipelines in the web UI](#) on page 201

1. In the Continuous Delivery for PE web UI, navigate to the control or module repo that has pipelines you want to manage with code.
2. At the top of the **Pipelines** section, click **Manage pipelines** > **Manage as code**.
3. Continuous Delivery for PE displays your pipelines in YAML format. Copy the YAML code by clicking **Copy to clipboard** at the bottom of the code block.
4. Create a new `.yaml` file named `.cd4pe` and paste the YAML code into it.

Important: The complete file name must be `.cd4pe.yaml`. Make sure you do not miss the leading period.

5. Save the `.cd4pe.yaml` file to the repo's root directory (on any branch) and commit the change in your source control system.
6. Go to the control or module repo in the Continuous Delivery for PE web UI, click **Manage pipelines** > **Manage as code**, and, in the **Select branch** area, select the branch where you saved the `.cd4pe.yaml` file.
7. Click **Save settings**.

Continuous Delivery for PE reads the contents of the `.cd4pe.yaml` file before every pipeline run, and it uses the YAML code to render the pipelines' definitions in the web UI. Because you're now managing this repo's pipelines with code, the pipeline controls in the web UI are disabled for this repo.

If you need to change this repo's pipelines, [Update your pipelines with a new .cd4pe.yaml file](#) on page 205. If you want to go back to managing this repo's pipelines with the web UI, you can [Stop managing your pipelines with code](#) on page 206.

Update your pipelines with a new `.cd4pe.yaml` file

You can use a `.cd4pe.yaml` file to replace pipelines you created in the Continuous Delivery for Puppet Enterprise (PE) web UI or to update pipelines you're already managing with code.

Before you begin

You need a properly formatted `.cd4pe.yaml` file containing the definitions of the pipelines you want to create or update. To learn about `.cd4pe.yaml` file syntax, go to [.cd4pe.yaml file structure](#) on page 206. If you were previously managing pipelines in the web UI, you can [Convert your existing pipelines to code](#) on page 205 and then update the resulting `.cd4pe.yaml` file, rather than creating a completely new file.

1. Add the new or updated `.cd4pe.yaml` file to the root directory of your control repo or module repo, and commit the file to your source control system.

If you are updating an existing `.cd4pe.yaml` file, you can replace the previous file. If you save the updated file on a different branch, you need to change the branch in the Continuous Delivery for PE web UI.

If the repo does not already have a `.cd4pe.yaml` file, you can save the file to the root directory on any branch in the repo.

2. Go to the control or module repo in the Continuous Delivery for PE web UI, click **Manage pipelines** > **Manage as code**, and, in the **Select branch** area, select the branch where you saved the `.cd4pe.yaml` file.
3. Click **Save settings**.

Continuous Delivery for PE reads the contents of the `.cd4pe.yaml` file before every pipeline run, and it uses the YAML code to render the pipelines' definitions in the web UI. Because you're now managing this repo's pipelines with code, the pipeline controls in the web UI are disabled for this repo.

Repeat this process when you need to change this repo's pipelines. If you want to switch to managing this repo's pipelines with the web UI, you can [Stop managing your pipelines with code](#) on page 206.

Create new pipelines using a `.cd4pe.yaml` file

You can create pipelines-as-code for a new control repos or module repos that you haven't yet added to Continuous Delivery for Puppet Enterprise (PE). If a `.cd4pe.yaml` file exists in the repo when you first add it to Continuous Delivery for PE, the software detects the file and asks if you want to use it to build and manage your pipelines.

Before you begin

You need a properly formatted `.cd4pe.yaml` file containing the definitions of the pipelines you want to create. To learn about `.cd4pe.yaml` file syntax, go to [.cd4pe.yaml file structure](#) on page 206.

Note: This process only applies to new repos you haven't yet added to Continuous Delivery for PE. If you already added a repo to Continuous Delivery for PE, you need to [Convert your existing pipelines to code](#) on page 205 or [Update your pipelines with a new .cd4pe.yaml file](#) on page 205.

1. Add a `.cd4pe.yaml` file to the root directory of the main branch of your control repo or module repo. Commit the file to your source control system.
2. Add the control repo or module to Continuous Delivery for PE.
For instructions, go to [Add repositories](#) on page 156.
3. After adding the repo, Continuous Delivery for PE detects the `.cd4pe.yaml` file and asks if you want to use it to build and manage your pipelines. Click **Confirm**.

Continuous Delivery for PE reads the contents of the `.cd4pe.yaml` file before every pipeline run, and it uses the YAML code to render the pipelines' definitions in the web UI. Because you're managing this repo's pipelines with code, the pipeline controls in the web UI are disabled for this repo.

If you need to change this repo's pipelines, [Update your pipelines with a new .cd4pe.yaml file](#) on page 205. If you want to start managing this repo's pipelines with the web UI, you can [Stop managing your pipelines with code](#) on page 206.

Stop managing your pipelines with code

If you want to use the web UI, instead of a `.cd4pe.yaml` file, to manage your pipelines, you can quickly make this switch in the Continuous Delivery for Puppet Enterprise (PE) web UI.

1. In the Continuous Delivery for PE web UI, navigate to the control repo or module that has pipelines you are currently managing with a `.cd4pe.yaml` file.
2. At the top of the **Pipelines** section, click **Manage pipelines** > **Manage in the web UI**, and make any necessary adjustments to the pipeline settings.
3. Click **Save settings**.

The pipeline controls in the web UI are enabled. Continuous Delivery for PE ignores your `.cd4pe.yaml` file and you must make all pipeline changes in the web UI.

Tip: To avoid confusion, remove the `.cd4pe.yaml` file from the repo's root directory in source control.

If you want to go back to managing pipelines as code, you can [Convert your existing pipelines to code](#) on page 205.

`.cd4pe.yaml` file structure

When managing your pipelines with code, pipeline definitions are expressed in a structured format and stored in a `.cd4pe.yaml` file that is stored in your control repo or module repo.

Your `.cd4pe.yaml` file must use this structure:

- `spec_version`: The version of the pipelines-as-code file specification you've used to write this YAML file. Currently, only one specification version, `v1`, is available.

- `config`: Pipeline configuration settings for the control repo or module repo as a whole.
- `pipelines`: The names of pipelines you're creating, either the branch name (for [fixed-branch pipelines](#)) or the naming convention (for [regex branch pipelines](#)). Nested under each named pipeline are:
 - `triggers`: What source control activities the pipeline listens for.
 - `stages`: Parts of the pipeline. Each stage includes:
 - `name`: The stage's name.
 - `auto_promote`: The promotion condition that determines if and how the pipeline proceeds to the next stage after completing steps in the current stage.
 - `steps`: Details about the exact work the pipeline performs in each stage, such as jobs and deployments. Each step must specify a `type`. Additional specifications depend on the type.

Go to [.cd4pe.yaml file syntax](#) on page 210 for details about specific parameters and definitions. Go to [.cd4pe.yaml file validation](#) on page 209 to learn how to check if your file is well formed.

Here's an example of a complete `.cd4pe.yaml` file for a control repo. This file describes a regex branch pipeline and a main branch pipeline. Notice the [regex branch pipeline](#) is identified by the feature branch naming convention, `feature_.*/*`.

```
spec_version: v1
config:
  enable_pull_requests_from_forks: false
  deployment_policy_branch: production
  enable_pe_plans: true
pipelines:
  /feature_.*/*:
    triggers:
      - pull_request
      - commit
    stages:
      - name: "Lint/Parser validation"
        auto_promote: all_succeeded
        steps:
          - type: job
            name: control-repo-puppetfile-syntax-validate
          - type: job
            name: control-repo-template-syntax-validate
      - name: "Deploy feature environment"
        steps:
          - type: deployment
            pe_server: cdpe-delivery
            policy: feature_branch
  main:
    triggers:
      - pull_request
      - commit
    stages:
      - name: "Lint/Parser validation"
        auto_promote: all_succeeded
        steps:
          - type: job
            name: control-repo-puppetfile-syntax-validate
          - type: job
            name: control-repo-template-syntax-validate
          - type: job
            name: control-repo-hiera-syntax-validate
          - type: job
            name: control-repo-manifest-validate
      - name: "Impact Analysis"
        auto_promote: any_succeeded
        steps:
          - type: impact_analysis
```

```

    concurrent_compilations: 10
    deployments:
      - "Direct merge to Production"
      - "my custom deploy 1"
    - type: pull_request_gate
- name: "Deploy to lower UAT"
  auto_promote: false
  steps:
    - type: deployment
      policy: direct_merge
      name: "Direct merge to Production"
      target:
        type: node_group
        node_group_id: fcda068f-e499-44ef-81f2-255fc487b2e2
      pe_server: cdpe-delivery
      parameters:
        stop_if_nodes_fail: 10
        noop: false
        timeout: 60
    - type: deployment
      name: "my custom deploy 1"
      policy:
        source: control-repo
        name: deployments::canary
      target:
        type: node_group
        node_group_id: fcda068f-e499-44ef-81f2-255fc487b2e2
      pe_server: cdpe-delivery
      parameters:
        noop: true
        max_node_failure: 50
        canary_size: 4

```

Using a `.cd4pe.yaml` file with a module

A `.cd4pe.yaml` file for a module is almost identical to one for a control repo, with these exceptions:

- If the module pipeline includes an impact analysis step, you must include a pointer to the control repo used in the deployment associated with the impact analysis task.
- If the `.cd4pe.yaml` file includes a regex branch pipeline using the [Feature branch deployment policy](#) on page 220, the deployment step must include the name of the control repo associated with the module and the name of the control repo branch on which to base the feature branches Continuous Delivery for PE creates.

Here's an example of a complete `.cd4pe.yaml` file for a for a module with an impact analysis step. Notice, at the end of the pipeline, the `target` parameter for the `type: deployment` step includes a `control_repo` pointer for the impact analysis task.

```

spec_version: v1
config:
  enable_pull_requests_from_forks: false
pipelines:
  staging:
    triggers:
      - commit
    stages:
      - name: "Impact analysis"
        auto_promote: false
        steps:
          - type: impact_analysis
            deployments:
              - "Deployment to staging-rustler-1"
            concurrent_compilations: 10

```

```

    all_deployments: false
  - name: "Deployment to staging"
    steps:
      - type: deployment
        name: "Deployment to staging-rustler-1"
        policy:
          name: eventual_consistency
          timeout: 3600000
          concurrent_compilations: 0
          all_deployments: false
          pe_server: pe-github
          target:
            type: node_group
            node_group_id: 250fd263-8456-4114-b559-9c6d9fa27748
            control_repo: cdpe-code-rustler-app-2020

```

Here's an example of a module's regex branch pipeline using the feature branch deployment policy. The pipeline's name is the feature branch naming convention. At the end of the pipeline, the `control_repo` and `base_feature_branch` parameters in the `type: deployment` step are required for this type of pipeline. These parameters specify which control repo is associated with this module and which branch of that control repo is used to create feature branches when this pipeline is triggered.

```

spec_version: v1
config:
  enable_pull_requests_from_forks: false
pipelines:
  /feature_.*/:
    triggers:
      - commit
    stages:
      - name: "Validation stage"
        auto_promote: any_completed
        steps:
          - type: job
            name: "module-pdk-validate"
      - name: "Feature branch deployment stage"
        auto_promote: false
        steps:
          - type: deployment
            name: "Deployment to hot-dog-prod-2"
            policy:
              name: "cd4pe_deployments::feature_branch"
              all_deployments: false
              pe_server: "hot-dog-prod-2"
              control_repo: "cc-hot-dog-app"
              base_feature_branch: "production"

```

.cd4pe.yaml file validation

Every time you commit a change to a `.cd4pe.yaml` file in your control repo or module repo, Continuous Delivery for PE uses the YAML code to render the pipelines' definitions in the web UI. Before committing changes to your repos, use the Continuous Delivery for PE validation tool to make sure your YAML code is well-formed.

1. In the Continuous Delivery for PE web UI, go to a control repo or module where you're managing pipelines with code.
2. At the top of the **Pipelines** section, click **Manage pipelines**.
3. Select **Manage as code**. Continuous Delivery for PE displays your current pipelines in YAML format.
4. Update the code in the window with the changes you wish to make. Alternatively, you can delete the contents of the window and paste in code you've written elsewhere.

5. To check the syntax of your code, click **Validate**.
 - If your syntax is invalid, an error message about the location of the issue appears.
 - If your syntax is valid, **Copy to clipboard** is activated.
6. Once your changes are complete and successfully validated, copy them into the `.cd4pe.yaml` file in your source control or module and commit your change.

.cd4pe.yaml file syntax

Your `.cd4pe.yaml` file must be formatted properly.

Spec version

Every `.cd4pe.yaml` file begins with a `spec_version` declaration expressed as `v<VERSION_NUMBER>`. This sets the version of the Continuous Delivery for Puppet Enterprise (PE) pipelines-as-code specification used to write the file.

As there is only one version of the Continuous Delivery for PE pipelines-as-code specification, your file must begin with:

```
---
spec_version: v1
```

Config

The `config` section of the `.cd4pe.yaml` file defines global pipeline configuration settings for all pipelines for the control or module repo where the file is stored. You can use these keys in the `config` section:

config key	Description	Value	Default
<code>enable_pull_requests_from_forks</code>	Controls whether the repo's pipelines can be triggered by pull requests from forks	Boolean	<code>false</code>
<code>deployment_policy_branch</code>	(Optional) Specifies a branch where custom deployment policies are kept. If custom deployment policies exist on the specified branch, you can use them when building pipelines.	A branch name, as a string	N/A
<code>enable_pe_plans</code>	(Optional) Controls whether Bolt tasks can be included in custom deployment policies used in this repo's pipelines.	Boolean	<code>true</code>

For example, this `config` section is for a control repo. It allows pull requests from forks and serves custom deployment policies from the `production` branch:

```
config:
  enable_pull_requests_from_forks: true
  deployment_policy_branch: production
  enable_pe_plans: true
```

Pipelines

The `pipelines` section names each pipeline you're creating for a control repo or module. It requires key/value pairs where each key corresponds to a pipeline and the values are triggers, stages, and other pipeline contents. The key must use either a specific branch's name or a regular expression (for [regex branch pipelines](#)).

Important: Only one regular expression (regex) entry is allowed for each control/module repo.

For example, this skeleton `pipelines` section has two pipelines. The first pipeline is for the repo's `main` branch, whereas the second is a regex branch pipeline that applies to any branches whose names start with `feature_`. Notice the regex branch pipeline's key is the full regular expression surrounded by forward slashes.

```
pipelines:
  main:
    triggers:
    stages:

  /feature_.*/:
    triggers:
    stages:
```

Here is the same `pipelines` section with `triggers`, `stages`, and `steps` in each stage (each of these components are explained below):

```
pipelines:
  main:
    triggers:
      - pull_request
      - commit
    stages:
      - name: "Deploy to production"
        auto_promote: all_succeeded
        steps:
          - type: deployment
            policy: cd4pe_deployments::direct
            name: "Direct merge to Production"
            target:
              type: node_group
              node_group_id: fcda068f-e499-44ef-81f2-255fc487b2e2
            pe_server: cdpe-delivery
            parameters:
              stop_if_nodes_fail: 10
              noop: false

  /feature_.*/:
    triggers:
      - pull_request
      - commit
    stages:
      - name: "Lint/Parser validation"
        auto_promote: all_succeeded
        steps:
          - type: job
            name: control-repo-puppetfile-syntax-validate
```

Triggers

Each pipeline must have a `triggers` section that specifies the events in your source control system that start the pipeline.

The only allowed values are `pull_request` and `commit`. You can set either or both of these values. If no value is set, the pipeline does not run unless triggered manually. For example, this `main` branch pipeline is triggered by both pull requests and commits:

```
pipelines:
  main:
    triggers:
      - pull_request
      - commit
```

If you want your pipelines to be triggered by pull requests from forks, you must specify this in the `config` section.

Stages

Each pipeline must have a `stages` section.

The `stages` section accepts an array where each item is a stage in the pipeline. Each stage has a name, instructions about when and whether to auto-promote to the next stage in the pipeline, and a list of steps to be executed in parallel.

The `name` key is the stage's name. It accepts a string in quotes, and the default value is "Stage <STAGE_NUMBER>".

The `auto_promote` key provides instructions on when and if you want the pipeline to automatically proceed to the next stage. It accepts one of the following values:

- `false`: Requires manual promotion. The pipeline does not auto-promote.
- `all_succeeded`: Auto-promote only if all steps succeed. This is the default value.
- `any_succeeded`: Auto-promote if **any** step succeeds.
- `all_completed`: Auto-promote only if all steps complete (succeed or fail) and no steps are canceled.
- `any_completed`: Auto-promote if **any** step completes (succeeds or fails). Does not promote if all steps are canceled.

Steps

Each stage must have at least one step. A step defines a particular job to be performed in a pipeline stage.

Important: All steps run concurrently, so the order of steps in a stage doesn't matter. If you need certain steps to occur after others, use stages to separate them.

Every step is defined using a hash of key/value pairs. The `type` key is always required. Requirements for other keys depend on the step's `type`. The `type` key accepts these values:

- `job`: Specify a [job](#) available in the workspace where this pipeline exists.
- `pull_request_gate`: A conditional checkpoint that forcefully pauses the pipeline if the pipeline was triggered by a pull request. If the pipeline was triggered by a commit, the pipeline continues beyond the pull request gate.
- `impact_analysis`: Reports the potential impact the code change might have on nodes and configurations in deployments defined later in the pipeline.
- `deployment`: A Puppet code deployment.

For type: job steps: You must specify the job name as listed on the workspace's **Jobs** page in the web UI. For example, this step uses a job called `control-repo-puppetfile-syntax-validate`:

```
steps:
  - type: job
    name: control-repo-puppetfile-syntax-validate
```

For type: `pull_request_gate` steps: No additional keys are needed or available. To add a pull request gate to your pipeline, add this code to the `steps`:

```
steps:
  - type: pull_request_gate
```

For type: `impact_analysis` steps: There are three requirements. First, you must use one of these keys to specify deployments for which you want to access the potential impact:

- `all_deployments`: Assess the impact to all deployments in the pipeline. This key accepts a Boolean value, and the default value is `false`.
- `deployments`: List specific deployments you want to assess. Specify an array of strings where each item is the name of a deployment in your pipeline definition.
- `percentage_node_filter`: Run impact analysis on a percentage of nodes impacted by the incoming changes. This key accepts floating point values between 1 to 100 inclusive.

In addition to specifying deployments to assess, the pipeline must include a `type: deployment` step, and you must set the `control_repo` key on the deployment associated with the impact analysis step. For more information, refer to the information about `type: deployment` and the `target` key at the end of this page.

You can also set these optional keys on the `type: impact_analysis` step:

- `concurrent_compilations`: How many catalog compilations to perform at the same time. The default value is 10.

Tip: If your compilers are hitting capacity when performing an impact analysis, lower this number. However, lowering this number increases the impact analysis run time.

- `puppetdb_connection_timeout_sec`: The timeout period (in seconds) for requests to PuppetDB during an impact analysis task. The default value is 120.

The following example illustrates one impact analysis step that runs 10 compilations at a time on two specific deployments, and a second impact analysis step that runs on all deployments and times out PuppetDB requests after three minutes:

```
steps:
  - type: impact_analysis
    concurrent_compilations: 10
    deployments:
      - "Direct merge to Production"
      - "My PCI deployment"
  - type: impact_analysis
    all_deployments: true
    puppetdb_connection_timeout_sec: 180
```

For type: `deployment` steps: Specify these required keys:

- `policy`: The deployment policy used. For built-in deployment policies, provide one of the following policy key-value pairs:
 - `cd4pe_deployments::direct`
 - `cd4pe_deployments::eventual_consistency`
 - `cd4pe_deployments::feature_branch`
 - `cd4pe_deployments::rolling`

For custom deployment policies, provide a key-value pair defining the policy to be used.



CAUTION: Do not include sensitive parameters in custom deployment policies used in pipelines managed with code.

- `name`: The name, as a string in quotes, that you're giving to the deployment.

- `target`: The infrastructure (node group) the deployment targets. Requires the `type: node_group` key and one or more node group IDs as key/value pairs. In a module pipeline with an impact analysis task, this also requires a `control_repo` key. More information about this key, including how to find node group IDs, is provided below.

Restriction: Do not use this key for deployments from regex branch pipelines.

- `pe_server`: The name, as a string, of the PE instance the deployment targets. You must use the name shown in the Continuous Delivery for PE web UI at **Settings > Puppet Enterprise**.

The `parameters` key is optional. It accepts key/value pairs specifying relevant parameters defined by the deployment policy. To find out which parameters the policy accepts, go to [Built-in deployment policies](#) on page 216 or refer to the custom deployment policy's documentation.

For module regex branch pipelines using the feature branch deployment policy, you must also specify the `control_repo` (the name, as a string, of the control repo a module is associated with) and the `base_feature_branch` (the name, as a string, of the control repo branch you want Continuous Delivery for PE to use to create feature branches).

For example, this deployment step uses a built-in deployment policy:

```
steps:
  - type: deployment
    policy: cd4pe_deployments::direct
    name: "Direct merge to Production"
    target:
      type: node_group
      node_group_id: fcda068f-e499-44ef-81f2-255fc487b2e2
    pe_server: cdpe-delivery
    parameters:
      stop_if_nodes_fail: 10
      noop: false
      timeout: 60
```

This deployment step uses a custom deployment policy:

```
steps:
  - type: deployment
    policy:
      name: deployments::custom_policy1
      source: name-of-control-repo
    name: "Direct merge to Production"
    target:
      type: node_group
      node_group_id: fcda068f-e499-44ef-81f2-255fc487b2e2
    pe_server: cdpe-delivery
    parameters:
      policy_parameter1: 10
      policy_parameter2: false
```

This deployment step belongs to a regex branch pipeline and uses the feature branch deployment policy:

```
steps:
  - type: deployment
    policy:
      name: "cd4pe_deployments::feature_branch"
    name: "Deployment to hot-dog-prod-2"
    target:
      type: node_group
    pe_server: "hot-dog-prod-2"
    control_repo: "cc-hot-dog-app"
    base_feature_branch: "production"
```

The `target` key format depends on other characteristics of the pipeline, however; the `target` key is never used for deployments from regex branch pipelines.

When used for a module repo that does not include impact analysis or any control repo pipeline, the `target` key must include the `type` and `node_group_id`, such as:

```
target:
  type: node_group
  node_group_id: fcda068f-e499-44ef-81f2-255fc487b2e2
```

Presently, `node_group` is the only available `type`. To locate the `node_group_id`:

1. In the PE console, click **Node groups** (or **Classification** in PE versions prior to 2019.8.1).
2. Locate your target node group and click its name. A page with information about the node group opens.
3. In the page URL, locate and copy the alphanumeric string that follows `/groups/`. This is your node group ID.

When used for a module pipeline that includes an impact analysis step, the `target` key must also include the `control_repo` (in addition to the `type` and `node_group_id`), such as:

```
target:
  type: node_group
  node_group_id: fcda068f-e499-44ef-81f2-255fc487b2e2
  control_repo: cdpe-2018-1-pe-master-1-control
```

The `control_repo` value is the name of the control repo used in the deployment associated with the impact analysis task.

Deploy Puppet code

Use deployment policies to control how and when code changes are deployed. You can use built-in deployment policies or create your own custom policies. To apply a deployment policy and deploy code, you can either manually deploy code or use pipelines to automate deployments.

Deployment policies can't be triggered on their own. You specify which policy to use when you [Deploy code manually](#) on page 222 or [Deploy code automatically with a pipeline](#) on page 203. When you set up a deployment, you also specify a repository, branch, commit, environment, or other parameters that determine where to deploy your code, which version of your code to deploy, and how and where the deployment policy operates.

- [Built-in deployment policies](#) on page 216

Deployment policies are prescriptive workflows for deploying Puppet code that are built into Continuous Delivery for Puppet Enterprise (PE).

- [Custom deployment policies](#) on page 220

If the built-in deployment policies included with Continuous Delivery for Puppet Enterprise (PE) don't align with how your organization uses PE, you can write custom deployment policies tailored to your needs.

- [Deploy code manually](#) on page 222

Use the manual deployment workflow to push a code change to a specific node group on demand.

- [Deploy module code](#) on page 222

You can use a Continuous Delivery for Puppet Enterprise (PE) module pipeline to deploy new module code to your Puppet environments. To do this, you must add a `:branch => :control_branch` declaration to the module's entry in your control repo's Puppetfile.

- [Require approval for deployments to protected Puppet environments](#) on page 223

If your organization's business processes require manual review and approval before deploying Puppet code to certain environments, you can create an approval group consisting of users with the authority to review proposed deployments and manually approve or decline them.

Related information

[Deploy code automatically with a pipeline](#) on page 203

You can use a control repo pipeline to automatically deploy new code to a specified set of nodes every time a commit is made.

Built-in deployment policies

Deployment policies are prescriptive workflows for deploying Puppet code that are built into Continuous Delivery for Puppet Enterprise (PE).

When you set up a code deployment, you select the best deployment policy for your situation, and then Continuous Delivery for PE does all the Git heavy lifting to deploy your code to the correct nodes.

Deployment policies can't be triggered on their own. You specify which policy to use when you [Deploy code manually](#) on page 222 or [Deploy code automatically with a pipeline](#) on page 203. When you set up a deployment, you also specify a repository, branch, commit, environment, or other parameters that determine where to deploy your code, which version of your code to deploy, and how and where the deployment policy operates.

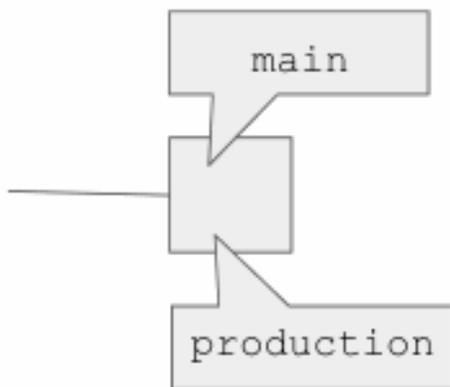
Deployment policy comparison

Continuous Delivery for Puppet Enterprise (PE) has four built-in deployment policies, each with a different branching workflow. The following table compares the deployment policy options to help you select the best policy for your circumstances.

Deployment policy	Description	Use cases
Direct deployment policy on page 216	Deploy code to one node group.	<ul style="list-style-type: none"> • Small or trivial changes • Changes that you are confident aren't going to cause issues when deployed • Deploying a specific commit to one environment
Temporary branch policy on page 218	Create a temporary environment and then deploy code to batches of nodes. You can monitor the outcome of each batch.	<ul style="list-style-type: none"> • Fully-tested changes • Validating changes by deploying to a temporary test environment • Workflows that don't require extensive historical data logging
Eventual consistency policy on page 219	Deploy new code and allow nodes to pick up the new code at their regular Puppet run intervals, rather than forcing the change through an immediate orchestrator job.	<ul style="list-style-type: none"> • Fully-tested changes • Large PE installations that have regularly-scheduled Puppet runs • When running additional orchestrator jobs is undesirable
Feature branch deployment policy on page 220	Deploy code from a feature branch to a dedicated feature branch environment.	Restriction: Only available for regex branch pipelines .
Custom deployment policies on page 220	Use Bolt plans to create a unique policy.	You have a special use case that isn't addressed by any of the built-in deployment policies.

Direct deployment policy

The direct deployment policy is the most basic of the four built-in deployment policies. Use this policy when you want to deploy a specific commit to a specific environment, and then use a Puppet run to deploy the change on only that specific environment.



When a deployment uses the direct deployment policy, Continuous Delivery for Puppet Enterprise (PE) performs the following actions:

1. **Synchronize code:** Using Code Manager, Continuous Delivery for PE synchronizes your selected code change with your PE instance by running `puppet-code deploy` on Puppet Server.
2. **Create temporary environment node group:** Continuous Delivery for PE creates a new environment node group as a child of the node group you selected when setting up the deployment. This new child node group inherits all the rules, configuration settings, and variables of its parent node group. A Puppet Query Language (PQL) query pins all nodes associated with the parent node group to the child node group.

Note: This environment node group is temporary, and it is automatically deleted after the code deployment. It is unlikely you'll ever interact directly with this environment node group.

3. **Run Puppet:** Continuous Delivery for PE uses the orchestrator to start a Puppet run on the nodes in the new environment node group. This Puppet run applies the new code to the nodes as quickly as Puppet Server's concurrency limits allow. You can monitor the Puppet run's progress on the deployment details page in the Continuous Delivery for PE web UI.
4. **Clean-up:** After the Puppet run, Continuous Delivery for PE deletes the temporary environment node group and moves the HEAD of the target branch (in your source control repository) to the commit you chose to deploy.

Direct deployment policy parameters

You can use the following optional parameters to customize your direct deployment. In the web UI, you can set these parameters when configuring a pipeline or manual deployment that uses the direct deployment policy. For pipelines-as-code, you can add these parameters to the your `.cd4pe.yaml` file as part of a `type: deployment step` declaration.

Parameter	Type	Description	Default value
<code>max_node_failure</code>	Integer	The maximum number of nodes that can fail before the deployment is stopped and reported as a failure.	Empty (unlimited nodes can fail)
<code>noop</code>	Boolean	Indicates whether to perform the Puppet run in no-op mode.	<code>false</code>

Parameter	Type	Description	Default value
<code>fail_if_no_nodes</code>	Boolean	Indicates whether to stop the deployment and report a failure if the selected node group does not contain any nodes.	<code>false</code>

Related information

[Deploy code manually](#) on page 222

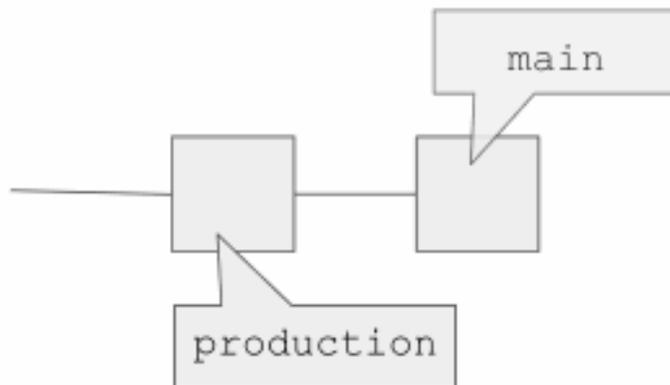
Use the manual deployment workflow to push a code change to a specific node group on demand.

[Deploy code automatically with a pipeline](#) on page 203

You can use a control repo pipeline to automatically deploy new code to a specified set of nodes every time a commit is made.

Temporary branch policy

When a deployment uses the temporary branch policy, Continuous Delivery for Puppet Enterprise (PE) creates a temporary Git branch (that contains the Puppet code you want to deploy) and a temporary environment node group (that contains the nodes you want to deploy the code to). The new code is then deployed to the nodes in the temporary environment node group in batches. This allows you to monitor the outcome of each batch and be certain that your new code works with each node before deploying the changes to the actual environment.



When a deployment uses the temporary branch policy, Continuous Delivery for PE performs these actions:

1. **Create a temporary branch:** Based on the commit you selected, Continuous Delivery for PE creates a new, temporary branch named `rolling_deployment_<ID>` in your source control repository.

Note: This branch is temporary, and it is automatically deleted after the code deployment. You'll probably never directly interact directly with temporary branches.

2. **Synchronize code:** Using Code Manager, Continuous Delivery for PE synchronizes your code change with your PE instance by running `puppet-code deploy` on Puppet Server.
3. **Create a temporary environment node group:** Continuous Delivery for PE creates a new, temporary environment node group as a child of the node group you selected when setting up the deployment. This new child

node group inherits all the rules, configuration settings, and variables of its parent node group. A Puppet Query Language (PQL) query pins all nodes associated with the parent node group to the child node group.

Note: This environment node group is temporary, and it is automatically deleted after the code deployment. You'll probably never directly interact with temporary environment node groups.

4. **Run Puppet:** Continuous Delivery for PE uses the PE orchestrator to run Puppet on the nodes in the new environment node group. Puppet runs on a few nodes at a time, based on the stagger settings you specified when setting up the deployment. You can monitor each Puppet run's progress on the deployment details page in the Continuous Delivery for PE web UI.
5. **Clean up:** After all Puppet runs end, Continuous Delivery for PE deletes the temporary environment node group, moves the HEAD of the target branch (in your source control repository) to point to your chosen commit, and deletes the temporary Git branch that was created at the beginning of this process.

Temporary branch deployment policy parameters

You can use the following optional parameters to customize your temporary branch deployment. In the web UI, you can set these parameters when configuring a pipeline or manual deployment that uses the temporary branch deployment policy. For pipelines-as-code, you can add these parameters to the your `.cd4pe.yaml` file as part of a `type: deployment` step declaration.

Parameter	Type	Description	Default value
<code>batch_delay</code>	Integer	How many seconds to wait between each deployment batch.	60
<code>batch_size</code>	Integer	How many nodes to include in each deployment batch.	10
<code>max_node_failure</code>	Integer	The maximum number of nodes that can fail before the deployment is stopped and reported as a failure.	Empty (unlimited nodes can fail)
<code>noop</code>	Boolean	Indicates whether to perform the Puppet runs in no-op mode.	false
<code>fail_if_no_nodes</code>	Boolean	Indicates whether to stop the deployment and report a failure if the selected node group or target environment does not contain any nodes.	true

Related information

[Deploy code manually](#) on page 222

Use the manual deployment workflow to push a code change to a specific node group on demand.

[Deploy code automatically with a pipeline](#) on page 203

You can use a control repo pipeline to automatically deploy new code to a specified set of nodes every time a commit is made.

Eventual consistency policy

Use the eventual consistency policy to deploy new Puppet code that nodes can pick up during their next scheduled check-in, rather than immediately applying the changes with a dedicated Puppet run. With this policy, the impacted nodes pick up the new code over time and eventually all become consistent with the new code.



CAUTION: If you have nodes with infrequent check-in schedules, deployments using the eventual consistency policy can take a long time to reach those nodes. If you need code changes applied more rapidly, consider using the [Temporary branch policy](#) on page 218 or [Direct deployment policy](#) on page 216.

When a deployment uses the eventual consistency policy, Continuous Delivery for Puppet Enterprise (PE) performs the following actions:

1. **Synchronize code:** Using Code Manager, Continuous Delivery for PE synchronizes your selected code change with your PE instance by running `puppet-code deploy` on Puppet Server.
2. **Apply changes during scheduled Puppet runs:** Rather than using the orchestrator to start a Puppet run to specifically deploy the new code, the code is distributed to nodes over time. The new code is delivered to each impacted node according to their regular check-in schedule.

Eventual consistency deployment policy parameters

There are no additional parameters associated with the eventual consistency deployment policy.

Related information

[Deploy code manually](#) on page 222

Use the manual deployment workflow to push a code change to a specific node group on demand.

[Deploy code automatically with a pipeline](#) on page 203

You can use a control repo pipeline to automatically deploy new code to a specified set of nodes every time a commit is made.

Feature branch deployment policy

The feature branch deployment policy deploys code changes you've made on a feature branch to a Puppet environment with the same name as the feature branch.

Restriction: The feature branch deployment policy is only available for [regex branch pipelines](#).

When a deployment in a regex branch pipeline uses the feature branch deployment policy, Continuous Delivery for Puppet Enterprise (PE) deploys code to the feature branch environment. Using Code Manager, Continuous Delivery for PE deploys the commit in the pipeline to a Puppet environment with the same name as the feature branch that triggered the pipeline run. If an environment with the feature branch's name doesn't already exist, Continuous Delivery for PE creates this environment during the deployment.

Feature branch deployment policy parameters

There are no additional parameters associated with the feature branch deployment policy.

Related information

[Deploy code manually](#) on page 222

Use the manual deployment workflow to push a code change to a specific node group on demand.

[Deploy code automatically with a pipeline](#) on page 203

You can use a control repo pipeline to automatically deploy new code to a specified set of nodes every time a commit is made.

Custom deployment policies

If the built-in deployment policies included with Continuous Delivery for Puppet Enterprise (PE) don't align with how your organization uses PE, you can write custom deployment policies tailored to your needs.

When you create a custom deployment policy, you declare the steps that need to happen to deploy your Puppet code changes. Because custom deployment policies use Bolt plans, you can configure your policies to automate your deployment workflow processes. This means your custom policies can:

- Include Git processes in your chosen source control provider.
- Use Puppet orchestrator to make changes to your node groups.
- Run Bolt tasks and plans

- Issue automatic notifications via the third-party services, such as Slack and ServiceNow

Custom deployment policies use the `puppetlabs-cd4pe_deployments` module, which is built into Continuous Delivery for PE. This means you don't need to download or install anything in order to use custom deployment policies. For a full list of functions and variables you can use in your custom deployment policies, refer to the `puppetlabs-cd4pe_deployments` module's [README](#) and [REFERENCE](#) on GitHub. Also in the module's GitHub repository, we've [rewritten the built-in deployment policies as custom policies](#) so you can use these examples when writing your own policies.

Warnings and restrictions



CAUTION: Custom deployment policies are a beta feature. Documentation might be incomplete and you might experience some unexpected behavior or bugs. Please explore them at your own risk.

Restriction: To make your custom deployment policies available in the Continuous Delivery for PE web UI, you must have enabled webhooks between Continuous Delivery for PE and your source control system.



CAUTION: Do not include sensitive parameters in custom deployment policies that you use in pipelines-as-code.

Related information

[Integrate with source control](#) on page 146

Integrate your source control system with Continuous Delivery for Puppet Enterprise (PE) by following the instructions for your source control provider.

[Construct pipelines from code](#) on page 204

If you manage your pipelines with code, rather than in the web UI, you can maintain a record of pipeline changes over time. When you choose to manage pipelines with code, a `.cd4pe.yaml` file, containing the pipelines' definitions for a control repo or module, is stored in your source control system alongside the Puppet code for that control repo or module.

[Built-in deployment policies](#) on page 216

Deployment policies are prescriptive workflows for deploying Puppet code that are built into Continuous Delivery for Puppet Enterprise (PE).

Add a custom deployment policy to your control repo

Each custom deployment policy file must be stored in a module, named `deployments`, in the control repo that uses the policy.

Before you begin

By default, Continuous Delivery for Puppet Enterprise (PE) looks for custom deployment policies on the `production` branch. If your control repo doesn't have a `production` branch, or you want to store custom deployment policies on a different branch, you must **manage your pipelines as code** and **tell Continuous Delivery for PE where to find your custom deployment policy files**.

To direct Continuous Delivery for PE to branch other than `production`:

1. Open the pipeline's `.cd4pe.yaml` file.
2. Locate the `config` section.
3. Set `deployment_policy_branch` to the branch where the custom deployment policy files are stored.

For information about pipelines-as-code and `.cd4pe.yaml` files, refer to [Construct pipelines from code](#) on page 204 and [.cd4pe.yaml file structure](#) on page 206.

Important: If your control repo doesn't have a `production` branch, or you want to store custom deployment policies on a different branch, you must use pipelines-as-code and specify the `deployment_policy_branch` setting, as explained above.

These requirements don't apply if your control repo has a `production` branch **and** you'll store your custom deployment policy files on that branch.

1. Write your custom deployment policy, as explained in the `puppetlabs-cd4pe_deployments` module's [README](#) on GitHub.
2. In your control repo, go to the branch where your custom deployment policy files are stored. This is either the `production` branch or another branch you specified in your pipeline's `.cd4pe.yaml` file.

Remember: If your control repo doesn't have a `production` branch, or you want to store custom deployment policies on a different branch, you must use `pipelines-as-code` and specify the `deployment_policy_branch` setting, as explained above.

3. Create a new Puppet module named `deployments`, and store it in one of the following directories in your control repo:

- `/modules`
- `/site`
- `/site-modules`

You must store the module in one of these specific directories so Continuous Delivery for PE can find and run your custom deployment policies.

4. Add your custom deployment policy file to the `/plans` directory in the `deployments` module.

Deploy code manually

Use the manual deployment workflow to push a code change to a specific node group on demand.

Tip: You can also [Deploy code automatically with a pipeline](#) on page 203.

1. In the Continuous Delivery for Puppet Enterprise (PE) web UI, go to **Control repos** and select the control repo you want to deploy code from.
2. Click **Manual actions** > **New deployment**.
3. Select the branch that has the code changes you want to deploy, select the commit you want to deploy, and then select your PE instance.

Tip: When setting up a manual deployment for a regex branch pipeline, the commit at the HEAD of your feature branch is automatically selected.

4. Select the Puppet environment you want to deploy code to.
5. Select a deployment policy.
You can use one of the [Built-in deployment policies](#) on page 216 or create [Custom deployment policies](#) on page 220.
6. Optional: Depending on the selected deployment policy, set termination conditions for the deployment, and choose the number of nodes that can fail before the deployment is stopped.
7. Give the deployment a name, and then click **Deploy**.
You can monitor the deployment's progress on the **Deployment details** page that opens after you click **Deploy**.

Related information

[Deploy code automatically with a pipeline](#) on page 203

You can use a control repo pipeline to automatically deploy new code to a specified set of nodes every time a commit is made.

Deploy module code

You can use a Continuous Delivery for Puppet Enterprise (PE) module pipeline to deploy new module code to your Puppet environments. To do this, you must add a `:branch => :control_branch` declaration to the module's entry in your control repo's Puppetfile.

Continuous Delivery for PE uses the [Eventual consistency policy](#) on page 219 to deploy module code to your Puppet environments. When you trigger a module code deployment, Continuous Delivery for PE creates a new branch in your module repository with the same name as your target Puppet environment. This new branch contains the module code you want to deploy. Then, Continuous Delivery for PE triggers Code Manager. Code Manager reads the module's `:branch => :control_branch` declaration in the control repo's Puppetfile and adds the new module code to the control repo. The new module code is delivered to each node in the specified Puppet environment during each node's next scheduled Puppet run.

1. Open the Puppetfile that includes the module you want to deploy. Add a `:branch => :control_branch` declaration to the module's section of the Puppetfile. For example:

```
mod 'apache',
  :git => 'https://github.com/puppetlabs/puppetlabs-apache',
  :branch => :control_branch,
  :default_branch => 'main'
```

To learn more about the `:branch => :control_branch` declaration, refer to [Declare content from a relative control repo branch](#) in PE's Code Manager documentation.

2. **If you are running PE version 2019.0.z or earlier:** You must make two additional changes to the Puppetfile.
 - a) Add the following code to the top of the Puppetfile:

```
def default_branch(default)
  begin
    match = /(.)_(cdpe|cdpe_ia)_\d+
    $/.match(@librarian.environment.name)
    match ? match[1]:default
  rescue
    default
  end
end
```

- b) Add the following `:branch` and `:default_branch` entries to the module's entry in the Puppetfile:

```
:branch => :control_branch,
:default_branch => default_branch('main')
```

3. In the Continuous Delivery for PE web UI, go to **Modules** and select the module you want to deploy.
4. If you haven't already done so, [Create a pipeline](#) on page 201 for your module.
5. Click **Add stage** and select **Deployment**.
6. Select your PE instance and choose the Puppet environment where you want to deploy the module code.
7. Click **Add deployment**.

Your module pipeline can now deploy new module code to the chosen Puppet environments when the pipeline is triggered.

Require approval for deployments to protected Puppet environments

If your organization's business processes require manual review and approval before deploying Puppet code to certain environments, you can create an approval group consisting of users with the authority to review proposed deployments and manually approve or decline them.

Before you begin

- Make sure you have super user permissions. You can't create approval groups if you don't have super user permissions.
- If it is not already configured, [Configure SMTP](#) on page 161 for your Continuous Delivery for Puppet Enterprise (PE) installation.
- Identify the users you want to designate as authorized approvers.

- Identify the Puppet environments that need to require manual approval for code deployments.

Environments that require manual approval for deployments are known as *protected environments*. The authorized approvers (who belong to an approval group) review all proposed deployments to the protected environments, and they manually approve or decline each deployment. Deployments to protected environments don't proceed until an approver makes a decision.

You can create multiple approval groups if you want different users to authorize deployments to different environments.

1. Create an approval group:

- In the Continuous Delivery for PE web UI, go to **Settings**.
- Switch to the **Groups** tab and click **Create new group**.
- Enter a group name (such as `Approvers`) and description, then click **Save**.
If you're creating multiple approval groups, you might want more specific names, such as `Production Approvers`.
- In each permissions category, select the permissions you want to assign to the approval group, then click **Save and add users**.

Important: At minimum, the approval group must have the **List** permission for **Control repos** so they can view and approve/deny deployments.

- On the **Add users** page, select the individuals you want to review deployments to protected environments. You can search for users by user name or email address.
 - Repeat these steps if you want to create more approval groups.
- #### 2. Specify which Puppet environments require manual approval for deployments:
- In the Continuous Delivery for PE web UI, go to **Settings > Puppet Enterprise**.
 - Locate the **Protected environments** section for the relevant PE integration.
 - Click **Add protected environment**.
 - Select the Puppet environment for which you want to require manual approval for deployments.
 - Select the approval group you want to review deployments to this environment.
 - Click **Add Protected Environment**.
 - Repeat these steps to designate additional protected environments, then click **Done**.

After creating approval groups and designating protected environments, each time a deployment to a protected environment is triggered (either manually or through a pipeline), the members of the environment's designated approval group get an email and a message in the **Message center** alerting them that a deployment requires manual review.

One member of the approval group must review the deployment's **Details** page, click **Provide approval decision**, and choose whether to approve or decline the deployment. If approved, the deployment process proceeds. If declined, the process terminates. A record of the decision is added to the deployment's **Details** page.

Review node inventory

The nodes from all Puppet Enterprise (PE) instances integrated with a workspace are shown on that workspace's **Nodes** page. This page gives you a federated view of all the nodes that your workspace deploys code to.

Customize your node table

The **Nodes** page lists basic information about every node from every Puppet Enterprise (PE) instance integrated with your workspace. You can customize your page view to display node facts relevant to your work.

By default, the node table shows this information about each node:

- Node name.

Tip: Click the node name to view all facts and PE reports for the node.

- Time since the node's most recent PE report was generated.
- [Node run status](#) for the most recent Puppet run.
- The PE server the node is associated with.
- The value of the `ipaddress` fact.
- The value of the `operatingsystem` fact.

You can add additional columns to the node table to show other fact values (including structured fact values), and you can remove irrelevant columns.

1. In the Continuous Delivery for PE web UI, click **Nodes**.
2. Click **Add/Remove columns** to open the column selector.
3. To add a column, search for a fact value and click **Add**. The fact value appears in the list of columns.
4. Use the check marks next to the columns list to select the columns you want to show. Unchecked columns are hidden.
5. Click **Apply** to apply your changes to the table.

Create filters to focus on specific node sets

Build basic or compound (multi-element) filters to help you answer complex queries about your nodes. By combining multiple filters with the AND and OR logical operators, you can target specific node sets.

Important:

Due to a processing limitation, you can apply filters to a specific Puppet Enterprise (PE) primary server or to all integrated PE primary servers. Attempting to add a filter to two or more specific primary servers triggers a `Cannot add this filter` warning.

For example, the following filter is valid because it specifies only one primary server and filters the nodes on that server by their operating system.

```
PE primary server = my-server AND Operating system = purpleOS
```

The following filter is invalid:

```
PE primary server = my-server OR Operating system = purpleOS
```

While the `PE primary server` parameter only specifies one server, using the OR operator ignores the first filter and attempts to apply the `Operating system` filter to all primary servers.

1. In the Continuous Delivery for Puppet Enterprise (PE) web UI, click **Nodes**.

- Click **Create filter**, select a filter category, define the filter values:

Category	Values
PE primary server	Select from a list of integrated PE instances
Change status for the most recent Puppet run	Select from a list of statuses
No-op status for the most recent Puppet run	Noop or Enforced
Operating system	An operator and a case-sensitive value
Node group	Select a PE primary server and select from available node groups
Factor fact value	A fact name, an operator, and a case-sensitive value

- Click **Add**, and the filter appears on the **Nodes** page. Then:

- **If you only need one filter:** Go to step 4.
- **If you want to add another basic filter:** Click **Add new filter**, configure the filter, click **Add**, and then click the logical operator between the filters (**AND** or **OR**) to specify whether nodes must match one or both filters.
- **If you want to build a compound filter:** Click **Create compound filter**  on your basic filter, configure the filter, click **Add**, and then click the logical operator (**AND** or **OR**) to select whether nodes must match one or both filters. Compound filters are like parenthesis in your query.

Note: You cannot mix logical operators within a single compound filter. To create queries with more complex logic, use a combination of basic and compound filters.

- Click **Apply filters** to apply the filters to your node table.

To modify the filters, click **Edit filters**.

To remove filters, click **Edit filters** (if you're not already in edit mode), and click **Remove filter** (the small blue x icon to the right of each individual filter).

Build a fact chart

Fact charts are visual representations of the distribution of Factor fact values across all nodes managed by the Puppet Enterprise (PE) instances you've integrated with Continuous Delivery for PE. You can add custom fact charts to any saved view for quick reference.

Select the Factor facts that are important to your organization and visualize the distribution of unique values for a given fact in a fact chart. For example, you can create a fact chart showing how many Windows Server versions you currently have in your inventory and how many servers are on each unique version.

You can create a fact chart for any Factor fact, including custom facts and structured facts.

- In the Continuous Delivery for PE web UI, click **Nodes**.
- Click **Add fact chart** and enter the name of the fact you want to visualize.
- Select whether to exclude nodes that do not use the specified fact from the fact chart.

If the fact chart includes all nodes, the nodes without the specified fact are shown in the **Fact absent** section of the resulting fact chart.
- Click **Save**. The new fact chart appears in the fact chart section of the page.
- To edit or remove fact charts:
 - Click **More actions**  on the fact chart you want to edit or remove.
 - Click **Edit** or **Remove**.

In addition to the legend on each fact chart, you can hover over the fact charts to see details about the number and percentage of nodes with the specified fact value.

[Save custom views](#) on page 227 to save fact charts for future reference and share them with your colleagues.

Save custom views

After customizing the nodes table and creating fact charts, you can save your customized view for later reference. Saved views are automatically shared with all members of your workspace, and each member can save their favorite views for quick access.

1. Customize the **Nodes** page:

- [Customize your node table](#) on page 224
- [Create filters to focus on specific node sets](#) on page 225
- [Build a fact chart](#) on page 226

2. Click **Save** > **Save as new** to create a new saved view.

3. Enter a name and optional description for the view.

Saved views are added to the library of saved views created by everyone in your workspace. All workspace members can see and use these saved views.

4. Optional: Designate this saved view as one of your favorites.

Marking a saved view as a favorite prioritizes that view on the **Saved views** page and when you **Select a different view** from the **Nodes** page. Each workspace member has their own list of favorite views.

5. Click **Save**.

6. You're redirected to the **Saved views** page where you can access all saved views for this workspace. In the **Actions** column you can:

- Use the **Star** icon to add  or remove  views from your favorites.
- Use the **Trash** icon  to delete a view you created.

Remember: Because all workspace members share all saved views, make sure no one else is using a view before you delete it.

7. Click a view's name open the **Nodes** page with that view applied.

From the **Nodes** page, you can click **Select a different view** to quickly switch views.

Review activity

Activity values across all the Puppet Enterprise (PE) instances integrated within a workspace are shown on the **Activity** report.

Activity report

The activity report gives you a federated view of activity and changes that happen across your Puppet Enterprise (PE) instances. By default the report gathers data over the previous seven days. You can change this time period by selecting a start and end date.

Note: You can configure the activity report using the APIs available under the `ValueReportingV1` tag in the [public Continuous Delivery for PE API](#).

By default, the activity report shows information about:

Values over time

- **NODES PATCHED** – Number of nodes that have been patched.
- **TASKS RAN** – Number of times a task has been run.
- **PLANS RAN** – Number of times a plan has been run.
- **CORRECTIVE CHANGES** – Number of times a corrective change has been made.
- **INTENTIONAL CHANGES** – Number of times an intentional change has been made.

Time saved

Bar chart that depicts the number of hours saved per value. Each value has a default of 90 minutes associated with it to estimate the time saved. You can change the default via an API. Task and plan inputs can be granular to provide different inputs per task and per plan.

Automated values

- **MODULES DEPLOYED** – Average number of modules deployed.
- **PUPPET SOURCES MANAGED** – Average number of Puppet sources being managed.
- **NODES CHECKED FOR PATCHES** – Average number of nodes checked for patches.

1. In the Continuous Delivery for PE web UI, click **Activity**.
2. Select the **Start Date** for the time period you want to check.
3. Select the **End Date** for the time period you want to check. After you select the end date the page shows you the number of activities that have occurred in that period, the time saved for each activity, and the automated activities that have been managed.

Support and troubleshooting

This section includes guidance for troubleshooting issues with Continuous Delivery for Puppet Enterprise (PE), some less common configuration methods you might need to use in specific circumstances, and information about working with the Puppet Support team.

- [Troubleshooting Continuous Delivery for PE](#) on page 228

Use this guidance to troubleshoot issues you might encounter with your Continuous Delivery for Puppet Enterprise (PE) installation.

- [Getting support](#) on page 234

The Support team at Puppet provides support for all features and capabilities included in Continuous Delivery for Puppet Enterprise (PE).

Troubleshooting Continuous Delivery for PE

Use this guidance to troubleshoot issues you might encounter with your Continuous Delivery for Puppet Enterprise (PE) installation.

Important: If your PE instance has a replica configured for disaster recovery, Continuous Delivery for PE is not available if a partial failover occurs. Learn more at [What happens during failovers](#) in the PE documentation. To restore Continuous Delivery for PE functionality, you must promote the replica to primary server.

Look up a source control webhook

Continuous Delivery for PE creates a webhook and attempts to automatically deliver it to your source control system when you add a new control repo or module to your workspace. You can look up this webhook if you ever need to manually add (or re-add) it to your source control repository.

1. In the Continuous Delivery for PE web UI, click **Control Repos** or **Modules**, and select the control repo or module whose webhook you want to view.
2. In the **Pipelines** section, click **Manage pipelines**.
3. If you're using pipelines-as-code, click **Manage in the web UI**. This temporarily converts your pipeline code to the web UI format so you can copy the webhook. After copying the webhook, don't save any changes and make sure you switch back to **Manage as code** before exiting the page.

Tip: If you use pipelines-as-code, make sure you don't save any changes, and make sure you switch back to **Manage as code** before exiting the page. Your pipeline code isn't affected as long as you don't save.

4. In the **Automation webhook** section, copy the full webhook URL. This URL represents the unique webhook that connects this control repo or module in Continuous Delivery for PE with its corresponding repository in your source control system.

Add the webhook to the corresponding repository in your source control system, according to the source control provider's documentation. Usually, webhooks are managed in the repository's settings.

Manually configure a Puppet Enterprise integration

When you add credentials for a Puppet Enterprise (PE) instance, Continuous Delivery for PE attempts to look up the endpoints for PuppetDB, Code Manager, orchestrator, and node classifier, and it attempts to access the primary SSL certificate generated during PE installation. If this information can't be located, such as in cases where your PE instance uses customized service ports, you must enter it manually.

This task assumes you have completed the steps in [Add your Puppet Enterprise credentials](#) and have been redirected to the manual configuration page.

1. In the **Name** field, enter a unique friendly name for your PE installation.

Tip: If you need to work with multiple PE installations within Continuous Delivery for PE, the friendly names help you differentiate which installation's resources you're managing.

2. In the **API token** field, paste a PE access token for your "Continuous Delivery" user. Generate this token using the `puppet-access` command or the RBAC v1 API.

For instructions on generating an access token, see [Token-based authentication](#) in the PE documentation.

3. In the five **Service** fields, enter the endpoints for your PuppetDB, Code Manager, orchestrator, and node classifier services:
 - a) In the PE console, go to **Status** and click **Puppet Services status**.
 - b) Copy the endpoints from the Puppet Services status monitor and paste them into the appropriate fields in Continuous Delivery for PE. Omit the `https://` prefix for each endpoint, as shown in the table below:

Service	PE console format	Continuous Delivery for PE format
PuppetDB service	<code>https:// sample.host.puppet:8081</code>	<code>sample.host.puppet:8081</code>
Puppet Server service	<code>https:// sample.host.puppet:8140</code>	<code>sample.host.puppet:8140</code>
Code Manager service	<code>https:// sample.host.puppet:8170</code>	<code>sample.host.puppet:8170</code> Important: Use port 8170 for Code Manager in Continuous Delivery for PE.
Orchestrator service	<code>https:// sample.host.puppet:8143</code>	<code>sample.host.puppet:8143</code>
Classifier service	<code>https:// sample.host.puppet:4433</code>	<code>sample.host.puppet:4433</code>

Tip: The Puppet Server service is used for impact analysis, among other processes. You can run impact analysis tasks on a compiler or load balancer instead of the primary server. **This is strongly recommended for PE installations that use compilers or load balancers as part of their architecture.** To do this, in the **Puppet Server service** field, enter the hostname of the compiler or load balancer at `:8140`. For example: `loadbalancer.example.com:8140`

4. To locate the master SSL certificate generated when you installed PE, run:

```
curl https://<HOSTNAME>:8140/puppet-ca/v1/certificate/ca --insecure
```

The `<HOSTNAME>` is your PE installation's DNS name.

5. Copy the entire certificate (including the header and footer) and paste it into the **CA certificate** field in Continuous Delivery for PE.
6. Click **Save Changes**.
7. Optional: Once the main PE integration is configured, [Configure impact analysis](#) on page 192.

If you want code deployments to skip unavailable compilers, go to [Enable compiler maintenance mode](#) on page 166.

Restart Continuous Delivery for PE

Continuous Delivery for PE is run in a managed Kubernetes cluster, and restarting the pod is an appropriate first step when troubleshooting.

To restart the pod, run:

```
kubectl rollout restart deployment cd4pe
```

For more information about the `kubectl rollout` command, refer to the [Kubernetes documentation](#).

Stop Continuous Delivery for PE

In rare circumstances, you might need to shut down, or force stop, Continuous Delivery for Puppet Enterprise (PE).

**CAUTION:**

Force stopping Continuous Delivery for PE can cause errors. Only use these commands under specific circumstances, preferably with guidance from Support.

We recommend that you initially try to [Restart Continuous Delivery for PE](#) on page 230, which is different and less disruptive than a force stop.

1. To stop Continuous Delivery for PE, run:

```
kubectl scale deploy cd4pe --replicas=0
```

For more information about the `kubectl scale` command, refer to the [Kubernetes documentation](#).

2. To start Continuous Delivery for PE after a force stop, run:

```
kubectl scale deploy cd4pe --replicas=1
```

Logs

Because Continuous Delivery for PE is run in a managed Kubernetes cluster, you must use the `kubectl logs` command to access the logs.

To access the Continuous Delivery for PE logs, run:

```
kubectl logs deployment/cd4pe
```

To access your installation's PostgreSQL logs, run:

```
kubectl logs statefulset/postgres
```

Trace-level logging

To enable or disable trace-level logging:

1. In Puppet Application Manager (PAM), go to the **Config** page.
2. Locate the **Advanced configuration and tuning** section.
3. Toggle the **Enable trace logging** setting according to your preference.

PE component errors in logs

The logs include errors for both Continuous Delivery for PE and the numerous PE components used by Continuous Delivery for PE. Sometimes an error in the Continuous Delivery for PE logs might actually indicate an issue with Code Manager, r10k, or another PE component.

For example, this log output indicates a Code Manager issue:

```
Module Deployment failed for
  PEModuleDeploymentEnvironment[nodeGroupBranch=cd4pe_lab,
nodeGroupId=a923c759-3aa3-43ce-968a-f1352691ca02, nodeGroupName=Lab
environment,
peCredentialsId=PuppetEnterpriseCredentialsId[domain=d3, name=lab-MoM],
pipelineDestinationId=null, targetControlRepo=null,
targetControlRepoBranchName=null,
targetControlRepoHost=null, targetControlRepoId=null].
Puppet Code Deploy failure: Errors while deploying environment
'cd4pe_lab' (exit code: 1):
ERROR -> Unable to determine current branches for Git source 'puppet' (/etc/
puppetlabs/code-staging/environments)
```

```
Original exception: malformed URL 'ssh://git@bitbucket.org:mycompany/control_lab.git'
at /opt/puppetlabs/server/data/code-manager/worker-caches/deploy-pool-3/ssh---git@bitbucket.org-mycompany-control_lab.git
```

For help resolving issues with PE components, go to the PE [Troubleshooting](#) documentation.

Error IDs in web UI error messages

Occasionally, error messages shown in the Continuous Delivery for PE web UI include an error ID and instructions to contact the site administrator. For example:

Remove 'docker' capability?

Are you sure you want to remove this capability?

Please contact the site administrator for support along with `errorId=[md5:35910b8341bf229e7040af2a1ce4bfe7 2020-02-06 19:33 1e6gebei26uda0x1k8uxtmz4vu]`

Remove

Cancel

For security reasons, these errors don't report any additional details. If you have root access to the Continuous Delivery for PE host system, you can search for the error ID the logs to learn more.

Duplicate job logs after reinstall

Job logs are housed in object storage after jobs are complete. If you reinstall Continuous Delivery for PE and you reuse the same object storage without clearing it, you might notice logs for multiple jobs with the same job number, or you might notice job logs already present when a new job has just started.

To remove duplicate job logs and prevent creation of duplicate job logs, make sure you clear both the object storage and the database when reinstalling Continuous Delivery for PE.

Name resolution

Continuous Delivery for PE uses CoreDNS for name resolution. In the logs, many can't reach and timeout connecting errors are actually DNS lookup failures.

To add DNS entries to CoreDNS, run the following command to open the `configmap` file in a text editor:

```
kubectl -n kube-system edit configmaps coredns
```

In the `configmap` file, add a `hosts` stanza directly after the `kubernetes` stanza according to the following format:

```
hosts /etc/hosts <DOMAIN> {
  <IP ADDRESS> <HOSTNAME> [ALIASES]
  fallthrough
}
```

Here's an example of a `configmap` file with the `hosts` stanza added:

```
apiVersion: v1
data:
  Corefile: |
    .:53 {
      errors
      health {
        lameduck 5s
      }
      ready
```

```

kubernetes cluster.local in-addr.arpa ip6.arpa {
  pods insecure
  fallthrough in-addr.arpa ip6.arpa
  ttl 30
}
hosts /etc/hosts puppetdebug.vlan {
  10.234.4.29 pe-201922-master.puppetdebug.vlan pe-201922-master
  fallthrough
}
prometheus :9153
forward . /etc/resolv.conf
cache 30
loop
reload
loadbalance
}
kind: ConfigMap
metadata:
  creationTimestamp: "2020-08-25T17:34:17Z"
  name: coredns
  namespace: kube-system
  resourceVersion: "10464"
  selfLink: /api/v1/namespaces/kube-system/configmaps/coredns
  uid: ba2907be-0067-4382-b103-fc248974719a

```

Looking up information about Continuous Delivery for PE

Use `kubectl` commands to access information about your Continuous Delivery for PE installation.

Look up the environment variables in use

To list the environment variables in use on your installation, run:

```
kubectl describe deployments.apps cd4pe
```

Note: For information on using environment variables to tune your Continuous Delivery for PE installation (such as adjusting HTTP and job timeout periods, changing the size of LDAP server searches, or enabling Git repository caching), refer to [Advanced configuration](#) on page 163.

Look up your Continuous Delivery for PE version

To print the version of Continuous Delivery for PE running in your installation, run:

```
kubectl get deployment cd4pe -o
  jsonpath='{.spec.template.spec.containers[0].image}' ; printf "\n"
```

Drain a node

Drain impacted nodes when performing maintenance on your Kubernetes cluster, such as upgrading system packages or rebooting the system.

To drain a node so you can perform system maintenance, run:

```
/opt/ekco/shutdown.sh
```

Resize an existing volume

The Continuous Delivery for PE database stores historical data and over time the accumulation of this data can exhaust disk space allocated to container volumes. You can adjust the size of existing volumes as needed to adjust for this data storage.

Before you begin

A PersistentVolumeClaim (PVC) can be increased in size, but not reduced. Attempting to reduce the size of a PVC results in an error.

The current free space in each PVC can be monitored on the administration console dashboard in the **Volume Available Storage (%)** Prometheus graph.

Ensure there is sufficient storage available to allocate the newly configured storage amount.

1. Set the PVC to be increased:

```
export PVC="data-minio-0"
```

2. Make sure the current size of the PVC is what you are expecting:

```
kubectl get pvc ${PVC} -o jsonpath="{.spec.resources.requests.storage}"
```

3. Update the PVC size. For example, to update the PVC storage to 20 GiB:

```
kubectl patch pvc ${PVC} -p '{"spec":{"resources":{"requests":{"storage":"20Gi"}}}}'
```

4. Restart the StatefulSet that use that PVC:

```
kubectl get pod -o json | jq ".items[] | select
  (.spec.volumes[]?.persistentVolumeClaim.claimName == \"${PVC}\")
  | .metadata.ownerReferences[] | select (.kind == \"StatefulSet\")
  | .name" | sort | uniq | xargs -rt kubectl delete sts --cascade=orphan
```

If there is no output from this command, then the StatefulSet could not be found.

5. Delete the Continuous Delivery for PE migration job:

```
kubectl delete job -l app.kubernetes.io/name=cd4pe-migrate
```

6. In the **Advanced configuration and tuning** section, update the storage size based on the PVC you updated. The possible target settings are:

- data-minio-0 = "Object store capacity"
- data-postgres-0 = "CD4PE PostgreSQL capacity"
- data-query-postgres-0 = "Estate Reporting PostgreSQL capacity"

7. Deploy the application. As the application is re-deployed, there may be disruption to the application as pods are restarted.

Reset root account password

If you need to reset your Continuous Delivery for PE root account password, use this process.

1. Log into the PAM UI and navigate to **Config**.
2. Update the password in the **Root account password** field.
3. Save the configuration and re-deploy Continuous Delivery for PE to update the password.

Getting support

The Support team at Puppet provides support for all features and capabilities included in Continuous Delivery for Puppet Enterprise (PE).

Report your support issues through our confidential [Support Portal](#). When you purchase Continuous Delivery for PE, you receive an account and log-on details for the portal, which includes access to the Support Knowledge Base.

Supported installation methods

The Puppet Support team can assist with the Continuous Delivery for PE installation process using the documented installation methods for the 4.x series:

- [Deploy Continuous Delivery for PE](#) on page 130
- [Deploy Continuous Delivery for PE offline](#) on page 131

The Puppet Support team only supports the approved installation methods listed on this page. They cannot assist you with Kubernetes configuration or container runtime issues unrelated to Continuous Delivery for PE.

Application support

The Puppet Support team supports all Continuous Delivery for PE features and capabilities, regardless of where the application is running. However, if a feature is not working correctly due to your runtime environment, the Support team reserves the right to reject the support request.

Creating a support bundle

When seeking support, you might be asked to generate and provide a support bundle. This bundle collects a large amount of logs, system information, and Continuous Delivery for PE diagnostics.

To create a support bundle:

1. In Puppet Application Manager (PAM), click **Troubleshoot** > **Generate a support bundle**.
2. You can generate the bundle automatically or manually:
 - **Automatic generation:** Click **Analyze Continuous Delivery for Puppet Enterprise** and wait while PAM generates the support bundle and uploads it to the **Troubleshoot** page.
 - **Manual generation:** Click the prompt to generate a custom command for your installation, run the command on your cluster, and follow the prompts to upload the bundle to PAM.
3. **Important:** Review the data collected in the bundle before forwarding it to the Support team. The bundle can contain sensitive information that you might want to redact.
4. Return to the **Troubleshoot** page, download the support bundle, and send it to your Puppet Support contact.

Create a support bundle from the command line

To collect the Continuous Delivery for PE support bundle from the command line, run:

```
kubectl support-bundle --load-cluster-specs --redactors=configmap/default/kotsadm-redact-spec/redact-spec,configmap/default/kotsadm-cd4pe-redact-spec/redact-spec
```

If you do not already have the support-bundle kubectl plugin installed, install it by using the command below:

```
curl https://krew.sh/support-bundle | bash
```

Or by installing [krew2](#) and running:

```
kubectl krew install support-bundle
```

Airgap bundles

This page lists available 4.x series `.airgap` bundles, which are used when installing Continuous Delivery for PE in an offline environment.

Version	Release date	Airgap bundle
4.35.0	29 April 2025	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.35.0.airgap
4.34.0	30 January 2025	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.34.0.airgap
4.33.0	30 October 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.33.0.airgap
4.32.0	29 August 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.32.0.airgap
4.31.0	10 July 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.31.0.airgap
4.30.2	14 June 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.30.2.airgap
4.30.1	22 May 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.30.1.airgap
4.30.0	7 May 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.30.0.airgap
4.29.2	21 March 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.29.2.airgap
4.29.1	21 February 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.29.1.airgap
4.29.0	8 February 2024	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.29.0.airgap
4.28.0	30 November 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.28.0.airgap
4.27.1	11 October 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.27.1.airgap
4.27.0	4 October 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.27.0.airgap
4.26.2	7 September 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.26.2.airgap
4.26.1	23 August 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.26.1.airgap
4.26.0	22 August 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.26.0.airgap
4.25.1	24 July 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.25.1.airgap
4.25.0	11 July 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.25.0.airgap
4.24.1	28 June 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.24.1.airgap
4.24.0	31 May 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.24.0.airgap

Version	Release date	Airgap bundle
4.23.1	2 May 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.23.1.airgap
4.23.0	18 April 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.23.0.airgap
4.22.2	21 March 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.22.2.airgap
4.22.1	16 March 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.22.1.airgap
4.22.0	8 March 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.22.0.airgap
4.21.1	6 February 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.21.1.airgap
4.21.0	25 January 2023	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.21.0.airgap
4.20.0	15 November 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.20.0.airgap
4.19.0	4 October 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.19.0.airgap
4.18.1	27 September 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.18.1.airgap
4.18.0	8 September 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.18.0.airgap
4.17.0	11 August 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.17.0.airgap
4.16.1	14 July 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.16.1.airgap
4.16.0	12 July 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.16.0.airgap
4.15.1	14 June 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.15.1.airgap
4.15.0	7 June 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.15.0.airgap
4.14.0	5 May 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.14.0.airgap
4.13.0	5 April 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.13.0.airgap
4.12.1	7 March 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.12.1.airgap
4.12.0	2 March 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.12.0.airgap
4.11.5	22 February 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.11.5.airgap

Version	Release date	Airgap bundle
4.11.4	14 February 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.11.4.airgap
4.11.2	2 February 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.11.2.airgap
4.11.1	20 January 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.11.1.airgap
4.11.0	20 January 2022	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.11.0.airgap
4.10.5	20 December 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.10.5.airgap
4.10.4	17 December 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.10.4.airgap
4.10.3	10 December 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.10.3.airgap
4.10.2	9 December 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.10.2.airgap
4.10.1	11 November 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.10.1.airgap
4.10.0	9 November 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.10.0.airgap
4.9.0	8 September 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.9.0.airgap
4.8.2	31 August 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.8.2.airgap
4.8.1	24 August 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.8.1.airgap
4.8.0	10 August 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.8.0.airgap
4.7.2	26 July 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.7.2.airgap
4.7.1	12 July 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.7.1.airgap
4.7.0	8 July 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.7.0.airgap
4.6.1	16 June 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.6.1.airgap
4.6.0	3 June 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.6.0.airgap
4.5.2	11 May 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.5.2.airgap
4.5.1	27 April 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.5.1.airgap

Version	Release date	Airgap bundle
4.5.0	22 April 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.5.0.airgap
4.4.2	13 April 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.4.2.airgap
4.4.1	29 March 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.4.1.airgap
4.4.0	11 March 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.4.0.airgap
4.3.3	23 February 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.3.3.airgap
4.3.2	3 February 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.3.2.airgap
4.3.1	26 January 2021	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.3.1.airgap
4.2.4	17 December 2020	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.4.airgap
4.2.3	17 November 2020	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.3.airgap
4.2.2	12 November 2020	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.2.airgap
4.2.1	5 November 2020	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.1.airgap
4.2.0	3 November 2020	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.0.airgap
4.1.3	15 October 2020	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.1.3.airgap
4.1.2	8 October 2020	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.1.2.airgap
4.1.1	29 September 2020	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.1.1.airgap
4.0.1	14 September 2020	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.0.1.airgap
4.0.0	25 August 2020	https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.0.0.airgap

Copyright and trademark notices

© 2024 Puppet, Inc., a Perforce company. All rights reserved.

Puppet and other identified trademarks are the property of Puppet, Inc., Perforce Software, Inc., or an affiliate. Such trademarks are claimed and/or registered in the U.S. and other countries and regions. All third-party trademarks are the property of their respective holders. References to third-party trademarks do not imply endorsement or sponsorship of any products or services by the trademark holder. Contact Puppet, Inc., for further details.