Continuous Delivery

Contents

Welcome to Continuous Delivery for Puppet Enterprise..... 5

Release notes.....5

Continuous Delivery for PE release notes.....6
Continuous Delivery for PE known issues..... 12
Release notes archive..... 13

Getting started with Continuous Delivery for PE.....20

Step 1: Install Continuous Delivery for PE..... 20
Step 2: Create your user account and set up a workspace..... 21
Step 3: Set up integrations and configure job hardware.....21
Step 4: Add a control repo.....22
Step 5: Set up a pipeline..... 22
Step 6: Set up an environment node group..... 22
Step 7: Deploy changes to your nodes..... 23
Step 8: Create an impact analysis report..... 23

Installing.....24

System requirements.....25
Continuous Delivery for PE architecture.....25
Hardware requirements.....26
Supported external databases.....26
Job hardware requirements.....26
Supported Puppet Enterprise versions.....27
Supported browsers.....27
Install Continuous Delivery for PE from the PE console.....28
Install Continuous Delivery for PE from the PE 2019.2.x or 2019.1.x console..... 28
Install Continuous Delivery for PE from the PE 2018.1.8+ console.....29
Advanced configuration options.....29
Automate upgrades of your Continuous Delivery for PE console installation..... 30
Install Continuous Delivery for PE using the `cd4pe` module.....31
Install Continuous Delivery for PE with the `cd4pe` module.....32
Configure Continuous Delivery for PE with a task.....33
Advanced configuration options.....35
Generate a trial license.....36
Alternative installation methods.....36
Install Continuous Delivery for PE using Docker..... 37
Configure a Continuous Delivery for PE module installation using classification.....39

Configuring and adding integrations..... 41

Integrate with source control.....42
Integrate with Azure DevOps.....42
Integrate with Bitbucket Cloud..... 42
Integrate with Bitbucket Server.....43
Integrate with GitHub.....44

Integrate with GitHub Enterprise.... 44
Integrate with GitLab.... 45
Integrate with Puppet Enterprise.... 45
Create a Continuous Delivery user and user role in PE.... 45
Add your Puppet Enterprise credentials.... 46
Configure impact analysis.... 47
Generate a dedicated Puppet Enterprise certificate.... 47
Install modules.... 48
Update classification.... 48
Add credentials to Continuous Delivery for PE.... 49
Configure job hardware.... 49
Configure job hardware with the web UI.... 50
Configure job hardware with `distelli.yml`.... 50
Configure global shared job hardware.... 51
Upgrade the Continuous Delivery agent.... 52
Analytics data collection.... 52
What data does Continuous Delivery for PE collect?.... 53
Opt out of analytics data collection when installing with the `cd4pe` module.... 53
Opt out of analytics data collection when installing with Docker.... 53
Configure LDAP.... 54
Create a new LDAP configuration.... 54
Create an LDAP group map.... 56
Configure SMTP.... 56
Configure SSL.... 57
Setting up a new SSL configuration.... 57

Managing workspaces.... 59
Best practices when creating workspaces.... 59
Set up a new workspace for a team.... 59

Testing Puppet code with jobs.... 60
What is a job?.... 60
Install job dependencies.... 61
Pre-built job reference.... 61
Sample non-Docker-based module jobs.... 63
Sample non-Docker-based control repo jobs.... 64

Constructing pipelines.... 65
Stages and tasks in pipelines.... 65
Set up a pipeline.... 66
Test code automatically with a pipeline.... 66
Test pull requests automatically.... 66
Deploy code automatically with a pipeline.... 67

Previewing the impact of code changes.... 68
Add impact analysis to a control repo pipeline.... 68
Add impact analysis to a module pipeline.... 69

Deploying Puppet code.... 70
Introducing deployments.... 70
Git branches and Continuous Delivery for PE.... 71

Create environment node groups....72
Deployment policies....73
Which deployment policy should I use?....73
Direct deployment policy....74
Temporary branch policy....75
Eventual consistency policy....76
Incremental branch policy....76
Blue-green branch policy....77
Deploy code manually....78
Deploy module code....78
Require approval for deployments to protected Puppet environments....79

Managing access....80
Create a user account....80
Adding users and creating groups....81
Add a user....81
Create a new group....81
Add or remove group members....81
Assign permissions to a group....82
Remove a user from a workspace....82
Permissions reference....82
Using the root console....84
Designate super users....84
Change a user's password....84
Reset a user's email address....85

Upgrading to 3.x....85

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.

To harness the full power of Puppet Enterprise, 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.

Helpful Continuous Delivery for PE docs links	Other useful places
Before you install Release notes - What's new, what's removed, what's resolved. System requirements - For software, browsers. **Install Continuous Delivery for PE** Install using the cd4pe module Install from the PE console - Simplified installation for users of current PE versions. Puppet License Manager - Generate a trial license for yourself or your team. **Configure and manage access** Configuring - Integrate with source control and PE, and configure impact analysis. User permissions and groups - Manage access with role-based access control. **Learn the basics** Getting started - Learn core concepts and basic workflows. **Continuously deliver Puppet code** Create a job to test code See the potential impact of new code Choose a deployment policy	**Docs for related Puppet products** Puppet Enterprise Open source Puppet Puppet Development Kit **Why and how people are using Continuous Delivery for PE** Read our ebook on Critical Considerations for Continuous Delivery Find Continuous Delivery for PE product information **Learn Continuous Delivery for PE and Puppet** Plan your skills-building path with our learning roadmap Find an online, in-person or self-paced class Get certified **Get support** Search the Support portal and knowledge base Upgrade your support plan **Share and contribute** Engage with the Puppet community Puppet Forge - Find modules you can use, and contribute modules you've made to the community. Open source projects from Puppet on GitHub

To send us feedback or let us know about a docs error, open a ticket (you'll need a Jira account) or email the docs team.

Release notes

New features, enhancements, known issues, and resolved issues for the Continuous Delivery for Puppet Enterprise 2.x release series.

- Continuous Delivery for PE release notes on page 6

These are the new features, enhancements, resolved issues, and deprecations for the Continuous Delivery for Puppet Enterprise 2.x release series.

- Continuous Delivery for PE known issues on page 12

These are the known issues for the Continuous Delivery for PE 2.x release series.

- Release notes archive on page 13

Release notes for Continuous Delivery for PE 2.x release series versions that are more than six months past the most recent 2.x release version are housed in the release notes archive.

Continuous Delivery for PE release notes

These are the new features, enhancements, resolved issues, and deprecations for the Continuous Delivery for Puppet Enterprise 2.x release series.

Version 2.18.4

Released 23 January 2020

New in this release:

- **Fewer stacktrace exceptions included in log files.** We've reduced the number of stacktrace exceptions that resulted from checking for dependencies and approvals. You'll no longer see long stacktrace errors for the following:

```
com.puppet.pipelines.cdpe.cdpeTaskUtils.CDPETaskInterruptedException:
 Dependency check attempt maxtime exceeded.
com.puppet.pipelines.cdpe.cdpeTaskUtils.CDPETaskInterruptedException:
 Approval check attempts maxiumum exceeded. Thread should yeild and try
 again.
```

Version 2.18.3

Released 31 October 2019

Resolved in this release:

- This release contains internal component upgrades that resolve high-risk vulnerabilities. For a complete list of upgraded components and resolved CVEs, see the CD4PE October 2019 Security Fixes page.

Version 2.18.2

Released 22 October 2019

Resolved in this release:

- This release contains internal component upgrades that resolve several high-risk vulnerabilities. For a complete list of upgraded components and resolved CVEs, see the CD4PE October 2019 Security Fixes page.

Version 2.18.1

Released 16 October 2019

Resolved in this release:

- Bitbucket Server push and pull request webhooks were not processed correctly.

Version 2.18.0

Released 16 September 2019

New in this release:

- **Impact analysis reports for module code.** Preview the potential impact to your infrastructure of a Puppet module code change before the new code is merged. After you add an impact analysis task to your module

pipeline, a detailed report is generated each time the pipeline is triggered, showing how the new module code will impact your nodes and resources. To get started, see Add impact analysis to a module pipeline.

Note: Impact analysis reports for module code are supported on PE 2018.1.9 and newer versions in the 2018.1 series, and PE 2019.1.1 and newer versions in the 2019.1 series.

Version 2.17.1

Released 3 September 2019

Resolved in this release:

- Module deployments could not be edited.

Version 2.17.0

Released 3 September 2019

New in this release:

- **Redesigned pipeline stage creation workflow.** We've updated and improved the processes for adding a new stage to your pipeline and adding new items to a pipeline stage. You no longer need to close and reopen the pipeline editing modal to add additional stages or items, and you can now add multiple jobs at once.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - The option to set a job to run on global shared job hardware, even if global shared job hardware is not available for your installation when the job is created.
  - When a change is manually promoted in a pipeline, the commit message is displayed.

Resolved in this release:

- SSH private keys for each workspace were not encrypted in the database.
- The deployment details page was not generated for manually triggered deployments.
- An error message was shown when you attempted to rename a workspace.
- When you added an impact analysis task to a pipeline, environments that do not support impact analysis were automatically selected.
- The stage number was incorrectly displayed when adding a deployment to a default pipeline.
- The full Continuous Delivery agent version number was not shown on the **Job Hardware** page or returned when you ran `distelli agent version`. Version 3.70.2 of the Continuous Delivery agent includes a fix for this issue.

Removed in this release:

- **Support for Puppet Enterprise version 2018.0.** PE 2018.0 has reached the end of its support lifecycle.

Version 2.16.0

Released 19 August 2019

New in this release:

- **Pipeline events summary on the control repo deployment details page.** To make it easier for deployment approvers to review the validation events performed on a proposed change, the new **What's happened in the pipeline so far?** section summarizes the successes and failures in each completed pipeline stage and includes the option to view further details.
- **Impact analysis reports include node names.** Impact analysis reports now show the names of the nodes impacted by a proposed change.

Resolved in this release:

- During the creation of LDAP group mappings, RBAC groups were not listed for renamed workspaces.
- The **Set up Continuous Delivery for PE** banner did not show that source control had been integrated after a Bitbucket Cloud instance was added to Continuous Delivery for PE.

- An inaccurate message about missing PE credentials was shown when you tried to add impact analysis to a pipeline before creating a deployment.
- In some cases, webhook auto-creation failed for Bitbucket Server.

Version 2.15.1

Released 6 August 2019

Resolved in this release:

- Deployments from Bitbucket Cloud failed for target branches that did not exist because Continuous Delivery for PE did not automatically create a target branch in these cases.

Version 2.15.0

Released 5 August 2019

New in this release:

- **Bitbucket Cloud support.** See Integrate with Bitbucket Cloud for instructions on how to create an OAuth application to start using your organization's Bitbucket Cloud repositories with Continuous Delivery for PE.
- **Delete workspaces.** A workspace's owner can now delete the workspace by visiting the **Workspace** tab in **Settings**. Super users can also delete workspaces.

Resolved in this release:

- Blank rows appeared in the workspaces menu.
- When configuring impact analysis after adding PE credentials on the **New Deployment** screen, a `PE credentials with the specified name already exist.` error was shown.

Version 2.14.2

Released 29 July 2019

Resolved in this release:

- The prefixed environment selection option did not appear for users with more than one PE instance integrated with Continuous Delivery for PE.

Version 2.14.1

Released 24 July 2019

Resolved in this release:

- If your trial or production license expired, you were unable to successfully upload a new license.

Version 2.14.0

Released 22 July 2019

New in this release:

- **Redesigned manual deployment workflow**. When you create a new manual deployment for control repos and modules, Continuous Delivery for Puppet Enterprise no longer prompts you to select a branch. The branch for your new deployment now automatically matches the pipeline you've selected. To override this setting, click **Edit** and select a new branch name.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - A larger log window on the **Job Details** page, allowing you to see more information before you have to scroll.

Resolved in this release:

- If you had only one PE instance integrated with Continuous Delivery for PE, and that PE instance contained prefixed environments, the prefixed environments did not appear when you created a deployment.
- Some buttons on the SAML, LDAP, and SSL configuration screens were blank.
- The option to add impact analysis task to a regex branch pipeline was included erroneously, and has been removed.
- Super users could not install global shared job hardware.

Version 2.13.4

Released 16 July 2019

Resolved in this release:

- In automated runs, module deployments to prefixed environments resulted in errors.

Version 2.13.3

Released 16 July 2019

Resolved in this release:

- Performing a manual module deployment to prefixed environments resulted in errors.

Version 2.13.2

Released 15 July 2019

Resolved in this release:

- In certain cases, impact analysis incorrectly reported that resources had changed.

Version 2.13.1

Released 12 July 2019

Resolved in this release:

- If any catalog did not successfully compile during an impact analysis task, the task failed.
- The presence of certain parameter value types in impact analysis reports caused page crashes.
- Changes to data types marked sensitive were shown as modified in some impact analysis reports.

Version 2.13.0

Released 11 July 2019

New in this release:

- **Redesigned impact analysis reports.** We've updated the format of impact analysis reports to improve usability and provide you with greater detail about how the resources on your nodes will be impacted as a result of code changes.
- **Generate impact analysis reports on demand.** You can now generate an impact analysis report for any change by clicking **New Impact Analysis** on a control repo's details page.
- **Trial mode.** When you set up Continuous Delivery for PE for the first time, you now have the option to select **Trial Mode**, which launches a seven-day trial period without requiring a license. At the end of the trial period, you are prompted to generate and upload a free 30-day trial license.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - When logging back into Continuous Delivery for PE, you are automatically returned to the workspace you last used.
  - When an upstream pipeline dependency fails, downstream jobs and deployments are now labeled **canceled** instead of **failed**.
  - An improved error message if you create a username that includes non-supported characters.

Resolved in this release:

- Impact analysis required reconfiguring after you upgraded Puppet Enterprise to version 2019.1.

Removed in this release:

- **Chronological event view.** We've removed the option to view a chronological event history for your control repo or module.

Version 2.12.2

Released 21 June 2019

Resolved in this release:

- If you did not create any groups in your Continuous Delivery for PE user account before upgrading to version 2.12.0 or 2.12.1, you were unable to access the workspace that was automatically created for your user account as part of the upgrade.

Version 2.12.1

Released 19 June 2019

Resolved in this release:

- If you renamed a workspace and then navigated to a page in the **Settings** menu, you were redirected your default workspace.
- User names were listed in the workspaces menu.
- The link to a newly created workspace in the workspaces menu didn't work properly until the page was refreshed.

Version 2.12.0

Released 17 June 2019

New in this release:

- **Workspaces.** Teams using Continuous Delivery for PE need to share common resources, such as control repos, modules, pipelines, and jobs. Workspaces is a redesign of the users and team model in previous versions of Continuous Delivery for PE, and was created to make sharing resources with team members simpler and more intuitive. To learn more about how to set up and use workspaces, see Managing workspaces.

  Note: If you're upgrading to version 2.12.0 from an older version of Continuous Delivery for PE, your existing user accounts have been converted to workspaces.
- **Redesigned navigation.** We've updated the navigation bar and moved it to the left side of your screen, bringing Continuous Delivery for PE into better visual and interface alignment with Puppet Enterprise.
- **Global shared job hardware.** You can now set up global shared job hardware servers that are available to all workspaces in your Continuous Delivery for PE installation. Once global shared job hardware is configured, you can instruct a job to use shared hardware when creating or editing the job. To get started, see Configure global shared job hardware.
- **Link to the root console for super users.** Super users will now find a **Root Console** link in the workspaces menu at the top of the navigation bar.
- **Add all members of an Active Directory or LDAP group to a workspace.** You can add existing LDAP or Active Directory groups to your installation's workspaces by clicking **Manage Groups** in the **Single Sign On** tab in the root console's **Settings** area.
- **Support for unauthenticated SMTP.** When configuring SMTP, you are no longer required to enter a username and password.
- **Select from available hardware capabilities.** When creating a new job, you're now shown a checklist list of available job hardware capabilities. This list is automatically populated by the capabilities you've set on your workspace's job hardware, as well as the capabilities on any available global shared job hardware.

- **Usability improvements.** We've made numerous small usability improvements, including:
  - In the **Edit Job** pane, pressing **Enter** no longer saves and exits an in-progress job.
  - When you click the link to a commit SHA on a module's deployment details page, the link opens in a new browser tab.
  - Deployments to protected environments now wait until all dependencies are complete before issuing an approval request.
  - You'll see more helpful error messages when an impact analysis task fails.

Resolved in this release:

- Prefixed environments failed to deploy successfully in certain circumstances.
- The status of a commit in GitLab was not updated correctly once a pipeline stage completed.
- In certain cases, impact analysis reports displayed an error when nodes in an environment had the potential to be impacted, but no impact was actually detected.
- If you added new PE credentials and gave them the same name as existing credentials, the existing credentials were overwritten.
- Users for whom Continuous Delivery for PE is not connected to the internet and analytics data collection is disabled received Google communication log warnings.

Version 2.11.1

Released 10 June 2019

New in this release:

- **Bitbucket Server webhooks created automatically.** Webhooks are now created automatically when you integrate Bitbucket Server (5.x series and newer versions) with Continuous Delivery for PE. You no longer need to manually configure plugins for webhooks or pull requests, as was required with the 4.x series.

Resolved in this release:

- In some circumstances, PostgreSQL database users' jobs could not exit a pending state.

Removed in this release:

- **Bitbucket Server 4.x series support for new users.** The Bitbucket Server 4.x series is no longer supported for new users who are setting up Continuous Delivery for PE for the first time.

Version 2.11.0

Released 28 May 2019

New in this release:

- **Deploy module code from a regex branch pipeline to a feature branch Puppet environment.** You can now deploy module code from a regex branch pipeline. When triggered, this deployment uses the Feature Branch deployment policy to deploy the new code to a Puppet environment with the same name as the branch where the new code is housed. If a Puppet environment by this name doesn't exist at the time of deployment, Continuous Delivery for PE creates it for you.

Resolved in this release:

- You were unable to connect Continuous Delivery for PE to an LDAP server when using an insecure LDAP connection.

Version 2.10.1

Released 20 May 2019

Resolved in this release:

- Module deployment reports didn't work with disk storage. Disk storage users were not able to create or download module deployment reports.
- Users could not render the upload-license page. This change fixes a previously unreported issue where a user could not visit the upload license page.

Version 2.10.0

Released 13 May 2019

Important security notice:

- **Select data was collected on accounts that have opted out of data collection.** We recently discovered that Continuous Delivery for PE was erroneously collecting select analytics data from accounts that had opted out of analytics data collection. We resolved this issue and have destroyed all analytics data collected on or before 9 May 2019, regardless of whether the data was collected as a result of this issue. To learn more about the data we collect, see Analytics data collection.

New in this release:

- **Module deployment reports.** You can now generate a module deployment report that lists the most recent commit SHA in each environment in Continuous Delivery for PE for the 10 most recently deployed modules. The module deployment report is a comma-separated value (CSV) file, and can be downloaded and imported to the tool of your choice. To generate a module deployment report, click the reports icon on the**Modules** page.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - All forms can now be submitted by pressing the Enter key.
  - You can now show or hide the value of your Artifactory access token when adding credentials.
  - Success messages are no longer shown in red text in logs.
  - Improved error messaging when an invalid private key is added during impact analysis configuration.

Resolved in this release:

- When you added a new control repo or module and were prompted to integrate your source control system, the newly added source control system was not selected correctly when you returned to the module or control repo creation screen.

Removed in this release:

- **Job history.** We've removed the job history timeline from the **Jobs** page.

Continuous Delivery for PE known issues

These are the known issues for the Continuous Delivery for PE 2.x release series.

Module names are limited to 32 characters

If the name you give your module exceeds 32 characters, an `Invalid name. Valid names must only include printable ASCII characters` error is displayed.

Automatic PE integration fails if the value of `puppetdb_port` is set as a string

If the `puppetdb_port` parameter's value in the `puppet_enterprise` class in the PE Infrastructure node group is set as a string, automatic integration of PE fails with an `Automatic configuration failed` error. To work around this issue, use the PE console to change the `puppetdb_port` parameter's value to an integer.

Regex branch module deployments fail if the `:control_branch` pattern is used for multiple modules

Deploying a module from a regex branch pipeline fails if more than one module in your Puppetfile uses the `:branch => :control_branch` pattern. To work around this issue, make sure that the `default_branch`

parameter is set in the Puppetfile for every Git-sourced module that uses the `:branch => :control_branch` pattern.

Docker configuration changes to jobs are not immediately available

When you update the Docker configuration for a job, several minutes must elapse for your changes to take effect. To work around this issue, wait at least five minutes after making a Docker configuration change before attempting to run the job.

Add new workspace screen inaccessible to users removed from all workspaces

If you delete or are removed from all workspaces of which you are a member, you are directed to the **Add New Workspace** screen. If you log out or navigate away from this screen without creating a new workspace, you are unable to access any workspaces or get back to the **Add New Workspace** screen until invited to an existing workspace by another user. To work around this issue, create a new workspace when prompted, or request an invitation to an existing workspace.

Release notes archive

Release notes for Continuous Delivery for PE 2.x release series versions that are more than six months past the most recent 2.x release version are housed in the release notes archive.

Version 2.9.1

Released 30 April 2019

Resolved in this release:

- MySQL users were unable to run Continuous Delivery for PE version 2.9.0.

Version 2.9.0

Released 30 April 2019

New in this release:

- **Pipeline stage status shown in your source control system.** The status of each stage of the relevant pipeline is now shown alongside a commit or pull request in your source control system.
- **Master branch creation for new modules.** When you add a new module to Continuous Delivery for PE, the software now checks to see if the module contains a master branch. If no master branch is present, Continuous Delivery for PE offers to create one for you from an existing branch of your choosing. For more on the master branch and its importance, see Git branches and Continuous Delivery for PE.
- **Name a newly created pipeline stage.** To assign a custom name to a newly created pipeline stage, fill in the **Name** field when adding the new stage.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - If an impact analysis task fails, a **Failed** message is now shown in the pipeline and events views.
  - You can now show or hide the value of your GitHub Enterprise token when adding credentials.

Resolved in this release:

- Due to a Firefox known issue, Firefox users saw a blank screen when opening a message in the message center.
- When editing the later stages of a long pipeline, the editing modal often appeared above the visible area of the browser window.
- Clicking **Edit Deployment** too soon after a new page load caused an error.
- A non-descriptive error was shown when you attempted to deploy to a node group that didn't contain any nodes.
- When adding new PE credentials, relevant fields were not displayed for manual entry of information not automatically discovered by Continuous Delivery for PE.

- The **Edit Impact Analysis** screen did not accurately show the environments you chose to run impact analysis tasks on.

Version 2.8.2

Released 25 April 2019

Resolved in this release:

- The Continuous Delivery agent was unable to successfully clone repositories from GitHub. To fix this issue, Update the Continuous Delivery agent to the latest version.

Version 2.8.1

Released 18 April 2019

Important security notice:

- **Root passwords were exposed in the PE console for some users.** If you used the **cd4pe::root_configuration** task to configure your Continuous Delivery for PE installation, the root user's username and password were exposed in the job's **Job Details** pane in the PE console. To fix this issue, upgrade the `puppetlabs-cd4pe` module to version 1.2.1 and reset the root user's password.

Version 2.8.0

Released 15 April 2019

Important notice for users upgrading to version 2.8.0:

- **A Continuous Delivery agent upgrade is required on all SELinux-enabled job hardware.** This version features improved support for distribution-packaged Docker on hosts with SELinux enabled. Mounting volumes from the host to the Continuous Delivery for PE container now tells SELinux to allow the container access to the volume mount. If you have SELinux enabled, you must upgrade your Continuous Delivery agents. For instructions, see Upgrade the Continuous Delivery agent.

New in this release:

- **View a control repo or module's event history sorted by pipeline run.** You can now choose to view the event history for your control repo or module by pipeline run, rather than by chronological event. To get started, choose **View history by: Pipeline** on your control repo or module's **Events** pane.

  Note: If you're upgrading to Continuous Delivery for PE version 2.8.0, please note that we are unable to retroactively sort your event history from previous versions of the software into pipeline runs. Only pipeline runs triggered using version 2.8.0 and newer versions are shown in the pipeline view.
- **Automatic Git branch creation when deploying a module to an environment without a matching branch.** When you deploy a module to an environment for which a matching Git branch does not yet exist, Continuous Delivery for PE now creates the branch for you.
- **Set a deployment's timeout duration.** When creating a new deployment, you can now specify the length of the deployment's timeout period. By default, deployments time out after 60 minutes.
- **Name your pipeline stages.** You can now provide custom names for the stages in your control repo and module pipelines. To name or rename a pipeline stage, click the **More Actions** ··· menu and select**Rename Stage**.
- **Support for custom CA certificates for GitHub Enterprise.** When setting up your GitHub Enterprise integration, you now have the option to provide a custom CA certificate, such as a self-signed certificate.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - The description you provide for a manually triggered module deployment is now shown in the module's events timeline and on the deployment's details page.

Resolved in this release:

- If you enabled SAML and then ran a configuration test, SAML was disabled until you re-saved the integration.

- A display error occurred if you reordered the stages of a default pipeline while the pipeline contained stage placeholders.
- When using GitHub Enterprise with self-signed certificates, an error was shown when you ran a job.

Removed in this release:

- **Dispatch a job.** We've removed the ability to manually dispatch a job from the **Jobs** page.

Version 2.7.1

Released 2 April 2019

Resolved in this release:

- When manually triggering a regex branch pipeline for a control repo, the branch selection section of the **Run Pipeline** page was unresponsive.

Version 2.7.0

Released 1 April 2019

New in this release:

- **Default pipelines.** When creating a new pipeline for a control repo or module, Continuous Delivery for PE now offers to set up a default pipeline for you. Default pipelines contain recommended stages along with prompts to build out the pipeline's functionality. You can edit and rearrange the elements of the default pipeline to suit your needs. To get started, click **+ Add default pipeline** when creating a new pipeline.
- **SSL support.** To learn how to set up SSL for your Continuous Delivery for PE instance, see Configure SSL.
- **Notification of upcoming token expiration.** You'll receive a notification in the message center when yourPE token is two weeks from its expiration date, and another notification if the token expires.
- **Updated guidance and new default queries for setting up regex branch pipelines.** These changes are designed to help you avoid accidentally triggering the regex branch pipeline when changes are deployed to your environment branches.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - You are no longer required to generate and enter a new PE token when adding or editing impact analysis credentials after integrating your PE instance.
  - Improved messaging when configuring storage in the root console.

Deprecated in this release:

- **Incremental branch and blue-green branch deployment policies.** The incremental branch and blue-green branch deployment policies have been deprecated. These deployment policies will be removed from Continuous Delivery for PE in a future release.
- **Support for Puppet Enterprise version 2017.3.** PE 2017.3 has reached the end of its support lifecycle.

Version 2.6.0

Released 18 March 2019

New in this release:

- **Module deployments.** You can now deploy module changes to your Puppet environments using your module's pipeline. In order to deploy module code, you must first add a `:branch => :control_branch` declaration to the module's section of your Puppetfile. See Deploy module code for more information.
- **Master branch creation for new control repos.** When you add a new control repo to Continuous Delivery for PE, the software now checks to see if the control repo contains a master branch. If no master branch is present, Continuous Delivery for PE offers to create one for you from an existing branch of your choosing. For more on the master branch and its importance, see Git branches and Continuous Delivery for PE.

- **No-op runs for deployments.** When creating a new deployment for a control repo, you now have the option to set the deployment to run in no-op mode. No-op deployments are not available for regex pipelines or deployments using the eventual consistency deployment policy.
- **Redesigned Puppet Enterprise account settings page.** The **Puppet Enterprise** account settings page now displays each connected PE instance's node classifier address, number of protected environments, and token expiration date.
- **Specify the lifetime of your PE token.** When adding new PE credentials using basic authorization, you can now customize the lifetime of the PE token Continuous Delivery for PE generates for you.
- **Regenerate a PE token.** On the Puppet Enterprise account settings page, click **Regenerate** to enter a new API token or provide your PE credentials to have a new token generated for you.
- **Run job hardware on Ubuntu 18.04.** You can now install the Continuous Delivery agent on systems running Ubuntu 18.04.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - Improved messaging when setting up a deployment using the feature branch policy.
  - No longer requiring a deployment description when adding a deployment to a pipeline stage,
  - Reorganized the steps for creating a new deployment for a more logical flow.

Resolved in this release:

- All deployments with stagger settings ran batches of 10 nodes at a time, rather than the number of nodes specified when the deployment was created.
- The **Update Ref** section of the deployment details page was not displayed for deployments using the eventual consistency deployment policy.
- Terminate conditions were not shown on the deployment details page for deployments using the direct merge deployment policy.

Version 2.5.0

Released 4 March 2019

New in this release:

- **Deploy code from a regex branch pipeline to a feature branch Puppet environment.** You can now add a deployment to your regex branch pipeline. When triggered, this deployment uses the Feature Branch deployment policy to deploy the new code to a Puppet environment with the same name as the branch where the new code is housed. If a Puppet environment by this name doesn't exist at the time of deployment, Continuous Delivery for PE creates it for you.
- **Manually trigger a regex branch pipeline.** You can now start a pipeline run on a regex branch pipeline by clicking **Run Pipeline** .
- **Onboarding guide for first-time users.** New users are now greeted by a guided path through the integrations and setup tasks required to get started with Continuous Delivery for PE.
- **Reorder pipeline stages.** You can now change the order in which the stages in your pipeline appear. Click the **More Actions** menu and select**Reorder Stages**.
- **Support for pull requests from forked repositories.** The Continuous Delivery for PE pull request (PR) gate workflow now accepts PRs from forked repositories in GitHub, GitLab, and Azure DevOps. Pull requests from forked repositories on Bitbucket Server are not currently supported.

Resolved in this release:

- The **New LDAP Configuration** screen's close control was not operational.

Version 2.4.0

Released 21 February 2019

New in this release:

- **Fewer inputs required for PE integration.** We've simplified the process for integrating your PE instance with Continuous Delivery for PE, and no longer require you to manually generate an API token, enter your endpoints, or paste in your CA certificate. See Add your Puppet Enterprise credentials for details.
- **Regex branch pipelines.** You can now create a pipeline for branches in your control repo or module that aren't associated with an existing pipeline, and have names matching the regular expression you set. Only one regex branch pipeline is allowed per module or control repo. To create a regex branch pipeline, click **+ Add Pipeline** and select **Branch Regex**.
- **Improved controls for adding pipeline stages.** Previously, you could add stages only to the bottom of your pipeline. This release introduces the **More Actions** ··· menu to every pipeline stage, which includes the options to add a stage before or after any stage.
- **Hiera 3 support.** Continuous Delivery for PE now supports use of Hiera 3.
- **Deployment approval notifications in the message center.** Notifications of deployments to protected environments that require approval are now shown in the message center. Members of the approval group will also continue to receive email notifications.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - Disk storage is now shown first in the list of available storage options on the **Settings** page.
  - A default name for the disk storage bucket is prepopulated when configuring disk storage.
  - Clearer controls and guidance when adding impact analysis credentials.
  - You can now delete messages from the message center.

Resolved in this release:

- If you entered data into one of the fields on the storage settings page, then changed to a different storage type, the data was shown in the fields for the new storage type.
- A recent breaking change to the Azure DevOps API caused new Azure DevOps integrations with Continuous Delivery for PE versions 2.3.0 and earlier to fail.

Version 2.3.0

Released 4 February 2019

Important notice for users upgrading to version 2.3.0:

- **A Continuous Delivery agent upgrade is required on all Docker job hardware.** Due to this version's significant improvements to job execution in Docker containers, an upgrade of the Continuous Delivery agent is required on all Docker-capable job hardware. For instructions, see Upgrade the Continuous Delivery agent.

New in this release:

- **Improved support for LDAP.** We've made significant improvements to the Continuous Delivery for PE LDAP integration, including the ability to integrate multiple LDAP servers, the option to perform recursive group lookups, and support for LDAP group mapping. For more information, see Configure LDAP.
- **Disk storage.** Continuous Delivery for PE now supports disk storage, eliminating the requirement to use Amazon S3 or Artifactory for external object storage of logs, job manifests, and artifacts. To use disk storage, you must mount a volume to the Continuous Delivery for PE Docker container. For more information, see Install Continuous Delivery for PE using Docker.
- **Eventual consistency deployment policy.** Continuous Delivery for PE now provides the option to deploy new Puppet code without launching any orchestrator jobs. Use the eventual consistency deployment policy when you prefer to use your regularly scheduled Puppet runs to pick up the changes.
- **Links to associated PE jobs.** A deployment's details page now displays links to the Puppet jobs that were created as part of the deployment in the PE console.
- **Faster Docker-based jobs.** We've made significant improvements to the execution speed of jobs run in Docker containers.
- **Docker run arguments included in Docker-based jobs.** When creating a job that runs inside a Docker container, you now have the option to pass Docker run arguments to the container. Find the optional **Docker Run Arguments** field in the **Docker Configuration** section of the **New Job** and **Edit Job** panes.

- **Message center.** Open the message center in the navigation bar to view important notifications about required actions following an upgrade ofContinuous Delivery for PE.
- **Analytics data collection.** In order to help us better serve you, our users, Continuous Delivery for PE is now gathering information on how you work with the software. No personally identifiable information is ever collected, and the data we gather is never used or shared outside Puppet. To learn about what data we collect and find instructions for opting out, see Analytics data collection.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - When displayed, the Continuous Delivery agent secret key is now shown in a modal window.
  - Improved error messaging for attempted deployments to environment node groups with classes.
  - Your job hardware server's friendly name is shown on the job details page, rather than the server's ID.
  - An error message is shown if you attempt to add an impact analysis task to a pipeline for a PE instance on which impact analysis has not been enabled.

Resolved in this release:

- Pre-built jobs were not displayed if you navigated to the **Jobs** page within 30 seconds of completing the initial setup and configuration process.
- Changes to SMTP settings did not take effect immediately.
- Nodes selected for a deployment did not respect all the rules of a target node group's hierarchy.

Version 2.2.1

Released 19 December 2018

Resolved in this release:

- An incomplete deployment policy was accidentally enabled in version 2.2.0. The full feature will be available in an upcoming version.

Version 2.2.0

Released 17 December 2018

New in this release:

- **Azure DevOps support.** See Integrate with Azure DevOps for instructions on how to create an OAuth application to start using your organization's Azure DevOps repositories with Continuous Delivery for PE.
- **Manual approval workflow for protected Puppet environments.** You can now designate protected Puppet environments and create groups of users authorized to manually approve or decline deployments to those protected environments. See Require approval for deployments to protected Puppet environments for instructions on configuring and using this workflow.
- **Configurable catalog compilation batch size for impact analysis tasks.** When setting up an impact analysis task, you can now specify how many catalogs should be compiled simultaneously.
- **Configurable SSH port number for GitLab integrations.** When setting up your GitLab integration, you now have the option to specify a custom SSH port number.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - If you have more than one pipeline for a control repo or module, the pipeline you most recently viewed is still displayed after you refresh the page.

Resolved in this release:

- If your session timed out during the initial installation of Continuous Delivery for PE, you were unable to complete the installation process.

Version 2.1.1

Released 7 December 2018

Resolved in this release:

- Any pipeline containing five or more deployment or impact analysis tasks froze when triggered.
- An incomplete feature on the **Modules** page was accidentally enabled in version 2.1.0. The full feature will be available in an upcoming version.

Version 2.1.0

Released 29 November 2018

New in this release:

- **Support for prefixed environment names**. Continuous Delivery for PE now supports use of environment name prefixes, which are set using the `prefix` subsetting of the r10k `sources` parameter. No additional setup or configuration is required to use prefixed environment names with Continuous Delivery for PE.
- **Enable or disable LDAP nested group search.** When configuring your LDAP single sign-on credentials, you now can choose whether to perform recursive queries to search nested LDAP groups. To change your LDAP settings, navigate to the root console and click **Single Sign On**.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - Modal styling on all configuration and creation screens.
  - Validation of newly created usernames to ensure they contain only allowed characters.

Resolved in this release:

- The **Settings** page in the root console crashed if your chosen storage provider, Artifactory or S3, could not be validated.

Version 2.0.3

Released 8 November 2018

New in this release:

- **Support for smaller screens.** The Continuous Delivery for PE web UI now scales more responsively when you work with the software on a smaller screen or in a smaller browser window.

Version 2.0.2

Released 31 October 2018

Resolved in this release:

- Control repos with matching names were automatically selected from all available organizations for GitHub Enterprise users.

Version 2.0.1

Released 25 October 2018

New in this release:

- **Gravatars.** Each user account now shows its associated Gravatar image in the navigation bar of the web UI and on the user's **Profile** page.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - You'll now see your changed resources first when opening an impact analysis report.
  - Clearer guidance messaging when adding impact analysis tasks to a pipeline.
  - Control repo names can now contain more than 30 characters.

Resolved in this release:

- False positives in impact analysis reports that were shown when an `if` statement in the Puppet code was updated but the node's resources remained unchanged.

- Pipeline run errors created when an impact analysis task was the only item in a pipeline stage and **Auto Promote** to the next stage was disabled.
- Code Manager errors created when an impact analysis task was added to a pipeline stage containing a deployment.
- Incomplete list of job hardware shown after an item was removed from the list of job hardware.
- Impact analysis failure when run against an environment containing no nodes.
- **Update Ref** field on the deployment details page shown as **Waiting** after deployment completion.

Version 2.0.0

Released 9 October 2018

New in this release:

- **Impact analysis.** Preview the potential impact to your infrastructure of a Puppet code change before the new code is merged. Each time a pull request is opened against your pipeline, impact analysis produces a detailed report on how the new code will impact your nodes, resources, and classes. To get started with impact analysis, see Configure impact analysis.
- **Docker-based jobs.** Direct any job to run inside a Docker container on your job hardware by enabling the switch in the **Docker Configuration** section of the **New Job** screen and specifying the Docker image name.
- **Pre-built jobs.** This release introduces six pre-built, Docker-based jobs for testing your control repos and modules. For new users, these jobs are available on the **Jobs** screen upon installation. Existing users who upgrade to version 2.0.0 can create the jobs using the Pre-built job reference.
- **Rerun a job.** Click **Rerun Job** on a control repo or module's**Overview** screen or on a job's **Job Details** screen to run a job again on the same commit.
- **Open virtual appliance (OVA).** The Continuous Delivery for PE OVA offers a simplified set up process so you can run a trial or proof of concept. See Install the OVA for instructions. The OVA is not suitable for production use.
- **Usability improvements.** We've made numerous small usability improvements, including:
  - The confirmation button on all editing interactions in the web UI is now **Save Changes**.
  - The `https://` prefix is allowed when entering PE service endpoints during the PE integration process.

Resolved in this release:

- 403 errors when a super user attempted to navigate to a user's account from the root console.
- Incorrect user name in pipeline item callback URLs when created by super users in other users' accounts.
- Failure to load the list of commits for pipelines triggered while the **Edit Deployment** window was open.
- Temporary removal of multiple PE credentials when a user deleted a single PE credential.
- Inability to scroll through lists of more than 10 modules.
- Incorrect **Control Repo Name** label on the **Add Module** screen.

Getting started with Continuous Delivery for PE

Greetings! Welcome to Continuous Delivery for PE. If you're trying out the software for the first time, this getting started guide is for you. As a new user, you'll need to perform some initial workspace setup tasks, and then we'll show you how to begin using core features of Continuous Delivery for PE.

You're just a few steps away from a more streamlined, powerful, and flexible Puppet code delivery process. Ready to get started?

Step 1: Install Continuous Delivery for PE

We'll start by installing Continuous Delivery for PE directly from the PE console.

These installation instructions assume you are running PE version 2019.1.x or 2019.2.x on an Enterprise Linux 7 node. If this is not the case, see Which installation method should I choose? and follow the instructions for the installation method that suits your infrastructure.

1. In the PE console, click **Integrations**.
2. In the **Continuous Delivery for PE host** field, enter the name of the node where you wish to install Continuous Delivery for PE.

   Note: This node must be running Enterprise Linux 7, and must have Puppet installed.
3. In the **Administrator email** field, enter an email address for the Continuous Delivery for PE administrator, also known as the root user. This email address is used as your username when signing in to Continuous Delivery for PE as the root user.
4. In the **Administrator password** field, enter a strong password for the Continuous Delivery for PE administrator account.
5. Click **Install** to perform a default installation of Continuous Delivery for PE.

   Important: Your Puppet certificate name must be a resolvable DNS hostname.
6. When the job is complete, navigate to the URL printed on the task page. Click **Trial Mode** to start a free seven-day trial.

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

Think of a workspace like a neighborhood within the Continuous Delivery for PE 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 the members of your team to join your workspace.

1. If you haven't already done so, navigate to the Continuous Delivery for PE web UI address you received at the end of the installation process in Step 1.
2. Create your 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**.
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**.

Step 3: Set up integrations and configure job hardware

Next, it's time to set up integrations with your PE instance and the source control system where you keep your Puppet code. You'll also configure a job hardware server, where your code will be tested before deployments.

We're sending you to our integration docs to complete these tasks.

1. Follow our Integrate with Puppet Enterprise instructions.

   Note: This guide assumes you're integrating a PE version 2019.1.x or 2019.2.x instance, which automatically configures impact analysis for you. If you're integrating a different version of PE, follow the Configure impact analysis instructions once your integration is complete.

2. Select your source control system from the list below and follow the integration instructions:
   - Azure DevOps
   - Bitbucket Cloud
   - Bitbucket Server
   - GitHub
   - GitHub Enterprise
   - GitLab
3. Follow our Configure job hardware with the web UI instructions to set up a job hardware server.

Step 4: Add a control repo

A control repo in Continuous Delivery for 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 master Git branch.

When you set up your new control repo, Continuous Delivery for PE adds a webhook to the associated 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.

1. In the Continuous Delivery for PE web UI, click **Control Repos**, then click **Add Control Repo**.
2. Follow the prompts to select your source control, organization, and your chosen repository.
3. The master branch of your repository is automatically selected for you. If your control repo does not currently contain a master branch, follow the prompts to let Continuous Delivery for PE create one for you, and select the branch that's under active development to create the master branch from.

   Important: When working with Continuous Delivery for PE, commit only to the master branch and to any feature branches (which are eventually merged back into the master branch). Do not push code changes to any of your other Git branches, as doing so can create conflicts with Continuous Delivery for PE workflows.
4. Optional: Edit the name of your new control repo.

   Tip: The control repo name must contain only alphanumeric characters, dashes, and underscores.
5. Click **Add**. The control repo is now shown in the master list on the **Control repos** page.

Step 5: Set up a pipeline

Pipelines in Continuous Delivery for PE 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**. Click the name of the control repo you added in Step 4.
2. On the right side of the web UI, you'll see the space where we'll create your pipeline. Click **+ Add default pipeline**.
3. Your default pipeline is automatically created for your master branch. This pipeline contains three stages:
   - The **Code Validation stage** includes two jobs (tests for Puppet code).
   - The **Impact Analysis stage** includes an impact analysis task with a pull request gate (more on this later).
   - The **Deployment stage**, where we'll add deployment instructions in step 7.

Step 6: Set up an environment node group

To give us a place to demonstrate how Continuous Delivery for PE deploys new code to your Puppet Enterprise-managed nodes, we'll next set up a small environment node group to use for deployment testing.

1. In your Git repository, create a new branch called **cd4pe-testing**. This will represent the environment node group.
2. On your master, run `puppet code deploy cd4pe-testing`.
3. In the PE console, click **Classification**.
4. Click **Add group...** and create a new node group with the following specifications:

   - Parent name: All Environments
   - Group name: CD4PE test group
   - Environment: cd4pe_testing
   - Environment group: yes
   - Description: Node group used for Continuous Delivery for PE testing

   Click **Add**.
5. In the list of node groups, click **CDPE 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.

Step 7: Deploy changes to your nodes

Once you've added a deployment to your pipeline, you can automatically move code changes to your nodes following 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, then select the **CDPE test group** node group we created in Step 5.
3. Select the **Direct Deployment Policy**. Don't worry about setting special conditions for this deployment.
4. Click **Add Deployment to Stage**. On the confirmation screen, click **Done**.
5. Now we need a change to pass through to production. In your Git repository, on the master branch, 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 the two tests on your code, skips over the impact analysis stage that we haven't yet set up, and stops, as the pipeline is set up not to autopromote into the Deployment stage. Click **Promote** to continue the pipeline run and launch the deployment. The results of the pipeline run are logged in the **Events** area on the left of the screen.

Step 8: Create an impact analysis report

An impact analysis report shows the potential impact that new Puppet code will have on the nodes and resources you're managing with PE. You can add impact analysis tasks to your pipeline, and you can generate impact analysis reports on demand, as we'll do in this step.

1. First, create a change for the report to analyze. We'll generate an impact analysis report to review how this change impacts the nodes in your CDPE test group. In your Git repository, create a feature branch from your master branch
2. On the feature branch, make a code change, such as updating a package version.
3. Commit the change on the feature branch and then return to the Continuous Delivery for PE web UI.
4. In the Continuous Delivery for PE web UI, click **New Impact Analysis**.
5. In the **New Impact Analysis** window, select your feature branch, then select the commit you just made. Select your PE instance and the CDPE test group node group.
6. Click **Analyze** to generate the report.

   Continuous Delivery for PE will now generate a new catalog containing your commit, and will compare this new catalog to the current catalog of the nodes in the CDPE test group.

7. Click **View Impact Analysis**. The report shows how the change on your feature branch would impact your nodes and resources if it was merged to the master branch.

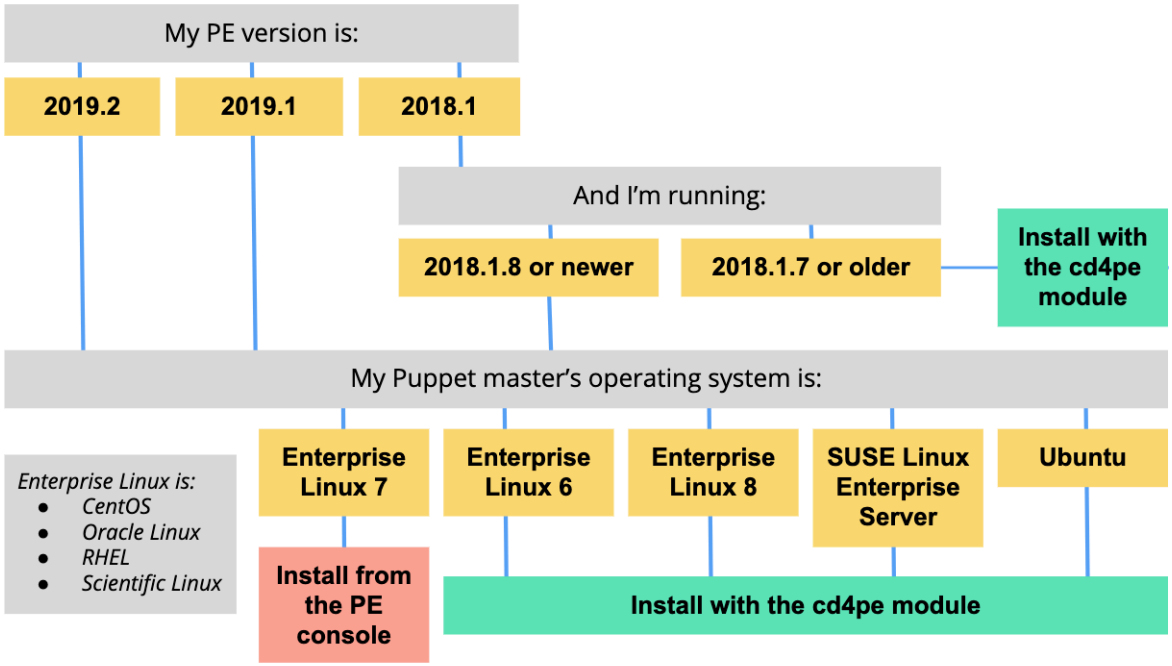
Congratulations! You've reached the end of this introductory guide. You're now familiar with some of the core features of Continuous Delivery for PE, and have a basic understanding of how the software helps you deploy Puppet code and preview the impact of changes.

Installing

In order for your organization to begin using Continuous Delivery for Puppet Enterprise, you must first complete the initial installation and setup process.

Which installation method should I use?

The best method for your Continuous Delivery for PE installation depends on your PE version and the operating system running on your Puppet master. Use the following chart to select the best installation method for your circumstances.



- System requirements on page 25

Refer to these system requirements for your Continuous Delivery for Puppet Enterprise (PE) installation.

- Install Continuous Delivery for PE from the PE console on page 28

Users of supported PE versions who are running Enterprise Linux 7 on their Puppet master can install Continuous Delivery for PE directly from the PE console. This installation method is available for PE 2019.2.x, 2019.1.x, and PE 2018.1.8 and later versions in the 2018.1.x series.

- Install Continuous Delivery for PE using the cd4pe module on page 31

Use version 1.1.0 or later of the `puppetlabs-cd4pe` module to install and configure Continuous Delivery for Puppet Enterprise (PE). This module installs Docker, configures the Continuous Delivery for PE Docker image and service for you, and creates a Docker volume for disk storage. You can also use the module to configure your Continuous Delivery for PE root account and to manage a MySQL container.

- Generate a trial license on page 36

Puppet License Manager is the centralized hub for purchasing and tracking licenses for many Puppet products, including Continuous Delivery for PE. Use Puppet License Manager to generate a free 30-day trial license for Continuous Delivery for PE.

- Alternative installation methods on page 36

This section includes alternative methods for installing Continuous Delivery for Puppet Enterprise (PE).

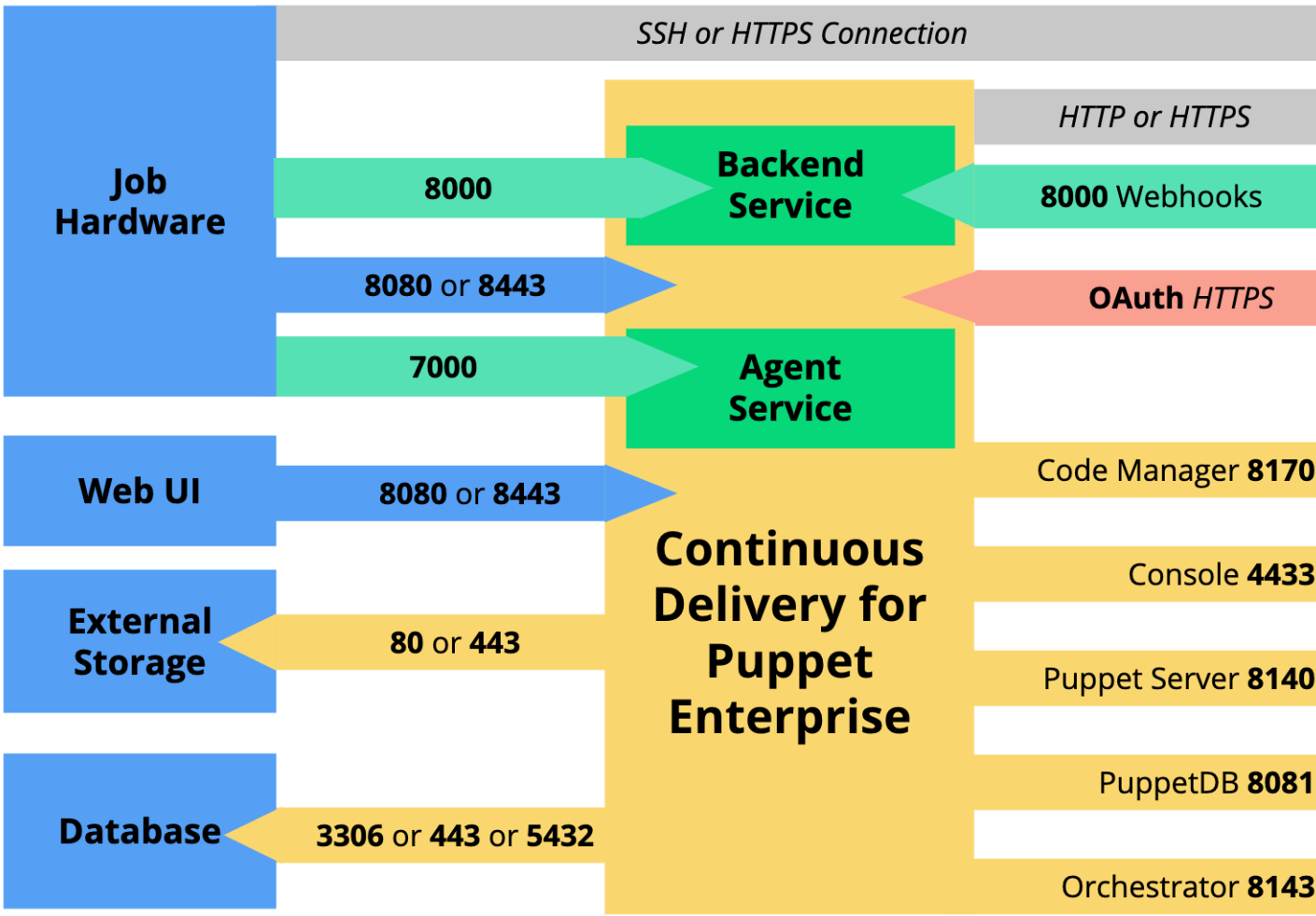
System requirements

Refer to these system requirements for your Continuous Delivery for Puppet Enterprise (PE) installation.

Continuous Delivery for PE architecture

Continuous Delivery for Puppet Enterprise (PE) communicates with your PE installation, your source control system, and the servers you've designated as job hardware, as well as with the various components of the software.

The following diagram shows the architecture and port requirements of a typical Continuous Delivery for PE installation.



Important: Continuous Delivery for PE uses TCP (Transmission Control Protocol) connections.

Job hardware and web UI configuration	Default port number
HTTP	8080

Job hardware and web UI configuration	Default port number
HTTPS	8443

External storage configuration	Default port number
Artifactory using HTTP	80
Artifactory using HTTPS	443
Amazon S3	443

Database configuration	Default port number
MySQL (external)	3306
Amazon DynamoDB (external)	443
PE-PostgreSQL (local)	5432

Hardware requirements

Before installing Continuous Delivery for PE, ensure that your system meets these requirements.

Installation architecture	Memory	Storage	CPUs
Installation using disk object storage	8 GB	100 GB	4
Installation using external (Artifactory or Amazon S3) object storage	8 GB	50 GB	4

Supported external databases

Continuous Delivery for PE uses a database to persist information. By default, the `puppetlabs-cd4pe` module creates a new installation of PE-PostgreSQL on the node where you install Continuous Delivery for PE. If you prefer to use an external database, the following are supported.

External database	Supported versions	Notes
MySQL	5.7	Your MySQL database must use the `latin1` character set and `latin1_swedish_ci` collation.
Amazon DynamoDB	n/a	Based on your usage, you might need to tune your tables' read/write capacity in order to reduce page load times.

Job hardware requirements

System requirements for your job hardware vary considerably based 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 that server's resources need to be. Determining that load involves a huge number of variables, from the number of jobs that run concurrently to the languages those jobs are written in. As a result, it's nearly impossible to provide one-size-fits-all system requirements for job hardware.

Instead, we've developed a sizing chart based on the estimated number of concurrent jobs a job hardware server is expected to regularly handle. While this chart represents our best estimates and understanding, it's provided only as a starting point. Testing and experience will help you fine-tune your job hardware and determine the optimum resource configuration for your installation's unique circumstances.

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 these facts 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 will run more slowly.

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 is our preferred way to install Docker and keep it up to date.

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 by following the Docker documentation.

Supported Puppet Enterprise versions

The following versions of Puppet Enterprise (PE) are supported for use with Continuous Delivery for PE.

PE version
2019.2.x
2019.1.x
2018.1.x

PE version-specific feature limitations

A small number of Continuous Delivery for PE features are not available on all supported versions of PE.

- Impact analysis reports for module code are supported on PE 2018.1.9 and newer versions in the 2018.1 series, PE 2019.1.1 and newer versions in the 2019.1 series, and all 2019.2.x series versions.

Supported browsers

The following browsers are supported for use with the Continuous Delivery for PE web 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

Install Continuous Delivery for PE from the PE console

Users of supported PE versions who are running Enterprise Linux 7 on their Puppet master can install Continuous Delivery for PE directly from the PE console. This installation method is available for PE 2019.2.x, 2019.1.x, and PE 2018.1.8 and later versions in the 2018.1.x series.

Before you begin

Review the system requirements and ensure you have:

- A designated node where you'll install Continuous Delivery for PE. This node must be running Enterprise Linux 7, and must have Puppet installed.

  Important: To install Continuous Delivery for PE from the PE console, you must be running Enterprise Linux 7 in one of the following distributions on your Puppet master.

  - CentOS
  - Oracle Linux
  - Red Hat Enterprise Linux
  - Scientific Linux

  For all other operating systems and versions, see Install Continuous Delivery for PE using the cd4pe module.
- The PE user permission to run tasks on this node.

CAUTION: Code Manager webhooks are not compatible with Continuous Delivery for PE. If your organization currently uses Code Manager webhooks to deploy code, you must dismantle these webhooks before installing Continuous Delivery for PE.

Install Continuous Delivery for PE from the PE 2019.2.x or 2019.1.x console

Users of the PE 2019.1.x or 2019.2.x series can install Continuous Delivery for PE directly from the PE console with just a few clicks.

1. In the PE console, click **Integrations**.
2. In the **Continuous Delivery for PE host** field, enter the name of the node where you wish to install Continuous Delivery for PE.

   Note: This node must be running Enterprise Linux 7, and must have Puppet installed.
3. In the **Administrator email** field, enter the email address for the Continuous Delivery for PE administrator, also known as the root user. This email address is used as your username when signing in to Continuous Delivery for PE as the root user.
4. In the **Administrator password** field, enter a strong password for the Continuous Delivery for PE administrator account.
5. To perform a default installation of Continuous Delivery for PE, click **Install**. If you need to customize your installation, click **Advanced options** and see Advanced configuration options.

   Important: In order to use the default installation, your Puppet certificate name must be a resolvable DNS hostname. If that is not the case, you must set the `resolvable_hostname` parameter to a resolvable address where the Continuous Delivery for PE server is reachable. You can use the `${trusted[certname]}` fact to set this parameter.
6. When the job is complete, navigate to the URL printed on the task page. Click **Trial Mode** to start a free seven-day trial. Once this period is complete, you'll be prompted to generate and upload a license. See Generate a license for instructions on creating a free 30-day trial license.

Now that Continuous Delivery for PE is installed, create your individual user account and then move on to these next steps:

- Integrate your source control system.
- Integrate with Puppet Enterprise, which automatically configures impact analysis in PE 2019.1.x and 2019.2.x.

- Configure job hardware, which is used when testing your Puppet code.
- Follow our Getting started with Continuous Delivery for PE guide to learn about core workflows and capabilities.

Install Continuous Delivery for PE from the PE 2018.1.8+ console

If you're running PE version 2018.1.8 or a newer version in the 2018.1 series, install Continuous Delivery for PE by running a pre-populated task in the PE console.

1. In the PE console, click **Integrations**.
2. In the **Continuous Delivery for Puppet Enterprise** area, click **Install**. The **Run a task** page opens with the `pe_installer_cd4pe::install` task pre-populated.
3. In the **Value** field for the required `cd4pe_admin_email` parameter, enter the email address for the Continuous Delivery for PE administrator, also known as the root user. This email address is used as your username when signing in to Continuous Delivery for PE as the root user.
4. In the **Value** field for the required `cd4pe_admin_password` parameter, enter a strong password for the Continuous Delivery for PE administrator account.
5. **Optional.** Customize your Continuous Delivery for PE installation by setting any of the parameters listed in Advanced configuration options. If none of these parameters are set, your installation proceeds with the default settings.

   Important: In order to use the default installation, your Puppet certificate name must be a resolvable DNS hostname. If that is not the case, you must set the `resolvable_hostname` parameter to a resolvable address where the Continuous Delivery for PE server is reachable. You can use the `${trusted[certname]}` fact to set this parameter.
6. In the **Select targets** area, specify the node on which you want to install Continuous Delivery for PE.

   Note: This node must be running Enterprise Linux 7, and must have Puppet installed.
7. Click **Run Job**.
8. When the job is complete, navigate to the URL printed on the task page. Click **Trial Mode** to start a free seven-day trial. Once this period is complete, you'll be prompted to generate and upload a license. See Generate a license for instructions on creating a free 30-day trial license.

Now that Continuous Delivery for PE is installed, create your individual user account and then move on to these next steps:

- Integrate your source control system.
- Integrate with Puppet Enterprise.
- Configure impact analysis.
- Configure job hardware, which is used when testing your Puppet code.
- Follow our Getting started with Continuous Delivery for PE guide to learn about core workflows and capabilities.

Advanced configuration options

Customize your Continuous Delivery for PE installation from the PE console by setting any of the following parameters. If none of these parameters are set, your installation proceeds with the default settings.

Parameters to configure the Docker image and version	
The following two parameters are concatenated by the `puppetlabs-cd4pe` module as follows: `image => "${cd4pe_image}:${cd4pe_version}",`	
cd4pe_image	Set this parameter if you use an internal Docker registry for mirroring containers. Use this parameter to set the image name; use cd4pe_version to set a tag.
cd4pe_version	Use this parameter to specify a particular version of the Continuous Delivery for PE Docker container. Default is `latest`.

Parameters to configure the database	
By default, the `puppetlabs-cd4pe` module (version 1.3.0 and newer) creates a new installation of PE-PostgreSQL on the node where you installed Continuous Delivery for PE. If you prefer to use Amazon DynamoDB or MySQL, set the parameters in this section. **CAUTION:** Changing any of these parameters post-install creates a new database and destroys all data kept in the previous database.	
manage_database	Set this parameter to `false` to use an external DynamoDB or MySQL server. Set this parameter to `true` to use Continuous Delivery for PE-managed PostgreSQL or MySQL.
db_provider	Enter `mysql` if you're using MySQL. Do not set this parameter if using DynamoDB.
db_host	Enter the address of the database. (Required for external MySQL and DynamoDB.)
db_name	Enter the name of the database. (Required for external MySQL and DynamoDB.)
db_pass	Enter the password for the database. (Required for external MySQL and DynamoDB.) **CAUTION:** To set your password successfully, you must set the root_password parameter to `Sensitive` in Hiera. For instructions, see Setting sensitive parameters in Hiera.
db_port	**Optional.** Enter the port the database listens on.
db_prefix	**Optional.** If you'd like your database tables to share a prefix, such as `cdpe-`, enter it here.

Parameters to configure the port mappings	
agent_service_port	Defaults to 7000.
backend_service_port	Defaults to 8000.
web_ui_port	Defaults to 8080.

Other optional parameters	
cd4pe_docker_extra_params	To pass any additional arguments to the Docker process running the Continuous Delivery for PE container, specify them as an array.
analytics	To opt out of analytics data collection, set this parameter to `false`. To learn about what data we collect, see Analytics data collection.

Automate upgrades of your Continuous Delivery for PE console installation

New versions of Continuous Delivery for Puppet Enterprise (PE) are released regularly. Automating upgrades ensures you're always taking advantage of the latest features, fixes, and improvements.

Install the `puppetlabs-cd4pe` module, then create and classify a node group to automate management of your Continuous Delivery for PE installation's version.

1. Add the module and its dependencies to your Puppetfiles.

   The `puppetlabs-cd4pe` module must be used with seven dependent modules. The eight modules and their required versions are as follows:

Module	Required version
`puppetlabs-cd4pe`	1.1.0 or later
`puppetlabs-stdlib`	4.19.0 or later
`puppetlabs-puppet_authorization`	0.5.0
`puppetlabs-hocon`	0.9.3 or later in the 0.x or 1.x series
`puppetlabs-concat`	1.1.1 or later in the 1.x, 2.x, 3.x, or 4.x series
`puppetlabs-docker`	3.3.0 or later
`puppetlabs-apt`	4.4.1 or later
`puppetlabs-translate`	1.1.0 or later

2. Add the eight modules listed above to the Puppetfile for each environment against which your compilers compile catalogs.

   A sample Puppetfile entry:

```
mod 'puppetlabs-cd4pe', :latest
# Requirements for cd4pe
mod 'puppetlabs-concat', '4.2.1'
mod 'puppetlabs-hocon', '1.0.1'
mod 'puppetlabs-puppet_authorization', '0.5.0'
mod 'puppetlabs-stdlib', '4.25.1'
mod 'puppetlabs-docker', '3.3.0'
mod 'puppetlabs-apt', '6.2.1'
mod 'puppetlabs-translate', '1.1.0'
```

3. Deploy the `puppetlabs-cd4pe` module and its dependencies to the `production` environment by running `puppet code deploy production`.
4. In the PE console, click **Classification**. Click **Add group** and create a new node group with the following specifications.
   - **Parent name**: PE Infrastructure
   - **Group name**: Continuous Delivery for PE
   - **Environment**: production
   - **Environment group**: Do not select this option
5. Open the newly created Continuous Delivery for PE node group. Add your Continuous Delivery for PE host server to the node group by either creating a rule or pinning the node.
6. Click Configuration. In the **Add new class** field, select the **cd4pe** class and click **Add class**.

   Note: If the cd4pe class isn't available, click **Refresh** to update the available class definitions.
7. To automate upgrades of Continuous Delivery for PE to the latest available version, set the `cd4pe_version` parameter to `latest`. Add the parameter and commit your change.

After the classification change is applied on the next Puppet run, your Continuous Delivery for PE installation automatically upgrades itself whenever a new version is available.

Install Continuous Delivery for PE using the cd4pe module

Use version 1.1.0 or later of the `puppetlabs-cd4pe` module to install and configure Continuous Delivery for Puppet Enterprise (PE). This module installs Docker, configures the Continuous Delivery for PE Docker image

and service for you, and creates a Docker volume for disk storage. You can also use the module to configure your Continuous Delivery for PE root account and to manage a MySQL container.

CAUTION: Code Manager webhooks are not compatible with Continuous Delivery for PE. If your organization currently uses Code Manager webhooks to deploy code, you must dismantle these webhooks before installing Continuous Delivery for PE.

Install Continuous Delivery for PE with the cd4pe module

Use the puppetlabs-cd4pe module version 1.1.0 or later to install Continuous Delivery for PE.

The `puppetlabs-cd4pe` module must be used with seven dependent modules. The eight modules and their required versions are as follows:

Module	Required version
`puppetlabs-cd4pe`	1.1.0 or later
`puppetlabs-stdlib`	4.19.0 or later
`puppetlabs-puppet_authorization`	0.5.0
`puppetlabs-hocon`	0.9.3 or later in the 0.x or 1.x series
`puppetlabs-concat`	1.1.1 or later in the 1.x, 2.x, 3.x, or 4.x series
`puppetlabs-docker`	3.3.0 or later
`puppetlabs-apt`	4.4.1 or later
`puppetlabs-translate`	1.1.0 or later

1. Add the eight modules listed above to the Puppetfile for each environment against which your compilers compile catalogs.

   A sample Puppetfile entry:

   ```
   mod 'puppetlabs-cd4pe', :latest
   # Requirements for cd4pe
   mod 'puppetlabs-concat', '4.2.1'
   mod 'puppetlabs-hocon', '1.0.1'
   mod 'puppetlabs-puppet_authorization', '0.5.0'
   mod 'puppetlabs-stdlib', '4.25.1'
   mod 'puppetlabs-docker', '3.3.0'
   mod 'puppetlabs-apt', '6.2.1'
   mod 'puppetlabs-translate', '1.1.0'
   ```

2. Deploy the updated code to the relevant environments by running `puppet code deploy <ENVIRONMENT>`.
3. In the PE console, click **Classification**. Click **Add group** and create a new node group with the following specifications.
   - **Parent name**: PE Infrastructure
   - **Group name**: Continuous Delivery for PE
   - **Environment**: production
   - **Environment group**: Do not select this option
4. Open the newly created Continuous Delivery for PE node group. Add the server you wish to use as your Continuous Delivery for PE host server to the node group by either creating a rule or pinning the node.
5. Click Configuration. In the **Add new class** field, select the **cd4pe** class and click **Add class**.

   Note: If the cd4pe class isn't available, click **Refresh** to update the available class definitions.

6. To automate upgrades of Continuous Delivery for PE to the latest available version, set the `cd4pe_version` parameter to `latest`. Add the parameter and commit your change.
   After the classification change is applied on the next Puppet run, your Continuous Delivery for PE installation automatically upgrades itself whenever a new version is available.
7. **Optional.** Customize your Continuous Delivery for PE installation by setting any of the parameters listed in Advanced configuration options. If none of these parameters are set, your installation proceeds with the default settings.

   Important: In order to use the default installation, your Puppet certificate name must be a resolvable DNS hostname. If that is not the case, you must set the `resolvable_hostname` parameter to a resolvable address where the Continuous Delivery for PE server is reachable. You can use the `${trusted[certname]}` fact to set this parameter.
8. Run Puppet on your Continuous Delivery for PE host server.

   Note: Once the Puppet agent run is complete, Docker downloads the Continuous Delivery for PE image, which can take up to a few minutes.

Continuous Delivery for PE is now installed. Next, complete the configuration of the software.

Configure Continuous Delivery for PE with a task

Once you've completed the installation of Continuous Delivery for PE using the `cd4pe` module, run a task to configure the software.

Tip: If you prefer to use classification, rather than a task, to configure your Continuous Delivery for PE installation, see Configure a Continuous Delivery for PE module installation using classification.

1. In the PE console, click **Classification**. Expand the **PE Infrastructure** node group and click **Continuous Delivery for PE**.
2. Click **Run** and select **Task**.
3. In the **Task** field, select `cd4pe::root_configuration`.

4. Enter parameters for the task, as follows:

Parameter	Value	Notes
root_email	The email address to associate with the root account.	Required.
root_password	The password to associate with the root account.	Required.
resolvable_hostname	The resolvable hostname where the Continuous Delivery for PE container can be reached. For example, if the container resides on a Docker host named `mydockerengine.myinc.com`, set resolvable_hostname to `http://mydockerengine.myinc.com`.	Required only if the agent certificate is not the machine's resolvable internet address.
agent_service_endpoint	The endpoint where the agent service can be reached, in the form `http://<resolvable_hostname>:<port>`.	Required if you set the agent_service_port parameter in the cd4pe class during installation.
backend_service_endpoint	The endpoint where the back end service can be reached, in the form `http://<resolvable_hostname>:<port>`.	Required if you set the backend_service_port parameter in the cd4pe class during installation.
web_ui_endpoint	The endpoint where the web UI can be reached, in the form `http://<resolvable_hostname>:<port>`.	Required if you set the web_ui_port parameter in the cd4pe class during installation.
storage_provider	Which object store provider to use. Must be one of: DISK, ARTIFACTORY or S3.	Defaults to DISK.
storage_bucket	The name of the bucket used for object storage.	Required if using Amazon S3 or Artifactory for object storage.
storage_endpoint	The URL of the storage provider.	Required if using Amazon S3 or Artifactory for object storage.
storage_prefix	For Amazon S3: the subdirectory of the bucket to use. For Artifactory: the top level of the Artifactory instance.	Optional.
s3_access_key	The AWS access key that has access to the bucket.	Required if using Amazon S3.
s3_secret_key	The AWS secret key that has access to the bucket.	Required if using Amazon S3.
artifactory_access_token	API token for your Artifactory instance.	Required if using Artifactory.

5. Click **Run job**.

6. When the job is complete, navigate to the URL printed on the task page. Click **Trial Mode** to start a free seven-day trial. Once this period is complete, you'll be prompted to generate and upload a license. See Generate a license for instructions on creating a free 30-day trial license.

Now that Continuous Delivery for PE is installed and configured, create your individual user account and then move on to these next steps:

- Integrate your source control system.
- Integrate with Puppet Enterprise.
- Configure impact analysis.
- Configure job hardware, which is used when testing your Puppet code.
- Follow our Getting started with Continuous Delivery for PE guide to learn about core workflows and capabilities.

Advanced configuration options

Customize your Continuous Delivery for PE installation from the PE console by setting any of the following parameters. If none of these parameters are set, your installation proceeds with the default settings.

Parameters to configure the Docker image and version	
The following two parameters are concatenated by the `puppetlabs-cd4pe` module as follows: `image => "${cd4pe_image}:${cd4pe_version}",`	
cd4pe_image	Set this parameter if you use an internal Docker registry for mirroring containers. Use this parameter to set the image name; use cd4pe_version to set a tag.
cd4pe_version	Use this parameter to specify a particular version of the Continuous Delivery for PE Docker container. Default is `latest`.

Parameters to configure the database	
By default, the `puppetlabs-cd4pe` module (version 1.3.0 and newer) creates a new installation of PE-PostgreSQL on the node where you installed Continuous Delivery for PE. If you prefer to use Amazon DynamoDB or MySQL, set the parameters in this section.  **CAUTION:** Changing any of these parameters post-install creates a new database and destroys all data kept in the previous database.	
manage_database	Set this parameter to `false` to use an external DynamoDB or MySQL server. Set this parameter to `true` to use Continuous Delivery for PE-managed PostgreSQL or MySQL.
db_provider	Enter `mysql` if you're using MySQL. Do not set this parameter if using DynamoDB.
db_host	Enter the address of the database. (Required for external MySQL and DynamoDB.)
db_name	Enter the name of the database. (Required for external MySQL and DynamoDB.)
db_pass	Enter the password for the database. (Required for external MySQL and DynamoDB.)  **CAUTION:** To set your password successfully, you must set the root_password parameter to `Sensitive` in Hiera. For instructions, see Setting sensitive parameters in Hiera.

Parameters to configure the database	
db_port	**Optional.** Enter the port the database listens on.
db_prefix	**Optional.** If you'd like your database tables to share a prefix, such as `cdpe-`, enter it here.

Parameters to configure the port mappings	
agent_service_port	Defaults to 7000.
backend_service_port	Defaults to 8000.
web_ui_port	Defaults to 8080.

Other optional parameters	
cd4pe_docker_extra_params	To pass any additional arguments to the Docker process running the Continuous Delivery for PE container, specify them as an array.
analytics	To opt out of analytics data collection, set this parameter to `false`. To learn about what data we collect, see Analytics data collection.

Generate a trial license

Puppet License Manager is the centralized hub for purchasing and tracking licenses for many Puppet products, including Continuous Delivery for PE. Use Puppet License Manager to generate a free 30-day trial license for Continuous Delivery for PE.

Important: Puppet offers a free 30-day trial license for Continuous Delivery for PE. When you're ready to purchase a production license, please contact us.

1. Navigate to Puppet License Manager and click **Sign Up**. Fill in the form to create your Puppet account.

   If you already have a Puppet account, log in with your credentials.
2. Click **Get License**.
3. Locate Continuous Delivery for Puppet Enterprise Enterprise Edition and click **30-day Free Trial**.
4. Fill out the license registration form and click **Start 30-day Trial**.

   Your trial license is created, and the main license management screen opens. You can view a list of your licenses here, along with each one's status and expiration date.
5. Click **Download** and save the license file in a secure, memorable location. You'll be asked to upload it when your initial Continuous Delivery for PE seven-day trial period expires.

Alternative installation methods

This section includes alternative methods for installing Continuous Delivery for Puppet Enterprise (PE).

- Install Continuous Delivery for PE using Docker on page 37

Installing Continuous Delivery for Puppet Enterprise (PE) requires you to pull and run a Docker container, then follow our guided setup to create the root account and configure your installation.

- Configure a Continuous Delivery for PE module installation using classification on page 39

Once you've completed the installation of Continuous Delivery for PE using the `cd4pe` module, you can use classification to configure the software.

Install Continuous Delivery for PE using Docker

Installing Continuous Delivery for Puppet Enterprise (PE) requires you to pull and run a Docker container, then follow our guided setup to create the root account and configure your installation.

Before you begin

Review the system requirements and ensure you have:

- The ability to run a Docker image.

  Note: Some operating systems ship with older versions of Docker that do not work well with Continuous Delivery for PE. For best results, install Docker Community Edition (CE) or Docker Enterprise Edition (EE) using packages distributed directly by Docker.
- A database for persisting information, either MySQL version 5.7 or Amazon DynamoDB.

  Note: Your MySQL database must use the `latin1` character set and `latin1_swedish_ci` collation.

CAUTION: Code Manager webhooks are not compatible with Continuous Delivery for PE. If your organization currently uses Code Manager webhooks to deploy code, you must dismantle these webhooks before installing Continuous Delivery for PE.

1. Continuous Delivery for PE is delivered as a Docker container. Pull down the latest version by running:

```
docker pull puppet/continuous-delivery-for-puppet-enterprise:latest
```

2. Run the image with the command appropriate to your database infrastructure, replacing the options with the appropriate values, as per the instructions below.

   For MySQL:

   ```
   docker run --rm --name cd4pe -v cd4pe:/disk -e
    DB_ENDPOINT=<MY_DATABASE_ENDPOINT> -e DB_USER=<MY_DB_USERNAME> -e
    DB_PASS=<MY_DB_PASS> -e DB_PREFIX=<OPTINAL_PREFIX_TO_DB_TABLES> -e
    DUMP_URI=<DUMP_URI> -e PFI_SECRET_KEY=<KEY_FOR_DB_SECRETS_ENCRYPTION>
    -p 8080:8080 -p 8000:8000 -p 7000:7000 puppet/continuous-delivery-for-
   puppet-enterprise:latest
   ```

   For Amazon DynamoDB:

   ```
   docker run --rm --name cd4pe -v cd4pe:/disk -e DB_ENDPOINT=ddb://
   <AWS_REGION> -e DB_USER=<AWS_ACCESS_KEY> -e DB_PASS=<AWS_SECRET_KEY> -e
    DUMP_URI=<DUMP_URI> -e PFI_SECRET_KEY=<KEY_FOR_DB_SECRETS_ENCRYPTION>
    -p 8080:8080 -p 8000:8000 -p 7000:7000 puppet/continuous-delivery-for-
   puppet-enterprise:latest
   ```

   `--name`

   A friendly name you give to the running image. We used `cd4pe` in the example above.

   `-v cd4pe:/disk`

   Creates a new Docker volume called `cd4pe` for the user, which will be used for storing logs, job manifests, and artifacts. To learn more about this Docker volume, use the command `docker volume inspect cd4pe`. To learn more about Docker volumes in general, see the Docker documentation.

   Important: If you wish to use Amazon S3 or Artifactory as an external storage provider, do not include this option in your `docker run` command.

   `DB_ENDPOINT`

   The `mysql://` or `ddb://` endpoint used to connect to your database. For example, `mysql://samplehost:3306/cdpe`

   `DB_USER` and `DB_PASS`

   For MySQL: Login credentials for your MySQL user. For security purposes, the database user you select should be able to connect to only this database.

   For Amazon DynamoDB: Access and secret keys for your DynamoDB user.

   Important: Make sure you generate credentials with full create, read, update, and delete permissions for DynamoDB resources.

   For security purposes, select a database user you select who can connect to only this database.

   `DB_PREFIX`

   When starting up, Continuous Delivery for PE creates tables in MySQL or DynamoDB. If you'd like the tables to share a prefix, such as `cdpe-`, enter it here.

   Tip: If you wish to simulate a fresh installation of a given version of Continuous Delivery for PE, entering a new database table prefix causes all the database tables to regenerate.

   `DUMP_URI`

   How to address port 7000 of this container, which is the endpoint used by the Continuous Delivery for PE web UI to connect to the correct instance of the Continuous Delivery agent service. In a typical installation, this value is `dump://localhost:7000`.

   `PFI_SECRET_KEY`

   A 16-byte secret key used for AES encryption of secrets (such as PE access tokens) supplied to Continuous Delivery for PE.

   If you're a *nix user, generate this key by running:

   ```
   dd bs=1 if=/dev/urandom count=16 2>/dev/null | base64
   ```

   If you're a Windows user, generate this key by running:

3. Point your web browser to `localhost:8080` to access the Continuous Delivery for PE web UI. Click **Get Started**.
4. Follow the prompts to configure your installation. You'll be asked to:
   a. Select the email address and password to associate with the root account.

      Note: The root account is used for installing and configuring your organization's Continuous Delivery for PE instance.
   b. Configure the endpoints for Continuous Delivery for PE services.

      Important: For production-grade systems supporting multiple Continuous Delivery for PE users, configure these services to point to the DNS endpoints of load balancers. This is especially vital if you plan to run multiple container replicas of Continuous Delivery for PE.
   c. If desired, integrate an external storage provider (Amazon S3 or Artifactory) by adding credentials. You can skip this step if you are using disk storage.
   d. Click **Trial Mode** to start a free seven-day trial. Once this period is complete, you'll be prompted to generate and upload a license. See Generate a license for instructions on creating a free 30-day trial license.
5. When configuration is complete, you'll see a success message. Click **Create your user account** to sign out of the root account.

   Tip: If your organization uses GitHub, return to the root console and set up your GitHub OAuth application before signing out as root. See Integrate with GitHub for instructions.

Now that Continuous Delivery for PE is installed, create your individual user account and move on to these next steps:

- Integrate your source control system.
- Integrate with Puppet Enterprise.
- Configure impact analysis.
- Configure job hardware, which is used when testing your Puppet code.
- Follow our Getting started with Continuous Delivery for PE guide to learn about core workflows and capabilities.

Configure a Continuous Delivery for PE module installation using classification

Once you've completed the installation of Continuous Delivery for PE using the `cd4pe` module, you can use classification to configure the software.

1. In the PE console, click **Classification**. Expand the **PE Infrastructure** node group and click **Continuous Delivery for PE**.
2. Click **Configuration**. In the **Add new class** field, select **cd4pe::root_config**.

3. Enter parameters for the class, as follows:

Parameter	Value	Notes
root_email	The email address to associate with the root account.	Required.
root_password	The password to associate with the root account. **CAUTION:** To set your password successfully, you must set the root_password parameter to `Sensitive` using Hiera. This parameter cannot be set successfully in the PE console. For instructions, see Setting sensitive parameters in Hiera.	Required. **Note:** If you need to update the root account password, first do so in the Continuous Delivery for PE web UI, and then update the password in the PE console using this parameter.
storage_provider	Which object store provider to use. Must be one of: DISK, ARTIFACTORY or S3.	Defaults to DISK.
storage_bucket	The name of the bucket used for object storage.	Required if using Amazon S3 or Artifactory for object storage.
storage_endpoint	The URL of the storage provider.	Required if using Amazon S3 or Artifactory for object storage.
storage_prefix	For Amazon S3: the subdirectory of the bucket to use. For Artifactory: the top level of the Artifactory instance.	Optional.
s3_access_key	The AWS access key that has access to the bucket.	Required if using Amazon S3.
s3_secret_key	The AWS secret key that has access to the bucket.	Required if using Amazon S3.
artifactory_access_token	API token for your Artifactory instance.	Required if using Artifactory.

4. Commit your changes and run Puppet on the node group.
5. When the Puppet run is complete, navigate to port 8080 of the Continuous Delivery for PE host server. Click **Trial Mode** to start a free seven-day trial. Once this period is complete, you'll be prompted to generate and upload a license. See Generate a license for instructions on creating a free 30-day trial license.

Now that Continuous Delivery for PE is installed and configured, create your individual user account and then move on to these next steps:

- Integrate your source control system.
- Integrate with Puppet Enterprise.
- Configure impact analysis.
- Configure job hardware, which is used when testing your Puppet code.
- Follow our Getting started with Continuous Delivery for PE guide to learn about core workflows and capabilities.

Setting sensitive parameters in Hiera

When passing sensitive information, such as the root password for Continuous Delivery for PE, as parameters, set the value of these parameters as `Sensitive` in Hiera.

To set the value of parameters from `String` to `Sensitive` in Hiera, add the parameters to a `lookup_options` section in the `common.yaml` file. For example:

```
---
lookup_options:
  '^cd4pe::db_pass$':
    convert_to: 'Sensitive'
  '^cd4pe::root_config::root_password$':
    convert_to: 'Sensitive'
  '^cd4pe::root_config::s3_secret_key$':
    convert_to: 'Sensitive'
  '^cd4pe::root_config::artifactory_access_token$':
    convert_to: 'Sensitive'
```

Configuring and adding integrations

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.

- Integrate with source control on page 42

Integrate your source control system with Continuous Delivery for Puppet Enterprise (PE) by following the appropriate set of instructions on this page.

- Integrate with Puppet Enterprise on page 45

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.

- Configure impact analysis on page 47

In order to use impact analysis, you must configure both your Puppet Enterprise (PE) and Continuous Delivery for PE instances. This process involves generating a new certificate, installing a dedicated module, updating classification of your Puppet master, and adding credentials to Continuous Delivery for PE.

- Configure job hardware on page 49

Job hardware, or the servers Continuous Delivery for Puppet Enterprise (PE) uses to run tests on your Puppet code, must be configured before you can begin running tests or using pipelines.

- Analytics data collection on page 52

Continuous Delivery for Puppet Enterprise (PE) automatically collects data about how you use the software. If you want to opt out of providing this data, you can do so when installing Continuous Delivery for PE.

- Configure LDAP on page 54

Continuous Delivery for Puppet Enterprise (PE) supports use of the Lightweight Directory Access Protocol (LDAP) for managing user authentication. Once an LDAP configuration is in place, 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 56

Configure SMTP for your Continuous Delivery for Puppet Enterprise (PE) installation so that users can receive email notifications from the software.

- Configure SSL on page 57

Continuous Delivery for Puppet Enterprise (PE) supports the use of Secure Sockets Layer (SSL) for enhanced security when using the software.

Integrate with source control

Integrate your source control system with Continuous Delivery for Puppet Enterprise (PE) by following the appropriate set of instructions on this page.

Integrate with Azure DevOps

Continuous Delivery for PE works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. Create an Azure DevOps OAuth application in order to integrate your Azure DevOps instance with Continuous Delivery for PE and start using these tools.

Before you begin

An administrator on your team must create an Azure DevOps OAuth application for Continuous Delivery for PE.

1. Sign into Continuous Delivery for PE as the root user.
2. Click **Settings**, then click **Integrations**.

   Tip: The authorization callback URL required to create your OAuth app is shown in the root console.
3. Go to https://app.vsaex.visualstudio.com/app/register. 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 root console.
7. In the **Authorized scopes** section, select **Code (read and write)**.
8. Click **Create Application**. Your new application is created, and a new page showing the application's settings is displayed.

   Important: Leave this page open. You'll 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 OAuth application and click **Add**.

Once an Azure DevOps OAuth application is established for your organization, each workspace must be authenticated with the application in order to integrate the Continuous Delivery for PE instance with Azure DevOps. This process involves granting code read and write permissions and adding a public SSH key, which enables cloning of modules and control repos during automated tasks.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. Click **Source Control**, then click **Azure DevOps**.
3. Click **Add Credentials** to give Continuous Delivery for PE code read and write permissions for your Azure DevOps account. You are directed to a Microsoft page.
4. Click **Accept**. You are directed back to the **Source Control** page.
5. Next, add the SSH key. Still in the Continuous Delivery for PE web UI, click **SSH Key**.
6. Click **Show** to display your public SSH key. Click **Copy**.
7. In the Azure DevOps web UI, open the user menu and click **Security**, then click **SSH public keys**.
8. Click **Add** and paste your public SSH key into the **Key Data** field. Add a description and click **Save**.

Integrate with Bitbucket Cloud

Continuous Delivery for PE works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. Create a Bitbucket Cloud OAuth application in order 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.

2. Click **Settings**, then click **Integrations**.

   Tip: The authorization callback URL required to create your OAuth consumer is shown in the root console.
3. In your organization's Bitbucket Cloud account, create an OAuth consumer. See Create a consumer in the Bitbucket Cloud documentation for instructions.
4. When your OAuth application is created, note the key and secret shown on the OAuth settings page in the Bitbucket Cloud web UI.
5. 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 a Bitbucket Cloud OAuth application is established for your organization, each workspace must be authenticated with the application in order to integrate the Continuous Delivery for PE instance with Bitbucket Cloud.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. Click **Source Control**, then click **Bitbucket Cloud**.
3. Click **Add Credentials** to give Continuous Delivery for PE code read and write permissions for your Azure DevOps account. You are directed to a Microsoft page.
4. Click **Add Credentials**.

   At this point you'll be redirected to Bitbucket Cloud to authorize the OAuth application set up by your workspace administrator.

   Give Continuous Delivery for PE the following permissions on your Bitbucket Cloud account:

   - Access organizations, teams, and membership (read-only)
   - Access user email addresses (read-only)
   - Access public and private repositories
5. Click **Authorize application**.

Integrate with Bitbucket Server

Continuous Delivery for 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 in order to start using these tools.

Note: Continuous Delivery for PE supports Bitbucket Server 5.0 and newer versions.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. Click **Source Control**, then click **Bitbucket Server**.
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 wish 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. Enter the SSH base URL for your Bitbucket Server if it is different from the host 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**.

Integrate with GitHub

Continuous Delivery for PE works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. Create a GitHub OAuth application in order 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.
2. Click **Settings**, then click **Integrations**.

   Tip: The authorization callback URL required to create your OAuth app is shown in the console.
3. In your organization's GitHub account, create an OAuth application. See Creating an OAuth App in the GitHub documentation for instructions.
4. Once your OAuth application is created, note the Client ID and Client Secret shown on the application's page in the GitHub UI.
5. 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 a GitHub OAuth application is established for your organization, each workspace must be authenticated with the application in order to integrate the Continuous Delivery for PE instance with GitHub.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. Click **Source Control**, then click **GitHub**.
3. Click **Add Credentials**.

   At this point you'll be redirected to GitHub to authorize the OAuth application set up by your team's administrator.

   Give Continuous Delivery for PE the following permissions on your GitHub account:

   - Access organizations, teams, and membership (read-only)
   - Access user email addresses (read-only)
   - Access public and private repositories
4. Click **Authorize application**.

Integrate with GitHub Enterprise

Continuous Delivery for 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 in order to start using these tools.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. Click **Source Control**, then click **GitHub Enterprise**.
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 your profile photo, then click **Settings**.
   b) Click **Personal access tokens**, and click **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 certificate in full in the **Custom CA Certificate** field.

7. Click **Add Credentials**.

Integrate with GitLab

Continuous Delivery for 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 in order to start using these tools.

Important: In order for Continuous Delivery for PE to work correctly with GitLab, make sure that none of your environment branches (the branches Continuous Delivery for PE works with when deploying new code to nodes) are GitLab protected branches.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. Click **Source Control**, and then click **GitLab**.
3. In the **Host** field, enter the public IP or DNS for your GitLab instance.
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.
   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. Optional: Add the SSH user's credentials in the **SSH User** field.
7. Optional: In the **SSH Port** field, specify the port on which your GitLab server listens for SSH requests. The default port number is 22.
8. Click **Add Credentials**.

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.

Create a Continuous Delivery user and user role in PE

Create a "Continuous Delivery" user and user role in PE. This allows you to view a centralized log of the activities Continuous Delivery for PE performs on your behalf. You'll also use this user account when generating the PE authentication token required by the setup process.

1. To begin, create a new user. 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 (optional) description for new role, such as `CDPE User Role`, then click **Add role**.
5. Select the user role you've just created from the list on the **User roles** page.

6. Click **Permissions**. Assign the following 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	-
Note: This permission is only available in PE 2019.1.x and newer versions. If you are using an older version of PE, you do not need to set this permission.		

7. Once all the permissions have been added, click **Commit changes**.
8. Add your Continuous Delivery user to the user role. Click **Member users**. Select the name of the user you created earlier, and click **Add user**, then commit your change.
9. Your user is now set up and has been given the permissions needed to operate Continuous Delivery for PE. Before proceeding, create a password for the Continuous Delivery user.
   a) Return to the **Users** page.
   b) Find and click the full name of your newly created user, then click **Generate password reset**.
   c) Follow the link created and create a password for the user. You'll use this password when adding your PE credentials to 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.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. Click **Puppet Enterprise**. Click **Add Credentials**.
3. In the **New Puppet Enterprise Credentials** pane, enter a unique friendly name for your Puppet Enterprise installation.

   If you need to work with multiple PE installations within Continuous Delivery for PE, these friendly names help you to differentiate which installation's resources you're managing, so choose them carefully.
4. Enter the web address you use to access the PE console. This address must match the certname of your PE master or an alias included in the `dns_alt_names` entry in your `puppet.conf` file.

5. 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. Optionally, change the token's lifetime by clicking **Edit**.
   - For **API Token**, generate a PE access token for your "Continuous Delivery" user using `puppet-access` or the RBAC v1 API, and paste it in the **API Token** field.

     For instructions on generating an access token, see Token-based authentication.

     Tip: To avoid unintended service interruptions, create an access token with a multiyear lifespan.
6. Click **Save Changes**.

Continuous Delivery for PE uses the information you provide to look up the endpoints for PuppetDB, Code Manager, orchestrator, and node classifier, and to access the master SSL certificate generated during PE installation. Once your credentials are successfully added, click **Edit Credentials** to view this information.

Your PE instance is now integrated with Continuous Delivery for PE.

If you're using PE version 2019.1.x or 2019.2.x, impact analysis was automatically configured for you during this integration. For users of all older PE versions: to enable impact analysis for this instance, see Configure impact analysis.

Configure impact analysis

In order to use impact analysis, you must configure both your Puppet Enterprise (PE) and Continuous Delivery for PE instances. This process involves generating a new certificate, installing a dedicated module, updating classification of your Puppet master, and adding credentials to Continuous Delivery for PE.

For best results, work through the configuration process in the order presented here.

What is impact analysis?

Impact analysis is a Continuous Delivery for PE tool that lets you see the potential impact that new Puppet code will have on your PE-managed infrastructure, even before the new code is merged. When you add impact analysis to a control repo's pipeline, Continuous Delivery for PE automatically generates a report on every proposed code change to that control repo.

See Previewing the impact of code changes for more on impact analysis.

Before you begin

Make sure you're ready to get started by completing the following:

- Install Continuous Delivery for PE.

  Important: If you are running PE version 2019.1.x or 2019.2.x and have integrated your PE instance with Continuous Delivery for PE, **you do not need to complete this configuration**. Impact analysis was automatically configured for you during the integration.
- Integrate with your source control system.
- Integrate with Puppet Enterprise.

Generate a dedicated Puppet Enterprise certificate

A dedicated PE certificate is required to authenticate with the endpoints used for impact analysis. For security purposes, make sure you generate a new certificate.

Note: This certificate is not counted as one of your PE node licenses.

On the node hosting your PE certificate authority service (typically the Puppet master), run the command appropriate to your version of PE.

- For PE version 2019.1 or 2019.2: `puppetserver ca generate --certname <CERTIFICATE_NAME>`
- For PE version 2018.1: `puppet cert generate <CERTIFICATE_NAME>`

Replace `<CERTIFICATE_NAME>` with the name you wish to give the certificate, such as `cd4pe-impact-analysis`.

Install modules

Impact analysis requires you to install the `puppetlabs-cd4pe` module and its dependent modules.

The `puppetlabs-cd4pe` module must be used with seven dependent modules. The eight modules and their required versions are as follows:

Module	Required version
`puppetlabs-cd4pe`	1.1.0 or later
`puppetlabs-stdlib`	4.19.0 or later
`puppetlabs-puppet_authorization`	0.5.0
`puppetlabs-hocon`	0.9.3 or later in the 0.x or 1.x series
`puppetlabs-concat`	1.1.1 or later in the 1.x, 2.x, 3.x, or 4.x series
`puppetlabs-docker`	3.3.0 or later
`puppetlabs-apt`	4.4.1 or later
`puppetlabs-translate`	1.1.0 or later

1. Add the eight modules listed above to the Puppetfile for each environment against which your compilers compile catalogs.

   A sample Puppetfile entry:

   ```
   mod 'puppetlabs-cd4pe', :latest
   # Requirements for cd4pe
   mod 'puppetlabs-concat', '4.2.1'
   mod 'puppetlabs-hocon', '1.0.1'
   mod 'puppetlabs-puppet_authorization', '0.5.0'
   mod 'puppetlabs-stdlib', '4.25.1'
   mod 'puppetlabs-docker', '3.3.0'
   mod 'puppetlabs-apt', '6.2.1'
   mod 'puppetlabs-translate', '1.1.0'
   ```

2. Deploy the updated code to the relevant environments by running `puppet code deploy <ENVIRONMENT>`.

   Remember: Continuous Delivery for PE performs this step for you if you've configured it to deploy code changes automatically to the relevant environments.

Update classification

Once `puppetlabs-cdpe` and its dependencies have been deployed, you must update classification of your nodes and whitelist the certificate you generated for impact analysis.

1. In the PE console, click **Classification** and open the **PE Infrastructure** group.
2. Select the **PE Master** group and click **Configuration**.
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. In the newly added **cd4pe::impact_analysis** class, select the **whitelisted_certnames** parameter.

5. In the **Value** field, enter an array containing the name of the certificate you generated earlier, using the following format: `["<CERTIFICATE_NAME>"]`.

   CAUTION: Do not include angle brackets (< and >) in the array.

6. Run Puppet on the nodes in the **PE Master** group.

   Important: This Puppet run restarts the pe-puppetserver service.

Add credentials to Continuous Delivery for PE

The final step in the impact analysis configuration process is to update your PE instance's credentials in the Continuous Delivery for PE web UI.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. Click **Puppet Enterprise**. Click **Edit** for the PE instance you're adding impact analysis to.
3. Click **Impact Analysis Credentials** to display the configuration controls.
4. Enter the endpoint for the Puppet Server service. You can locate this endpoint with the PE console.
   a) In the PE console, click **Overview**, then click **Puppet Services status**.
   b) Copy the Puppet Server endpoint from the Puppet Services status monitor.
   c) In the Continuous Delivery for PE web UI, paste the endpoint into the **Puppet Server Service** field.
5. Retrieve the new PE certificate you generated earlier and its private key from your Puppet Server certificate authority service by running the following:

   ```
   cat /etc/puppetlabs/puppet/ssl/certs/<CERTIFICATE_NAME>.pem
   cat /etc/puppetlabs/puppet/ssl/private_keys/<CERTIFICATE_NAME>.pem
   ```

6. Copy the certificate in its entirety (including the header and footer) and paste it in the **Impact Analysis Certificate** field. Then copy the private key in its entirety (including the header and footer) and paste it in the **Impact Analysis Private Key** field.
7. Click **Save Changes**.

Continuous Delivery for PE validates that everything is configured correctly. Once you receive a success message, you're ready to use impact analysis.

To get started, see Previewing the impact of code changes.

Configure job hardware

Job hardware, or the servers Continuous Delivery for Puppet Enterprise (PE) uses to run tests on your Puppet code, must be configured before you can begin running tests or using pipelines.

Important: Due to a Docker known issue, running job hardware in the container where Continuous Delivery for PE is installed results in a `Failed to connect to <ADDRESS> port <PORT NUMBER>: Host is unreachable` error. To work around this issue, install your job hardware in a separate container, or update your Docker firewall settings to allow the two services to communicate.

Selecting a job hardware installation method

There are two methods for authenticating your account when configuring job hardware. Which you choose to employ is mainly a question of how many servers you're designating as job hardware.

When installing the Continuous Delivery agent, you must enter the email address and password associated with your Continuous Delivery for PE account. Doing this manually is fine if you're setting up a server or two, but if you're configuring a larger number of servers, adding these credentials can quickly become cumbersome. In this case, using a `distelli.yml` file to automatically pass your credentials to the agent is the more efficient method.

Configure job hardware with the web UI

To designate a server as job hardware, install the Continuous Delivery agent on the server and mark it as active job hardware in the web UI.

Tip: This method is best to use if you're configuring a small number of servers.

1. In the Continuous Delivery for PE web UI, click **Hardware**.
2. Click **Add Job Hardware**.
3. Select **Linux / MacOS** or **Windows**. Install the Continuous Delivery agent by running the commands shown in the web UI on the machine you've designated as job hardware.

   Important: On Linux hosts, the agent installation process adds a sudoers rule to `/etc/sudoers.d/distelli` that enables the `distelli` user to run any command with sudo privileges and without requiring a password.
4. When prompted for your email and password, enter the credentials associated with your Continuous Delivery for PE account.

   Important: Do not enter the root account credentials.
5. At the prompt `To which hardware collection should this agent be added?` select the relevant workspace.
6. In the Continuous Delivery for PE web UI, close the **Add Job Hardware** pane. You might need to refresh the **Job Hardware** page to see your newly configured job hardware.
7. Click the **Job Hardware Active** toggle to mark your new job hardware as active.
8. Click **+ Add Capability** and enter up to three capabilities associated with this piece of job hardware, such as `PUPPET-AGENT` or `DOCKER`, pressing **Save** after each addition.

   The Continuous Delivery agent automatically detects and displays four reserved capabilities: **WINDOWS**, **LINUX**, **DARWIN**, and **RASPBIAN**.

   When creating a job, you can specify which job hardware capabilities are required. This allows you to ensure that jobs requiring specific conditions, such as a particular operating system or piece of software, are run only on job hardware meeting those conditions.

   Important: In order to run jobs in Docker containers (including pre-built jobs), make sure Docker is installed on your job hardware and set **DOCKER** as a capability.

   Note: Running Docker-enabled job hardware on an Alpine Linux base image requires the libgcc, bash, wget, and ca-certificates packages.

Configure job hardware with `distelli.yml`

By setting up a `distelli.yml` file containing your user-specific agent credentials, you can configure job hardware without manually entering your account credentials.

Tip: This method is best to use if you're configuring a large number of servers and don't want to enter your credentials manually each time.

1. Create your agent credentials.
   a) In the Continuous Delivery for PE web UI, click **Settings**.
   b) Click **Hardware Agents**, then click **Create Agent Credentials**.
   c) Continuous Delivery for PE creates an access token and secret key for your account. Click **Show** to display your secret key. Leave this page open while you set up your `distelli.yml` file.
2. Create a file named `distelli.yml` and store it in a convenient location.

3. Add the following to your `distelli.yml` file, pasting in the access token and secret key you generated in Step 1:

   ```
   DistelliAccessToken: <MY_ACCESS_TOKEN>
   DistelliSecretKey: <MY_SECRET_KEY>
   ```

   Save the file and exit.
4. In the Continuous Delivery for PE web UI, click **Hardware**.
5. Click **Add Job Hardware**.
6. Select **Linux / MacOS** or **Windows**. SSH into the machine you've chosen to designate as job hardware. In sudo or Administrator mode, install the Continuous Delivery agent by running the commands shown in the web UI, adding `-conf <PATH_TO_DISTELLI.YML_FILE>` to the end of the `distelli agent install` command.
   For example, `/usr/local/bin/distelli agent install -conf <PATH_TO_DISTELLI.YML_FILE>`.

   Important: On Linux hosts, the agent installation process adds a sudoers rule to `/etc/sudoers.d/distelli` that enables the `distelli` user to run any command with sudo privileges and without requiring a password.
7. At the prompt `To which hardware collection should this agent be added?` select the relevant workspace.
8. In the Continuous Delivery for PE web UI, click **Hardware**.
9. Click the **Job Hardware Active** toggle to mark your new job hardware as active.
10. Click **+ Add Capability** and enter up to three capabilities associated with this piece of job hardware, such as `PUPPET-AGENT` or `DOCKER`, clicking **Save** after each addition.

    The Continuous Delivery agent automatically detects and displays four reserved capabilities: **WINDOWS**, **LINUX**, **DARWIN**, and **RASPBIAN**.

    When creating a job, you can specify which job hardware capabilities are required. This allows you to ensure that jobs requiring specific conditions, such as a particular operating system or piece of software, are run only on job hardware meeting those conditions.

    Important: In order to run jobs in Docker containers (including pre-built jobs), make sure Docker is installed on your job hardware and set **DOCKER** as a capability.

    Note: Running Docker-enabled job hardware on an Alpine Linux base image requires the libgcc, bash, wget, and ca-certificates packages.

Configure global shared job hardware

Global shared job hardware can be used by all workspaces in your Continuous Delivery for PE installation. A super user must set up these special job hardware severs in the root console.

A super user must perform this action. Once set up, all super users and the root user can view global shared job hardware in the root console.

1. Log into the root console by selecting **Root Console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar.
2. Click **Hardware**, then click **Add Job Hardware**.
3. Select **Linux / MacOS** or **Windows**. Install the Continuous Delivery agent by running the commands shown in the web UI on the machine you've designated as global shared job hardware.

   Important: On Linux hosts, the agent installation process adds a sudoers rule to `/etc/sudoers.d/distelli` that enables the `distelli` user to run any command with sudo privileges and without requiring a password.
4. When prompted for your login email and password, enter the credentials associated with your super user account.

5. At the prompt `To which hardware collection should this agent be added?` select `Global shared hardware`.
6. In the root console, close the **Add Job Hardware** pane. You might need to refresh the **Job Hardware** page to see your newly configured global shared job hardware.
7. Click the **Job Hardware Active** toggle to mark your new job hardware as active.
8. Click **+ Add Capability** and enter up to three capabilities associated with this piece of job hardware, such as `PUPPET-AGENT` or `DOCKER`, pressing **Save** after each addition.

   The Continuous Delivery agent automatically detects and displays four reserved capabilities: **WINDOWS**, **LINUX**, **DARWIN**, and **RASPBIAN**.

   When creating a job, users can specify which job hardware capabilities are required. This ensures that jobs requiring specific conditions, such as a particular operating system or piece of software, are run only on job hardware meeting those conditions.

   Important: In order to run jobs in Docker containers (including pre-built jobs), make sure Docker is installed on your job hardware and set **DOCKER** as a capability.

   Note: Running Docker-enabled job hardware on an Alpine Linux base image requires the libgcc, bash, wget, and ca-certificates packages.

Now that shared global job hardware is available, a **Use shared hardware** option is shown when creating or editing a job.

Upgrade the Continuous Delivery agent

To upgrade the Continuous Delivery agent to a more recent version, reinstall the agent on top of the existing agent.

1. In the Continuous Delivery for PE web UI, click **Hardware**.
2. Click **Add Job Hardware**.
3. Select **Linux / MacOS** or **Windows**. SSH into the machine you're using as job hardware. In sudo or Administrator mode, install the Continuous Delivery agent by running the commands shown in the web UI.

   Important: If you created a distelli.yml file to store your agent credentials, add `-conf <PATH_TO_DISTELLI.YML_FILE>` to the end of the `distelli agent install` command and skip to step 5 below.
4. When prompted for your email and password, enter the credentials associated with your Continuous Delivery for PE account.

   Important: Do not enter the root account credentials.
5. In the Continuous Delivery for PE web UI, click **Hardware** and locate the newly upgraded server.
6. Click the **Job Hardware Active** toggle to mark the upgraded job hardware as active.
7. Click **+ Add Capability** and enter up to three capabilities associated with this piece of job hardware, such as `PUPPET-AGENT` or `DOCKER`, clicking **Save** after each addition.

Analytics data collection

Continuous Delivery for Puppet Enterprise (PE) automatically collects data about how you use the software. If you want to opt out of providing this data, you can do so when installing Continuous Delivery for PE.

Continuous Delivery for PE collects analytics data in order to better understand how our customers are using 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 the most and the least used helps us decide where to prioritize new functionality.

What data does Continuous Delivery for PE collect?

Continuous Delivery for PE collects analytics data when you use the software. No personally identifiable information is collected, and the data we collect is never used or shared outside Puppet.

The following data is collected when the software starts for the first time or restarts after an upgrade, and once per week 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)

The following data is collected while Continuous Delivery for PE is in use:

- 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 personally identifiable information 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 when installing with the `cd4pe` module

To opt out of analytics data collection, use the PE console to set the `analytics` parameter on the `cd4pe` class.

1. Follow steps 1 through 5 of the installation instructions in Install Continuous Delivery for PE with the `cd4pe` module.
2. On the **Configuration** page of the Continuous Delivery for PE node group, add the **analytics** parameter to the **cd4pe** class. Set the parameter's value to `false`.
3. Run Puppet on your Continuous Delivery for PE node group to apply the change.
   You have opted out of data collection, and your new instance of Continuous Delivery for PE will not send analytics data to Puppet.

Opt out of analytics data collection when installing with Docker

To opt out of analytics data collection, set the `ANALYTICS` environment variable to `false` during the initial Docker run of the Continuous Delivery for Puppet Enterprise container.

1. Pull down the latest version of the Docker container by running:

```
docker pull puppet/continuous-delivery-for-puppet-enterprise:latest
```

2. Follow the instructions in step 2 of Install Continuous Delivery for PE to run the image appropriate to your database infrastructure, adding `-e ANALYTICS=false` as an environment variable.

   For MySQL:

   ```
   docker run --rm --name cd4pe -v cd4pe:/disk -e
    DB_ENDPOINT=<MY_DATABASE_ENDPOINT> -e DB_USER=<MY_DB_USERNAME> -e
    DB_PASS=<MY_DB_PASS> -e DB_PREFIX=<OPTINAL_PREFIX_TO_DB_TABLES> -e
    DUMP_URI=<DUMP_URI> -e PFI_SECRET_KEY=<KEY_FOR_DB_SECRETS_ENCRYPTION> -e
    ANALYTICS=false -p 8080:8080 -p 8000:8000 -p 7000:7000 puppet/continuous-
   delivery-for-puppet-enterprise:latest
   ```

   For Amazon DynamoDB:

   ```
   docker run --rm --name cd4pe -v cd4pe:/disk -e DB_ENDPOINT=ddb://
   <AWS_REGION> -e DB_USER=<AWS_ACCESS_KEY> -e DB_PASS=<AWS_SECRET_KEY> -e
    DUMP_URI=<DUMP_URI> -e PFI_SECRET_KEY=<KEY_FOR_DB_SECRETS_ENCRYPTION> -e
    ANALYTICS=false -p 8080:8080 -p 8000:8000 -p 7000:7000 puppet/continuous-
   delivery-for-puppet-enterprise:latest
   ```

3. Continue with the installation process as described in the documentation.
   You have opted out of data collection, and your new instance of Continuous Delivery for PE will not send analytics data to Puppet.

Configure LDAP

Continuous Delivery for Puppet Enterprise (PE) supports use of the Lightweight Directory Access Protocol (LDAP) for managing user authentication. Once an LDAP configuration is in place, 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 your 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.

Create a new LDAP configuration

Add an LDAP configuration to Continuous Delivery for PE by providing key information on the mapping of user and group attributes in your LDAP server implementation.

1. Log into the root console by adding `/root/login` to the end of the base URL of the Continuous Delivery for PE web UI and signing in as the root user.
2. Click **Settings**, then click **Single Sign On**.
3. Select **LDAP** and click **+ Add LDAP Configuration**.

4. In the **New LDAP Configuration** pane, enter the requested information as per the instructions below.

   Name

   A friendly identifier for this LDAP configuration. This name cannot be changed, so select it carefully.

   Endpoint URL

   The endpoint URL of the LDAP server, 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 will bind as when performing LDAP operations. An admin account or a service account created specifically for Continuous Delivery for PE is generally used here. If you choose to create a new account, ensure that 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 the LDAP configuration is updated.

   User base DN

   The LDAP base DN that informs Continuous Delivery for PE where users are located in the directory. The more specific the DN, the better your LDAP search performance will be.

   User attribute

   Which LDAP user attribute to use to map users. Since a user's email address is required to log into Continuous Delivery for PE, use a user attribute with an email address for a value (most commonly `mail`). If your LDAP users have duplicate emails, they will need to be changed.

   User base filter

   A filter that Continuous Delivery for PE can use to restrict user search results. This is useful in edge cases where certain users should be included or excluded.

   Group base DN

   The LDAP base DN that informs Continuous Delivery for PE where groups are located in the directory. The more specific the DN, the better your LDAP search performance will be.

   Note: If users and groups are stored in the same location, the value in this field will be the same as the value in the **User Attribute** field.

   Group user attribute

   The user attribute that group entries use to identify users. In most cases the value of this option is `dn`.

   Group member attribute

   The group attribute that maps to a group member. In most cases the value of this option is either `member` or `uniqueMember`.

   Group name attribute

   The group attribute that identifies the group name. In most cases the value of this option is `cn`.

   Group base filter

   A filter that Continuous Delivery for PE can use to restrict group search results. This is useful in edge cases where certain groups should be included or excluded.

   User object class

   Specifies the value of the `objectClass` attribute that allows Continuous Delivery for PE to query user entries. In most cases the value of this option is `user` or `person`.

   Group object class

   Specifies the value of the `objectClass` attribute that allows Continuous Delivery for PE to query group entries. In most cases the value of this option is `group` or `groupOfUniqueNames`.

   User member attribute

   Specifies the user attribute that can be used to identify the membership of a group. If present, the value of this option is generally `memberOf`.

5. Select the **Priority** number for this LDAP configuration. This number indicates the order in which your LDAP configurations are used to search for users during login and when synchronizing groups.
6. 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.
7. 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. Set the toggle to **Enable LDAP**, then click **Run Configuration Test** to check the connection between Continuous Delivery for PE and the LDAP server.

   Note: This configuration test checks to see if the LDAP server can be reached; it does not test the other configuration options.
9. Click **Save Configuration**. Your new LDAP configuration is now shown on the **Single Sign On** settings page, where you have the option to edit or delete the configuration.

   Important: Once you enable an LDAP configuration, Continuous Delivery for PE will automatically disable all local Continuous Delivery for PE accounts other than the root account, and will attempt to use LDAP authentication. If your LDAP authentication fails, sign in as the root user by adding `/root/login` to the end of the base URL of the Continuous Delivery for PE web UI and adjust your settings.

Create an LDAP group map

Once you add an LDAP configuration to Continuous Delivery for PE, use a group map to map your existing LDAP groups to Continuous Delivery for PE RBAC groups. This makes it possible to mirror group membership defined in LDAP to groups in Continuous Delivery for PE.

Before you begin

Add at least one LDAP configuration to your Continuous Delivery for PE instance.

1. Log into the root console by adding `/root/login` to the end of the base URL of the web UI and signing in as the root user.
2. Click **Settings**, then click **Single Sign On**.
3. Click **LDAP**, then click **Manage Groups**. Click **+ Add LDAP Group Mapping**
4. Select the LDAP configuration to associate the group mapping with.
5. From the list of available LDAP groups, select the group to be used to perform the mapping.
6. Select a Continuous Delivery for PE account that is associated with the RBAC group you wish to map to the selected LDAP group.
7. From the list of available Continuous Delivery for PE RBAC groups, select the RBAC group you wish to map to the selected LDAP group.
8. Click **Add Group Mapping**.
   Once a group map is set up, Continuous Delivery for PE synchronizes with your LDAP groups based on the mapping you create.

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 complete this process.

1. Log into the root console by selecting **Root Console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar or signing in as the root user.
2. Click **Settings** > **SMTP**.
3. If your SMTP server requires authentication, enter the **SMTP Username** and **SMTP Password** in the relevant fields.

4. Enter the name of your **SMTP Host** and the **SMTP Port** number in the relevant fields.

   Note: Contact your email server administrator if you need help determining the correct SMTP port.
5. If your server requires TLS authentication, click the toggle to **Enable TLS**.
6. Click **Save Settings**.
7. To test your new SMTP configuration, generate a password reset email.
   a) Log out of Continuous Delivery for PE. On the sign-in screen, click **Forgot your password?**
   b) Enter the email address associated with your Continuous Delivery for PE account and click **Send Reset Instructions**.

      Important: You can safely ignore the reset password instructions and keep your current password.

Once SMTP is configured, Continuous Delivery for PE sends email in these situations:

- Password reset instructions when requested by a user
- Password reset confirmation once a password is updated
- Deployment approval request notifications when a deployment to a protected environment is proposed

Configure SSL

Continuous Delivery for Puppet Enterprise (PE) supports the use of Secure Sockets Layer (SSL) for enhanced security when using the software.

SSL requirements

Before enabling SSL on your system, review the following important information.

1. Enabling SSL requires super user permissions.
2. After configuring SSL, you must reinstall the Continuous Delivery agent on all job hardware. For instructions, see Configure job hardware.
3. Enabling SSL requires a restart of the Continuous Delivery for PE Docker container.

Setting up a new SSL configuration

Configure your Continuous Delivery for PE instance to use SSL by entering the relevant certificates in the root console and then updating your web UI endpoint to reflect the new DNS host and SSL port.

Before you begin

Review the SSL requirements section above.

1. Log into the root console by selecting **Root Console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar or signing in as the root user.
2. Click **Settings** and make sure you're viewing the **Endpoints** tab.
3. In the Configure SSL area, paste in the CA certificate, server certificate, and server private key for your Continuous Delivery for PE host.
4. Optional: Click the toggle to **Enable SSL**.

   Note: You can leave your SSL configuration disabled and save the information you've entered. If SSL information is entered and saved but not enabled, your certificates are saved and the private key is saved in an encrypted format until you're ready to enable SSL.
5. Click **Save SSL Settings**. If you've enabled SSL, proceed to the next step.
6. Stop the Continuous Delivery for PE container by running `docker stop <CONTAINER NAME>`.

7. Restart Continuous Delivery for PE with the command appropriate to your database infrastructure, replacing the options with the appropriate values, as per the instructions below.

   For MySQL:

   ```
   docker run --rm --name cd4pe -v cd4pe:/disk -e SSL_PORT=8443 -e
    DB_ENDPOINT=<MY_DATABASE_ENDPOINT> -e DB_USER=<MY_DB_USERNAME> -e
    DB_PASS=<MY_DB_PASS> -e DB_PREFIX=<OPTINAL_PREFIX_TO_DB_TABLES> -e
    DUMP_URI=<DUMP_URI> -e PFI_SECRET_KEY=<KEY_FOR_DB_SECRETS_ENCRYPTION>
    -p 8443:8443 -p 8000:8000 -p 7000:7000 puppet/continuous-delivery-for-
   puppet-enterprise:latest
   ```

   For Amazon DynamoDB:

   ```
   docker run --rm --name cd4pe -v cd4pe:/disk -e SSL_PORT=8443
    -e DB_ENDPOINT=ddb://<AWS_REGION> -e DB_USER=<AWS_ACCESS_KEY>
    -e DB_PASS=<AWS_SECRET_KEY> -e DUMP_URI=<DUMP_URI> -e
    PFI_SECRET_KEY=<KEY_FOR_DB_SECRETS_ENCRYPTION> -p 8443:8443 -p 8000:8000
    -p 7000:7000 puppet/continuous-delivery-for-puppet-enterprise:latest
   ```

   `--name`

   A friendly name you give to the running image. We used `cd4pe` in the example above.

   `-v cd4pe:/disk`

   Creates a new Docker volume called `cd4pe` for the user, which will be used for storing logs, job manifests, and artifacts. To learn more about this Docker volume, use the command `docker volume inspect cd4pe`. To learn more about Docker volumes in general, see the Docker documentation.

   Important: If you wish to use Amazon S3 or Artifactory as an external storage provider, do not include this option in your `docker run` command.

   `SSL_PORT`

   Sets the SSL port. By default, Continuous Delivery for PE uses port 8443 for SSL.

   `DB_ENDPOINT`

   The `mysql://` or `ddb://` endpoint used to connect to your database. For example, `mysql://samplehost:3306/cdpe`

   `DB_USER` and `DB_PASS`

   For MySQL: Login credentials for your MySQL user. For security purposes, select a database user who can connect to only this database.

   For Amazon DynamoDB: Access and secret keys for your DynamoDB user.

   Important: Make sure you generate credentials with full create, read, update, and delete permissions for DynamoDB resources.

   For security purposes, select a database user who can connect to only this database.

   `DB_PREFIX`

   When starting up, Continuous Delivery for PE creates tables in MySQL or DynamoDB. If you'd like the tables to share a prefix, such as `cdpe-`, enter it here.

   Tip: If you wish to simulate a fresh installation of a given version of Continuous Delivery for PE, entering a new database table prefix causes all the database tables to regenerate.

   `DUMP_URI`

   How to address port 7000 of this container, which is the endpoint used by the Continuous Delivery for PE web UI to connect to the correct instance of the Continuous Delivery agent service. In a typical installation, this value is `dump://localhost:7000`.

   `PFI_SECRET_KEY`

   A 16-byte secret key used for AES encryption of secrets (such as PE access tokens) supplied to Continuous Delivery for PE.

   If you're a *nix user, generate this key by running:

   ```
   dd bs=1 if=/dev/urandom count=16 2>/dev/null | base64
   ```

8. In the root console, update the WebUI endpoint in the **Configure Endpoints** area. The format for the new WebUI endpoint is `https://<DNS_HOST>:<SSL_PORT>`.
9. **Azure DevOps users:** Update the backend service endpoint to use `https`. This change allows Azure DevOps webhooks to function correctly.

You can now access Continuous Delivery for PE over SSL by pointing your web browser to the new WebUI endpoint entered above. Access over both `https` and `http` is allowed.

Now that SSL is configured on your system, you must reinstall the Continuous Delivery agent on all job hardware. For instructions, see Configure job hardware.

Managing workspaces

Workspaces enable you to share access to key Continuous Delivery for PE resources, such as control repos, modules, and jobs, with the other members of your team. Once you set up a workspace, add your team members to that workspace and give them the user permissions needed to do their work.

Best practices when creating workspaces

This section offers suggestions and best practices for creating workspaces. This guidance is intended to help you and your organization understand Continuous Delivery for PE workspaces and use them effectively.

Workspaces support teams writing Puppet code and deploying that code to nodes managed by PE. Whether your organization tasks one team with writing and deploying all Puppet code, or has organized multiple teams to write and test Puppet code while a central deployment team pushes those changes to production, workspaces can help you ensure that each team member has exactly the Continuous Delivery for PE resources they need.

If your organization uses...	Workspace recommendation
One team to write, test, and deploy all Puppet code A **single source control repository** to store all Puppet code	• Use one workspace for the whole team • Set permissions carefully to ensure that each team member has access to the resources they need
Multiple teams to write and test Puppet code **Multiple source control repositories** to store Puppet code A **deployment team** responsible for getting Puppet code changes into production	• Set up separate workspaces for each writing and testing team, ideally one workspace for each control repository • Create a separate workspace for the deployment team, and allow these team members access to all other workspaces • Set permissions carefully to ensure that each team member has access to the resources they need

Set up a new workspace for a team

Perform these steps to set up a single workspace for a small, centralized team, or to create one of several team workspaces for a larger, less centralized organization.

Before you begin

Make sure all members of the team have created individual Continuous Delivery for PE accounts.

Note: When new users reach the **Choose a workspace** screen in the new account creation process, ask them to log out of Continuous Delivery for PE until the team workspace is fully set up.

1. Create a new workspace by navigating to **Manage Workspaces** at top of the navigation bar in the Continuous Delivery for PE web UI and clicking **+ Add New Workspace**. Give the workspace a descriptive name (if necessary, you can change this later in the **Settings** menu).

   When you create a workspace, you are automatically set as the owner of the workspace. Other Continuous Delivery for PE super users can also access your workspace by navigating to the root console and clicking the name of your workspace in the **Workspaces** tab.
2. Switch to the newly created workspace by clicking its name in the list of **My workspaces**, or by selecting it from the workspaces menu in the Continuous Delivery for PE web UI.
3. Follow the new workspace prompts at the top of the screen to set up the required resources for the workspace:
   a) If necessary, integrate Puppet Enterprise
   b) Integrate with your source control system
   c) Set up job hardware for this workspace or use global shared job hardware
4. Add the control repo where your team keeps its Puppet code to Continuous Delivery for PE. Click **Control Repos**, then click **Add Control Repo**.
5. Add the members of your team as users of the workspace. Click **Settings**, then **Users**, and add each team member.
6. Click **Groups** and create one or more new groups, assigning appropriate permissions and users to each group. Make sure that every team member is assigned to at least one group.

   If a user is added to a workspace but not assigned to any groups, they will see a 403 error when they sign into Continuous Delivery for PE.
7. Invite the members of the team to sign into Continuous Delivery for PE. They'll now see your newly created team workspace in the workspaces area of the Continuous Delivery for PE. web UI.

Team members are now able to view and interact with the resources you've added to the workspace according to the permissions you've assigned.

You can delete unneeded workspaces by navigating to **Settings** > **Workspace** and clicking **Delete Workspace**. Only the workspace owner or a super user can perform this action.

Testing 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 wish, from module validation to linting.

Important: Make sure you have set up job hardware and installed any needed software (such as Docker, Puppet Development Kit or the Puppet agent) before you attempt add one of these jobs to a pipeline.

What is a job?

In Continuous Delivery for PE, jobs are tests for your Puppet code. Jobs are created and stored in the web UI, where you can run them on demand and add them to an automated pipeline that runs 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 be written to take advantage of 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

It's important to be aware of any prerequisites or dependencies needed to run your jobs, and to ensure that your job hardware has the necessary software, operating systems, and other resources installed before attempting to run the job.

Install job dependencies

In order to run a job successfully, the selected job hardware must have any required dependencies installed, such as software or operating systems.

Continuous Delivery for PE comes with six jobs pre-installed, so you don't need to create new jobs from scratch in order to get started testing your code. View these jobs by clicking **Jobs** in the web UI.

In order to run the pre-installed jobs, Docker must be installed on your job hardware.

1. Follow the Installing Docker instructions appropriate to your job hardware.
2. In the Continuous Delivery for PE web UI, click **Hardware**. Locate your job server.
3. Click **+ Add Capability** and enter `DOCKER`, then click **Save**.

Pre-built job reference

Continuous Delivery for PE offers users six pre-built Docker-based jobs for testing control repos and modules. These jobs run inside a Docker container, and must be run on job hardware on which you've set a Docker capability.

Validate the syntax of a control repo's Puppetfile

To add this job to your Continuous Delivery for PE instance, click **New Job**. Fill in the fields indicated with the information provided, then click **Create Job**.

Field	Content
Job Name	control-repo-puppetfile-syntax-validate
Description	Validate that a control repo’s Puppetfile is syntactically correct
Commands: Job	`rake -f /Rakefile r10k:syntax`
Docker Configuration	Run this job in a Docker container
Docker Image Name	puppet/puppet-dev-tools
Job Hardware: Capabilities	DOCKER

Before running this job, configure job hardware and ensure that Docker is installed.

Validate the syntax of a control repo's Puppet templates

To add this job to your Continuous Delivery for PE instance, click **New Job**. Fill in the fields indicated with the information provided, then click **Create Job**.

Field	Content
Job Name	control-repo-template-syntax-validate
Description	Validate that a control repo’s Puppet templates are syntactically correct
Commands: Job	`rake -f /Rakefile syntax:templates`
Docker Configuration	Run this job in a Docker container
Docker Image Name	puppet/puppet-dev-tools
Job Hardware: Capabilities	DOCKER

Before running this job, configure job hardware and ensure that Docker is installed.

Validate the syntax of a control repo's Hiera data

To add this job to your Continuous Delivery for PE instance, click **New Job**. Fill in the fields indicated with the information provided, then click **Create Job**.

Field	Content
Job Name	control-repo-hiera-syntax-validate
Description	Validate that a control repo's Hiera data is syntactically correct
Commands: Job	`rake -f /Rakefile syntax:hiera`
Docker Configuration	Run this job in a Docker container
Docker Image Name	puppet/puppet-dev-tools
Job Hardware: Capabilities	DOCKER

Before running this job, configure job hardware and ensure that Docker is installed.

Validate the syntax of a control repo's Puppet manifest code

To add this job to your Continuous Delivery for PE instance, click **New Job**. Fill in the fields indicated with the information provided, then click **Create Job**.

Field	Content
Job Name	control-repo-manifest-validate
Description	Validate that a control repo's Puppet manifest code is syntactically correct
Commands: Job	`rake -f /Rakefile syntax:manifests`
Docker Configuration	Run this job in a Docker container
Docker Image Name	puppet/puppet-dev-tools
Job Hardware: Capabilities	DOCKER

Before running this job, configure job hardware and ensure that Docker is installed.

Validate the syntax of a module's Puppet manifest code

To add this job to your Continuous Delivery for PE instance, click **New Job**. Fill in the fields indicated with the information provided, then click **Create Job**.

Field	Content
Job Name	module-pdk-validate
Description	Validate that a module's Puppet manifest code is syntactically correct
Commands: Job	`pdk validate --parallel`
Docker Configuration	Run this job in a Docker container
Docker Image Name	puppet/puppet-dev-tools
Job Hardware: Capabilities	DOCKER

Before running this job, configure job hardware and ensure that Docker is installed.

Run rspec-puppet unit tests on a module

To add this job to your Continuous Delivery for PE instance, click **New Job**. Fill in the fields indicated with the information provided, then click **Create Job**.

Field	Content
Job Name	module-rspec-puppet
Description	Run rspec-puppet unit tests on a module
Commands: Job	`pdk test unit`
Docker Configuration	Run this job in a Docker container
Docker Image Name	puppet/puppet-dev-tools
Job Hardware: Capabilities	DOCKER

Before running this job, configure job hardware and ensure that Docker is installed.

Sample non-Docker-based module jobs

This section lists sample jobs that you can use with Continuous Delivery for PE. These jobs can be entered as-is into the **New Job** creation screen in the Continuous Delivery for PE web UI, or you can make alterations to suit your deployment's needs.

Puppet Development Kit validation tests

Use the sample jobs below to validate your module code against PDK. Select the version appropriate to your operating system.

In order to use this job successfully, you must:

- Install PDK on your job hardware
- Install `puppet-agent` on your job hardware

Job name	Description	Commands	Capabilities
module-pdk-validate-linux	Validate via PDK	`pdk validate`	LINUX
module-pdk-validate-windows	Validate via PDK	`powershell.exe -c "pdk validate"`	WINDOWS

Puppet Development Kit `rspec-puppet` tests

Use the sample jobs below to run unit tests on your module code with `rspec-puppet`. Select the version appropriate to your operating system.

In order to use this job successfully, you must:

- Install PDK on your job hardware
- Install `puppet-agent` on your job hardware

Job name	Description	Commands	Capabilities
module-pdk-test-unit-linux	Run unit tests via PDK	`pdk test unit`	LINUX
module-pdk-test-unit-windows	Run unit tests via PDK	`powershell.exe -c "pdk test unit"`	WINDOWS

Sample non-Docker-based control repo jobs

This section lists sample jobs that you can use with Continuous Delivery for PE. These jobs can be entered as-is into the **New Job** creation screen in the Continuous Delivery for PE web UI, or you can make alterations to suit your deployment's needs.

Syntax validation

This job validates the syntax of everything in your control repo.

In order to use this job successfully, you must:

- Use a *nix host
- Install `puppet-agent` on your job hardware

Job name	Description	Commands	Capabilities
control-repo-validate-linux	Validate syntax	[See below]	LINUX

```
#!/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.

In order to use this job successfully, you must:

- Use a *nix host
- Install `puppet-agent` on your job hardware

Job name	Description	Commands	Capabilities
control-repo-lint-linux	Lint Puppet code	[See below]	LINUX

```
#!/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
```

Constructing pipelines

Once you set up a pipeline for a given branch of your control repo or module, every time you commit code to that branch, the tasks you've added to the pipeline run automatically.

Stages and tasks in pipelines

Pipelines in Continuous Delivery for PE 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.

If you want to set up some logic of the "if this job succeeds, then run the next one, otherwise stop and tell me what happened," variety, use stages to break up the pipeline into a series of incremental steps. You can set your pipeline to require manual promotion before moving on to the next stage, or to promote automatically when certain conditions are met.

Set up a pipeline

Set up a pipeline to enable automatic testing of newly added code by adding stages and jobs in the Continuous Delivery for PE web UI.

1. In the Continuous Delivery for PE web UI, click **Control Repos**. Click the name of the control repo you wish to set up a pipeline for.

   Continuous Delivery for PE automatically creates a pipeline for the branch you selected when setting up the control repo (the master branch). You'll see this branch name at the top of the **Control Repo Pipeline** area of the web UI.

   Tip: You can also set up pipelines for other branches in this repo, in order to automatically run tests on the code changes that Continuous Delivery for PE makes on your behalf. To create a pipeline for a different branch, click **Add Pipeline**. You can undo this action with **Delete Pipeline**.
2. Every pipeline needs at least one stage. Click **+ Add Stage** and select **Job**.
3. Select a job from the list, and click **Add Stage**. Your job is added to the pipeline. Click **Done**.

   If you need to run a single test on your new code, your pipeline can be this simple. But if you wish to add additional tests, you can either add them to the existing stage by clicking **Add Job**, or create another stage.
4. Create a second stage in your pipeline by clicking **Add another stage** and selecting **Job**. Select a job from the list and click **Add Stage**, then **Done**.

   Tip: If you wish to remove or reorder the stages in your pipeline, click **More Actions** ···. To delete all stages in a pipeline and start fresh, click **Delete Pipeline**.
5. You now have a two-stage pipeline with a promotion step between the stages. Select **Auto Promote** and choose a promotion condition from the options.

Your pipeline is ready for use.

Test code automatically with a pipeline

Every time you commit new code to your development or master branch, the pipeline runs the jobs you've set up to test the code and reports any errors or failures.

1. To see the pipeline in action, make a trivial change (such as adding a new line to the README file) in the master branch of your control repo. Commit your change.
2. In the Continuous Delivery for PE web UI, click **Control Repos**, then click the name of the control repo you just committed code to.
3. The **Overview** area now shows you the push event initiated by your code commit, and an entry for each job in your pipeline. The jobs report their status here, and update to show their success or failure when the job run is complete.

   Each event summary includes a link to the commit in the control repo. Job events also include a job number; clicking on this number takes you to the **Job Details** page, where you can see the job's log for its run and review error messages.

Test pull requests automatically

When "pull request" is enabled as a pipeline trigger, your pipeline automatically tests new code each time a pull request is opened against the relevant branch. By setting up a PR gate in the pipeline, you can ensure that the relevant tests run on the new code, but stop the pipeline before it proceeds to further tests or a deployment.

Note: If you're a GitLab user, you might be more familiar with the term "merge request" than "pull request," which is used by GitHub and GitHub Enterprise. Although the two terms can be used interchangeably, for the sake

of simplicity and consistency, the Continuous Delivery for PE web UI and these instructions use the term "pull request" (PR).

1. Add a pull request (PR) gate to your pipeline. A PR gate indicates that any pipeline runs triggered by a pull request stops at a certain point in the pipeline. This allows you to automatically run acceptance tests on new pull requests, but to manage additional testing or deployment of the new code manually based on the results of the initial automated tests.
   a) In your pipeline's first stage, click **More Actions** ···. Click **Add Item to Stage**, and select **Pull Request Gate**.
   b) Click **Add PR Gate**. When the gate has been added, you'll see a success message. Click **Done**.

   Important: You can add only one PR gate to a pipeline. If you decide to move the placement of a PR gate, delete the existing gate and create a new one in the desired stage.
2. To see the PR gate in action, switch to a branch in your control repo other than master. Make a trivial change (such as adding a new line to the README file) in your control repo, and commit your change, then open a pull request against the master branch for this commit.

   Remember: The pull request must be made against the branch the pipeline is set up to use in order to trigger the pipeline.
3. In the Continuous Delivery for PE web UI, click **Control Repos**, then click the name of the control repo you just committed code to.
4. The **Overview** area now shows you the pull request event initiated by your code commit, and an entry for the job in your pipeline that precedes the PR gate. Note that the job in the pipeline's second stage, beyond the PR gate, has 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.

1. In the Continuous Delivery for PE web UI, click **Control Repos**. 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**.
4. Select your Puppet Enterprise instance and the node group you wish to deploy changes to every time this pipeline runs.
5. Select a deployment policy. For purposes of this getting started guide, select **Direct Deployment**.

   See Deployment policies to learn more about the four deployment policies and how they work.
6. 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 close the **Add new Stage** window. Your new stage, showing the details of the deployment, is added to the pipeline.
8. To see the pipeline in action, make a trivial change (such as adding a new line to the README file) on the master branch of your control repo. Commit your change.

   The control repo **Overview** area now shows you the push event initiated by your code commit, and a deployment event. The deployment reports its status here, and updates to show its success or failure when the deployment is complete.

   Each event summary includes a link to the commit in the control repo. Deployment events also include a deployment number; clicking on this number takes you to the **Deployment Details** page, where you can see more information.

9. You can run the same deployment again from the web UI by retriggering the webhook. In the **Overview** area, locate the push event you wish to rerun, and click **View Webhook** .

   The webhook's request and response data is shown in the **Webhook Data** pane, which can be useful for troubleshooting.

10. Click **Redeliver Webhook**, and confirm your action. The **Overview** area now shows the new event triggered by the redelivery of the webhook.

    Note: If you've made changes to your pipeline since the last time this webhook was delivered, the redelivered webhook follows the current pipeline's sequence and rules.

Previewing the impact of code changes

Continuous Delivery for Puppet Enterprise (PE) lets you see the potential impact that new Puppet code will have on your PE-managed infrastructure even before the new code is merged. You can generate an impact analysis report for any commit, or add impact analysis to a control repo's pipeline to automatically generate a report on all proposed code changes.

Important information about the limitations of impact analysis

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 designed to be an exhaustive report — do not use it as a substitute for more exhaustive testing. For instance, 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 that these limitations exist, and that you remain alert to the possibility that code changes might have consequences for your infrastructure beyond what impact analysis shows.

The following is a non-exhaustive list of some of the limitations of impact analysis. This list might be updated from time to time, but is not complete.

- **Changes to Hiera data.** The imapct of changes to Hiera data cannot be analyzed.
- **Changes to an environment's `environment.conf` file.** The downstream impact of changes to an `environment.conf` file cannot be analyzed.
- **Brand-new code.** New Puppet classes, facts, or functions that have never before been applied to any of your nodes cannot be analyzed.
- **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 the names of classes that have been 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, the impact to nodes that receive those exported resources cannot be analyzed.
- **Sensitive data types.** Any changes to data types marked as sensitive will not be analyzed.
- **Alias metaparameter.** Any resources using the alias metaparameter cannot be analyzed.

Remember: Reports generated by impact analysis are provided "AS-IS."

Add impact analysis to a control repo pipeline

Once you add an impact analysis task to your control repo pipeline, the impact analysis report is automatically generated each time the pipeline is triggered and the conditions you've set are met.

Before you begin

1. Configure impact analysis.
2. Set up a pipeline for your control repo that includes at least one deployment.

CAUTION: Impact analysis fails if you include the `$environment` variable in your Puppet manifest. Instead, use Hiera and class parameters.

Impact analysis reports are generated by diffing a newly generated catalog for the deployment conditions you've set against the current catalog for the same deployment. The results of this process are shown in the impact analysis report.

You can add as many impact analysis tasks to your pipeline as you wish, but each stage in the pipeline can have only one impact analysis task. An impact analysis task cannot be in the same stage as a deployment.

1. In the Continuous Delivery for PE web UI, click **Control Repos**. Click the name of the control repo you wish to add impact analysis to.
2. Select the pipeline you wish to add impact analysis to. Make sure there is at least one deployment present in the pipeline.

   Impact analysis is calculated for your pipeline using the deployment conditions you've chosen.
3. Add impact analysis to a stage that does not contain a deployment. Click **More Actions** ···. Select **Add item to Stage**, then select **Impact Analysis**.

   Tip: For best results, add impact analysis to a stage in your pipeline that comes before a PR gate and before any deployment stages.
4. Set the catalog compilation batch size. By default, Continuous Delivery for PE compiles 10 catalogs at a time when performing an impact analysis task.
5. Determine which environments you want to generate an impact analysis report for.
   - Select **Run for all environments in the pipeline** to generate impact analysis for all environments used by all deployments in the pipeline.
   - Select **Run for selected environments** to choose which environments to run impact analysis on from among those used by the pipeline's deployments.
6. Click **Add Impact Analysis**.

Impact analysis is now enabled for your control repo pipeline. An impact analysis report is generated each time the pipeline runs.

Add impact analysis to a module pipeline

Once you add an impact analysis task to your module pipeline, the impact analysis report is automatically generated each time the pipeline is triggered and the conditions you've set are met.

Before you begin

1. Configure impact analysis.
2. Set up a pipeline for your module that includes at least one deployment.

CAUTION: Impact analysis fails if you include the `$environment` variable in your Puppet manifest. Instead, use Hiera and class parameters.

Impact analysis reports are generated by diffing a newly generated catalog for the deployment conditions you've set against the current catalog for the same deployment. The results of this process are shown in the impact analysis report.

You can add as many impact analysis tasks to your pipeline as you wish, but each stage in the pipeline can have only one impact analysis task. An impact analysis task cannot be in the same stage as a deployment.

Important: Impact analysis reports for module code are supported on PE 2018.1.9 and newer versions in the 2018.1 series, PE 2019.1.1 and newer versions in the 2019.1 series, and all versions in the 2019.2 series.

1. In the Continuous Delivery for PE web UI, click **Modules**. Click the name of the module you wish to add impact analysis to.
2. Select the pipeline you wish to add impact analysis to. Make sure there is at least one deployment present in the pipeline.

   Impact analysis is calculated for your pipeline using the deployment conditions you've chosen.
3. Add impact analysis to a stage that does not contain a deployment. Click **More Actions** ···. Select **Add item to Stage**, then select **Impact Analysis**.

   Tip: For best results, add impact analysis to a stage in your pipeline that comes before a PR gate and before any deployment stages.
4. Set the catalog compilation batch size. By default, Continuous Delivery for PE compiles 10 catalogs at a time when performing an impact analysis task.
5. Select the environments you want to generate an impact analysis report for.
6. For each selected environment, choose the control repo where the code associated with that environment is stored.
7. Click **Add Impact Analysis**.

Impact analysis is now enabled for your module pipeline. An impact analysis report is generated each time the pipeline runs.

Deploying Puppet code

- Introducing deployments on page 70

Deployments in Continuous Delivery for PE use the PE tools you're familiar with--Code Manager, orchestration, and Puppet environments--to deploy your new code, but don't require you to use Git to move your commits between Puppet environments. Instead, Continuous Delivery for PE does that work for you.

- Create environment node groups on page 72

In order for code deployments managed by Continuous Delivery for PE to work correctly, your environment node groups should be set up in a specific hierarchy.

- Deployment policies on page 73

Deployment policies are prescriptive workflows for Puppet code deployment that are built into Continuous Delivery for Puppet Enterprise (PE). You select the best deployment policy for your situation, and Continuous Delivery for PE does all the Git heavy lifting for you, deploying your code to the right nodes.

- Deploy code manually on page 78

Use the manual deployment workflow to push a code change to a specified group of nodes on demand.

- Deploy module code on page 78

You can deploy new module code to your Puppet environments via a Continuous Delivery for PE module pipeline. To do so, you must first 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 79

If your organization's business processes require manual review and approval before Puppet code is deployed to certain environments, set up an approval group of individuals with the authority to provide the needed review and sign-off. These approvers are contacted each time a deployment to a protected environment is proposed.

Introducing deployments

Deployments in Continuous Delivery for PE use the PE tools you're familiar with--Code Manager, orchestration, and Puppet environments--to deploy your new code, but don't require you to use Git to move your commits between Puppet environments. Instead, Continuous Delivery for PE does that work for you.

Before we learn how deployments work, it's important to define some key terms.

Environment node group

An environment is a set of nodes used for a particular purpose. An environment node group is node group you create in Puppet to represent an environment. Environment node groups define which nodes belong to which environment.

Environment node groups can be broad, such as Staging--all nodes used to stage changes prior to shipping them to production. They can also be smaller and more specific, such as Denver Production Forge Servers. When Continuous Delivery for PE deploys changes, it deploys them to one environment node group at a time.

A node cannot be in two environment node groups.

Puppet environment

A Puppet environment is a Git branch that gets turned into a directory on your Puppet master.

A Puppet environment specifies the resources that the Puppet master uses when compiling catalogs for agent nodes. The code management tools built into Puppet Enterprise create and maintain your Puppet environments based on the branches in your control repo.

For example, if your control repo has a production branch, a development branch, and a testing branch, code management creates a production Puppet environment, a development Puppet environment, and a testing Puppet environment.

Continuous Delivery for PE manages the journey from changes made in version control to checkout in a Puppet environment to deployment on nodes in an environment node group. When you kick off a deployment, you're in essence telling Continuous Delivery for PE that "I want to deploy the changes in commit #x on the master branch to nodes in my environment node group #y." Continuous Delivery for PE then talks to your version control system, to Code Manager, to the node classifier, and to the orchestrator to guide and automate deployment of the change you've requested to the right group of nodes.

The basic steps in this process are as follows:

1. You initiate a deployment and specify the commit to be deployed and the target environment node group.
2. Continuous Delivery for PE updates the target branch and Puppet environment.
3. The orchestrator runs Puppet against the impacted nodes to apply the code change.

Git branches and Continuous Delivery for PE

Continuous Delivery for PE is designed to watch for Puppet code changes you make in Git and help you deploy those changes to your environment node groups. You and Continuous Delivery for PE work together, so it's important to understand what 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 master 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 master branch is designated as the branch to file pull requests against, tag releases off of, and treat as the leading edge of new development.

It is also your responsibility to create new environment branches when you first set up environment node groups. Environment branches are long-lived Git branches used to control which version of code is made available in corresponding Puppet environments. Continuous Delivery for PE owns environment branches after they're created, deploying changes to them and keeping them up to date. You and your team won't make changes to these branches directly, because manual changes to environment branches get overwritten whenever an automated deployment occurs.

While you focus on making commits to master and getting pull requests merged, Continuous Delivery for PE works in your environment branches on your behalf, making sure your commits or changes are tested, deployed, and promoted to your environment node groups in accordance with the control repo pipelines you create and your manual deployment directives.

Create environment node groups

In order for code deployments managed by Continuous Delivery for PE to work correctly, your environment node groups should be set up in a specific hierarchy.

Continuous Delivery for PE deploys changes to environment node groups. By setting up environment node groups, you define the groups of nodes that you can choose to deploy changes to.

1. In the PE console, click **Classification**.
2. Click **Add group...** and create a new node group with the following specifications:
   - Parent name: All Nodes
   - Group name: All Environments
   - Environment: production
   - Environment group: yes
   - Description: Environment group parent and default
3. Open the new All Environments node group and add a new rule:
   - Fact: name
   - Operator: ~
   - Value: `.*`
4. Edit the Agent-specified environment node group so that All Environments is its parent. This group should have no rules, and won't match any nodes.
5. Edit the Production environment node group so that All Environments is its parent. If necessary, modify its rules so that it matches only the correct nodes.
6. For each of your environment groups (such as testing, staging, and production), create an environment node group.
   a) Create a Git branch to represent the environment.
   b) Run `puppet code deploy <ENVIRONMENT_NAME>`.
   c) Create a new environment node group with the following specifications:
      - Parent name: All Environments
      - Group name: All <ENVIRONMENT_NAME>
      - Environment: <ENVIRONMENT>
      - Environment group: Yes
   d) Associate the relevant nodes with the environment group by creating rules or pinning nodes.

      Best practices for associating nodes with environment node groups:

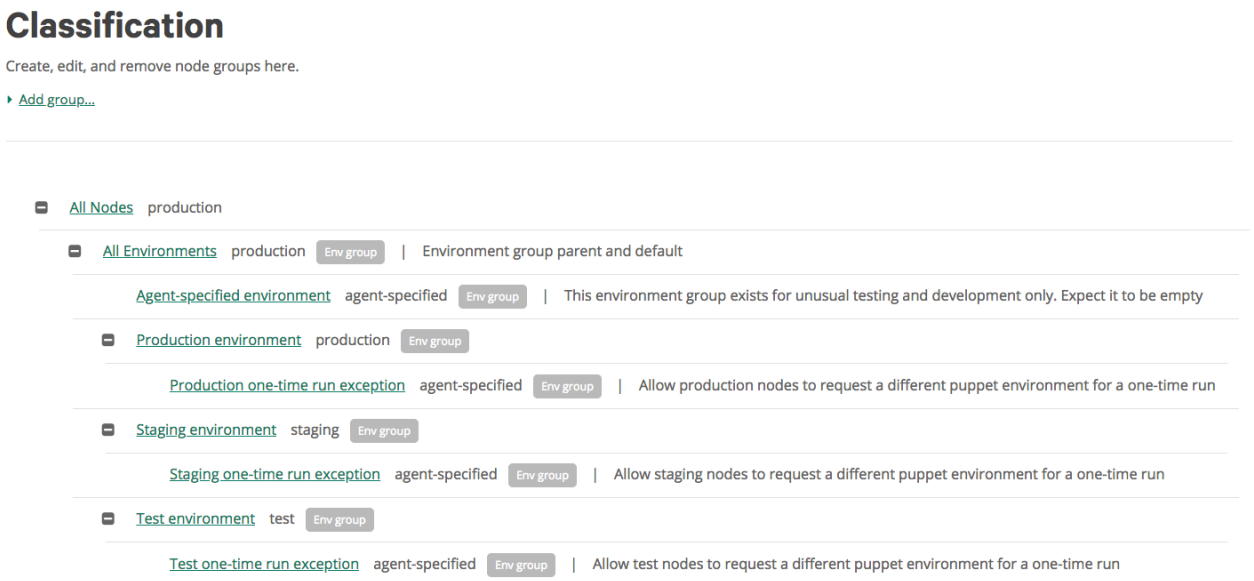
      - Use the `pp_environment` trusted fact, or a similar custom fact, to define which environment each node belongs to. Write a rule in each environment group that uses `pp_environment` or your custom fact to match nodes.
      - See if other facts or trusted facts can be used to create rules that match nodes to one and only one environment group
      - If trusted facts, custom facts, or other facts cannot be used to determine node environments, use pinning. Pin each node to only one environment group.
   e) Specify the Git branch corresponding to the environment.

7. Optional: For each of your newly created environment groups, create a child environment group. Nodes from the parent environment group are allowed to drop into this exception group to test code from Git feature branches. Give each child environment group the following specifications:

   - Parent name: All <ENVIRONMENT>
   - Group name: <ENVIRONMENT> one-time run exception
   - Environment: Agent-specified
   - Environment group: Yes
   - Description: Allow <ENVIRONMENT> nodes to request a different Puppet environment for a one-time run

   Once the child environment node group is set up, give it the rule `(agent_specified_environment ~ . +)`. Do not pin any nodes to this node group.

The resulting environment node groups has a format similar to this:



Now that your environment nodes groups are configured, we can deploy new code to your nodes.

Deployment policies

Deployment policies are prescriptive workflows for Puppet code deployment that are built into Continuous Delivery for Puppet Enterprise (PE). You select the best deployment policy for your situation, and Continuous Delivery for PE does all the Git heavy lifting for you, deploying your code to the right nodes.

Which deployment policy should I use?

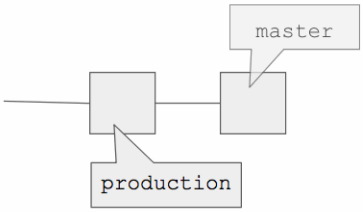
Continuous Delivery for PE includes built-in deployment policies, each with a different branching workflow. Use the table below to select the best deployment policy for your circumstances.

Deployment policy	**Best for...**
Direct deployment policy	• Small or trivial changes • Changes that you have a high level of confidence will not cause issues when deployed

Deployment policy	Best for...
Temporary branch policy	• Fully tested changes • Changes you'd like to have validated through deployment • Workflows that don't require extensive logging of historic data
Eventual consistency policy	• Fully tested changes • Large PE installations with regularly scheduled Puppet runs • Situations in which running additional orchestrator jobs is undesirable
DEPRECATED: Incremental branch policy	• Fully tested changes • Changes you'd like to have validated through deployment • Workflows that require robust historic data and the ability to pinpoint the date and time a particular change was implemented
DEPRECATED: Blue-green branch policy	• Fully tested changes • Changes you'd like to have validated through deployment • Workflows that require robust historic data and the ability to pinpoint the date and time a particular change was implemented • Workflows that benefit from an automatically maintained redundant fallback branch

Direct deployment policy

The direct deployment policy is the most basic of the four deployment policies offered in Continuous Delivery for PE. This policy is best to use if you wish to deploy a certain commit to a specified environment, then run Puppet on that environment (and only that environment) to deploy your changes.



When you kick off a deployment using the direct deployment policy, Continuous Delivery for PE performs the following steps.

Step 1: `puppet-code deploy`

Using Code Manager, Continuous Delivery for PE synchronizes the selected code change with your PE instance by running `puppet-code deploy` on Puppet Server.

Step 2: Temporary environment group

Continuous Delivery for PE creates a new environment group that's a child of the node group you selected when setting up your deployment. The child node group inherits all the rules, configuration settings, and variables of its parent node group. A Puppet Query Language query pins all the nodes associated with the parent node group to the child node group.

This environment node group is temporary; when the deployment is complete, it is automatically deleted. It's likely that you won't ever need to interact directly with this environment group.

Step 3: Running Puppet

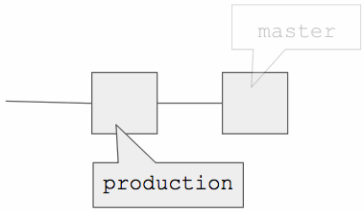
Continuous Delivery for PE uses the orchestrator to kick off a Puppet run on the nodes in the child node group. The orchestrated Puppet run updates nodes as quickly as the concurrency limits of Puppet Server allow. You can monitor the progress of this process on the deployment details page in the Continuous Delivery for PE web UI.

Step 4: Clean-up

When the Puppet run is complete, Continuous Delivery for PE deletes the child environment group and moves the HEAD of your target branch in your source control repository to the commit you chose to deploy.

Temporary branch policy

During a deployment using the temporary branch policy, Continuous Delivery for PE creates a temporary Git branch containing the code you wish to deploy, and a temporary environment group containing the nodes you're deploying code to. Your new code is then mapped to the temporary environment group in batches, allowing you to be certain that your new Puppet code works with each node before the changes are deployed to the full environment.



When you kick off a deployment using the temporary branch policy, Continuous Delivery for PE performs the following steps.

Step 1: New branch

Continuous Delivery for PE creates a new branch in your source control repository with the commit you've selected at its tip. This branch is temporary; when the deployment is complete, it will be automatically deleted. It's likely that you won't ever need to interact directly with this temporary branch.

Step 2: `puppet-code deploy`

Using Code Manager, Continuous Delivery for PE synchronizes the code changes with your PE instance by running `puppet-code deploy` on Puppet Server.

Step 3: Temporary environment group

Continuous Delivery for PE creates a new environment group that's a child of the node group you selected when setting up your deployment. The child node group inherits all the rules, configuration settings, and variables of its parent node group. A Puppet Query Language query pins all the nodes associated with the parent node group to the child node group.

This environment node group is temporary; when the deployment is complete, it is automatically deleted. It's likely that you won't ever need to interact directly with this environment group.

Step 4: Running Puppet

Continuous Delivery for PE uses the orchestrator to kick off a Puppet run on the nodes in the child 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 the progress of this process on the deployment details page in the Continuous Delivery for PE web UI.

Step 5: Clean-up

When all the Puppet runs have completed successfully (or in the event that the termination conditions you specified are met), Continuous Delivery for PE deletes the child environment 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 created at the beginning of this process.

Eventual consistency policy

The eventual consistency policy is designed for situations in which you wish to deploy new Puppet code, but prefer to wait for nodes to check in on their regular schedule to pick up the changes, rather than kicking off a dedicated Puppet run with the orchestrator. By using this policy, impacted nodes all eventually reach consistency with the new code.

When you kick off a deployment using the eventual consistency policy, Continuous Delivery for PE performs the following steps:

Step 1: `puppet-code deploy`

Using Code Manager, Continuous Delivery for PE synchronizes the selected code change with your PE instance by running `puppet-code deploy` on Puppet Server.

Step 2: Code changes are implemented during the next round of scheduled Puppet runs

The new code is delivered to the impacted nodes the next time they check in according to their established schedule.

Incremental branch policy

The incremental branch policy functions much like the temporary branch policy, with one exception: the Git branch created when the deployment is initiated isn't destroyed when the deployment is complete. Instead, this new branch remains in your source control system, providing a historic record of the changes made to your environments.

CAUTION: The incremental branch policy has been deprecated, and will be removed from Continuous Delivery for PE in a future version.

During a deployment using the incremental branch policy, Continuous Delivery for PE creates a new Git branch containing the code you wish to deploy, and a temporary environment group containing the nodes you're deploying code to. Your new code is then mapped to the temporary environment group in batches, allowing you to be certain that your new Puppet code works with each node before the changes are deployed to the full environment.

When you kick off a deployment using the incremental branch policy, Continuous Delivery for PE performs the following steps.

Step 1: New branch

Continuous Delivery for PE creates a new branch in your source control repository with the commit you've selected at its tip. The name of this new branch contains the environment you're deploying to and the deployment's ID number.

Step 2: `puppet-code deploy`

Using Code Manager, Continuous Delivery for PE synchronizes the code changes with your PE instance by running `puppet-code deploy` on Puppet Server.

Step 3: Temporary environment group

Continuous Delivery for PE creates a new environment group that's a child of the node group you selected when setting up your deployment. The child node group inherits all the rules, configuration settings, and variables of its parent node group. A Puppet Query Language query pins all the nodes associated with the parent node group to the child node group.

This environment node group is temporary; when the deployment is complete, it will be automatically deleted. It's likely that you won't ever need to interact directly with this environment group.

Step 4: Running Puppet

Continuous Delivery for PE uses the orchestrator to kick off a Puppet run on the nodes in the child 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 the progress of this process on the deployment details page in the Continuous Delivery for PE web UI.

Step 5: Clean-up

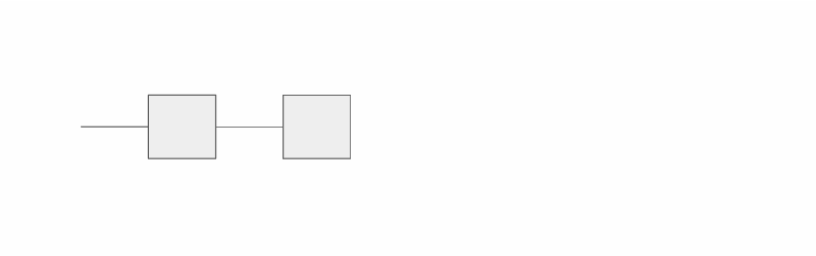
When all the Puppet runs have completed successfully (or in the event that the termination conditions you specified are met), Continuous Delivery for PE deletes the child environment group. Because you selected the incremental branch policy, the Git branch created at the beginning of this process is not deleted.

CAUTION: Using the incremental branch policy means that you will amass a collection of formerly used branches in your source control system. These branches are intended for use as a historic record of past changes to your environments. Garbage collection of old branches isn't performed by Continuous Delivery for PE; you must manually delete unneeded branches according to your company's policies about historic data retention.

Blue-green branch policy

A blue-green branch deployment policy uses redundancy to minimize risk when deploying code to production. Continuous Delivery for PE automatically alternates the use of two branches, deploying your changes by mapping the nodes in your target environment to the active branch while the passive branch remains stable.

CAUTION: The blue-green branch policy has been deprecated, and will be removed from Continuous Delivery for PE in a future version.



In contrast to the temporary branch policy, which deletes each deployment branch when the deployment is complete, or the incremental branch policy, which creates and preserves a new branch for each deployment, the blue-green branch policy creates and preserves two branches (`_blue` and `_green`), and reuses these two branches in alternating order.

When you kick off a deployment using the blue-green branch policy, Continuous Delivery for PE performs the following steps.

Step 1: Blue and green branches

During your first deployment for a given environment using the blue-green branch policy, Continuous Delivery for PE creates a new Git branch (the `_blue` branch) containing the code you wish to deploy, and follows the process outlined below. During your second deployment with this policy, a new `_green` branch is created and used. From this point on, the branches are reused in alternating order; the third deployment uses the `_blue` branch, the fourth uses the `_green` branch, and so on.

Step 2: `puppet-code deploy`

Using Code Manager, Continuous Delivery for PE synchronizes the code changes with your PE instance by running `puppet-code deploy` on Puppet Server.

Step 3: Temporary environment group

Continuous Delivery for PE creates a new environment group that's a child of the node group you selected when setting up your deployment. The child node group inherits all the rules, configuration settings, and variables of its parent node group. A Puppet Query Language query pins all the nodes associated with the parent node group to the child node group.

This environment node group is temporary; when the deployment is complete, it is automatically deleted. It's likely that you won't ever need to interact directly with this environment group.

Step 4: Running Puppet

Continuous Delivery for PE uses the orchestrator to kick off a Puppet run on the nodes in the child 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 the progress of this process on the deployment details page in the Continuous Delivery for PE web UI.

Step 5: Clean-up

When all the Puppet runs have completed successfully (or in the event that the termination conditions you specified are met), Continuous Delivery for PE deletes the child environment group. Because you selected the blue-green branch policy, the Git branch is not deleted, but is preserved for later reuse.

Deploy code manually

Use the manual deployment workflow to push a code change to a specified group of nodes on demand.

1. In the Continuous Delivery for PE web UI, click **Control Repos**, then select the control repo you wish to deploy from.
2. Click **New Deployment** .
3. Select the branch on which you made the change you're going to deploy (master), the commit you want to deploy, and your Puppet Enterprise instance.
4. Select the Puppet environment you wish to deploy the change to.
5. Select a deployment policy. For purposes of this getting started guide, select **Direct Deployment**.

   See Deployment policies to learn more about the deployment policies and how they work.
6. Set termination conditions for this deployment, and choose the number of nodes that can fail before the deployment is stopped.
7. Give the deployment a name, then click **Deploy**.

Monitor the progress of your deployment on the deployment details page that opens when you launch the deployment.

Deploy module code

You can deploy new module code to your Puppet environments via a Continuous Delivery for PE module pipeline. To do so, you must first add a `:branch => :control_branch` declaration to the module's entry in your control repo's Puppetfile.

Module code is deployed to your Puppet environments using the eventual consistency deployment policy. When you trigger a module 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 code to be deployed.

Continuous Delivery for PE then triggers Code Manager, which reads the `:branch => :control_branch` declaration referring to the module in the control repo's Puppetfile and adds the new module code to the control repo.

The new module code is now ready to be deployed to your chosen Puppet environments on the next scheduled Puppet run.

1. Open the Puppetfile that includes the module you wish to deploy. Add a `:branch => :control_branch` declaration to the module's section of the Puppetfile, as in the following example.

   ```
   mod 'apache',
     :git    => 'https://github.com/puppetlabs/puppetlabs-apache',
     :branch => :control_branch.
     :default_branch => 'master'
   ```

   To learn more about the `:branch => :control_branch` option, see Declare content from a relative control repo branch in the Code Manager documentation.
2. In the Continuous Delivery for PE web UI, click **Modules** and select the module you wish to deploy.
3. If you haven't already done so, create a pipeline for your module.
4. Click **+ Add Stage**. Select **Module Deployment**.
5. Select your PE instance and choose the Puppet environment the module code will be deployed to.
6. Click **Add Deployment**.

Your module pipeline is now set up to deploy new module code to your chosen Puppet environments any time the pipeline is triggered.

Require approval for deployments to protected Puppet environments

If your organization's business processes require manual review and approval before Puppet code is deployed to certain environments, set up an approval group of individuals with the authority to provide the needed review and sign-off. These approvers are contacted each time a deployment to a protected environment is proposed.

Before you begin

Make sure SMTP has been configured for your Continuous Delivery for PE installation. For instructions, see Configure SMTP.

Enabling a manual approval checkpoint on deployments to protected Puppet environments is a two-step process. First, designate the Continuous Delivery for PE users with the authority to approve or reject deployment requests. Next, designate the Puppet environments that require manual deployment approval.

Important: Designating approvers requires super user permissions.

1. Create an approval group. The members of this group review all proposed deployments to the environments you designate as protected and manually approve or decline each deployment.
   a) In the Continuous Delivery for PE web UI, click **Settings**.
   b) In the **Groups** tab, click **+ Create Group**.
   c) Enter a name (such as `Approval`) and description for your new group, then click **Create Group**.
   d) Click **Go to Groups** and click **View Group Details** for the group you just created.
   e) In each permissions category, click **+ Set Permissions**. Select the permissions you wish to assign to the approval group and click the blue **Set Permissions** button.

      Important: At a minimum, the approval group must have the **List** permission for **Control Repos** in order to view and approve or deny deployments.
   f) In **Group Members** column, add the individuals with the authority to approve or deny deployments to protected environments.

      - To add a new member to the group, click **+ Add Member**, and search for the user you wish to add by username or email address. When you locate the user you wish to add, click **Add User**.
      - To remove an existing member from the group, click **Remove Group Member** and confirm your action.

2. Designate which Puppet environments require deployment approval.
   a) Click the **Puppet Enterprise** tab.
   b) Click the number (likely "0") in the **Protected Environments** column for your PE instance.
   c) Select the Puppet environment that requires deployment approval.
   d) Select the approval group you created in step 1.
   e) Click **Add**.
   f) If necessary, repeat these steps to designate additional environments as protected, then click **Done**.

Now that this set-up process is complete, each time a deployment to the protected environment is triggered, either manually or through a pipeline run, the members of the approval group receive an email and a message in the message center alerting them that approval of the deployment is required.

A member of the approval group must review the deployment's details page and click **Provide Approval Decision**. After they approve or decline the deployment, a record of their decision is added to the deployment's details page.

Managing access

Use the user access and management tools in Continuous Delivery for Puppet Enterprise (PE) to ensure that each user has the access and permissions appropriate to their role on the team, from read-only users to super users.

- Create a user account on page 80
After Continuous Delivery for Puppet Enterprise (PE) has been installed by an administrator on your team, each person who will use the software must create an individual user account.
- Adding users and creating groups on page 81
Continuous Delivery for Puppet Enterprise (PE) is built to facilitate collaboration between the members of your team. You can share resources by adding users to your team's account, set up groups to facilitate collaboration, and manage the permissions granted to each member of your team.
- Permissions reference on page 82
The table below lists the group permissions available in Continuous Delivery for Puppet Enterprise (PE), along with a short explanation of each permission's scope.
- Using the root console on page 84
Super users and the root user can access the root console in Continuous Delivery for Puppet Enterprise (PE). Use the root console to manage users' account credentials, change super user permissions, configure single sign-on, access all workspaces associated with the installation, and update your installation's settings.

Create a user account

After Continuous Delivery for Puppet Enterprise (PE) has been installed by an administrator on your team, each person who will use the software must create an individual user account.

Before you begin

Obtain the Continuous Delivery for PE web UI endpoint from the administrator on your team 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 **Sign Up**.

Congratulations! You now have a Continuous Delivery for PE account.

View your account credentials by clicking your username in the navigation bar.

Adding users and creating groups

Continuous Delivery for Puppet Enterprise (PE) is built to facilitate collaboration between the members of your team. You can share resources by adding users to your team's account, set up groups to facilitate collaboration, and manage the permissions granted to each member of your team.

Tip: Set up an account for your team or organization, and add individual users to that account. This allows you to create groups based on user roles and manage each group's level of permissions in Continuous Delivery for PE.

Add a user

Adding users allows you to collaborate on projects and share account resources such as jobs and pipelines. You can add users in the **Account 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**.
2. In the **Users** tab, click **+ Add User(s)**. The **Add User** pane opens.
3. Enter the username or email address of the user you wish to add to your account. Partial matches are allowed.
4. When you locate the user you wish to add to your account, click **Add User** and confirm your action.
5. Repeat these steps to add additional users.

When you're finished adding users, close the **Add User** pane to return to your **Users** screen, where you'll see the full list of users with access to your Continuous Delivery for PE instance.

Create a new group

You can set up groups to organize users by permission level, job focus, geography, or other criteria.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. In the **Groups** tab, click **+ Create Group**.
3. Enter a name and description for your new group.

   Tip: Use a name for your group that describes the group's function or permission level, such as "Read-only users" or "Module developers."

   CAUTION: Once submitted, group names and descriptions cannot be edited. If you wish to change the name or description of your group, you must delete the group and recreate it.
4. Click **Create Group**.

Add or remove group members

Grant users membership in groups by adding them in the Continuous Delivery for PE web UI. This is also where you can remove users from groups.

Before you begin

Add the users you wish to include in your group, and create a new group.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. In the **Groups** tab, locate the group whose membership you wish to update, and click **View Group Details**.

3. In the **Group Details and Permissions** pane, locate the **Group Members** column.
   - To add a new member to the group, click **+ Add Member**, and search for the user you wish to add by username or email address. When you locate the user you wish to add, click **Add User**.
   - To remove an existing member from the group, click **Remove Group Member** and confirm your action.

Assign permissions to a group

Permissions are the tasks members of a group can perform in Continuous Delivery for PE, such as adding new control repos, editing jobs, and deploying code changes. Assign permissions to a group based on that group's function and needs.

Before you begin

Create a group and add group members.

Group permissions are additive. If a user is a member of multiple groups, that user is able to perform all of the actions described by all of the permissions in all their assigned groups.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. In the **Groups** tab, locate the group whose permissions you wish to set, and click **View Group Details**.
3. In each area, such as Modules or Integrations, click **+ Set Permissions**. Select the permissions you wish to assign to the group and click the blue **Set Permissions** button.

   For an explanation of the scope of each permission, see the Permissions reference.

   Note: The permission to change a user's email address and password is granted only to super users. For more on super users, see Using the root console.
4. When you've finished setting permissions, click **Close** and review the summary of the permissions now granted to your group.

Remove a user from a workspace

If you wish to remove a user from Continuous Delivery for PE workspace, you can do so in the **Account Settings** area of the Continuous Delivery for PE web UI.

Note: If you remove a user from a Continuous Delivery for PE workspace, that user is automatically removed from all groups in that workspace.

1. In the Continuous Delivery for PE web UI, click **Settings** then click **Users**.
2. In the list of users, locate the user you wish to remove from your workspace.
3. Click **Remove User** and confirm your action.
4. Repeat these steps to remove additional users from the workspace.

Permissions reference

The table below lists the group permissions available in Continuous Delivery for Puppet Enterprise (PE), along with 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.

Type	Permission	Definition
Control Repos	Edit	The ability to edit existing control repos. The ability to deploy code.
Control Repos	List	The ability to view all existing control repos.
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 all existing module repos.
Jobs	Create	The ability to create new jobs.
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.
Jobs	Run	The ability to run all existing jobs.
Users	Edit	The ability to add new users to Continuous Delivery for PE, and to remove existing users.
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 create new source control or Puppet Enterprise integrations.

Type	Permission	Definition
Integrations	Disconnect	The ability to remove existing source control or Puppet Enterprise integrations.

Using the root console

Super users and the root user can access the root console in Continuous Delivery for Puppet Enterprise (PE). Use the root console to manage users' account credentials, change super user permissions, configure single sign-on, access all workspaces associated with the installation, and update your installation's settings.

To access the root console:

- If you are the **root user**, sign into Continuous Delivery for PE with the root account credentials established during installation.
- If you are a **super user**, select **Root Console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar.

To exit the root console, click the name of a workspace.

Designate super users

Any Continuous Delivery for PE user except the root user can be designated as a super user. Super users have access to the root console, where they can reset other users' credentials and update settings. Super users can also manage global shared job hardware and designate which users are able to manually approve deployments to protected Puppet environments.

You must have root or super user permissions to access the root console and perform this action.

1. Log into the root console by selecting **Root Console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar or signing in as the root user.
2. Click **Accounts**.
3. Locate the user's name in the list and select the **Super User** toggle, then confirm your action.

   The selected user is now a super user.

Change a user's password

Users can reset their own passwords at any time by clicking **Forgot your password?** on the Continuous Delivery for 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.

You must have root or super user permissions to access the root console and perform this action.

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 in order to revoke their access to Continuous Delivery for PE, remember that the user can still access their account until their session expires or they log out.

1. Log into the root console by selecting **Root Console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar or signing in as the root user.
2. Click **Accounts**.
3. Locate the user whose password you wish to update by searching by username or email address. and click **User Settings** ⚙.
4. Click **Change Password**. Enter and confirm the new password, then 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 selecting **Root Console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar or signing in as the root user.
2. Click **Accounts**.
3. Locate the user whose password you wish to update by searching by username or email address. and click **User Settings** ☼.
4. Enter the new email address for the user and click **Change Email**.

Upgrading to 3.x

The Continuous Delivery for Puppet Enterprise (PE) 3.x series is now available. When you're ready to upgrade from the 2.x series to the 3.x series, follow the directions on this page appropriate to your installation method.

The Continuous Delivery for PE 3.x series includes numerous exciting new features, including:

- View Hiera changes in impact analysis reports
- Manage your pipelines as a YAML file
- Compose your own custom deployment policies

For a full list of new features, enhancements, and resolved issues, see the 3.x series release notes.

Upgrades from the 2.x series to the 3.x series are not automatic. Instead, you must upgrade your version of the software by following the appropriate instructions on this page.

Upgrade to 3.x by updating your cd4pe module classification

If you installed Continuous Delivery for PE using the `cd4pe` module or from the PE console, follow these instructions to upgrade to the 3.x series.

1. In the PE console, click **Classification**.
2. Expand the **PE Infrastructure** group, and select the **Continuous Delivery for PE** node group.
3. Click **Configuration**.
4. Under **Class: cd4pe**, in the **Parameter** field, select **cd4pe_version**. In the **Value** field, enter `3.x`. Click **Add parameter** and commit your changes.
5. To apply the new version, run Puppet on the node group. At the top of the page, click the **Run** selector and choose **Puppet**. On the **Run Puppet** page that opens, make any adjustments and click **Run Job**.

When the Puppet run is complete, your Continuous Delivery for PE instance will be running the 3.x series. You'll now automatically upgrade to new 3.x series versions as they are released. See the 3.x series release notes for the latest new features, fixes, and enhancements.

Upgrade to 3.x by updating your Docker image

If you installed Continuous Delivery for PE using Docker, follow these instructions to upgrade to the 3.x series.

1. Update your container runtime of choice to use the following image:

```
docker pull puppet/continuous-delivery-for-puppet-enterprise:3.x
```

2. Deploy the new Continuous Delivery for PE containers.

When the deployment is complete, your Continuous Delivery for PE instance will be running the 3.x series. You'll now automatically upgrade to new 3.x series versions as they are released. See the 3.x series release notes for the latest new features, fixes, and enhancements.