Continuous Delivery
## Contents

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

**Release notes**................................................................................................................. 6
  Continuous Delivery for PE release notes................................................................. 6
  Continuous Delivery for PE known issues............................................................... 20

Getting started with Continuous Delivery for PE......................................................... 21

**Puppet Application Manager**......................................................................................... 25
  Welcome to Puppet Application Manager (PAM)....................................................... 26
  Release notes.................................................................................................................. 26
    PAM release notes....................................................................................................... 26
    Known issues............................................................................................................... 30
  PAM UI.......................................................................................................................... 31
  Architecture overview.................................................................................................. 32
  PAM system requirements............................................................................................. 34
  Install PAM..................................................................................................................... 45
    Install Puppet applications using PAM on a customer-supported Kubernetes cluster............. 46
    PAM HA online installation....................................................................................... 48
    PAM HA offline installation...................................................................................... 52
    PAM standalone online installation......................................................................... 55
    PAM standalone offline installation........................................................................... 58
    Automate PAM and Puppet application online installations...................................... 60
    Automate PAM and Puppet application offline installations..................................... 62
  Uninstall PAM................................................................................................................ 64
  Working with Puppet applications................................................................................ 65
    Install applications via the PAM UI.......................................................................... 65
    Update a license for online installations................................................................. 66
    Update a license for offline installations................................................................. 66
    Upgrade an automated online application installation............................................ 67
    Upgrade an automated offline application installation.......................................... 67
  Maintenance and tuning............................................................................................... 68
  Upgrading PAM on a Puppet-supported cluster......................................................... 69
  Upgrading PAM on a customer-supported cluster...................................................... 73
  Backing up PAM using snapshots............................................................................... 74
  Troubleshooting PAM.................................................................................................... 79

Set up Continuous Delivery for PE...................................................................................... 83
  Continuous Delivery for PE architecture...................................................................... 84
  Configure and deploy Continuous Delivery for PE.................................................... 87
  Configure and deploy Continuous Delivery for PE in an offline environment............. 88
  Supported PE versions and browsers.......................................................................... 90
  Analytics data collection............................................................................................. 91
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migrating 3.x data to 4.x</td>
<td>92</td>
</tr>
<tr>
<td>Upgrading Continuous Delivery for PE</td>
<td>96</td>
</tr>
<tr>
<td>Configuring and adding integrations</td>
<td>99</td>
</tr>
<tr>
<td>Integrate with Puppet Enterprise</td>
<td>100</td>
</tr>
<tr>
<td>Configure impact analysis</td>
<td>104</td>
</tr>
<tr>
<td>Integrate with source control</td>
<td>107</td>
</tr>
<tr>
<td>Configure job hardware</td>
<td>112</td>
</tr>
<tr>
<td>Job hardware requirements</td>
<td>112</td>
</tr>
<tr>
<td>Configure job hardware</td>
<td>113</td>
</tr>
<tr>
<td>Add job hardware capabilities</td>
<td>114</td>
</tr>
<tr>
<td>Configure global shared job hardware</td>
<td>114</td>
</tr>
<tr>
<td>Add repositories</td>
<td>115</td>
</tr>
<tr>
<td>Configure SAML</td>
<td>116</td>
</tr>
<tr>
<td>Configure LDAP</td>
<td>117</td>
</tr>
<tr>
<td>Configure SMTP</td>
<td>119</td>
</tr>
<tr>
<td>Set up external PostgreSQL</td>
<td>120</td>
</tr>
<tr>
<td>Advanced Continuous Delivery for PE configuration</td>
<td>120</td>
</tr>
<tr>
<td>Managing teams and team members</td>
<td>123</td>
</tr>
<tr>
<td>Managing workspaces</td>
<td>123</td>
</tr>
<tr>
<td>Managing access</td>
<td>124</td>
</tr>
<tr>
<td>Create a user account</td>
<td>125</td>
</tr>
<tr>
<td>Adding users and creating groups</td>
<td>125</td>
</tr>
<tr>
<td>Permissions reference</td>
<td>127</td>
</tr>
<tr>
<td>Using the root console</td>
<td>129</td>
</tr>
<tr>
<td>Reviewing node inventory</td>
<td>130</td>
</tr>
<tr>
<td>Testing Puppet code with jobs</td>
<td>133</td>
</tr>
<tr>
<td>Key concepts</td>
<td>139</td>
</tr>
<tr>
<td>Working with Git branches in Continuous Delivery for PE</td>
<td>139</td>
</tr>
<tr>
<td>Understanding the Continuous Delivery for PE workflow</td>
<td>141</td>
</tr>
<tr>
<td>Using the feature branch workflow</td>
<td>143</td>
</tr>
<tr>
<td>Constructing pipelines</td>
<td>146</td>
</tr>
<tr>
<td>Constructing pipelines in the web UI</td>
<td>148</td>
</tr>
<tr>
<td>Constructing pipelines from code</td>
<td>151</td>
</tr>
<tr>
<td>Configuring your pipelines for management with code</td>
<td>151</td>
</tr>
<tr>
<td>Structuring a .cd4pe.yaml file</td>
<td>154</td>
</tr>
<tr>
<td>Analyzing the impact of code changes</td>
<td>165</td>
</tr>
<tr>
<td>Generating impact analysis reports</td>
<td>165</td>
</tr>
</tbody>
</table>
Limitations of impact analysis......................................................................................................................... 168

Deploying Puppet code................................................................................................................................. 168
   Built-in deployment policies.......................................................................................................................... 169
   Creating custom deployment policies.......................................................................................................... 173
   Deploy code manually.................................................................................................................................. 174
   Deploy module code...................................................................................................................................... 175
   Require approval for deployments to protected Puppet environments............................................................ 176

Support and troubleshooting......................................................................................................................... 176
   Troubleshooting Continuous Delivery for PE.............................................................................................. 177
   Getting support........................................................................................................................................... 181
Welcome to Continuous Delivery for Puppet Enterprise

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

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

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

<table>
<thead>
<tr>
<th>Learn to use Continuous Delivery for PE</th>
<th>Other useful links</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Before you install</strong></td>
<td>Docs for related Puppet products</td>
</tr>
<tr>
<td>Release notes</td>
<td>Puppet Enterprise</td>
</tr>
<tr>
<td>Known issues</td>
<td>Open source Puppet</td>
</tr>
<tr>
<td>System requirements</td>
<td>Puppet Development Kit</td>
</tr>
<tr>
<td><strong>Learn the basics</strong></td>
<td>Why and how people use Continuous Delivery for PE</td>
</tr>
<tr>
<td>Getting started</td>
<td>Critical Considerations for Continuous Delivery ebook</td>
</tr>
<tr>
<td>Key concepts on page 139</td>
<td>Get support</td>
</tr>
<tr>
<td><strong>Install and set up</strong></td>
<td>Support portal and knowledge base</td>
</tr>
<tr>
<td>Install Puppet Application Manager</td>
<td>Upgrade your Support plan</td>
</tr>
<tr>
<td>Set up Continuous Delivery for PE on page 83</td>
<td><strong>Share and contribute</strong></td>
</tr>
<tr>
<td>If you are upgrading to the 4.x series from a version in the 3.x series: Migrating 3.x data to 4.x on page 92</td>
<td>The Puppet community</td>
</tr>
<tr>
<td><strong>Configure and manage access</strong></td>
<td>Puppet Forge</td>
</tr>
<tr>
<td>Configuring job hardware, impact analysis, and source control and PE integrations.</td>
<td>Puppet's open source projects on GitHub</td>
</tr>
<tr>
<td>User permissions and groups</td>
<td></td>
</tr>
<tr>
<td>Creating and managing workspaces</td>
<td></td>
</tr>
<tr>
<td><strong>Continuously deliver Puppet code</strong></td>
<td></td>
</tr>
<tr>
<td>Preview potential impacts of new code</td>
<td></td>
</tr>
<tr>
<td>Create jobs to test code</td>
<td></td>
</tr>
<tr>
<td>Choose deployment policies</td>
<td></td>
</tr>
</tbody>
</table>

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

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

Security and vulnerability announcements are posted on Puppet's Security and vulnerability announcements page.

- Continuous Delivery for PE release notes on page 6
  These are the new features, enhancements, resolved issues, and deprecations for the Continuous Delivery for Puppet Enterprise (PE) 4.x release series.
- Continuous Delivery for PE known issues on page 20
  These are the known issues for the Continuous Delivery for Puppet Enterprise (PE) 4.x release series.

Continuous Delivery for PE release notes

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

Note: Migrating 3.x data to 4.x explains how to upgrade to the Continuous Delivery for PE 4.x series from a version in the 3.x series.

Version 4.13.0

Released 5 April 2022.

New in this release:

- **Improved proxy handling.** Continuous Delivery for PE now uses relative, rather than absolute, lookup paths to construct all URLs in the UI. Previously, configuring a proxy required changing the Continuous Delivery for PE service endpoint to avoid CORS errors. This improvement is a more complete resolution to the fix included in version 4.12.1.

- **Accessibility improvements.** Version 4.13.0 introduces several improvements for accessibility in Continuous Delivery for PE, including:
  - Removed leading + symbols from some buttons, which could cause incongruity between the accessible name and visible label.
  - On the PE settings page, the icons to remove a protected environment and change a token's lifetime now have tooltips correctly describing their functions.
  - The exit icon now has a tooltip.
  - On the SSH key settings page, the **More actions** icon now has a tooltip explaining the contents of the menu associated with this button.

Resolved in this release:

- **Fixed navigation for super users without workspace membership.** A super user who is not a member of any workspace is directed to the root console after logging in. When a super user accesses a workspace they are not a member of, the navigation menu loads correctly.

- **Multi-node clusters don't prevent draining nodes with StatefulSets.** We modified PodDistruptionBudget to allow multi-node Continuous Delivery for PE clusters to drain all but the last node.
• **Security fixes.** Version 4.13.0 includes several bug fixes related to security, including:
  - The kurl-proxy and minIO containers have the latest OS patches.
  - Fixes that address CVE-2021-43858 and CVE-2022-0839.
  - Upgraded OpenTelemetry to upgrade a Commons-io dependency.
  - The build process removes test keys such as those left by the `public-encrypt` package and Bolt installation.

**Version 4.12.1**

Released 7 March 2022.

Resolved in this release:

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

**Version 4.12.0**

Released 2 March 2022.

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

New in this release:

• **Secrets management in Continuous Delivery for PE.** You can add secrets to Continuous Delivery for PE jobs, which jobs use while running. To learn more, go to Testing Puppet code with jobs on page 133.
• **Continuous Delivery for PE now supports Kubernetes 1.19 to 1.24.** Kubernetes 1.17 and 1.18 are no longer supported.
• **Usability improvements.** Version 4.12.0 introduces several improvements to the design and usability of Continuous Delivery for PE, including:
  - The Nodes page has improved button placement and text.
  - The Nodes page has a new view selector that allows you to pick which view you want to see.
  - Custom view names now become the page title when selected.
  - You can edit an existing view or save a new view by clicking Save or Edit in the drop down.

Resolved in this release:

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

**Version 4.11.5**

Released 22 February 2022.

Resolved in this release:

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

**Version 4.11.4**

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

Released 14 February 2022.
Resolved in this release:

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

**Version 4.11.2**

Released 2 February 2022.

Resolved in this release:

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

**Version 4.11.1**

Released 20 January 2022.

Resolved in this release:

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

**Version 4.11.0**

Released 20 January 2022.

New in this release:

- **Compound filters on the Nodes page.** You can now build multi-element filters that use logical operators (and and or) to answer complex queries about your nodes. For more information about creating and using compound filters, go to Create filters to focus on specific node sets.
- **Endpoint configuration clarified.** When you Configure and deploy Continuous Delivery for PE on page 87, you can use a NodePort or an Ingress for your webhook and local container registry endpoints. Previously, the Ingress option was only available if you had previously set it. Now, you can always choose from both options.
- **Retrieve impact analysis CSV files through an API call.** You can now reach the getImpactAnalysisCsvV1 endpoint from the Continuous Delivery for PE deployments module. This means your custom deployment policies can use this endpoint to retrieve impact analysis CSV exports.
- **Custom deployment policy logging.** You can add custom deployment events with message parameters to your custom deployment policies. These appear as arbitrary log messages and, ultimately, on the web UI. This facilitates debugging when creating custom deployment policies.
- **Usability improvements.** Version 4.11.0 introduces several improvements to the design and usability of Continuous Delivery for PE, including:
  - The ability to search for nodes in the table by name on the Nodes page.
  - To improve reliability of snapshot restores, PostgreSQL now initially listens on localhost during startup.

Resolved in this release:

- **Improved impact analysis report filtering, searching, and pagination.** When an impact analysis report has multiple pages, searching and filtering refreshes pagination.
- **Export impact analysis report functionality restored.** From the web UI, you can export CSV files of your impact analysis reports again.
- **Improved error handling when changing user email addresses.** Changing a user's email address to the user's existing email address no longer triggers an error. Attempting to change a user's email address to an email address belonging to another user returns an error message explaining that another user is using this email address.
- **SMTP "from" address defaults to root account's email.** If the Send emails from this address field is empty, Continuous Delivery for PE now uses the email address associated with the root user.
• **Impact analysis succeeds on GitHub repos with different repo and display names.** Impact analysis tasks are now performed correctly on GitHub repositories where the repo name does not match the display name set in Continuous Delivery for PE.

• **The Continuous Delivery for PE container restarts successfully.** The database migration lock is now automatically removed when the container stops, allowing for a successful restart without manually removing the lock.

**Version 4.10.5**  
Released 20 December 2021.  
Resolved in this release:

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

**Version 4.10.4**  
Released 17 December 2021.  
Resolved in this release:

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

**Version 4.10.3**  
Released 10 December 2021.  
Resolved in this release:

• This release contains fixes that address CVE-2021-44228.

**Version 4.10.2**  
Released 9 December 2021.  
Resolved in this release:

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

**Version 4.10.1**  
Released 11 November 2021.  
Resolved in this release:

• Requests to the /v1/authtokens endpoint are now processed correctly.

**Version 4.10.0**  
Released 9 November 2021.

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

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

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

New in this release:

- **Environments path setting for impact analysis.** To improve impact analysis performance for users with certain PE configurations, a new Environments path setting is now available in the Impact analysis credentials section of each Puppet Enterprise instance's credentials. Users who have configured PE to use lockless deploys MUST NOT set the environments path. For more information, go to Optional impact analysis settings.

Resolved in this release:

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

**Version 4.9.0**

Released 8 September 2021.

New in this release:

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

Resolved in this release:

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

**Version 4.8.2**

Released 31 August 2021.

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

Resolved in this release:

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

**Version 4.8.1**

Released 24 August 2021.

Resolved in this release:
• An endpoint that was accidentally removed in version 4.8.0 is now restored.

**Version 4.8.0**

Released 10 August 2021.

New in this release:

• **Configure snapshot timeouts.** You can now configure the length of time that Puppet Application Manager spends attempting to back up Continuous Delivery for PE components when creating a snapshot. For more information, go to Adjusting the timeout period for snapshots.

Resolved in this release:

• When impact analysis tasks are run on a compiler, the resulting report now shows the list of impacted nodes.
• The installation preflight check now correctly requires 50 GB of storage for Ceph.
• You can now successfully restore Continuous Delivery for PE from a snapshot on legacy installations of Puppet Application Manager.

**Version 4.7.2**

Released 26 July 2021.

Resolved in this release:

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

**Version 4.7.1**

Released 12 July 2021.

Resolved in this release:

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

**Version 4.7.0**

Released 8 July 2021.

New in this release:

• **Fact charts.** You can now see visual representations of Facter fact values on all nodes across the infrastructure you've integrated with Continuous Delivery for PE. The new Fact charts section of each view on the Nodes page displays the distribution of unique values across your inventory for your selected facts. We've included four fact charts to get you started, and you can build custom fact charts for the facts that are relevant to your business goals.

• **Usability improvements.** Version 4.7.0 introduces several improvements to the design and usability of Continuous Delivery for PE, including:
  • Several web UI pages have been updated with a cleaner, more streamlined design.
  • The Users page now shows the complete list (up to 1,000 users) of users in a workspace.

Resolved in this release:

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

Version 4.6.1
Released 16 June 2021.
Resolved in this release:
• Login attempts after upgrading to Continuous Delivery for PE 4.6.0 or higher with an older license no longer fail.
• Setting up an external PostgreSQL database no longer requires a separate configuration for the estate reporting service. The estate reporting service now defaults to sharing the Continuous Delivery for PE database. For more information, see Set up external PostgreSQL.

Version 4.6.0
 Released 3 June 2021.
New in this release:
• **Promote permission.** Previously, the permission to manually promote changes through pipeline stages was included in the Edit permission for control repos and modules. The Promote permission is now separate from the Edit permission, and you can grant or deny these permissions to groups as needed.

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

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

Resolved in this release:
• Custom deployment policies no longer require environment branches.
• If a workspace has no owner, the **Workspaces** page in the root console now loads correctly so that you can reassign the workspace to a new owner.
• Information about global shared hardware is now correctly displayed when you navigate to the **Hardware** page in the root console from the individual workspace's **Hardware** page.
Version 4.5.2
Released 11 May 2021.
Resolved in this release:
• If SAML is enabled for your Continuous Delivery for PE installation, a Log in using single sign-on option is now shown on the login screen, and the pod no longer falls into a restart loop.
• Docker runtime arguments are no longer passed if a job previously run on workspace hardware is updated to run on global shared hardware.

Version 4.5.1
Released 27 April 2021.
Resolved in this release:
• The links provided in deployment approval emails now resolve correctly.

Version 4.5.0
Released 22 April 2021.

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

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

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

Version 4.4.2
Released 13 April 2021.

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

Resolved in this release:

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

Version 4.4.1

Released 29 March 2021.

Resolved in this release:

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

Version 4.4.0

Released 11 March 2021.

New in this release:

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

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

- **Built-in user groups for new workspaces.** Newly created workspaces now include three built-in user groups: Administrators, Operators, and Viewers. See the **Permissions reference** for details on the permissions included in each built-in user group.

- **Streamlined workflow for adding users to a workspace.** As part of the process of adding a new user to a workspace, you are now prompted to assign the user to one or more user groups (either the new built-in user groups or those you've created).

- **Configure login attempt limits.** You can now configure the number of unsuccessful login attempts that a user can make on Continuous Delivery for PE before their account is locked, as well as the length of time the account is locked and the length of time before the login attempt counter resets. For more information, see **Configuring login attempt limits.**

- **OpenTelemetry.** You now have the option to use OpenTelemetry to perform distributed tracing on your Continuous Delivery for PE installation. OpenTelemetry configuration options are available on the **Config** page in the platform admin console.

  **Important:** When using OpenTelemetry, you can choose to export the gathered data to your logs, to Jaeger over gRPC, or via OTLP. Be aware that if you choose the logging exporter option, the size of your Continuous Delivery for PE logs increases significantly. OpenTelemetry data you collect is not shared with Puppet, except in one specific case: if, while using the logging exporter, you generate and send a support bundle to Puppet, the support bundle contains OpenTelemetry data for your installation.
• **Preflight check improvements.** Preflight checks now verify that schedulable CPU and memory capacity are available for performing upgrades, and that the system is running Kubernetes version 1.17.0 or newer.

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

Resolved in this release:

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

**Version 4.3.3**

Released 23 February 2021.

Resolved in this release:

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

**Version 4.3.2**

Released 3 February 2021.

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

New in this release:

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

Resolved in this release:

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

Removed in this release:
• **Support for Puppet Enterprise version 2018.1.** PE 2018.1 reached the end of its support lifecycle on 31 January 2021.

**Version 4.3.1**

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

Released 26 January 2021.

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

New in this release:

• **Support for Red Hat Enterprise Linux (RHEL) 8 and CentOS 8.** You can now run Continuous Delivery for PE on RHEL version 8 and CentOS version 8.

• **Ceph replaces MinIO for object storage.** Continuous Delivery for PE 4.x now uses Ceph for object storage instead of MinIO. New 4.3.1 installations use Ceph from the outset. For existing 4.x users, MinIO information is migrated to Ceph for you as part of the upgrade to version 4.3.1. To support this change, Ceph replication status is now collected as part of the support bundle.

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

• **Default job timeout period increased.** The default job timeout period is now 30 minutes. This change reduces the chance that complex jobs time out before completion. See Adjusting the timeout period for jobs to learn more.

• **Usability improvements.** Version 4.3.1 introduces several improvements to the design and usability of the web UI, including:
  - The delete module icon is now correctly labeled.
  - Control repo icons are displayed when selecting a custom deployment policy for a deployment.

Resolved in this release:

• When logging in, users are now correctly directed to the last workspace they visited.

• Long branch names no longer overlap event status indicators in the **Events** area.

• Users are now less likely to encounter Docker Hub rate limits.

• The object storage migration process is now more robust and issues found in version 4.3.0 have been resolved.

• If an impact analysis task is canceled in a pipeline stage with the "any completed" auto-promotion criteria set, the pipeline run now stops at the canceled stage and does not continue.

Security notice:

• **CVE-2020-7946.** Source control tokens were displayed in plain text when trace-level logging was enabled. This issue has been resolved.

• **CVE-2020-27218.** An Eclipse Jetty vulnerability has been resolved.

• **CVE-2020-29361, CVE-2020-29362, and CVE-2020-29363.** The version of PostgreSQL included in Continuous Delivery for PE has been upgraded to resolve CVE-2020-29361, CVE-2020-29362, and CVE-2020-29363.

**Version 4.2.4**

Released 17 December 2020.
Resolved in this release:

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

**Version 4.2.3**

Released 17 November 2020.

Resolved in this release:

- Webhooks now correctly trigger pipelines for GitLab repositories with names that include spaces or other unusual characters.
- The platform admin console now rate limits authentication attempts to prevent brute force attacks.

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

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

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

**Version 4.2.2**

Released 12 November 2020.

Resolved in this release:

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

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

**Version 4.2.1**

Released 5 November 2020.

Resolved in this release:

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

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

**Version 4.2.0**

Released 3 November 2020.

New in this release:

- **Available memory setting.** A new setting on the Config page in the platform admin console lets you tune the total memory available to the Continuous Delivery for PE application. For more on the Memory available for CD4PE setting, see Adjusting available memory.

- **Removal of harmful terminology.** Documentation for this release replaces the term “PE master” with “PE primary server,” and the term “master branch” with “main branch”. When adding a new control repo or module, Continuous Delivery for PE now looks for a “main” branch instead of a “master” branch. These changes are part of a company-wide effort to remove harmful terminology from our products.
Resolved in this release:

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

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

Security notice:

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

**Version 4.1.3**

Released 15 October 2020.

Resolved in this release:

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

**Version 4.1.2**

Released 8 October 2020.

**Note:** To upgrade to version 4.1.2 from version 4.0.1 or 4.0.0, you must first upgrade Continuous Delivery for PE and then upgrade the platform admin console. Offline users, please see Upgrade in an offline environment.

Resolved in this release:

- If your load balancer requires HTTP health checks, you can now opt into using Ingress settings that do not require Server Name Indication (SNI) for `/status`. Enable this setting in the Customize endpoints section of the Config tab in the platform admin console.
- Preflight checks for offline installations no longer hang with an ImagePullBackoff error on initial setup.
- Long-running deployments and jobs no longer fail with a 504 upstream request timeout error.
Version 4.1.1

Released 29 September 2020.

**Note:** To upgrade to version 4.1.1, you must first upgrade Continuous Delivery for PE and then upgrade the platform admin console. Offline users, please see Upgrade in an offline environment.

New in this release:

- **Filter the Nodes page.** You can now apply custom filter combinations to your nodes table and zero in on the node data that's most relevant to your work. Available filters include fact value, most recent node change status, operating system, PE server, node group, and no-op status.

- **Snapshots.** Snapshots are point-in-time backups of your Continuous Delivery for PE deployment, which can be used to roll back to a previous state. You can create snapshots manually or set up a schedule to capture them automatically. To get started, see Configure rollback snapshots.

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

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

Resolved in this release:

- Snapshots now successfully save to Amazon S3. In order to save your snapshots to an Amazon S3 bucket, you must upgrade the platform admin console to the latest version after upgrading to Continuous Delivery for PE version 4.1.1. See Upgrade the platform admin console for instructions.

- When exporting node table data, occasional failed queries to PuppetDB are now retried automatically, and no longer result in a failed export.

Version 4.0.1

Released 14 September 2020.

Resolved in this release:

- The export functionality on the 4.x Nodes page now works correctly.

- The container no longer hangs indefinitely in some circumstances after the host is rebooted.

- Network security rules now restrict inter-service communications.

- Local registry credentials are now stored as secrets.

- Certificate validation preflight checks now correctly refer to the local registry during offline installations.

Version 4.0.0

Released 25 August 2020.

New in this release:

- **New installer and administration platform.** The new Continuous Delivery for PE 4.x platform introduces a streamlined experience for installation, upgrades, license management, troubleshooting, and more. Use the new platform admin console to configure, monitor, upgrade to new versions in the 4.x series, back up, restore, and deploy your Continuous Delivery for PE installation.

- **Migrate your 3.x data to a 4.x installation.** To upgrade to the Continuous Delivery for PE 4.x series from a version in the 3.x series, see Migrating 3.x data to 4.x.

- **Update webhooks.** The new Webhooks tool in the root console updates your source control webhooks to point to the current installation. Use this tool as part of the 3.x to 4.x migration process, or any time you change the location of your Continuous Delivery for PE installation.

Removed in this release:
• **Continuous Delivery agent on job hardware.** Support for the Continuous Delivery agent was deprecated in version 3.4.0. Puppet agent-based job hardware is still supported.

• **Support for external Amazon DynamoDB and MySQL databases.** Support for external Amazon DynamoDB and MySQL databases was deprecated in version 3.1.0.

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

---

## Continuous Delivery for PE known issues

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

### Upgrading to 4.12.0 can delete the cd4pe Ingress

In busier Kubernetes clusters, upgrading to Continuous Delivery for PE version 4.1.2.0 can delete the cd4pe Ingress. If the application is unavailable after upgrading and `kubectl get ingress cd4pe` returns an empty list, redeploy the deployed version to recreate the Ingress.

### Impact analysis can require additional disk space

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

### Impact analysis tasks fail when using the satellite_pe_tools module

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

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

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

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

Puppet Application Manager versions 1.19 and 1.20 display an `Analyzer Failed: invalid analyzer` error during the preflight checks when deploying a new version of Continuous Delivery for PE. This error relates to analyzers supported by version 1.24 and newer versions of the Puppet Application Manager. The error can be safely ignored. To resolve the failure and take advantage of the new preflight checks, upgrade Puppet Application Manager to the latest version.

### Deployments might time out on node groups with complex rules

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

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

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

Jobs fail when using chained SSL certificates on Windows

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

Purging unmanaged firewall rules with the puppetlabs-firewall module deletes required firewall settings

If your Continuous Delivery for PE node uses the puppetlabs-firewall module to manage its firewall settings, and if a resources { 'firewall': purge => true } metaparameter is set on the node or at a higher level, Puppet removes the unmanaged Docker firewall rules Continuous Delivery for PE requires to run successfully. To work around this issue, disable unmanaged firewall rule purging for your Continuous Delivery for PE node by changing the metaparameter to resources { 'firewall': purge => false }.

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

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

Getting started with Continuous Delivery for PE

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

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

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

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

1. Make sure your system meets the PAM system requirements on page 34.
2. Follow the instructions for your environment to Install PAM on page 45.
3. Depending on your environment, either Configure and deploy Continuous Delivery for PE on page 87, or Configure and deploy Continuous Delivery for PE in an offline environment on page 88.

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

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

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

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

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

### Part 3: Set up integrations

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

1. Follow the instructions to **Integrate with Puppet Enterprise** on page 100.
2. Follow the instructions to **Configure impact analysis** on page 104.
3. Follow the integration instructions for your source control system:
   - Azure DevOps Services
   - Bitbucket Cloud
   - Bitbucket Server
   - GitHub
   - GitHub Enterprise
   - GitLab

### Part 4: Configure job hardware

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

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

      A sample Puppetfile entry:

      ```
      mod 'puppetlabs-cd4pe_jobs', '1.6.0'
      ```
   b) Run `puppet code deploy production --wait` to deploy the updated code to the production environment.

4. The job hardware server must be able to use the **pre-built jobs** included in Continuous Delivery for PE. Since these jobs are Docker-based, you must install and configure Docker on the node. You can find installation files and instructions on the [Docker website](https://www.docker.com).

© 2021 Puppet, Inc.
5. Tell Continuous Delivery for PE that this node is ready to be used as job hardware for Docker-based jobs by assigning it to a hardware capability. Capabilities organize your job hardware servers and ensure that jobs run on hardware with the right characteristics. Continuous Delivery for PE automatically creates a **Docker** hardware capability for you.
   a) In the Continuous Delivery for PE web UI, click **Hardware**.
   b) Locate the **Docker** capability and click **Edit**.
   c) Select the PE instance that manages the node you've selected as job hardware, and then select your job hardware node. The selected node appears in the **Hardware with this capability** list.
   d) Click **Save**.

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

### Part 5: Add a control repo

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

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

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

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

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

### Part 6: Set up a pipeline

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

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

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

- The **Code validation stage** includes two tasks (jobs to test Puppet code).
- The **Impact analysis stage** includes an impact analysis task with a pull request gate (more on this later).
- The **Deployment stage**, which is where you'll add deployment instructions in **Part 8**.
**Part 7: Set up an environment node group**

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

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

1. In your Git repository, create a new branch called **cd4pe_testing**. This represents the environment node group for the test.
2. Run `puppet code deploy cd4pe_testing` on your PE primary server.
3. In the PE console, click **Node groups** (or **Classification** in PE versions prior to 2019.8.1).
4. Click **Add group...** and create a new node group with these 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
5. Click **Add**.
6. In the node groups list, click **CD4PE test group**. In the **Rules** tab, pin two or three test nodes to the node group. Make sure these nodes are not assigned to any other node groups in your PE installation.

**Part 8: Deploy changes to your nodes**

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

1. In the Continuous Delivery for PE web UI, in your control repo's pipeline, click **Add a deployment**.
2. Select your PE instance and then select the **CD4PE test group** node group you created in Part 7.
3. Select 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. To trigger deployment, you need to create a change to pass through to production. On the main branch of your Git repository, make a code change, such as updating a package version. Commit the change and then return to the Continuous Delivery for PE web UI to watch your pipeline in action.

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

**Part 9: Create an impact analysis report**

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

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

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

3. In the **New impact analysis** window, select your feature branch, and select your new commit. Select your PE instance and the **CD4PE test group** node group.

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

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

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

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

---

**Puppet Application Manager**

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

**Note:** Puppet Application Manager was previously called the platform admin console in the Continuous Delivery for PE documentation.

**What does Puppet Application Manager do?**

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

In the Puppet Application Manager UI, you can configure Continuous Delivery for PE, monitor the cluster's activity, upgrade to the latest version of the software, and back up your installation.

**How do I use Puppet Application Manager to deploy Continuous Delivery for PE?**

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

- **Welcome to Puppet Application Manager (PAM)** on page 26
  Puppet Application Manager is an administrative console where you can install, access, and manage your Puppet applications. It is also where you can go to access upgrades to new Puppet applications releases.
- **PAM UI** on page 31
  The Puppet Application Manager (PAM) UI provides administration functionality where you can access and manage your Puppet applications.
- **Architecture overview** on page 32
  Puppet Application Manager runs on Kubernetes. Puppet provides several supported configurations for different use cases.
- **PAM system requirements** on page 34
  You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.
- **Install PAM** on page 45
  You can install Puppet-supported Puppet Application Manager on a single node or in an HA configuration. Both online and offline install packages are available. You can also install it on an existing Kubernetes cluster.
- **Working with Puppet applications** on page 65
  You can install and upgrade Puppet using the Puppet Application Manager UI.

© 2021 Puppet, Inc.
• **Maintenance and tuning** on page 68
  Follow these guidelines when you’re tuning or performing maintenance on a node running Puppet Application Manager (PAM).

• **Upgrading PAM on a Puppet-supported cluster** on page 69
  Upgrade Puppet Application Manager (PAM) on a Puppet-supported cluster to take advantage of new features and bug fixes, and to upgrade your cluster to the latest version of Kubernetes when one is available.

• **Upgrading PAM on a customer-supported cluster** on page 73
  Upgrade Puppet Application Manager (PAM) on your own Kubernetes cluster to take advantage of new features and bug fixes.

• **Backing up PAM using snapshots** on page 74
  Snapshots are point-in-time backups of your Puppet Application Manager (PAM) deployment, which can be used to roll back to a previous state or restore your installation into a new cluster for disaster recovery.

• **Troubleshooting PAM** on page 79
  Use this guide to troubleshoot issues with your Puppet Application Manager installation.

---

**Welcome to Puppet Application Manager (PAM)**

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

Useful links:

<table>
<thead>
<tr>
<th>Puppet Application Manager docs links</th>
<th>Other useful places</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Before you install</strong></td>
<td>Docs for related Puppet products</td>
</tr>
<tr>
<td>Release notes</td>
<td>Continuous Delivery for PE</td>
</tr>
<tr>
<td>System requirements</td>
<td>Comply</td>
</tr>
<tr>
<td><strong>Install Puppet Application Manager</strong></td>
<td>Get support</td>
</tr>
<tr>
<td>PAM standalone online installation on page 55</td>
<td>Support</td>
</tr>
<tr>
<td>PAM standalone offline installation on page 58</td>
<td>Upgrade your support plan</td>
</tr>
<tr>
<td>PAM HA online installation on page 48</td>
<td><strong>Share and contribute</strong></td>
</tr>
<tr>
<td>PAM HA offline installation on page 52</td>
<td>Engage with the Puppet community</td>
</tr>
<tr>
<td><strong>Upgrading, disaster recovery, and troubleshooting</strong></td>
<td>Puppet Forge</td>
</tr>
<tr>
<td>Upgrading PAM on a Puppet-supported cluster on page 69</td>
<td>Open source projects from Puppet on GitHub</td>
</tr>
<tr>
<td>Backing up PAM using snapshots on page 74</td>
<td><strong>External resources</strong></td>
</tr>
<tr>
<td>Troubleshooting PAM on page 79</td>
<td>Getting started with Kubernetes Off-The-Shelf software (KOTS)</td>
</tr>
</tbody>
</table>

---

**Release notes**

**PAM release notes**

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

Follow the instructions in **Upgrading Puppet Application Manager** to get the current version.

**12 April 2022 (Puppet Application Manager 1.68.0)**

New in this release:
• **Install a specific version of an application.** When installing a Puppet application using the automated installation method, you now have the option to specify the application's version by passing the `--app-version-label=<version>` flag to the `kubectl kots install` command. For more information, go to Automate PAM and Puppet application online installations on page 60.

• **Status reporting improvements.** The status reporting tools can now detect when an application is being upgraded.

• **Component upgrades to address CVEs.** To address various CVEs in Envoy, this version includes an upgrade of Project Contour to version 1.20.1.

• **Other component upgrades.** This version includes an upgrade of KOTS to version 1.68.0.

Resolved in this release:

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

1 March 2022 (Puppet Application Manager 1.64.0)

Resolved in this release:

• Diffs are now shown correctly in the PAM UI.

• The OpenSSL package is no longer a prerequisite for successful installation on newer Red Hat Enterprise Linux 7 systems.

• You can now successfully install Puppet Application Manager on Red Hat Enterprise Linux 8 systems without the need to force-install the kurl-local audit-libs library.

17 February 2022 (Puppet Application Manager 1.62.0)

New in this release:

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

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

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

  **Tip:** See How to determine your version of Puppet Application Manager if you're not sure which installation type you're running.

• **Prometheus enabled on standalone architecture.** Beginning with version 1.62.0 Prometheus is enabled by default on all new and existing standalone Puppet Application Manager installations. Prometheus requires an additional 350m CPU and 500MiB of memory, so ensure your system is properly sized before upgrading. Prometheus is an optional component; if you need to disable it to conserve resources, see Optional components on page 82.

• **Automatic certificate rotation.** By default, the self-signed certificates used by Project Contour and Envoy expire after one year. This version includes an update that auto-rotates those certificates before they expire.

• **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of containerd to version 1.4.12.

• **Other component upgrades.** This version includes an upgrade of KOTS to version 1.62.0.

Deprecated in this release:

• **Legacy architecture.** The legacy architecture, which was the version of Puppet Application Manager available for installation prior to May 2021, is now deprecated. (See How to determine your version of Puppet Application Manager...
Manager if you need to confirm whether you're running the legacy architecture.) The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. Puppet will continue to update legacy architecture components other than Kubernetes until 30 June 2022. If security advisories against Kubernetes 1.19 arise, the remediation path is to migrate to one of the newer architectures by following the instructions in Disaster recovery or migration using a snapshot on page 77.

Important: Before beginning the migration process from a legacy deployment you must upgrade to PAM version 1.62.0 with the force-reapply-addons flag included in the upgrade command. Find upgrade instructions at PAM legacy upgrades on page 71 and PAM offline legacy upgrades on page 72.

30 November 2021 (Puppet Application Manager 1.56.0)
This release includes an upgrade of KOTS to version 1.56.0, which adds the following improvements:

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

6 October 2021 (Puppet Application Manager 1.52.1)
New in this release:

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

Resolved in this release:

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

25 August 2021 (Puppet Application Manager 1.49.0)
New in this release:

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

30 June 2021 (Puppet Application Manager 1.44.1)
New in this release:

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

Resolved in this release:

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

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

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

26 May 2021

New in this release:

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

Known issues in this release:

• Running the KOTS installer with the `airgap` and `kurl-registry-ip` flags results in an error.

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

10 May 2021 (Puppet Application Manager 1.40.0)

New in this release:

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

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

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

For more information on legacy upgrades, see PAM legacy upgrades on page 71.

15 April 2021 (Puppet Application Manager 1.38.0)

New in this release:

• **Snapshots.** Puppet Application Manager now supports full (instance-level) snapshots, which can be used for application rollbacks and disaster recovery. For more information, see Backing up Puppet Application Manager using snapshots.

• **Component upgrades.** This version includes an upgrade of KOTS to version 1.38.0.

17 February 2021 (Puppet Application Manager 1.29.3)

New in this release:

• **Support for Ubuntu 20.04.** You can now run Puppet Application Manager on Ubuntu 20.04.

• **Component upgrades.** This version includes an upgrade of Prometheus to version 2.22.1 and Prometheus Operator to version 0.44.1, an upgrade of KOTS to version 1.29.3, an upgrade of Project Contour to version 1.12.0, and an upgrade of ekco to version 0.10.0.
New in this release:

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

Resolved in this release:

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

**7 December 2020**

New in this release:

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

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

**Related information**

- **Upgrading PAM on a Puppet-supported cluster** on page 69
  Upgrade Puppet Application Manager (PAM) on a Puppet-supported cluster to take advantage of new features and bug fixes, and to upgrade your cluster to the latest version of Kubernetes when one is available.

- **Backing up PAM using snapshots** on page 74
  Snapshots are point-in-time backups of your Puppet Application Manager (PAM) deployment, which can be used to roll back to a previous state or restore your installation into a new cluster for disaster recovery.

**Known issues**

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

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

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

**Installations fail on RHEL 8 systems**

Due to a conflict with the `audit-libs` library, installations of Puppet Application Manager version 1.62.0 or older fail on Red Hat Enterprise Linux 8 systems. To work around this issue, force installation of the kurl-local audit-libs library version by running `sudo yum install audit-libs-3.0-0.17.20191104git1c2f876.el8.1 --allowerasing`.

**OpenSSL package required for newer RHEL 7 systems**

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

**Package updates with yum or DNF fail after upgrading PAM**

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

```
    yum module reset kurl.local
```
PAM UI

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

PAM console menu

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

- **Use the Dashboard tab to:**
  - Manage your applications
  - See version history
  - Set application configuration settings
  - Access support bundles for troubleshooting
  - Manage licenses
  - View files
  - Configure registry settings

- **Use the Cluster Management tab to view current information on the nodes in your cluster. You can also use this tab to drain, and add nodes to your cluster.**

- **Use the Snapshots tab to create point-in-time backups of your deployment, which can be used to roll back to a previous state, or restore your installation into a new cluster for disaster recovery. For more information, see Backing up PAM using snapshots.**

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

Application monitoring graphs

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

- **Node CPU Usage (%)** shows when hosts are getting overwhelmed (high % usage).
- **Node Memory Usage (%)** shows when hosts are reaching full memory capacity that may result in processes being killed due to out-of-memory errors.
- **Node Available Storage (%)** shows when hosts are running out of storage. At 15%, pods may start to be evicted or reads/writes on databases are paused until more storage is made available.
- **Volume Available Storage (%)** shows when application persistent volumes are getting full (low %) that may lead to problems with a particular application. Note that -

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

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

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

Related information

**Backing up PAM using snapshots** on page 74
Snapshots are point-in-time backups of your Puppet Application Manager (PAM) deployment, which can be used to roll back to a previous state or restore your installation into a new cluster for disaster recovery.

### Architecture overview

Puppet Application Manager runs on Kubernetes. Puppet provides several supported configurations for different use cases.

Puppet Application Manager can run on Puppet-supported or customer-supported Kubernetes clusters. For more information on installing on a customer-supported Kubernetes cluster, see [Install Puppet applications using Puppet Application Manager and an existing Kubernetes cluster](#).

**Note:** This architecture overview deals with Puppet-supported deployments only.

#### Terminology

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

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

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

#### Standalone architecture

Standalone is optimized for limited resources, storing data directly on disk. If you need to remove optional components like Prometheus and Grafana to decrease resource utilization, see [Optional components](#) on page 82. While additional compute capacity can be added through secondary nodes, this does not provide increased resilience as data is only stored on the node where a component service runs.

For information on migrating data from standalone to HA deployments, see [Disaster recovery or migration using a snapshot](#) on page 77.

#### HA architecture

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

The HA architecture installs Prometheus and Alertmanager. These are used to provide system monitoring in the Puppet Application Manager UI. Prometheus and Alertmanager are unauthenticated on ports 30900 and 30903, and you are recommended to control access to these ports via firewall rules. For information on how to remove Prometheus and Alertmanager, see [Optional components](#) on page 82.

#### Puppet Application Manager architectures

The following outline some of the core components involved in standalone and HA architectures, and how they communicate.

**Note:** A complete list of the ports used by Puppet Application Manager is available in the #unique_5/unique_5_Connect_42_pam-port-reqs section of the system requirements documentation.
Internal ports

All backplane ports may also be used for inter-process communication within a single host.

The following ports are only used within a single host for inter-process communication:
- TCP 6781, 6782, 6784 (Weave)
For information on setting up health checks for your load balancer, see Load balancer health checks on page 69

**DEPRECATED: Legacy architecture**

**Note:** The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. Puppet will continue to update legacy architecture components other than Kubernetes until 30 June 2022. If security advisories against Kubernetes 1.19 arise, the remediation path is to migrate to one of the newer architectures by following the instructions in Disaster recovery or migration using a snapshot on page 77.

The Puppet Application Manager legacy architecture reflects an older configuration that used Ceph 1.0 which hosted data directly on the file system. Puppet no longer recommend this for new installs, but maintains it to support existing installs and ensure other components can be kept up-to-date.

The legacy architecture installs Prometheus and Alertmanager. These are used to provide system monitoring in the Puppet Application Manager UI. Prometheus and Alertmanager are unauthenticated on ports 30900 and 30903, and you are recommended to control access to these ports via firewall rules. For information on how to remove Prometheus and Alertmanager, see Optional components on page 82.

For information on upgrading to a newer version of the legacy architecture, see PAM legacy upgrades on page 71 and PAM offline legacy upgrades on page 72.

For information on migrating data from a legacy architecture to a standalone or HA architecture, see Disaster recovery or migration using a snapshot on page 77.

**Related information**

Reduce recovery time when a node fails on page 81

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

Install Puppet applications using PAM on a customer-supported Kubernetes cluster on page 46

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

PAM legacy upgrades on page 71

Use this method to upgrade Puppet Application Manager (PAM) from a version installed before May 2021.

PAM offline legacy upgrades on page 72

Use this method to upgrade Puppet Application Manager (PAM) on offline nodes from a version installed before April 2021.

Troubleshooting PAM on page 79

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

**PAM system requirements**

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

**Customer-supported cluster hardware requirements**

The following Kubernetes distributions are supported:

- Google Kubernetes Engine
- AWS Elastic Kubernetes Service
- Red Hat OpenShift

If you use a different distribution, contact Puppet Support for more information on compatibility with PAM.

Application requirements:
<table>
<thead>
<tr>
<th>Application</th>
<th>CPU</th>
<th>Memory</th>
<th>Storage</th>
<th>Ports</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Delivery for Puppet Enterprise (PE)</td>
<td>3 CPU</td>
<td>8 GB</td>
<td>280 GB</td>
<td>Ingress, NodePort 8000</td>
<td>Note: NodePort is configurable</td>
</tr>
<tr>
<td>Puppet Comply</td>
<td>6 CPU</td>
<td>6 GB</td>
<td>35 GB</td>
<td>Ingress, NodePort 30303</td>
<td>Note: NodePort is configurable</td>
</tr>
</tbody>
</table>

Make sure that your Kubernetes cluster meets the minimum requirements:

- Kubernetes version 1.17-1.21.
- A default storage class that can be used for relocatable storage.
- A standard Ingress controller that supports websockets (we have tested with Project Contour and NGINX).
- We currently test and support Google Kubernetes Engine (GKE) clusters.

**Cluster ports**

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

**Puppet-supported HA cluster hardware requirements**

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

**Continuous Delivery for Puppet Enterprise (PE)**

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

<table>
<thead>
<tr>
<th>CPU</th>
<th>Memory</th>
<th>Storage</th>
<th>Open ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 CPU</td>
<td>10 GB</td>
<td>100 GB on an unformatted storage device&lt;br&gt; An additional 100 GB for /var/lib. This is primarily divided among:&lt;br&gt;  - 2 GB for /var/lib/etcd&lt;br&gt;  - 10 GB for /var/lib/rook (plus buffer)&lt;br&gt;  - 32 GB for /var/lib/kubelet&lt;br&gt;  - 40 GB for /var/lib/containerd</td>
<td>TCP: 80, 443, 2379, 2380, 6443, 6783, 8000, 8800, and 10250&lt;br&gt; UDP: 6783, 6784</td>
</tr>
</tbody>
</table>

Note: The storage backend prefers the file system inhabited by /var/lib/rook to remain below 70% utilization.

SSDs (or similarly low-latency storage) are recommended for /var/lib/etcd and /var/lib/rook.
**Puppet Comply**

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

<table>
<thead>
<tr>
<th>CPU</th>
<th>Memory</th>
<th>Storage</th>
<th>Open ports</th>
</tr>
</thead>
</table>
| 6 CPU | 9 GB   | 100 GB on an unformatted storage device An additional 100 GB for /var/lib. This is primarily divided among:  
• 2 GB for /var/lib/etcd  
• 10 GB for /var/lib/rook (plus buffer)  
• 32 GB for /var/lib/kubelet  
• 40 GB for /var/lib/containerd | TCP: 80, 443, 2379, 2380, 6443, 6783, 8800, 10250, and 30303  
UDP: 6783, 6784 |

**Note:** The storage backend prefers the file system inhabited by /var/lib/rook to remain below 70% utilization.  
SSDs (or similarly low-latency storage) are recommended for /var/lib/etcd and /var/lib/rook.

---

**Continuous Delivery for Puppet Enterprise (PE) and Puppet Comply**

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

<table>
<thead>
<tr>
<th>CPU</th>
<th>Memory</th>
<th>Storage</th>
<th>Open ports</th>
</tr>
</thead>
</table>
| 7 CPU | 12 GB  | 150 GB on an unformatted storage device An additional 100 GB for /var/lib. This is primarily divided among:  
• 2 GB for /var/lib/etcd  
• 10 GB for /var/lib/rook (plus buffer)  
• 32 GB for /var/lib/kubelet  
• 40 GB for /var/lib/containerd | TCP: 80, 443, 2379, 2380, 6443, 6783, 8000, 8800, 10250, and 30303  
UDP: 6783, 6784 |

**Note:** The storage backend prefers the file system inhabited by /var/lib/rook to remain below 70% utilization.  
SSDs (or similarly low-latency storage) are recommended for /var/lib/etcd and /var/lib/rook.
For a detailed example of an HA configuration running Continuous Delivery for PE and Puppet Comply, see Example of an HA cluster that supports CDPE and Comply.

**Networking requirements**

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

**Cluster port requirements**

Puppet Application Manager (PAM) uses the following ports:

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Purpose</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Puppet application ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>443</td>
<td>TCP</td>
<td>Web UI (Relies on Server Name Indication to route requests to the application)</td>
<td>Browser</td>
</tr>
<tr>
<td><strong>Continuous Delivery for Puppet Enterprise (PE)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8000</td>
<td>TCP</td>
<td>Webhook service</td>
<td>Source control</td>
</tr>
<tr>
<td><strong>Puppet Comply</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30303</td>
<td>TCP</td>
<td>Communication with Puppet Enterprise (PE)</td>
<td>PE instance</td>
</tr>
<tr>
<td><strong>Platform ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2379</td>
<td>TCP</td>
<td>High availability (HA) communication</td>
<td>etcd on the Kubernetes host</td>
</tr>
<tr>
<td>2380</td>
<td>TCP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6443</td>
<td>TCP</td>
<td>Kubernetes API</td>
<td>Admin workstation</td>
</tr>
<tr>
<td>6783</td>
<td>TCP/UDP</td>
<td>Kubernetes networking - Weave</td>
<td>Kubernetes host</td>
</tr>
<tr>
<td>6784</td>
<td>UDP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8800</td>
<td>TCP</td>
<td>Puppet Application Manager</td>
<td>Admin browser</td>
</tr>
<tr>
<td>10250</td>
<td>TCP</td>
<td>Kubernetes cluster management</td>
<td>Kubernetes host</td>
</tr>
</tbody>
</table>

**IP address range requirements**

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

Ensure that IP address ranges 10.96.0.0/22 and 10.32.0.0/22 are locally accessible. See Resolve IP address range conflicts for instructions.

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

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

**Antivirus and antimalware considerations**

Antivirus and antimalware software can impact PAM and its applications or prevent them from functioning properly. To avoid issues, exclude the following directories from antivirus and antimalware tools that scan disk write operations:
• /var/lib/rook
• /var/lib/kubelet
• /var/lib/containerd

**Firewall modules**

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

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

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

For more information, see the PAM firewall module.

**Supported operating systems**

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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Linux</td>
<td>2</td>
</tr>
<tr>
<td>CentOS</td>
<td>7.4, 7.5, 7.6, 7.7, 7.8, 7.9</td>
</tr>
<tr>
<td></td>
<td>8.1, 8.2, 8.3, 8.4</td>
</tr>
<tr>
<td>Oracle Linux</td>
<td>7.4, 7.5, 7.6, 7.7, 7.8, 7.9</td>
</tr>
<tr>
<td></td>
<td>8.0, 8.1, 8.2, 8.3, 8.4, 8.5</td>
</tr>
<tr>
<td>Red Hat Enterprise Linux (RHEL)</td>
<td>7.4, 7.5, 7.6, 7.7, 7.8, 7.9</td>
</tr>
<tr>
<td></td>
<td>8.1, 8.2, 8.3, 8.4, 8.5</td>
</tr>
<tr>
<td>Ubuntu (General availability kernels)</td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td>18.04</td>
</tr>
<tr>
<td></td>
<td>20.04</td>
</tr>
</tbody>
</table>

**Puppet-supported standalone hardware requirements**

Here are the suggested configurations for standalone installations.
### Continuous Delivery for Puppet Enterprise (PE)

<table>
<thead>
<tr>
<th>CPU</th>
<th>Memory</th>
<th>Storage</th>
<th>Open ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 CPU</td>
<td>8 GB</td>
<td>130 GB for /var/lib and /var/openebsThis is primarily divided among:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2 GB for /var/lib/etcld</td>
<td>TCP: 80, 443, 2379, 2380,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 32 GB for /var/lib/kubelet</td>
<td>6443, 6783,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 40 GB for /var/lib/containerd</td>
<td>8000, 8800, and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 50 GB for /var/openebs</td>
<td>10250</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TCP: 80, 443, 2379, 2380, 6443, 6783, 8000, 8800, and 10250</td>
<td>UDP: 6783, 6784</td>
</tr>
</tbody>
</table>

### Puppet Comply

<table>
<thead>
<tr>
<th>CPU</th>
<th>Memory</th>
<th>Storage</th>
<th>Open ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 CPU</td>
<td>6 GB</td>
<td>130 GB for /var/lib and /var/openebsThis is primarily divided among:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2 GB for /var/lib/etcld</td>
<td>TCP: 80, 443, 2379, 2380,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 32 GB for /var/lib/kubelet</td>
<td>6443, 6783,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 40 GB for /var/lib/containerd</td>
<td>8000, 8800, and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 50 GB for /var/openebs</td>
<td>10250</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TCP: 80, 443, 2379, 2380, 6443, 6783, 8000, 8800, and 10250</td>
<td>UDP: 6783, 6784</td>
</tr>
</tbody>
</table>

### Cluster port requirements

Puppet Application Manager uses the following ports:

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Purpose</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Puppet application ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>443</td>
<td>TCP</td>
<td>Web UI (Relies on Server Name Indication to route requests to the application)</td>
<td>Browser</td>
</tr>
<tr>
<td><strong>Continuous Delivery for Puppet Enterprise (PE)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8000</td>
<td>TCP</td>
<td>Webhook service</td>
<td>Source control</td>
</tr>
<tr>
<td><strong>Puppet Comply</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30303</td>
<td>TCP</td>
<td>Communication with Puppet Enterprise</td>
<td>PE instance</td>
</tr>
<tr>
<td><strong>Platform ports</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6443</td>
<td>TCP</td>
<td>Kubernetes API</td>
<td>Admin workstation</td>
</tr>
<tr>
<td>6783</td>
<td>TCP/ UDP</td>
<td>Kubernetes networking - Weave</td>
<td>Kubernetes host</td>
</tr>
<tr>
<td>6784</td>
<td>UDP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8800</td>
<td>TCP</td>
<td>Puppet Application Manager</td>
<td>Admin browser</td>
</tr>
</tbody>
</table>

© 2021 Puppet, Inc.
### IP address range requirements

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

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

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

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

### Antivirus and antimalware considerations

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

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

- /var/openebs
- /var/lib/kubelet
- /var/lib/containerd

### Firewall modules

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

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

```
systemctl restart kubelet
[kubectl -n kube-system delete pod -l k8s-app=kube-proxy
ekubectl -n kube-system delete pod -l name=weave-net]
```

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

### Detailed hardware requirements

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

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

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

#### Puppet Application Manager

<table>
<thead>
<tr>
<th>Node type</th>
<th>CPU</th>
<th>Memory</th>
<th>Storage</th>
</tr>
</thead>
</table>
| Primary   | 4 CPU | 7 GB | At least 50 GB on an unformatted storage device in addition to application-specific storage (below) for the Ceph storage backend. This can be satisfied by multiple devices if more storage is needed later, but should be balanced across primaries. An additional 100 GB for /var/lib. This is primarily divided among:
- 2 GB for /var/lib/etcd
- 10 GB for /var/lib/rook (plus buffer)
- 32 GB for /var/lib/kubelet
- 40 GB for /var/lib/containerd

**Note:** Ceph storage backend prefers the file system inhabited by /var/lib/rook to remain below 70% utilization. SSDs (or similarly low-latency storage) are recommended for /var/lib/etcd and /var/lib/rook. |
| Secondary  | 1 CPU | 1.5 GB | 80 GB for /var/lib. This is primarily divided among:
- 32 GB for /var/lib/kubelet
- 40 GB for /var/lib/containerd |

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

Application-specific requirements:

<table>
<thead>
<tr>
<th>Application</th>
<th>CPU</th>
<th>Memory</th>
<th>Storage</th>
<th>Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Delivery for Puppet Enterprise (PE)</td>
<td>3 CPU</td>
<td>8 GB</td>
<td>50 GB</td>
<td>80, 443, 8000</td>
</tr>
<tr>
<td>Puppet Comply</td>
<td>6 CPU</td>
<td>6 GB</td>
<td>50 GB</td>
<td>80, 443, 30303</td>
</tr>
</tbody>
</table>

The minimum recommended size for a secondary node is 4 CPU and 8 GB of memory to allow some scheduling flexibility for individual services.
Example of an HA cluster capable of running Continuous Delivery for PE and Comply

An HA cluster capable of running both Continuous Delivery for Puppet Enterprise (PE) and Puppet Comply requires 9 CPU and 14 GB of application-specific memory in addition to per-node baselines. You can create a cluster from 4 CPU, 8 GB nodes. Each primary uses all CPU and 7 GB of memory for cluster services, providing 0 CPU and 1 GB of memory for application workloads; each secondary uses 1 CPU and 1.5 GB of memory for cluster services, providing 3 CPU and 6.5 GB of memory for application workloads. Create the cluster as follows:

- Three primaries provide an excess of 3 GB of memory for application workloads. Each primary must have 150 GB of storage in an unformatted, unpartitioned storage device for Ceph and 100 GB of storage for /var/lib.
- Three secondaries provide an excess of 9 CPU and 19.5 GB of memory for application workloads. Each secondary must have 80 GB of storage for /var/lib.
This diagram illustrates the suggested three-node configuration for a cluster capable of running Continuous Delivery for Puppet Enterprise (PE) and Puppet.
# Web URL and port requirements for firewalls

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

<table>
<thead>
<tr>
<th>Category</th>
<th>URLs</th>
</tr>
</thead>
</table>
| Puppet Application Manager and platform | • get.replicated.com  
• registry.replicated.com  
• proxy.replicated.com  
• api.replicated.com  
• k8s.kurl.sh  
• kurl-sh.s3.amazonaws.com  
• replicated.app  
• registry-data.replicated.com |
| Container registries                   | • gcr.io  
• docker.io  
• index.docker.io  
• registry-1.docker.io  
• auth.docker.io  
• production.cloudflare.docker.com  
• quay.io |
| Puppet Enterprise                     | • pup.pt  
• forgeapi.puppet.com  
• pm.puppetlabs.com  
• amazonaws.com  
• s3.amazonaws.com  
• rubygems.org |

## Supported browsers

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

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

Refer to the Architecture overview for guidance on choosing which Puppet-supported Kubernetes cluster configuration is most appropriate for your needs.

Important: The Puppet-supported Puppet Application Manager cluster brings its own container runtime as part of the kURL installation.

- Do not install a container runtime from your operating system (OS) vendor or third-party.
- Do not install PAM on a node that has previously hosted a container runtime from your OS vendor or third-party.

Installing a different container runtime on a node, even if you installed and removed the packages before you installed PAM, causes failures that persist even after you've uninstalled the runtime.

For information on installing Puppet Application Manager on an existing Kubernetes cluster, see Install Puppet applications using PAM on a customer-supported Kubernetes cluster on page 46.

- Install Puppet applications using PAM on a customer-supported Kubernetes cluster on page 46

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

- PAM HA online installation on page 48
The Puppet Application Manager (PAM) installation process creates a Kubernetes cluster for you and walks you through installing your Puppet application on the cluster.

- PAM HA offline installation on page 52
Use these instructions to install Puppet Application Manager (PAM) in an air-gapped or offline environment where the Puppet Application Manager host server does not have direct access to the internet.

- PAM standalone online installation on page 55
The Puppet Application Manager (PAM) installation process sets up the application manager (with a simple Kubernetes installation for container orchestration) for you and installs the application on the single-node cluster.

- PAM standalone offline installation on page 58
Use these instructions to install Puppet Application Manager (PAM) in an offline environment where the Puppet Application Manager host server does not have direct access to the internet.

- Automate PAM and Puppet application online installations on page 60
During a fresh online installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

- Automate PAM and Puppet application offline installations on page 62
During a fresh offline installation of Puppet Application Manager (PAM) and a Puppet application, you have the option to configure the software automatically rather than completing the installation script interview.

- Uninstall PAM on page 64
Different uninstall procedures are required for Puppet-supported and customer-supported clusters

Related information
Architecture overview on page 32
Puppet Application Manager runs on Kubernetes. Puppet provides several supported configurations for different use cases.
Install Puppet applications using PAM on a customer-supported Kubernetes cluster

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

Before you begin

1. If you haven’t already done so, install kubectl.
2. Puppet Application Manager is expected to work on any certified Kubernetes distribution that meets the following requirements. We validated and support:
   - Google Kubernetes Engine
   - AWS Elastic Kubernetes Service
   - Red Hat OpenShift
   
   If you use a different distribution, contact Puppet Support for more information on compatibility with PAM.
3. Make sure your Kubernetes cluster meets the minimum requirements:
   - Kubernetes version 1.17-1.21.
   - A default storage class that can be used for relocatable storage.
   - A standard Ingress controller that supports websockets (we have tested with Project Contour and NGINX).
   - We currently test and support Google Kubernetes Engine (GKE) clusters.

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

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

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

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

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

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

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

   ```bash
curl https://kots.io/install | bash
   ```
2. If you are performing an offline install, ensure the required images are available in a local registry.
   a) Download the release assets matching the CLI version using the following command:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

   a. On the Puppet Application Manager Dashboard page, under **Config > Optional configuration**, select **View options for using a proxy or external load balancer**.
   b. Enter a new value for **Webhook service port**.
5. To configure your installation further, click **Config**. On this tab, you can configure a public hostname, root user, and other settings. These are written as Kubernetes secrets in the deployment manifests. For information on how to configure your application, see the documentation for that application:

- Configure Continuous Delivery for PE in an online environment
- Configure Comply in an online environment

6. To use cert-manager, in the **Customize endpoints** section, select **I have cert manager** and in the annotations section, add yours. For example:

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

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

Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For more information, see Install applications via the PAM UI on page 65.

For more information on installing Continuous Delivery for PE online, see Install Continuous Delivery for PE.

For more information on installing Comply online, see Install Comply online.

**Related information**

- Upgrade PAM on a customer-supported online cluster on page 73
- Upgrade PAM on a customer-supported offline cluster on page 74

**PAM HA online installation**

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

**Before you begin**

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

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

   If you want to keep Firewalld enabled:

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

```
apiVersion: cluster.kurl.sh/v1beta1
type: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", ","-c"]
```
4. Ensure that IP address ranges 10.96.0.0/22 and 10.32.0.0/22 are locally accessible. See Resolve IP address range conflicts for instructions.

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

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

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

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

6. Set all nodes used in your HA implementation to the UTC timezone.

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

This installation process results in a Puppet Application Manager instance that is configured for high availability. Installation takes several (mostly hands-off) minutes to complete.
1. Install and configure a load balancer (or two if you want to segment internal and external traffic - for more information, see Architecture overview). Round-robin load balancing is sufficient. For an HA cluster, the following is required:

- A network (L4, TCP) load balancer for port 6443 across primary nodes. This is required for Kubernetes components to continue operating in the event that a node fails. The port is only accessed by the Kubernetes nodes and any admins using `kubectl`.
- A network (L4, TCP) or application (L7, HTTP/S) load balancer for ports 80, and 443 across all primaries and secondaries. This maintains access to applications in event of a node failure. Include 8800 if you want external access to the Puppet Application Manager UI.

**Note:** Include port 8000 for webhook callbacks if you are installing Continuous Delivery for PE.
2. From the command line of your first primary node, run the installation script:

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

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

**Note:** An unformatted, unpartitioned storage device is required.

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

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

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

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

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

**Note:** If you lose this password or wish to change it, see [Reset the Puppet Application Manager password](#) for instructions.

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

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

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

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

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

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

To add MASTER nodes to this installation, run the following script on your other nodes:

```
curl -sSL https://k8s.kurl.sh/puppet-application-manager-unstable/join.sh | sudo bash -s kubernetes-master-address=...
```

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

4. Add the two new nodes to your load balancer.
5. Navigate to the Puppet Application Manager UI using the address provided by the installation script (http://<PUPPET APPLICATION MANAGER ADDRESS>:8800) and follow the prompts.

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

Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For more information, see Install applications via the PAM UI on page 65.

For more information on installing Continuous Delivery for PE online, see Install Continuous Delivery for PE.

For more information on installing Comply online, see Install Comply online.

**Related information**

- Reset the PAM password on page 80
- As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

- PAM system requirements on page 34
- You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

- Resolve IP address range conflicts on page 79
- When installing Puppet Application Manager, IP address ranges 10.96.0.0/22 and 10.32.0.0/22 must not be used by other nodes on the local network.

- Architecture overview on page 32
- Puppet Application Manager runs on Kubernetes. Puppet provides several supported configurations for different use cases.

- Using sudo behind a proxy server on page 83
- Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

**PAM HA offline installation**

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

**Before you begin**

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

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

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

```yaml
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
```
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
           sysctl -p"]
  firewalldCmds:
    - ["--permanent", "--zone=trusted", "--add-interface=weave"]
    - ["--zone=external", "--add-masquerade"]
    # SSH port
    - ["--permanent", "--zone=public", "--add-port=22/tcp"]
    # HTTPS port
    - ["--permanent", "--zone=public", "--add-port=443/tcp"]
    # Kubernetes etcd port
    - ["--permanent", "--zone=public", "--add-port=2379-2830/tcp"]
    # Kubernetes API port
    - ["--permanent", "--zone=public", "--add-port=6443/tcp"]
    # Weave Net port
    - ["--permanent", "--zone=public", "--add-port=6783/udp"]
    # Weave Net port
    - ["--permanent", "--zone=public", "--add-port=6783-6874/tcp"]
    # CD4PE Webhook callback port ( uncomment line below if needed)
    # - ["--permanent", "--zone=public", "--add-port=8000/tcp"]
    # KOTS UI port
    - ["--permanent", "--zone=public", "--add-port=8800/tcp"]
    # CD4PE Local registry port ( offline only, uncomment line below if
    # needed)
    # - ["--permanent", "--zone=public", "--add-port=9001/tcp"]
    # Kubernetes component ports ( kubelet, kube-scheduler, kube-
    # controller)
    - ["--permanent", "--zone=public", "--add-port=10250-10252/tcp"]
    # Reload firewall rules
    - ["--reload"]
    bypassFirewalldWarning: true
    disableFirewalld: false
    hardFailOnFirewalld: false
    preserveConfig: false

4. Ensure that IP address ranges 10.96.0.0/22 and 10.32.0.0/22 are locally accessible. See Resolve IP address range conflicts for instructions.

   **Note:** The minimum size for CIDR blocks used by Puppet Application Manager are:
   - **Standalone** -/24 for pod and service CIDRs
   - **HA** -/23 for pod and service CIDRs
   - Default of /22 is recommended to support future expansion

5. Ensure that the nodes can resolve their own hostnames, through either local host mapping or a reachable DNS server.

6. Set all nodes used in your HA implementation to the UTC timezone.

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

8. If you're restoring a backup from a previous cluster, make sure you include the kurl-registry-ip=<YOUR_IP_ADDRESS> installation option. For more information, see Disaster recovery or migration using a snapshot on page 77.

This installation process results in a basic Puppet Application Manager instance that is configured for optional high availability. Installation takes several (mostly hands-off) minutes to complete.
1. Install and configure a load balancer (or two if you want to segment internal and external traffic - for more information, see Architecture overview). Round-robin load balancing is sufficient. For an HA cluster, the following is required:
   - A network (L4, TCP) load balancer for port 6443 across primary nodes. This is required for Kubernetes components to continue operating in the event that a node fails. The port is only accessed by the Kubernetes nodes and any admins using kubectl.
   - A network (L4, TCP) or application (L7, HTTP/S) load balancer for ports 80, and 443 across all primaries and secondaries. This maintains access to applications in event of a node failure. Include 8800 if you want external access to the Puppet Application Manager UI.

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

2. From a workstation with internet access, download the cluster installation bundle (note that this bundle is ~4GB):
   ```
   https://k8s.kurl.sh/bundle/puppet-application-manager.tar.gz
   ```

3. Copy the installation bundle to your primary and secondary nodes and unpack it:
   ```
   tar xzf puppet-application-manager.tar.gz
   ```

4. Run the installation command:
   ```
   cat install.sh | sudo bash -s airgap
   ```

   **Note:** An unformatted, unpartitioned storage device is required.

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

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

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

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

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

   **Note:** If you lose this password or wish to change it, see Reset the Puppet Application Manager password for instructions.

© 2021 Puppet, Inc.
5. Follow instructions outlined after the following line in the install script:

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

6. Add the two new nodes to your load balancer.

7. Navigate to the Puppet Application Manager UI using the address provided by the installation script (http://<PUPPET APPLICATION MANAGER ADDRESS>:8800) and follow the prompts.

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

Follow the instructions for installing your Puppet applications on Puppet Application Manager. For more information, see Install applications via the PAM UI on page 65.

For more information on installing Continuous Delivery for PE offline, see Install Continuous Delivery for PE in an offline environment.

For more information on installing Comply offline, see Install Comply offline.

**Related information**

- **Reset the PAM password** on page 80
  As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

- **PAM system requirements** on page 34
  You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

- **Resolve IP address range conflicts** on page 79
  When installing Puppet Application Manager, IP address ranges 10.96.0.0/22 and 10.32.0.0/22 must not be used by other nodes on the local network.

- **Architecture overview** on page 32
  Puppet Application Manager runs on Kubernetes. Puppet provides several supported configurations for different use cases.

- **Using sudo behind a proxy server** on page 83
  Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

**PAM standalone online installation**

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

**Before you begin**

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

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

If you want to keep Firewalld enabled:

a. Make sure Firewalld is installed on your system.

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

```yaml
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
           sysctl -p"]
  firewalldCmds:
    - ["--permanent", "--zone=trusted", "--add-interface=weave"]
    - ["--zone=external", "--add-masquerade"]
    # SSH port
    - ["--permanent", "--zone=public", "--add-port=22/tcp"]
    # HTTPS port
    - ["--permanent", "--zone=public", "--add-port=443/tcp"]
    # Kubernetes etcd port
    - ["--permanent", "--zone=public", "--add-port=2379-2830/tcp"]
    # Kubernetes API port
    - ["--permanent", "--zone=public", "--add-port=6443/tcp"]
    # Weave Net port
    - ["--permanent", "--zone=public", "--add-port=6783/udp"]
    # Weave Net port
    - ["--permanent", "--zone=public", "--add-port=7867/udp"]
    # CD4PE Webhook callback port ( uncomment line below if needed)
    - ["--permanent", "--zone=public", "--add-port=8000/tcp"]
    # KOTS UI port
    - ["--permanent", "--zone=public", "--add-port=8800/tcp"]
    # CD4PE Local registry port ( offline only, uncomment line below if needed)
    - ["--permanent", "--zone=public", "--add-port=9001/tcp"]
    # Kubernetes component ports ( kubelet, kube-scheduler, kube-controller)
    - ["--permanent", "--zone=public", "--add-port=10250-10252/tcp"]
    # Reload firewall rules
    - ["--reload"]
  bypassFirewalldWarning: true
  disableFirewalld: false
  hardFailOnFirewalld: false
  preserveConfig: false
```

4. Ensure that IP address ranges 10.96.0.0/22 and 10.32.0.0/22 are locally accessible. See Resolve IP address range conflicts for instructions.

5. If you use the `puppetlabs/firewall` module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the `puppetlabs/pam_firewall` module before installing Puppet Application Manager.
This installation process results in a basic Puppet Application Manager instance. Installation takes several (mostly hands-off) minutes to complete.

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

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

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

a) When the installation script prints the Puppet Application Manager address and password, make a careful note of these credentials:

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

**Note:** If you lose this password or wish to change it, see Reset the Puppet Application Manager password for instructions.

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

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

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

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

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

2. Navigate to the Puppet Application Manager UI using the address provided by the installation script (`http://<PUPPET APPLICATION MANAGER ADDRESS>:8800`) and follow the prompts.

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

Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For more information, see Install applications via the PAM UI on page 65.

For more information on installing Continuous Delivery for PE online, see Install Continuous Delivery for PE.

For more information on installing Comply online, see Install Comply online.

**Related information**

- [Reset the PAM password](#) on page 80
- [PAM system requirements](#) on page 34
- [Resolve IP address range conflicts](#) on page 79

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

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

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

© 2021 Puppet, Inc.
Puppet Application Manager runs on Kubernetes. Puppet provides several supported configurations for different use cases.

Using sudo behind a proxy server on page 83
Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`.

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

Before you begin
1. Review the Puppet Application Manager system requirements.
2. Note that Swap is not supported for use with this version of Puppet Application Manager (PAM). The installation script attempts to disable Swap if it is enabled.
3. (Optional) If necessary, prepare additional steps related to SELinux and Firewalld:
   The PAM installation script disables SELinux and Firewalld by default. If you want to keep SELinux enabled, append the `--preserve-selinux-config` switch to the PAM install command. This may require additional configuration to adapt SELinux policy to the installation.
   If you want to keep Firewalld enabled:
   a. Make sure Firewalld is installed on your system.
   b. To prevent the installation from disabling Firewalld, provide a patch file to the PAM install command using
      
      ```
      -s installer-spec-file=patch.yaml
      ```
      where `patch.yaml` is the name of your patch file. For reference, here's an example patch file that enables Firewalld during installation, starts the service if it isn't running, and adds rules to open relevant ports:

      ```yaml
      apiVersion: cluster.kurl.sh/v1beta1
      kind: Installer
      metadata:
        name: patch
      spec:
        firewalldConfig:
          firewalld: enabled
          command: ["/bin/bash", "-c"]
          args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
            sysctl -p"]
        firewalldCmds:
          - ["--permanent", "--zone=trusted", "--add-interface=weave"]
          - ["--permanent", "--zone=external", "--add-masquerade"]
          # SSH port
          - ["--permanent", "--zone=public", "--add-port=22/tcp"]
          # HTTPS port
          - ["--permanent", "--zone=public", "--add-port=443/tcp"]
          # Kubernetes etcd port
          - ["--permanent", "--zone=public", "--add-port=2379-2830/tcp"]
          # Kubernetes API port
          - ["--permanent", "--zone=public", "--add-port=6443/tcp"]
          # Weave Net port
          - ["--permanent", "--zone=public", "--add-port=6783/udp"]
          # Weave Net port
          - ["--permanent", "--zone=public", "--add-port=6783-6874/tcp"]
          # CD4PE Webhook callback port (uncomment line below if needed)
          - ["--permanent", "--zone=public", "--add-port=8000/tcp"]
          # KOTS UI port
          - ["--permanent", "--zone=public", "--add-port=8800/tcp"]
      ```
# CD4PE Local registry port (offline only, uncomment line below if needed)
# - ["--permanent", "--zone=public", "--add-port=9001/tcp"]
# Kubernetes component ports (kublet, kube-scheduler, kube-controller)
- ["--permanent", "--zone=public", "--add-port=10250-10252/tcp"]
# Reload firewall rules
- ["--reload"]
bypassFirewalldWarning: true
disableFirewalld: false
hardFailOnFirewalld: false
preserveConfig: false

4. Ensure that IP address ranges 10.96.0.0/22 and 10.32.0.0/22 are locally accessible. See Resolve IP address range conflicts for instructions.
5. Ensure that the nodes can resolve their own hostnames, through either local host mapping or a reachable DNS server.
6. If you use the puppetlabs/firewall module to manage your cluster's firewall rules with Puppet, be advised that purging unknown rules from changes breaks Kubernetes communication. To avoid this, apply the puppetlabs/pam_firewall module before installing Puppet Application Manager.
7. If you're restoring a backup from a previous cluster, make sure you include the kurl-registry-ip=<YOUR_IP_ADDRESS> installation option. For more information, see Disaster recovery or migration using a snapshot on page 77.

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

1. From a workstation with internet access, download the cluster installation bundle (note that this bundle is ~6GB):

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

2. Copy the installation bundle to the host node and unpack it:

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

3. Run the installation command:

   cat install.sh | sudo bash -s airgap

   a) The installation script prints the address and password (only shown once, so make careful note of it) for Puppet Application Manager:

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

   **Note:** If you lose this password or wish to change it, see Reset the Puppet Application Manager password for instructions.

4. Navigate to the Puppet Application Manager UI using the address provided by the installation script (http://<PUPPET APPLICATION MANAGER ADDRESS>:8800) and follow the prompts.

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

   Follow the instructions for configuring and deploying your Puppet applications on Puppet Application Manager. For more information, see Install applications via the PAM UI on page 65.
For more information on installing Continuous Delivery for PE offline, see Install Continuous Delivery for PE in an offline environment.

For more information on installing Comply offline, see Install Comply offline.

**Related information**

Reset the PAM password on page 80
As part of the installation process, Puppet Application Manager (PAM) generates a password for you. You can update this password to one of your choosing after installation.

PAM system requirements on page 34
You can install Puppet Application Manager (PAM) on a Puppet-supported cluster or add PAM to a customer-supported cluster. Before installing PAM, ensure that your system meets these requirements.

Resolve IP address range conflicts on page 79
When installing Puppet Application Manager, IP address ranges 10.96.0.0/22 and 10.32.0.0/22 must not be used by other nodes on the local network.

Architecture overview on page 32
Puppet Application Manager runs on Kubernetes. Puppet provides several supported configurations for different use cases.

Using sudo behind a proxy server on page 83
Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use sudo to elevate privileges. If you're running PAM behind a proxy, sudo might not work correctly. If you're having trouble running commands with sudo, and you're behind a proxy, try switching to the root user and running the command without sudo.

**Automate PAM and Puppet application online installations**

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

**Before you begin**

Ensure that your system meets the PAM system requirements on page 34.

1. Install Puppet Application Manager. For detailed instructions, see PAM HA online installation on page 48.
2. Define the configuration values for your Puppet application installation, using Kubernetes YAML format.

```yaml
apiVersion: kots.io/v1beta1
kind: ConfigValues
metadata:
  name: app-config
spec:
  values:
    accept_eula:
      value: has_accepted_eula
  annotations:
    value: "ingress.kubernetes.io/force-ssl-redirect: 'false'"
hostname:
  value: "<HOSTNAME>"
root_password:
  value: "<ROOT ACCOUNT PASSWORD>"

Tip: View the keyword names for all settings by clicking View files > upstream > config.yaml in Puppet Application Manager.

Replace the values indicated:

- Replace <HOSTNAME> with a hostname you want to use to configure an Ingress and to tell job hardware agents and web hooks how to connect to it. You might need to configure your DNS to resolve the hostname to your Kubernetes hosts.
- Replace <ROOT ACCOUNT PASSWORD> your chosen password for the application root account. The root account is used to administer your application and has full access to all resources and application-wide settings. This account must NOT be used for testing and deploying control repositories or modules.
- Optional. These configuration values disable HTTP-to-HTTPS redirection, so that SSL can be terminated at the load balancer. If you want to run the application over SSL only, change the force-ssl-redirect annotation to true.
- Optional. If your load balancer requires HTTP health checks, you can now enable Ingress settings that do not require Server Name Indication (SNI) for /status. To enable this setting, add the following to the config values statement:

```
enable_lb_healthcheck:
  value: "1"
```

Note: The automated installation automatically accepts the Puppet application end user license agreement (EULA). Unless Puppet has otherwise agreed in writing, all software is subject to the terms and conditions of the Puppet Master License Agreement located at https://puppet.com/legal.

3. Write your license file and the configuration values generated in step 1 to the following locations:

- Write your license file to ./replicated_license.yaml
- Write your configuration values to ./replicated_config.yaml

4. Add the Puppet application definition to Puppet Application Manager with the license file and configuration values, passing in the Puppet Application Manager password you set in step 4:

```
kubectl kots install <APPLICATION NAME> --namespace default --shared-password <YOUR CHOSEN PASSWORD> --port-forward=false --license-file ./replicated_license.yaml --config-values ./replicated_config.yaml
```

Note: If you want to install a specific version of the application, include the --app-version-label=<VERSION> flag in the install command.

5. Wait five minutes to allow the software time to process the change.

Your configuration values are applied, and if preflight checks have passed, the application is deployed and in the process of starting up.

The application's status on the Application tab is shown as **Missing** for several minutes while deployment is underway. To monitor the deployment's progress, run `kubectl get pods --watch`.

When the deployment is complete, the application status changes to **Ready**.

7. Update your DNS or `/etc/hosts` file to include the hostname you chose during configuration.

8. Installation is now complete! Navigate to https://<HOSTNAME> and sign into Puppet application.

**Related information**

- PAM HA online installation on page 48
  The Puppet Application Manager (PAM) installation process creates a Kubernetes cluster for you and walks you through installing your Puppet application on the cluster.

- Upgrade an automated online application installation on page 67
  If you installed a Puppet application following the automated online installation instructions, run a script to upgrade to the latest version.

**Automate PAM and Puppet application offline installations**

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

**Before you begin**

Ensure that your system meets the PAM system requirements on page 34.

1. Install Puppet Application Manager. For detailed instructions, see PAM HA offline installation on page 52.
2. Define the configuration values for your Puppet application installation, using Kubernetes YAML format.

```yaml
apiVersion: kots.io/v1beta1
kind: ConfigValues
metadata:
  name: app-config
spec:
  values:
    accept_eula:
      value: has_accepted_eula
    annotations:
      value: "ingress.kubernetes.io/force-ssl-redirect: 'false'"
    hostname:
      value: "<HOSTNAME>"
    root_password:
      value: "<ROOT ACCOUNT PASSWORD>"
```

**Tip:** View the keyword names for all settings by clicking **View files > upstream > config.yaml** in Puppet Application Manager.

Replace the values indicated:

- Replace `<HOSTNAME>` with a hostname you want to use to configure an Ingress and to tell job hardware agents and web hooks how to connect to it. You might need to configure your DNS to resolve the hostname to your Kubernetes hosts.
- Replace `<ROOT ACCOUNT PASSWORD>` your chosen password for the application root account. The root account is used to administer your application and has full access to all resources and application-wide settings. This account must NOT be used for testing and deploying control repositories or modules.
- **Optional.** These configuration values disable HTTP-to-HTTPS redirection, so that SSL can be terminated at the load balancer. If you want to run the application over SSL only, change the `force-ssl-redirect` annotation to `true`.
- **Optional.** If your load balancer requires HTTP health checks, you can now enable Ingress settings that do not require Server Name Indication (SNI) for `/status`. To enable this setting, add the following to the config values statement:

```yaml
enable_lb_healthcheck:
  value: "1"
```

**Note:** The automated installation automatically accepts the Puppet application end user license agreement (EULA). Unless Puppet has otherwise agreed in writing, all software is subject to the terms and conditions of the Puppet Master License Agreement located at https://puppet.com/legal.

3. Write your license file and the configuration values generated in step 1 to the following locations:

- Write your license file to `./replicated_license.yaml`
- Write your configuration values to `./replicated_config.yaml`

4. Download the application bundle:

```bash
curl -L <APPLICATION BUNDLE URL> -o <APPLICATION BUNDLE FILE>
```

5. Copy the application bundle to your primary and secondary nodes and unpack it:

```bash
tar xzf ./<APPLICATION BUNDLE FILE>
```
6. Run the application install command on your primary node. Replace the `<YOUR CHOSEN PASSWORD>`, `<APPLICATION NAME>`, `<APPLICATION BUNDLE FILE>` values in the example below with your own values:

```
KOTS_PASSWORD=<YOUR CHOSEN PASSWORD>
kubectl kots install <APPLICATION NAME> --namespace default --shared-password $KOTS_PASSWORD --license-file ./license.yaml --config-values ./config.yaml --airgap-bundle ./<APPLICATION BUNDLE FILE> --port-forward=false
# wait several minutes for the application to deploy; if it doesn’t show up, preflights or another error might have occurred
```

**Note:** If you want to install a specific version of the application, include the `--app-version-label=<VERSION>` flag in the install command.

**Related information**

**PAM HA offline installation** on page 52
Use these instructions to install Puppet Application Manager (PAM) in an air-gapped or offline environment where the Puppet Application Manager host server does not have direct access to the internet.

**Upgrade an automated offline application installation** on page 67
If you installed a Puppet application following the automated offline installation instructions, run a script to upgrade to the latest version.

**Uninstall PAM**

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

To uninstall Puppet Application Manager (PAM) from Puppet-supported online clusters, use:

```
curl <installer URL>/tasks.sh | sudo bash -s reset
```

To uninstall Puppet Application Manager from Puppet-supported offline clusters, use:

```
cat tasks.sh | sudo bash -s reset
```

**Important:** You must be aware that the commands above do not completely clean up everything. Specifically, they do not clean up packages and config files, and do not remove the container runtime. They do, however, leave the system in a state that would be safe to restart the installer.

**Uninstall PAM on customer-supported clusters**

To uninstall Puppet Application Manager from customer-supported clusters, use:

```
kubectl delete namespace <pam-namespace>
kubectl delete clusterrolebinding kotsadm-rolebinding
kubectl delete clusterrole kotsadm-role
```

**Related information**

**Using sudo behind a proxy server** on page 83
Many of the commands you run to install or configure Puppet Application Manager (PAM) require root access. In the PAM documentation, commands that require root access use `sudo` to elevate privileges. If you're running PAM behind a proxy, `sudo` might not work correctly. If you're having trouble running commands with `sudo`, and you're behind a proxy, try switching to the `root` user and running the command without `sudo`. 

© 2021 Puppet, Inc.
Working with Puppet applications

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

• **Install applications via the PAM UI** on page 65
  The process of adding an application once you've installed Puppet Application Manager is simple.
• **Update a license for online installations** on page 66
  If you have performed online installation of an application, you can use the Puppet Application Manager UI to update your license.
• **Update a license for offline installations** on page 66
  If you have performed offline installation of an application, you can use the Puppet Application Manager UI to update your license.
• **Upgrade an automated online application installation** on page 67
  If you installed a Puppet application following the automated online installation instructions, run a script to upgrade to the latest version.
• **Upgrade an automated offline application installation** on page 67
  If you installed a Puppet application following the automated offline installation instructions, run a script to upgrade to the latest version.

Install applications via the PAM UI

The process of adding an application once you’ve installed Puppet Application Manager is simple.

<table>
<thead>
<tr>
<th>Application</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Delivery for Puppet Enterprise</td>
<td>4.6.0 or later</td>
</tr>
<tr>
<td>Comply</td>
<td>1.0.4 or later</td>
</tr>
</tbody>
</table>

For information on installing Puppet applications via the command line, see **Automate PAM and Puppet application online installations** on page 60 and **Automate PAM and Puppet application offline installations** on page 62.

To install a Puppet application using the Puppet Application Manager UI:

1. Log into the Puppet Application Manager UI, and click **Add a new application**.
   • If you have not added a Puppet application before you are prompted to upload a license.
   • If you have already added a Puppet application, click **Add a new application**.
2. Upload your `replicated_license.yaml` file when requested.

**Note:** Once the license file is installed, if offline installations are enabled, you are presented with an option to proceed with an offline setup.

Add the following information to install an offline application:

- **Hostname** - the hostname you want to use to configure an Ingress and to tell job hardware agents and web hooks how to connect to it. You might need to configure your DNS to resolve the hostname to your Kubernetes hosts.

  **Important:** The hostname must be unique for each application you install.

- **Username/Password** - The username and password for the application root account. The root account is used to administer your application and has full access to all resources and application-wide settings. This account must NOT be used for testing and deploying control repositories or modules.

- **Registry namespace** - the registry namespace for the application, e.g. CD4PE or Comply.

- **Airgap bundle** - upload the relevant application bundle tarball. Click **Continue**.

3. Add any additional required information that is presented on the **Config** page. Configure any other settings on the page relevant to your installation, such as external databases, customized endpoints, a load balancer, or TLS certificates. Click **Save Config** when you are done.

Saving your new configuration settings prompts the creation of a new application version.

4. Click **Go to new version**, which redirects you to the **Version history** tab. The newly created version is shown in the **All versions** section of the page.

5. Monitor the new version's preflight checks. The **Running Checks** indicator is shown on the screen while your system is checked to make sure your cluster meets minimum system requirements. When the preflight check is complete:

   - If the status is **Checks Failed**, click **View preflights**. Correct the issues and click **Re-run**. Repeat this step as needed.

     **Important:** Do not move on until all preflight checks pass.

   - If the status is **Ready to Deploy**, move on to the next step.

6. Once the version is ready to deploy, click **Deploy**. On the **Application** tab, monitor the application for readiness. The application's status is shown as **Missing** for several minutes while deployment is underway. To monitor the deployment's progress, run `kubectl get pods --watch`.

   When the deployment is complete, the application status changes to **Ready**.

7. Navigate to `https://<HOSTNAME>` (using the hostname you entered on the **Config** screen) and sign into your application.

**Update a license for online installations**

If you have performed online installation of an application, you can use the Puppet Application Manager UI to update your license.

To update the license for an online application:

1. Log in to Puppet Application Manager, click the **License** tab, and then **Sync License**.

2. On the **Version history** tab, click **Deploy**.

   Puppet Application Manager adds “License Change” as the deployment cause on the **Version history** tab.

**Update a license for offline installations**

If you have performed offline installation of an application, you can use the Puppet Application Manager UI to update your license.

To update the license for an offline application:
1. Ask your Puppet sales representative to email you an updated license file.
2. Log in to Puppet Application Manager, click the License tab.
3. Drag and drop or upload the updated license file provided by your Puppet sales representative.
4. On the Version history tab, click Deploy.

Puppet Application Manager adds “License Change” as the deployment cause on the Version history tab.

**Upgrade an automated online application installation**

If you installed a Puppet application following the automated online installation instructions, run a script to upgrade to the latest version.

**Important:** Ensure that you are following an approved upgrade path for the application you want to upgrade. For more information, check the relevant application documentation.

1. From the command line of your primary (control plane) node, get the application slug for the application you want to upgrade:

```
kubectl kots --namespace <NAMESPACE> get apps
```

Replace `<NAMESPACE>` with the name of the namespace in which you installed PAM (usually default).

2. Run the upgrade script:

```
kubectl kots upstream upgrade <APPLICATION SLUG> --namespace <NAMESPACE> --deploy
```

Replace `<APPLICATION SLUG>` with the relevant application slug for the application you want to upgrade.

Replace `<NAMESPACE>` with the name of the namespace in which you installed PAM (usually default).

3. Wait five minutes to allow the software time to process the change.


If preflight checks have passed, the upgraded application is deployed and in the process of starting up. To monitor the deployment's progress, run:

```
kubectl get pods --watch
```

**Related information**

Automate PAM and Puppet application offline installations on page 62

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

**Upgrade an automated offline application installation**

If you installed a Puppet application following the automated offline installation instructions, run a script to upgrade to the latest version.

**Important:** Ensure that you are following an approved upgrade path for the application you want to upgrade. For more information, check the relevant application documentation.

1. Download the application bundle you want to upgrade to. Copy to your primary node.

2. From the command line of your primary (control plane) node, get the application slug for the application you want to upgrade:

```
kubectl kots --namespace <NAMESPACE> get apps
```

Replace `<NAMESPACE>` with the name of the namespace in which you installed PAM (usually default).
3. Run the upgrade script:

```
kubectl kots upstream upgrade <APPLICATION SLUG> --airgap-bundle ./<APPLICATION BUNDLE FILE> --kotsadm-namespace <REGISTRY NAMESPACE> --namespace <NAMESPACE> --deploy
```

- Replace `<APPLICATION SLUG>` with the relevant application slug for the application you want to upgrade.
- Replace `<APPLICATION BUNDLE FILE>` with the name of the application bundle file.
- Replace `<REGISTRY NAMESPACE>` with your Registry namespace where images are uploaded.
- Replace `<NAMESPACE>` with the name of the namespace in which you installed PAM (usually `default`).

4. Wait five minutes to allow the software time to process the change.

5. Navigate to `http://<NODE IP ADDRESS>:8800` and log in with the Puppet Application Manager password.

If preflight checks have passed, the upgraded application is deployed and in the process of starting up. To monitor the deployment's progress, run:

```
kubectl get pods --watch
```

**Related information**

**Automate PAM and Puppet application offline installations** on page 62

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

---

**Maintenance and tuning**

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

**How to determine your version of Puppet Application Manager**

You can use the following command to determine what version of Puppet Application Manager (PAM) you're using:

```
kubectl get installer --sort-by=.metadata.creationTimestamp -o jsonpath='{.items[-1:].metadata.name}' ; echo
```

Depending on which version you've installed, the command returns one of these values:

- **HA architecture**: puppet-application-manager
- **Standalone architecture**: puppet-application-manager-standalone
- **Legacy architecture**: Any other value, for example, puppet-application-manager-legacy, cd4pe, or comply

**Rebooting PAM nodes**

Where possible, avoid rebooting or shutting down a PAM node. Shutting down a legacy or HA PAM node incorrectly could result in storage volume corruption and the loss of data.

For tasks such as package updates or security patches, where you must perform a reboot or shut down, follow the procedure below to gracefully shut down the node and ensure that it is drained correctly.

To reboot a node:

1. Shut down services using Ceph-backed storage:

```
/opt/ekco/shutdown.sh
```
2. If you're using a high availability (HA) cluster, drain the node:

   kubectl drain <NODE NAME> --ignore-daemonsets --delete-local-data

3. Reboot the node.

Load balancer health checks
To set up health checks for the load balancer that your Puppet Application Manager (PAM) applications are running behind, set up rules for these applications and services.

<table>
<thead>
<tr>
<th>Application/service</th>
<th>URL/port</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Puppet application. For example,</td>
<td>https://&lt;CDPE HOSTNAME&gt;:443/status</td>
<td>Although Puppet applications might expose other ports (Continuous</td>
</tr>
<tr>
<td>Continuous Delivery for Puppet</td>
<td></td>
<td>Delivery for PE exposes ports 443, 80, and 8000), 443 is the HTTPS</td>
</tr>
<tr>
<td>Enterprise or Puppet Comply</td>
<td></td>
<td>endpoint, and is the best port to use for health checks.</td>
</tr>
<tr>
<td>Puppet Application Manager (PAM)</td>
<td>https://&lt;KUBERNETES PRIMARY IP&gt;:8800/healthz</td>
<td></td>
</tr>
<tr>
<td>External load balancer endpoint</td>
<td>Port 6443 or https://&lt;KUBERNETES PRIMARY IP&gt;:6443/livez</td>
<td>For information on setting up a TCP probe on an external load</td>
</tr>
<tr>
<td></td>
<td></td>
<td>balancer endpoint, consult the kURL load balancer documentation.</td>
</tr>
<tr>
<td>Local container registry (for offline</td>
<td>https://&lt;KUBERNETES PRIMARY IP&gt;:9001</td>
<td></td>
</tr>
<tr>
<td>installations)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

There are four possible upgrade types for Puppet Application Manager installations:

- **Online** - For standalone or HA installations with a connection to the internet.
- **Offline** - For air-gapped standalone or HA installations without a connection to the internet.
- **Online legacy** - For standalone or HA installations created prior to April 2021 with a connection to the internet.
- **Offline legacy** - For air-gapped standalone or HA installations created prior to April 2021 without a connection to the internet.

**Restriction:** You cannot upgrade from a legacy deployment to a non-legacy deployment, or from standalone to HA, or a HA to a standalone deployment.

How to determine your version of Puppet Application Manager
You can use the following command to determine what version of Puppet Application Manager (PAM) you're using:

   kubectl get installer --sort-by=.metadata.creationTimestamp -o jsonpath='{.items[-1:].metadata.name}' ; echo

Depending on which version you've installed, the command returns one of these values:

- **HA architecture**: puppet-application-manager
- **Standalone architecture**: puppet-application-manager-standalone

© 2021 Puppet, Inc.
• **Legacy architecture**: Any other value, for example, `puppet-application-manager-legacy`, `cd4pe`, or `comply`.

## Upgrade PAM online

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

### Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at [Backing up PAM using snapshots](#) on page 74.

1. On your first primary node, rerun the installation script, passing in any arguments you included when installing for the first time:

   For standalone deployments, use:
   ```bash
   curl -sSL https://k8s.kurl.sh/puppet-application-manager-standalone | sudo bash -s force-reapply-addons
   ```

   For HA deployments, use:
   ```bash
   curl -sSL https://k8s.kurl.sh/puppet-application-manager | sudo bash -s force-reapply-addons
   ```

2. If a new version of Kubernetes is available, the installer notes upgrade scripts to run on other nodes in an HA cluster.

   The installer also pauses before draining nodes as part of the Kubernetes upgrade. The node draining process can take several minutes to complete, during which time application workloads are stopped or migrated to other systems. This migration may cause several minutes of downtime while databases are rescheduled.

### Related information

[Using sudo behind a proxy server](#) on page 83

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

## Upgrade PAM offline

Users operating in environments without direct access to the internet must use the links below to upgrade to the latest version of Puppet Application Manager (PAM).

### Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at [Backing up PAM using snapshots](#) on page 74.

To upgrade Puppet Application Manager:
1. From a workstation with internet access, download the latest version of the installation bundle that is relevant for your installation type:
   For standalone installations, enter the following command (note that this bundle is ~6GB):
   ```
   https://k8s.kurl.sh/bundle/puppet-application-manager-standalone.tar.gz
   ```
   For HA installations, enter the following command (note that this bundle is ~4GB):
   ```
   https://k8s.kurl.sh/bundle/puppet-application-manager.tar.gz
   ```

2. Copy the installation bundle to your primary and secondary nodes and unpack it:
   For standalone installations, use:
   ```
   tar xzf puppet-application-manager-standalone.tar.gz
   ```
   For HA installations, use:
   ```
   tar xzf puppet-application-manager.tar.gz
   ```

3. On your primary node, rerun the installation script, passing in any arguments you included when installing for the first time:
   ```
   cat install.sh | sudo bash -s airgap force-reapply-addons
   ```
   **Note:** This script issues a prompt to run the `task.sh` and `upgrade.sh` scripts on your secondary nodes. Use the versions of these scripts from the downloaded bundle in step 2.

4. If a new version of Kubernetes is available, the installer systems provide upgrade scripts to run on other nodes in an HA cluster. The installer also pauses before draining nodes as part of the Kubernetes upgrade. Node draining is performed as part of a Kubernetes upgrade.
   The node draining process can take several minutes to complete, during which time application workloads are stopped or migrated to other systems. This migration may cause several minutes of downtime while databases are rescheduled.

   When the deployment is complete, sign into Puppet Application Manager - http://<PUPPET APPLICATION MANAGER ADDRESS>:8800 - and verify that the new version number is displayed in the bottom left corner of the web UI.

**PAM legacy upgrades**

Use this method to upgrade Puppet Application Manager (PAM) from a version installed before May 2021.

**Before you begin**

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at Backing up PAM using snapshots on page 74.

**Restriction:** It is not currently possible to upgrade from an online legacy install to a new offline install configuration. Similarly, upgrades from an offline legacy configuration to a new online install are not currently supported.

The legacy version reflects an older configuration that used Ceph 1.0 which hosted data directly on the file system. Puppet no longer recommend this for new installs, but maintains it to support existing installs and ensure other components can be kept up-to-date.

**Legacy architecture is deprecated:** The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. Puppet will continue to update legacy architecture components other than Kubernetes until **30 June 2022**. If security
advisories against Kubernetes 1.19 arise, the remediation path is to migrate to one of the newer architectures by following the instructions in Disaster recovery or migration using a snapshot on page 77.

To upgrade a legacy version of Puppet Application Manager on nodes with internet access:

1. On your node (or control plane node if you have a HA deployment), rerun the installation script, passing in any arguments you included when installing for the first time:
   - For standalone installs:
     ```bash
curl -sSL https://k8s.kurl.sh/puppet-application-manager-legacy | sudo bash -s force-reapply-addons
```
   - For HA installs:
     ```bash
curl -sSL https://k8s.kurl.sh/puppet-application-manager-legacy | sudo bash -s ha force-reapply-addons
```

2. If a new version of Kubernetes is available, the systems provide upgrade scripts to run on each node in your cluster.
   Node draining is performed as part of a Kubernetes upgrade. The node draining process can take several minutes to complete.

   **Note:** During the Kubernetes upgrade process, nodes are not able to properly route network connections. If you have a HA deployment, make sure you have load balancers or a multi-node fail-over process in place, or schedule downtime before upgrading.

**PAM offline legacy upgrades**

Use this method to upgrade Puppet Application Manager (PAM) on offline nodes from a version installed before April 2021.

**Before you begin**

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at Backing up PAM using snapshots on page 74.

**Restriction:** It is not currently possible to upgrade from an online legacy install to a new offline install configuration. Similarly, upgrades from an offline legacy configuration to a new online install are not currently supported.

**Legacy architecture is deprecated:** The legacy architecture utilizes Rook 1.0, which is incompatible with Kubernetes version 1.20 and newer versions. Kubernetes version 1.19 is no longer receiving security updates. Puppet will continue to update legacy architecture components other than Kubernetes until **30 June 2022**. If security advisories against Kubernetes 1.19 arise, the remediation path is to migrate to one of the newer architectures by following the instructions in Disaster recovery or migration using a snapshot on page 77.

To upgrade Puppet Application Manager on nodes without a connection to the internet:

1. From a workstation with internet access, download the latest version of the cluster installation bundle (note that this bundle is ~4GB):
   ```bash
   https://k8s.kurl.sh/bundle/puppet-application-manager-legacy.tar.gz
   ```

2. Copy the installation bundle to your primary and secondary Puppet Application Manager nodes and unpack it:
   ```bash
tar xzf puppet-application-manager-legacy.tar.gz
   ```

© 2021 Puppet, Inc.
3. Rerun the installation script. Don't forget to pass in any additional arguments you included when installing for the first time you installed the product:

   For standalone installs use:

   ```
   cat install.sh | sudo bash -s airgap force-reapply-addons
   ```

   For HA installs use:

   ```
   cat install.sh | sudo bash -s airgap ha force-reapply-addons
   ```

   **Note:** During the upgrade process, follow any prompts to run commands on your other cluster nodes.

   When the deployment is complete, sign into Puppet Application Manager and verify that the new version number is displayed in the bottom center of the web UI.

---

### Upgrading PAM on a customer-supported cluster

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

There are two possible upgrade types for customer-supported Puppet Application Manager deployments:

- **Online** - For standalone or HA installations with a connection to the internet.
- **Offline** - For air-gapped standalone or HA installations without a connection to the internet.

#### How to determine your version of Puppet Application Manager

You can use the following command to determine what version of Puppet Application Manager (PAM) you're using:

```bash
kubectl get installer --sort-by=.metadata.creationTimestamp -o jsonpath='{.items[-1].metadata.name}'
```

Depending on which version you've installed, the command returns one of these values:

- **HA architecture**: `puppet-application-manager`
- **Standalone architecture**: `puppet-application-manager-standalone`
- **Legacy architecture**: Any other value, for example, `puppet-application-manager-legacy`, `cd4pe`, or `comply`

---

### Upgrade PAM on a customer-supported online cluster

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

#### Before you begin

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at [Backing up PAM using snapshots](#) on page 74.

To upgrade Puppet Application Manager on a customer-supported online cluster:

1. Upgrade kubectl KOTS:

   ```
   curl https://kots.io/install | bash
   ```
2. Issue the following KOTS command:

```
kubectl kots admin-console upgrade --namespace <target namespace>
```

**Tip:** Run the `kubectl kots admin-console upgrade -h` command for more usage information.

## Upgrade PAM on a customer-supported offline cluster

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

**Before you begin**

Make sure you have captured an up-to-date snapshot of your PAM installation, which you can use to fall back the current version if there is an issue with the upgrade process. Learn more about snapshots at [Backing up PAM using snapshots](#) on page 74.

To upgrade Puppet Application Manager on a customer-supported offline cluster, perform the following steps from a workstation that has `kubectl` access to the cluster:

1. Upgrade kubectl KOTS:

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

2. Ensure the required images are available in your local registry. Download the release assets matching the CLI version using the following command:

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

3. Extract the images and push them to your private registry. Registry credentials provided in this step must have push access. These credentials are not stored anywhere or reused later.

   ```bash
   kubectl kots admin-console push-images ./kotsadm.tar.gz
   --registry-username <rw-username>
   --registry-password <rw-password>
   ```

4. After you push the images to your private registry, execute the upgrade command with registry read-only credentials:

   ```bash
   kubectl kots upgrade puppet-application-manager
   --kotsadm-namespace puppet-application-manager
   --kotsadm-registry <private.registry.host>/puppet-application-manager
   --registry-username <ro-username>
   --registry-password <ro-password>
   --namespace <target namespace>
   ```

## Backing up PAM using snapshots

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

**Related information**

[Using sudo behind a proxy server](#) on page 83

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

**Full and partial snapshots**

There are two options available when you're creating a snapshot for your Puppet Application Manager (PAM) deployment, full snapshots (also known as instance snapshots) and partial (or application) snapshots. For full disaster recovery, make sure you've configured and scheduled regular full snapshots stored on a remote storage solution such as an S3 bucket or NFS share.

Full snapshots offer a comprehensive backup of your PAM deployment, because they include the core PAM application together with the Puppet applications you've installed in your PAM deployment. You can use a full snapshot to restore your PAM deployment and all of your installed Puppet applications to a previous backup. For example, you could use a full snapshot to revert an undesired configuration change or a failed upgrade, or to migrate your PAM deployment to another Puppet-supported cluster.

Partial snapshots are available from the PAM console, but are limited in their usefulness. To restore from a partial snapshot, you must already have an installed and functioning version of PAM. A functioning PAM installation is needed because the option to restore a partial snapshot can only be accessed from the Snapshots section of the PAM admin console.

Partial snapshots only back up the Puppet application you specified when you configured the snapshot, for example, Continuous Delivery for Puppet Enterprise, or Puppet Comply. They do not back up the underlying PAM deployment. Partial snapshots are sometimes useful if you want to roll back to a previous version of a specific Puppet application that you've installed on your PAM deployment, but are far less versatile than full snapshots. To make sure that you have all disaster recovery options available to you, use a full snapshot wherever possible.

**Configure snapshots**

Before using snapshots, select a storage location, set a snapshot retention period, and indicate whether snapshots are created manually or on a set schedule.

**Note:** The snapshots feature was accidentally disabled on some application licenses issued prior to March 2021. If you do not see a Snapshots option in your Puppet Application Manager UI, and you would like to use this feature, please contact our Support team using the Zendesk account provided to you.

A beta version of the snapshots feature, which only supported rollback snapshots, was available prior to the 15 April Puppet Application Manager release. Some features or storage locations mentioned on this page are not available on older versions of Puppet Application Manager.

**Important:** Disaster recovery requires that the store backend used for backups is accessible from the new cluster. When setting up snapshots in an offline cluster, make sure to record the registry service IP address with the following command:

```
kubectl -n kurl get svc registry -o jsonpath='{.spec.clusterIP}'
```

be sure to record the value returned by this command as it is required when creating a new cluster to restore to as part of disaster recovery. For more information on restoring from snapshots, see Disaster recovery or migration using a snapshot on page 77.

1. In the upper navigation bar of the Puppet Application Manager UI, click Snapshots > Settings & Schedule.
2. The snapshots feature uses `https://velero.io`, an open source backup and restore tool. Click Check for Velero to determine whether Velero is present on your cluster, and to install it if needed.
3. Select a destination for your snapshot storage and provide the required configuration information. Supported destinations are listed below. We recommend using an external service or NFS, depending on what is available to you:

- Internal storage (default)
- Amazon S3
- Azure Blob Storage
- Google Cloud Storage
- Other S3-compatible storage
- Network file system (NFS)
- Host path

For **Amazon S3 storage**, provide the following information:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket</td>
<td>The name of the AWS bucket where snapshots are stored.</td>
</tr>
<tr>
<td>Region</td>
<td>The AWS region the bucket is available in.</td>
</tr>
<tr>
<td>Path</td>
<td><strong>Optional.</strong> The path within the bucket where all snapshots are stored.</td>
</tr>
<tr>
<td>Use IAM instance role?</td>
<td>If selected, an IAM instance role is used instead of an access key ID and secret.</td>
</tr>
<tr>
<td>Access key ID</td>
<td><strong>Required only if not using an IAM instance role.</strong> The AWS IAM access key ID that can read from and write to the bucket.</td>
</tr>
<tr>
<td>Access key secret</td>
<td><strong>Required only if not using an IAM instance role.</strong> The AWS IAM secret access key that is associated with the access key ID.</td>
</tr>
</tbody>
</table>

For **Azure Blob Storage**, provide the following information:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket</td>
<td>The name of the Azure Blob Storage container where snapshots are stored.</td>
</tr>
<tr>
<td>Path</td>
<td><strong>Optional.</strong> The path within the container where all snapshots are stored.</td>
</tr>
<tr>
<td>Subscription ID</td>
<td><strong>Required only for access via service principal or AAD Pod Identity.</strong> The subscription ID associated with the target container.</td>
</tr>
<tr>
<td>Tenant ID</td>
<td><strong>Required only for access via service principal.</strong> The tenant ID associated with the Azure account of the target container.</td>
</tr>
<tr>
<td>Client ID</td>
<td><strong>Required only for access via service principal.</strong> The client ID of a Service Principle with access to the target container.</td>
</tr>
<tr>
<td>Client secret</td>
<td><strong>Required only for access via service principal.</strong> The Client Secret of a Service Principle with access to the target container.</td>
</tr>
<tr>
<td>Cloud name</td>
<td>The Azure cloud for the target storage (options: AzurePublicCloud, AzureUSGovernmentCloud, AzureChinaCloud, AzureGermanCloud)</td>
</tr>
<tr>
<td>Resource group</td>
<td>The resource group name of the target container.</td>
</tr>
<tr>
<td>Storage account</td>
<td>The storage account name of the target container.</td>
</tr>
</tbody>
</table>

**Note:** Only connections via service principals are currently supported.

For **Google Cloud Storage**, provide the following information:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucket</td>
<td>The name of the GCS bucket where snapshots are stored.</td>
</tr>
<tr>
<td>Path</td>
<td><strong>Optional.</strong> The path within the bucket where all snapshots are stored.</td>
</tr>
<tr>
<td>Service account</td>
<td>The GCP IAM Service Account JSON file that has permissions to read from and write to the storage location.</td>
</tr>
</tbody>
</table>
4. Click **Update storage settings** to save your storage destination information. Depending on your chosen storage provider, saving and configuring your storage provider might take several minutes.

5. Optional: To automatically create new snapshots on a schedule, select **Enable automatic scheduled snapshots** on the **Full snapshots (instance)** tab. (If desired, you can also set up a schedule for capturing partial (application-only) snapshots.)

You can schedule a new snapshot creation for every hour, day, or week, or you can create a custom schedule by entering a cron expression.

6. Finally, set the retention schedule for your snapshots by selecting the time period after which old snapshots are automatically deleted. The default retention period is one month.

**Note:** A snapshot's retention period cannot be changed once the snapshot is created. If you update the retention schedule, the new retention period applies only to snapshots created after the update is made.

7. Click **Update schedule** to save your changes.

Snapshots are automatically created according to your specified schedule and saved to the storage location you selected. You can also create an unscheduled snapshot at any time by clicking **Start a snapshot** on the **Dashboard** or on the **Snapshots** page.

### Roll back changes using a snapshot

When necessary, you can use a snapshot to roll back to a previous version of your Puppet Application Manager set-up without changing the underlying cluster infrastructure.

To roll back changes:

1. In console menu of the Puppet Application Manager UI, click **Snapshots > Full Snapshots (Instance)**.
2. Select the snapshot you wish to roll back to from the list of available snapshots and click **Restore from this backup**.
3. Follow the instructions to complete either a partial restore or a full restore.

A full restore is useful if you need to stay on an earlier version of an application and want to disable automatic version updates. Otherwise, a partial restore is the quicker option.

### Disaster recovery or migration using a snapshot

Full disaster recovery is possible using a snapshot. You must create a new cluster to recover to, then follow the process outlined below to recover your instance from a snapshot. You can also use this workflow to migrate data from legacy to standalone deployments, legacy to HA deployments, and between standalone and HA deployments.

**Before you begin**

- On the original system, Puppet Application Manager (PAM) must be configured to support **Full Snapshots (Instance)**.
- Velero must be configured to use an external snapshot destination accessible to both the old and new clusters, such as S3 or NFS.
- Both the old and new clusters must have the same connection status (online/offline). Migrating from offline to online clusters or vice versa is not supported.
- You must use the **30 June 2021** or later release of PAM.
- For offline installs, both the old and new clusters must use the same version of PAM.
- **If you are migrating from a legacy deployment**, before beginning the migration process you must upgrade to PAM version 1.62.0 with the `force-reapply-addons` flag included in the upgrade command.

To perform data migration or disaster recovery:
1. On the original system, find the version of kURL your deployment is using. Save the version information for the next step:

```
kubectl get configmap -n kurl kurl-current-config -o jsonpath="{.data.kurl-version}" && echo ""
```

2. Set up a new cluster to house the recovered instance following the system requirements for your applications. **Do not install any Puppet applications after the PAM installation completes. You'll recover your Puppet applications later in the process.**

   - Install PAM using the version of kURL you retrieved earlier:
     - For online installs:
       ```
curl -sSL https://k8s.kurl.sh/version/<VERSION STRING>/puppet-application-manager | sudo bash <-s options>
```
     - For offline installs:
       ```
curl -O https://k8s.kurl.sh/bundle/version/<VERSION STRING>/puppet-application-manager.tar.gz
```
     - When setting up a new offline cluster as part of disaster recovery, add `kurl-registry-ip=<IP>` to the install options, replacing `<IP>` with the value you recorded when setting up snapshots.

   **Note:** If you do not include the `kurl-registry-ip=<IP>` flag, the registry service will be assigned a new IP address that does not match the IP on the machine where the snapshot was created. You must align the registry service IP address on the new offline cluster to ensure that the restored configuration can pull images from the correct location.

3. To recover using a snapshot saved to a **host path**, ensure user/group 1001 has full access on all nodes by running:

```
chown -R 1001:1001 /<PATH/TO/HOSTPATH>
```

4. Configure the new cluster to connect to your snapshot storage location. Run the following to see the arguments needed to complete this task:

```
kubectl kots -n default velero configure-{hostpath,nfs,aws-s3,other-s3,gcp} --help
```

5. Run `kubectl kots get backup` and wait for the list of snapshots to become available. This might take several minutes.

6. Start the restoration process by running `kubectl kots restore --from-backup <BACKUP NAME>`. The restoration process takes several minutes to complete. When the PAM UI is available, use it to monitor the status of the application.

   **Note:** When restoring, wait for all restores to complete before making any changes. The following command waits for pods to finish restoring data from backup. Other pods may not be ready until updated configuration is deployed in the next step:

```
kubectl get pod -o json | jq -r ".items[] | select(.metadata.annotations."backup.velero.io/backup-volumes") | .metadata.name" | xargs kubectl wait --for=condition=Ready pod --timeout=20m
```

   This command requires the `jq` CLI tool to be installed. It is available in most Linux OS repositories.

© 2021 Puppet, Inc.
7. After the restoration process completes, save your config and deploy:
   a) From the PAM UI, click **Config**.
   b) (Optional) If the new cluster's hostname is different from the old one, update the **Hostname**.
   c) Click **Save Config**.
   d) Deploy the application. You must save your config and deploy even if you haven't made any changes.

   **Note:** If you have installed Continuous Delivery for PE and changed the hostname, you need to update the webhooks that connect Continuous Delivery for PE with your source control provider. For information on how to do this, see [Update webhooks](#).

---

**Troubleshooting PAM**

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

**How to determine your version of Puppet Application Manager**

You can use the following command to determine what version of Puppet Application Manager (PAM) you're using:

```
kubectl get installer --sort-by=.metadata.creationTimestamp -o jsonpath='{.items[-1:].metadata.name}' ; echo
```

Depending on which version you've installed, the command returns one of these values:

- **HA architecture**: puppet-application-manager
- **Standalone architecture**: puppet-application-manager-standalone
- **Legacy architecture**: Any other value, for example, puppet-application-manager-legacy, cd4pe, or comply

**Resolve IP address range conflicts**

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

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

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

To resolve IP address range conflicts, create a `patch.yaml` file and add the `installer-spec-file=patch.yaml` argument when running the installation script (see below):

1. If you use IP addresses internally that overlap **10.32.0.0/22**, add the following to your `patch.yaml` file (10.40.0.0/23 used here as an example range):

   ```yaml
   apiVersion: cluster.kurl.sh/v1beta1
   kind: Installer
   metadata:
     name: patch
   spec:
     weave:
       podCIDR: 10.40.0.0/23
       podCidrRange: "/23"
   ```
2. If you use IP addresses internally that overlap `10.96.0.0/22`, add the following to your `patch.yaml` file (10.100.0.0/23 used here as an example range):

```yaml
spec:
  ...
  kubernetes:
    serviceCIDR: 10.100.0.0/23
    serviceCidrRange: "/23"
```

**CAUTION:** The podCIDR and serviceCIDR ranges must not overlap.

3. Once your `patch.yaml` file is set up, add the `installer-spec-file=patch.yaml` argument when you run the installation script:

```bash
cat install.sh | sudo bash -s airgap installer-spec-file=patch.yaml
```

**Remember:** Add the `installer-spec-file=patch.yaml` argument any time you re-run the installation script, such as when reinstalling to upgrade to a new version.

**Related information**

*Using sudo behind a proxy server* on page 83

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

**Reset the PAM password**

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

1. To reset the Puppet Application Manager password, run:

```bash
kubectl -n default kots reset-password
```

The system prompts you to enter a new password of your choosing.

2. If the command fails with an unknown command "kots" for "kubectl" error, it's because `/usr/local/bin` is not in the path. To address this error, either update the path to include `/usr/local/bin`, or run:

```bash
/usr/local/bin/kubectl-kots reset-password default
```

**Update the PAM TLS certificate**

A self-signed TLS certificate secures the connection between your browser and Puppet Application Manager (PAM). Once the initial Puppet Application Manager setup process is complete, you can upload new certificates by enabling changes to the installation’s Kubernetes secrets.

Use this process if you chose not to add a TLS certificate when installing Puppet Application Manager, or if you need to update your existing TLS certificate.

1. Enable changes to your installation’s `kotsadm-tls` Kubernetes secret by running:

```bash
kubectl -n default annotate secret kotsadm-tls acceptAnonymousUploads=1
```
2. Restart the `kurl-proxy` pod to deploy the change by running:

```bash
kubectl delete pods $(kubectl get pods -A | grep kurl-proxy | awk '{print $2}')
```

3. Once the `kurl-proxy` pod restarts and is back up and running, navigate to `https://<HOSTNAME>:8800/tls` and upload your new TLS certificate.

**Reduce recovery time when a node fails**

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

How much downtime depends on the following factors:

- Timeout for communication between Kubernetes services (at least one minute to mark the node as unreachable).
- Timeout for the ekco service to determine that pods need to be rescheduled. The default is five minutes after node is marked unreachable.
- Time to restart services (at least two minutes, possibly up to five minutes, if there are complex dependencies).

The ekco service can be configured to reschedule pods more quickly by configuring the installation with a `patch.yaml` similar to the following:

```yaml
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  ekco:
    nodeUnreachableToleration: 1m
```

Apply the patch during an install or upgrade by including `installer-spec-file=patch.yaml` as an install option.

**Important:** This patch needs to be included during all future upgrades to avoid resetting the option.

**PAM components**

Puppet Application Manager (PAM) uses a range of mandatory and optional components.

**Support services**

- Database: PostgreSQL (single instance) - [https://www.postgresql.org/](https://www.postgresql.org/)
- Object storage: previously MinIO - [https://min.io](https://min.io), now Ceph - [https://ceph.io](https://ceph.io)
- tlser for basic TLS cert management - [https://github.com/puppetlabs/tlser](https://github.com/puppetlabs/tlser)
- `kurl_proxy` for HTTPS proxying outside the Ingress (ports besides 80/443): [https://github.com/replicatedhq/kots/tree/v1.36.1/kurl_proxy](https://github.com/replicatedhq/kots/tree/v1.36.1/kurl_proxy)

**Kubernetes components**

- Storage (CSI): Rook - [https://rook.io](https://rook.io), Ceph - [https://ceph.io](https://ceph.io)
- Ingress: Project Contour - [https://projectcontour.io](https://projectcontour.io)
- Kubernetes Cluster: kURL - [https://kurl.sh](https://kurl.sh)
- Embedded kURL Cluster Operator: ekco - [https://github.com/replicatedhq/ekco](https://github.com/replicatedhq/ekco)
- Admin Console: KOTS - [https://kots.io](https://kots.io)
- Snapshots: Velero - [https://velero.io](https://velero.io), Restic - [https://restic.net](https://restic.net)
- Monitoring: Prometheus - [https://prometheus.io](https://prometheus.io)
- Registry: Docker Registry - [https://docs.docker.com/registry/](https://docs.docker.com/registry/)
Optional components

Prometheus (+Grafana) and Velero (+Restic) are optional components:

- Prometheus + Grafana uses 112m/node + 600m CPU, 200MiB/node + 1750MiB RAM
- Velero + Restic uses 500m/node + 500m CPU, 512MiB/node + 128MiB RAM

If you do not need these optional components, they can be omitted from the initial install and further upgrades with a patch similar to the following:

```yaml
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  prometheus:
    version: ''
  velero:
    version: ''
```

**Important:** This patch needs to be included during upgrades to avoid adding the components later.

If you want to remove optional components that are already installed, use the following command:

```
kubectl delete ns/monitoring ns/velero
```

Load balancing

The following load balancer requirements are needed for a HA install:

- A network (L4, TCP) load balancer for port 6443 across primary nodes. This is required for Kubernetes components to continue operating in the event that a node fails. The port is only accessed by the Kubernetes nodes and any admins using `kubectl`.
- A network (L4, TCP) or application (L7, HTTP/S) load balancer for ports 80, and 443 across all primaries and secondaries. This maintains access to applications in event of a node failure. Include 8800 if you want external access to the Puppet Application Manager UI.

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

**Important:** If you are using application load balancing, be aware that Ingress items use Server Name Indication (SNI) to route requests, which may require additional configuration with your load balancer. If your load balancer does not support SNI for health checks, enable **Enable load balancer HTTP health check** in the Puppet Application Manager UI **Config** page.

Generate a support bundle

When seeking support, you might be asked to generate and provide a support bundle. This bundle collects a large amount of logs, system information and application diagnostics.

To create a support bundle:

1. In Puppet Application Manager UI, click **Troubleshoot > Generate a support bundle**.
2. Select a method for generating the support bundle:

   - **Generate the bundle automatically.** Click Analyze `<APPLICATION NAME>` (<APPLICATION NAME> is replaced in the UI by the name of the Puppet application you have installed), and Puppet Application Manager generates the bundle for you and uploads it to the **Troubleshoot** page.
   - **Generate the bundle manually.** Click the prompt to generate a custom command for your installation, then run the command on your cluster. Follow the prompts to upload the bundle to Puppet Application Manager.

© 2021 Puppet, Inc.
3. Review the collected data before forwarding it to Puppet, as it may contain sensitive information that you wish to redact.

4. Return to the Troubleshoot page, download the newly created support bundle, and send it to your Puppet Support contact.

Create a support bundle from the command line

If installation of the Puppet Application Manager, or upload of an app, on an embedded kURL cluster fails, it may not be possible to access the UI to generate a support bundle.

You can generate a support bundle by using the default kots.io spec. To do this, run the following command:

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

On an offline server, you can copy the default kots.io spec by using the following command:

```
curl -o spec.yaml https://kots.io -H 'User-agent:Replicated_Troubleshoot/v1beta1'
```

The spec can then be uploaded to the server. Use the local spec by running:

```
kubectl support-bundle /path/to/spec.yaml
```

If the Puppet Application Manager UI is working and the app is installed, you can use:

```
kubectl support-bundle http://<server-address>:8800/api/v1/troubleshoot/<app-slug>
```

If the app is not installed but the Puppet Application Manager UI is running:

```
kubectl support-bundle http://<server-address>:8800/api/v1/troubleshoot
```

If you do not already have the support-bundle kubectl plugin installed, install it by using the command below:

```
curl https://krew.sh/support-bundle | bash
```

Or by installing krew2 and running:

```
kubectl krew install support-bundle
```

Using sudo behind a proxy server

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

Set up Continuous Delivery for PE

The initial installation and set up process for Continuous Delivery for Puppet Enterprise (PE) includes understanding the architecture, selecting an appropriate TLS configuration, configuring and deploying Continuous Delivery for PE, ensuring your PE version and browser are supported, and configuring your analytics data collection preferences.

Note: Migrating 3.x data to 4.x explains how to upgrade to the Continuous Delivery for PE 4.x series from a version in the 3.x series.
• **Continuous Delivery for PE architecture** on page 84
  Continuous Delivery for Puppet Enterprise (PE) communicates with your PE installation, your source control system, the servers you've designated as job hardware, and the browser you use to connect to the web UI and Puppet Application Manager (PAM).

• **Configure and deploy Continuous Delivery for PE** on page 87
  After installing Puppet Application Manager (PAM), configure Continuous Delivery for Puppet Enterprise (PE) and deploy the application for the first time.

• **Configure and deploy Continuous Delivery for PE in an offline environment** on page 88
  Use these instructions to install Continuous Delivery for Puppet Enterprise (PE) in an air-gapped or offline environment where the Continuous Delivery for PE host server does not have direct access to the internet.

• **Supported PE versions and browsers** on page 90
  The Continuous Delivery for Puppet Enterprise (PE) 4.x series is designed for use with currently-supported versions of PE and current versions of major web browsers.

• **Analytics data collection** on page 91
  Continuous Delivery for Puppet Enterprise (PE) automatically collects data about how you use the software. You can opt out of providing this data when installing Continuous Delivery for PE.

---

### Continuous Delivery for PE architecture

Continuous Delivery for Puppet Enterprise (PE) communicates with your PE installation, your source control system, the servers you've designated as job hardware, and the browser you use to connect to the web UI and Puppet Application Manager (PAM).

This diagram shows the architecture and port requirements for a Continuous Delivery for PE 4.x installation.

**Important:** Continuous Delivery for PE uses TCP (Transmission Control Protocol) connections.
Port | Use
---|---
443 | On this port, Continuous Delivery for PE job hardware servers communicate with the Continuous Delivery for PE application, and users access the Continuous Delivery for PE application's web UI over HTTPS.
4433 | The Continuous Delivery for PE application uses this PE port to communicate with the node classifier and the PE console (for authentication).
8000 | This is the default port where source control provider webhooks send traffic to Continuous Delivery for PE. You can change this port in PAM under Optional configuration.
8081 | The Continuous Delivery for PE application uses this PE port to send queries to PuppetDB.
8140 | The Continuous Delivery for PE application and Continuous Delivery for PE job hardware servers use this PE port to communicate with Puppet Server.
8142 | Continuous Delivery for PE job hardware servers and PE communicate through Puppet Agent on this port.
8143 | The Continuous Delivery for PE application uses this PE port to communicate with Puppet Orchestrator.
8170 | The Continuous Delivery for PE application uses this PE port to communicate with Code Manager.
8800 | PAM's web UI accepts HTTPS traffic from users on this port.

You can configure ports 4433, 8081, 8140, 8143, and 8170 in the PE integration settings.

For additional information about each Continuous Delivery for PE port's source and destination, refer to the PAM system requirements on page 34. For more information about PE ports, refer to PE documentation, such as the PE Firewall configuration diagrams.

TLS configuration

You can choose from several TLS configuration options when installing Continuous Delivery for Puppet Enterprise (PE). Select the installation architecture that best meets your security needs and limitations.

Basic installation (default configuration)

This installation architecture works for single-node clusters and multi-node clusters that utilize automatic load balancing or are set up for Ingress load balancing, such as with Google Kubernetes Engine (GKE). The webhook callback listens on port 8000. This architecture uses public certificates for machine-to-machine communication, and it's common to use TLS configured at the Ingress, whether in manual-entry, self-signed, or certificate manager form.
### Port Use

<table>
<thead>
<tr>
<th>Port</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>443</td>
<td>Continuous Delivery for PE job hardware servers communicate with the Continuous Delivery for PE application over this port, and users access the Continuous Delivery for PE web UI over HTTPS on this port.</td>
</tr>
<tr>
<td>5000</td>
<td>In offline installations, NodePort communicates with the registry on this port.</td>
</tr>
<tr>
<td>8000</td>
<td>This is the default port where source control provider webhooks send traffic to Continuous Delivery for PE. You can change this port in Puppet Application Manager (PAM) under <strong>Optional configuration</strong>.</td>
</tr>
<tr>
<td>8080</td>
<td>Ingress forwards traffic to the Continuous Delivery for PE web UI and communicates with <code>query-service</code> on this port.</td>
</tr>
<tr>
<td>9001</td>
<td>Continuous Delivery for PE job hardware servers communicate with NodePort on this port.</td>
</tr>
</tbody>
</table>

**Installation with a proxy or load balancer using enhanced TLS**

This installation architecture uses public certificates for external proxy TLS termination and internal certificates for machine-to-machine communication. The Ingress is used for routing rules to the various services within Continuous Delivery for PE. You must configure the backend and web UI endpoints separately from the Ingress hostnames.
Configure and deploy Continuous Delivery for PE

After installing Puppet Application Manager (PAM), configure Continuous Delivery for Puppet Enterprise (PE) and deploy the application for the first time.

Before you begin
Install PAM on page 45.

1. In PAM, upload your Continuous Delivery for PE license and follow the prompts to set up SSL certificates and make sure your infrastructure meets Continuous Delivery for PE system requirements.

---

Port | Use
--- | ---
443 | On this port, Continuous Delivery for PE job hardware servers communicate with the Continuous Delivery for PE application, and users access the Continuous Delivery for PE web UI over HTTPS.
5000 | In offline installations, NodePort communicates with the registry on this port.
8000 | This is the default port where source control provider webhooks send traffic to Continuous Delivery for PE. You can change this port in Puppet Application Manager (PAM) under Optional configuration.
8080 | Ingress forwards traffic to the Continuous Delivery for PE web UI and communicates with query-service on this port.
9001 | Continuous Delivery for PE job hardware servers communicate with the proxy/load balancer registry and NodePort on this port.
2. To configure your Continuous Delivery for PE installation, click **Config**, and:
   a) Enter a hostname for the Continuous Delivery for PE installation.
   b) Set a password for the Continuous Delivery for PE root user account.
   c) Select where to host the webhook (for source control integration) and local container registry (in Puppet-supported offline installs) endpoints.
      - **Use a Nodeport**: All Kubernetes hosts listen on this NodePort and forward traffic to the appropriate endpoint.
      - **Use an Ingress with a hostname**: The endpoint listens on ports 80 (HTTP) and 443 (HTTPS) for requests using the specified hostname.
   d) Configure any other settings on this page relevant to your installation, such as external databases, customized endpoints, a load balancer, or TLS certificates.
   e) When you've finished editing the configuration, click **Save config**. Saving new configuration settings initiates creation of a new Continuous Delivery for PE version.

3. Click **Go to new version**, which redirects you to the **Version history** tab, and locate the new version in the **All versions** section of the page.

4. Monitor the new version's preflight checks. The **Running Checks** indicator is visible while Continuous Delivery for PE checks your system to make sure your cluster meets minimum system requirements. When the preflight checks are complete:
   - If the status is **Checks Failed**, click **View preflights**, correct the issues, and click **Re-run**. Repeat this step as needed.

   **Important**: Do not move on until all preflight checks pass.
   - If the status is **Ready to Deploy**, move on to the next step.

5. Once the version is ready to deploy, click **Deploy** and monitor the application's status on the **Application** tab. During deployment, the application is in **Missing** status for several minutes. You can run `kubectl get pods --watch` to monitor the deployment's progress.

   When deployment is complete, the application's status changes to **Ready**.

6. Navigate to `https://<HOSTNAME>` (using the hostname you entered on the **Config** screen) and sign into Continuous Delivery for PE.

   If you're upgrading to the 4.x series from a version in the 3.x series, see **Migrating 3.x data to 4.x**.

   If you're new to Continuous Delivery for PE, see **Getting started with Continuous Delivery for PE** on page 21 to learn about the software's core features and workflows.

---

**Configure and deploy Continuous Delivery for PE in an offline environment**

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

**Before you begin**

Install Puppet Application Manager (PAM).

1. In PAM, upload your Continuous Delivery for PE license and follow the prompts to set up SSL certificates and make sure your infrastructure meets Continuous Delivery for PE system requirements.
2. When prompted, upload an `.airgap` bundle for the most recent version of Continuous Delivery for PE. These are the available bundles:

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Airgap bundle</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.11.5</td>
<td>22 February 2022</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.11.5.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.11.5.airgap</a></td>
</tr>
<tr>
<td>4.7.0</td>
<td>8 July 2021</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.7.0.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.7.0.airgap</a></td>
</tr>
</tbody>
</table>
3. To configure your Continuous Delivery for PE installation, click **Config**, and:
   a) Enter a hostname for the Continuous Delivery for PE installation.
   b) Set a password for the Continuous Delivery for PE root user account.
   c) Select where to host the webhook (for source control integration) and local container registry (in Puppet-supported offline installs) endpoints.
      - **Use a Nodeport**: All Kubernetes hosts listen on this NodePort and forward traffic to the appropriate endpoint.
      - **Use an Ingress with a hostname**: The endpoint listens on ports 80 (HTTP) and 443 (HTTPS) for requests using the specified hostname.
   d) Configure any other settings on the page relevant to your installation, such as external databases, customized endpoints, a load balancer, or TLS certificates.
   e) When you've finished editing the configuration, click **Save config**. Saving new configuration settings initiates creation of a new Continuous Delivery for PE version.

4. Click **Go to new version**, which redirects you to the **Version history** tab, and locate the new version in the **All versions** section of the page.

5. Monitor the new version's preflight checks. The **Running Checks** indicator is visible while Continuous Delivery for PE checks your system to make sure your cluster meets minimum system requirements. When the preflight checks are complete:
   - If the status is **Checks Failed**, click **View preflights**, correct the issues, and click **Re-run**. Repeat this step as needed.
   - If the status is **Ready to Deploy**, move on to the next step.

6. Once the version is ready to deploy, click **Deploy** and monitor the application's status on the **Application** tab.
   During deployment, the application is in **Missing** status for several minutes. You can run `kubectl get pods --watch` to monitor the deployment's progress.
   When deployment is complete, the application's status changes to **Ready**.

7. Navigate to `https://<HOSTNAME>` (using the hostname you entered on the **Config** screen) and sign into Continuous Delivery for PE.
   If you're upgrading to the 4.x series from a version in the 3.x series, see **Migrating 3.x data to 4.x**.
   If you're new to Continuous Delivery for PE, see **Getting started with Continuous Delivery for PE** on page 21 to learn about the software's core features and workflows.

---

**Supported PE versions and browsers**

The Continuous Delivery for Puppet Enterprise (PE) 4.x series is designed for use with currently-supported versions of PE and current versions of major web browsers.

**Supported PE versions**

You can use Continuous Delivery for Puppet Enterprise (PE) with these PE versions.

- 2021.4
- 2021.3
- 2021.2
- 2021.1
- 2019.8.x (LTS)

**Important**: You must upgrade to Continuous Delivery for PE version 4.8.2 before installing PE version 2021.3, 2019.8.8, or newer versions in their respective release series. Version 4.8.2 resolves a PuppetDB issue that prevented generating new fact charts on the **Nodes** page.
You can learn more about PE versions on the Puppet Enterprise support lifecycle page.

**Supported browsers**

You can use the Continuous Delivery for PE web UI with these browsers.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Supported versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Chrome</td>
<td>Current version as of release</td>
</tr>
<tr>
<td>Mozilla Firefox</td>
<td>Current version as of release</td>
</tr>
<tr>
<td>Microsoft Edge</td>
<td>Current version as of release</td>
</tr>
<tr>
<td>Apple Safari</td>
<td>Current version as of release</td>
</tr>
</tbody>
</table>

**Analytics data collection**

Continuous Delivery for Puppet Enterprise (PE) automatically collects data about how you use the software. You can opt out of providing this data when installing Continuous Delivery for PE.

Continuous Delivery for PE collects analytics data in order to better understand how our customers use the software. For example, knowing how many control repos you manage helps us develop more realistic product testing, and learning which source control systems are most and least popular helps us decide where to prioritize new functionality.

**What data does Continuous Delivery for PE collect?**

Continuous Delivery for Puppet Enterprise (PE) collects analytics data when you use the software. It does not collect any personally identifiable information (PII). The data we collect is never used or shared outside Puppet, Inc.

Continuous Delivery for PE collects the following data when you start Continuous Delivery for PE for the first time, when Continuous Delivery for PE restarts after an upgrade, and weekly thereafter:

- License UUID
- For each active user account:
  - Number of control repos
  - Number of control repo pipelines
  - Number of control repo pipelines with impact analysis enabled
  - Number of modules
  - The deployment policy selected for each pipeline deployment
  - Configured integrations (PE and source control systems)

Continuous Delivery for PE collects the following data while you use Continuous Delivery for PE:

- Pageviews
- Progress through the initial installation and setup process
- Progress through the integration configuration process (PE and source control systems)

Continuous Delivery for PE does not collect:

- Any PII about a user, such as name, email address, password, or company name.
- User inputs such as usernames, control repo names, module names, job names, or job hardware information.

**Opt out of analytics data collection**

You can opt out of Continuous Delivery for Puppet Enterprise (PE) analytics data collection on the Puppet Application Manager (PAM) configuration settings page.

1. In PAM, click **Config**.
2. In the **Required setup** section of the page, deselect **Report analytics data to Puppet**.
3. Scroll to the bottom of the page and click **Save config**.
4. When the success message appears, click **Go to new version**.
5. On the **Version history** page, click **Deploy**.

You have opted out of data collection, and your instance of Continuous Delivery for PE no longer sends analytics data to Puppet.

## Migrating 3.x data to 4.x

The Continuous Delivery for PE 4.x series runs on a managed Kubernetes cluster, so you must migrate your 3.x series data to the new installation.

The migration process has four main steps:

1. Upgrade your existing 3.x installation to version 3.13.8.
2. Create a new installation of Continuous Delivery for PE version 4.x.
3. From the version 3.x root console, run the migration task.
4. Restart the 4.x installation, which is now populated with your 3.x data.

### What is migrated?

The migration task **copies** and migrates to 4.x:

- The database, which contains all your Continuous Delivery for PE 3.x installation’s information, users, integrations, history, and artifacts (except job commands and job logs).

  **Note:** If you configured your 4.x installation to use an externally managed PostgreSQL database, this step is skipped.

- The object store, which contains all of your 3.x installation’s job commands and job logs.

  **Note:** The 4.x series replaces external Artifactory and Amazon S3 object storage with a built-in highly available object storage system. Your 3.x object store is migrated to the built-in 4x object store.

- Any of these configuration settings, if in use on 3.x:

  ```
  CD4PE_REPO_CACHE_RETRIEVAL_TIMEOUT_MINUTES
  CD4PE_LDAP_GROUP_SEARCH_SIZE_LIMIT
  CD4PE_ENABLE_REPO_CACHING
  CD4PE_HTTP_CONNECTION_TIMEOUT_SEC
  CD4PE_HTTP_READ_TIMEOUT_SEC
  CD4PE_HTTP_WRITE_TIMEOUT_SEC
  CD4PE_HTTP_REQUEST_TIMEOUT_SEC
  CD4PE_INCLUDE_GIT_HISTORY_FOR_CD4PE_JOBS
  CD4PE_JOB_GLOBAL_TIMEOUT_MINUTES
  CD4PE_JOB_HTTP_READ_TIMEOUT_MINUTES
  ```

The migration task **updates** in the 4.x installation:

- The webhooks between your source control system and the Continuous Delivery for PE backend endpoint URL.

  **Note:** This step is optional when running the migration task. The migration task is designed safely run multiple times, so you can make sure your 4.x installation is ready for full-time use before you enable webhook updates.

The migration task **does not copy** or move to 4.x:

- The 3.x root user, because you create a new root user when installing 4.x.
- The **JVM_ARGS** configuration setting, if in use on 3.x.
These backup tables created for you by Continuous Delivery for PE during the 3.x series:

- app-pipelines-3-0-backup.pfi, which was created during the upgrade from 2.x to 3.x
- pe-ia-node-results-cve-2020-7944-backup.pfi and pe-ia-resource-results-cve-2020-7944-backup.pfi, which were created as part of the CVE 2020-7944 remediation in version 3.4.0

Migration cautions

A super user or the root user must perform this process.

Migrating from a local database in a 3.x installation to an external database in a 4.x installation is not supported.

Important: Once you initiate the migration task, any new data created in the 3.x installation may not be migrated to the 4.x installation. To avoid losing data, restrict access to Continuous Delivery for PE during migration.

Run the migration task

The migration task uses temporary credentials to establish a connection between your 3.x and 4.x installations, then migrates your 3.x installation data to your new 4.x installation.

Before you begin

1. **Upgrade your 3.x installation to version 3.13.8.** The migration task fails if you run it on any other 3.x version.
2. Create a new 4.x installation.
   a. Review the Puppet Application Manager (PAM) system requirements.
   b. Follow the instructions for your environment to Install PAM on page 45.
   c. Depending on your environment, either Configure and deploy Continuous Delivery for Puppet Enterprise (PE), or Configure and deploy Continuous Delivery for PE in an offline environment on page 88.

   **CAUTION:** Do not configure integrations or create new data in the 4.x installation until you complete migration.

3. Make sure your 3.x installation can reach the Kubernetes API running on your 4.x installation.
4. If your 4.x installation is behind a proxy, set HTTP_PROXY and HTTPS_PROXY as values for the cd4pe_docker_extra_params parameter on the cd4pe class for your 3.x installation, passing in the URL of your proxy.

   **Note:** To pass additional arguments to the Docker process running the Continuous Delivery for PE container, you must specify them as an array. For example: 
   
   ```
   -e HTTPS_PROXY=https://proxy.example.com, -e HTTP_PROXY=https://proxy.example.com
   ```

1. In PAM, click **Config**.
2. In the Migration area, select the **We're migrating an existing Continuous Delivery for PE 3.x instance** option. Scroll to the bottom of the page and click **Save config**.

   Enabling this setting allows data to move between your 3.x and 4.x installations.
3. When the success message appears, click **Go to new version**. On the Version history page, click **Deploy**.
4. To generate temporary credentials to connect your two installations, run the following script on the server where you installed 4.x. If necessary, change the value of the CD4PE_NAMESPACE variable to your installation's namespace. Most installations use the default namespace.

```bash
#!/bin/bash
C_DEFAULT="\e[0m"
C_GREEN="\e[0;32m"

CD4PE_NAMESPACE="default"

K8S_SERVER_URL="$(kubectl config view --minify -o jsonpath='{.clusters[*].cluster.server}')"
K8S_TOKEN="$(kubectl -n $CD4PE_NAMESPACE get secret $(kubectl -n $CD4PE_NAMESPACE get secrets | grep ^cd4pe-migration | cut -f1 -d ' ') -o jsonpath='{["data"]["token"]}')"
if [ -z "${K8S_TOKEN}" ]; then
    printf "Kubernetes token not found. Check to ensure 'Enable migration of an existing CD4PE instance' is checked in the application configuration"
    exit -1
fi

K8S_CA_CERT="$(kubectl config view --raw --minify -o jsonpath='{.clusters[*].cluster.certificate-authority-data}')"

printf "$C_GREEN"Kubernetes API server URL:$(C_DEFAULT)\n\n"${K8S_SERVER_URL}"

printf "$C_GREEN"Kubernetes service account token:$(C_DEFAULT)\n\n"${K8S_TOKEN}"

printf "$C_GREEN"Kubernetes API CA certificate:$(C_DEFAULT)\n\n"${K8S_CA_CERT}"
```

5. In the Continuous Delivery for PE 3.x root console, go to Settings > Migration. Copy the temporary credentials you generated in the previous step into the appropriate fields.

6. Click Start migration and confirm your settings.

7. Monitor the status of the migration task in the 3.x Docker logs. Run docker logs cd4pe to access the logs. The migration task first migrates your database tables, then migrates your object store, and lastly updates your webhooks, if applicable. When complete, Ran migration task in <elapsed time> prints to the log.

8. Redeploy the 4.x installation. In PAM, click Config.

9. In the Migration area, deselect the We're migrating an existing Continuous Delivery for PE 3.x instance option. Click Save config.

10. When the success message appears, click Go to new version. On the Version history page, locate the new version and click Deploy.

11. Go to your 4.x installation, which now is populated with migrated data, and log in with your 3.x user credentials (not the root account).

**Test the new 4.x installation**

You must thoroughly test the 4.x installation to make sure the data was properly migrated and all functionality is working as expected.
1. Test the data migration to make sure your 3.x installation's data is present in your 4.x installation. For each workspace in your 4.x installation:
   a) Review event logs for all control repos and modules.
   b) Check that all jobs and hardware connections are present.
   c) Check that all users and user groups are present.
2. If any data is missing or looks incorrect, rerun the migration task.
3. Test manual pipeline runs to make sure your 4.x installation is functioning properly. For each workspace in your 4.x installation, trigger a manual pipeline run for all control repos and modules. To do this, select Trigger pipeline from the Manual actions menu above each pipeline.

Once you've confirmed your 3.x data is present in your 4.x installation and all manual pipeline runs succeed, Update webhooks on page 95.

### Update webhooks

As the final step in the 3.x to 4.x data migration process, update the webhooks that connect Continuous Delivery for Puppet Enterprise (PE) to your source control system.

Follow these steps if you're updating webhooks after migrating data from 3.x to 4.x. There are separate instructions to Update webhooks on page 112 in other scenarios.

1. In the Continuous Delivery for PE 3.x root console, navigate to Settings > Endpoints and copy the backend service endpoint.
2. In the Continuous Delivery for PE 4.x root console, click Settings > Webhooks, and enter the backend service endpoint you copied from the 3.x root console.
3. Click Update webhooks.

Continuous Delivery for PE updates all webhooks to point to the current installation.

⚠️ **CAUTION:** After migrating data from 3.x to 4.x, once you update webhooks to point to your 4.x installation, do not create new data in the 3.x installation, and do not run the migration task again.

### Post-migration guidance and troubleshooting

After migrating Continuous Delivery for Puppet Enterprise (PE) data from 3.x to 4.x, thoroughly testing your 4.x installation, and updating webhooks, you're ready to begin using 4.x with your team. However, do not immediately decommission your 3.x installation. You may need to roll back to 3.x if issues arise.

**Important:** Keep your 3.x installation intact until you are fully confident that all features of the new 4.x installation are working as expected. **As a precaution, we strongly advise maintaining your inactive 3.x installation for the first two months of using 4.x.**

Maintaining your dormant 3.x installation for an extended test period allows you to roll back to 3.x and prevent production environment disruptions if an issue arises with your 4.x migration.

### Rolling back to 3.x

If necessary, you can temporarily roll back to your 3.x installation by switching the active webhooks from the 4.x installation to the 3.x installation and generating new access tokens for all PE instances connected with the 3.x installation.

1. In your Continuous Delivery for PE 3.x installation's web UI, navigate to a control repo's pipeline and click Manage pipelines.
2. Select Manage in the web UI and locate the Automation webhook section where the webhook (including token) for your control repo is shown.
3. Copy the webhook and navigate to the corresponding repository in your source control system. Add the 3.x webhook to the repository and remove the 4.x webhook.

4. Repeat this process for each control repo and module repo in your 3.x installation.

5. Generate a new access token for each PE instance connected to the 3.x installation. In the Continuous Delivery for PE web UI, navigate to Settings > Puppet Enterprise and click Edit credentials.

6. Select Basic authorization or API token and enter the required information:

   • For Basic authorization, enter the username and password for your "Continuous Delivery" user. Continuous Delivery for PE uses this information to generate an API token for you. The username and password are not saved.
   • For API token, generate a PE access token for your "Continuous Delivery" user using puppet-access or the RBAC API v1 service, and paste it into the API token field. For information about generating PE access tokens and the RBAC API v1 service, see Token-based authentication in the PE documentation.

   Note: To avoid unintended service interruptions, create access tokens with a multi-year lifespans.

Once the 3.x webhooks and new PE access tokens are in place, you can resume using the 3.x installation.

When you're ready to return to 4.x, repeat the migration process, starting with Run the migration task on page 93.

Upgrading Continuous Delivery for PE

We regularly release new versions of Continuous Delivery for Puppet Enterprise (PE). Upgrading to the current version ensures you have the latest features, fixes, and improvements.

Upgrade paths

These are valid upgrade paths for Continuous Delivery for Puppet Enterprise (PE).

<table>
<thead>
<tr>
<th>If you're on version...</th>
<th>Upgrade to version...</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.13.0</td>
<td>You're up to date!</td>
<td></td>
</tr>
<tr>
<td>4.x (any version in the series)</td>
<td>4.13.0 (current version)</td>
<td>You can safely upgrade from any 4.x version to any newer 4.x version.</td>
</tr>
<tr>
<td>3.13.8</td>
<td>4.x (any version in the series)</td>
<td>You must upgrade to the latest 3.13.x version in order to access the 3.x to 4.x data migration tool, and then you can upgrade past 3.13.x.</td>
</tr>
<tr>
<td>3.13.3 or earlier</td>
<td>3.13.8</td>
<td></td>
</tr>
</tbody>
</table>

Upgrade Continuous Delivery for PE

Check for, download, and deploy updates from the Version history tab in Puppet Application Manager (PAM). These instructions apply to Continuous Delivery for Puppet Enterprise (PE) installations in online environments.

1. In PAM, click Version history.
2. Click Check for updates.

Tip: You can click Configure automatic updates to automatically check for updates hourly, every four hours, daily, weekly, or at a custom interval.
3. If there is an update, PAM downloads it and performs preflight checks to make sure your cluster meets system requirements for the new version. Click View preflight to review the results.

4. When you're ready to upgrade to the new version of Continuous Delivery for PE, click Deploy.

When deployment is complete, sign in to Continuous Delivery for PE and verify that the new version number is shown in the bottom left corner of the web UI.

**Upgrade Continuous Delivery for PE in an offline environment**

Users operating in environments without direct access to the internet must download and install an airgap bundle to upgrade to the latest version of Continuous Delivery for Puppet Enterprise (PE).
1. Download the `.airgap` bundle for the latest version (or the version you want to upgrade to, depending on your upgrade path). These are the available bundles:

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Airgap bundle</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.11.5</td>
<td>22 February 2022</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.11.5.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.11.5.airgap</a></td>
</tr>
<tr>
<td>4.7.0</td>
<td>8 July 2021</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.7.0.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.7.0.airgap</a></td>
</tr>
</tbody>
</table>
2. In Puppet Application Manager (PAM), click **Version history** > **Upload new version**, and upload the .airgap bundle. The upload can take several minutes.

3. When you're ready to upgrade to the new version of Continuous Delivery for PE, click **Deploy**.

When deployment is complete, sign into Continuous Delivery for PE and verify that the new version number is shown in the bottom left corner of the web UI.

---

**Upgrade an automated installation**

If you used automated installation to install Continuous Delivery for Puppet Enterprise (PE), run a script to upgrade to the latest version.

1. From the command line of your primary (control plane) node, run the upgrade script:
   
   ```
   kubectl kots upstream upgrade cd4pe --namespace default --deploy
   ```

2. Wait five minutes to allow the software time to process the change.

   
   If preflight checks passed, the upgraded application is deployed and in the process of starting up. Run `kubectl get pods --watch` to monitor the deployment's progress.

When the deployment is complete, sign into Continuous Delivery for PE and verify that the new version number is shown in the bottom left corner of the web UI.

---

**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 Puppet Enterprise** on page 100
  
  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 104
  
  Impact analysis is a Continuous Delivery for Puppet Enterprise (PE) tool that shows you the potential impact that new Puppet code can have on your PE-managed infrastructure, without actually merging the new code. When you add impact analysis to a control repo's pipeline, Continuous Delivery for PE automatically generates a report on every proposed code change to that control repo.

- **Integrate with source control** on page 107
  
  Integrate your source control system with Continuous Delivery for Puppet Enterprise (PE) by following the instructions for your source control provider.

- **Configure job hardware** on page 112
  
  The servers Continuous Delivery for Puppet Enterprise (PE) uses to run tests on your Puppet code are known as job hardware. You must configure job hardware before you can run tests or use pipelines.

- **Add repositories** on page 115
  
  You must tell Continuous Delivery for Puppet Enterprise (PE) which repositories in your source control system are your control repos and module repos. This tells Continuous Delivery for PE where to listen for code changes.

- **Configure SAML** on page 116
  
  Continuous Delivery for Puppet Enterprise (PE) supports the use of Security Assertion Markup Language (SAML) authentication from a SAML identity provider (IDP). Once you configure your SAML IDP to integrate with Continuous Delivery for PE, you can use your chosen single sign-on tool to authenticate users to Continuous Delivery for PE.
Continuous Delivery for Puppet Enterprise (PE) supports use of the Lightweight Directory Access Protocol (LDAP) for managing user authentication. After configuring LDAP, use group mapping to associate your existing LDAP groups with role-based access control (RBAC) groups in Continuous Delivery for PE.

Configure SMTP on page 119
Configure SMTP for your Continuous Delivery for Puppet Enterprise (PE) installation so that users can receive email notifications from the software.

Set up external PostgreSQL on page 120
To use an external PostgreSQL database with Continuous Delivery for Puppet Enterprise (PE), set up the connection on the Config page in Puppet Application Manager (PAM).

Advanced Continuous Delivery for PE configuration on page 120
Advanced configuration settings for Continuous Delivery for PE help you fine-tune aspects of the software that can impact runtime and operation speed.

Integrate with Puppet Enterprise
To set up an integration between your Puppet Enterprise (PE) instance and Continuous Delivery for PE, you must first set up a dedicated PE user with appropriate permissions, then add your PE instance's credentials to Continuous Delivery for PE.

Before you begin
You must enable Code Manager on the PE instance before integrating with Continuous Delivery for PE. For instructions, see Configure Code Manager.

The Puppet primary server certificate must have dns_alt_names configured. To confirm your certificate configuration, run:

```
openssl x509 -in $(puppet config print hostcert) -text | grep -A 1 "Subject Alternative Name"
```

If this returns no output, you must regenerate the primary server certificate before integrating PE with Continuous Delivery for PE. For instructions, go to Regenerate primary server certificates in the PE documentation.

Note: If your PE instance has a replica configured for disaster recovery, Continuous Delivery for PE is not available if a partial failover occurs. Learn more at What happens during failovers in the PE documentation. To restore Continuous Delivery for PE functionality, you must promote the replica to primary server.

Create a Continuous Delivery user and user role in PE
Create a "Continuous Delivery" user and user role in PE. You'll use this to generate the PE authentication token required during the setup process and to view a centralized log of the activities Continuous Delivery for PE performs on your behalf.

1. In the PE console, click Access control > Users.
2. Enter a full name (such as Continuous Delivery User) and a login name (such as cdpe_user) and click Add local user.
3. Next, create a user role containing the permissions the Continuous Delivery user needs when operating Continuous Delivery for PE. In the PE console, click Access control > User roles.
4. Enter a name and an optional description for new role, such as CDPE User Role. Then, click Add role.
5. Select the new user role from the list on the User roles page.
6. Click **Permissions**, and assign these permissions to the user role:

<table>
<thead>
<tr>
<th>Type</th>
<th>Permission</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job orchestrator</td>
<td>Start, stop, and view jobs</td>
<td>-</td>
</tr>
<tr>
<td>Node groups</td>
<td>Create, edit, and delete child groups</td>
<td>All</td>
</tr>
<tr>
<td>Node groups</td>
<td>View</td>
<td>All</td>
</tr>
<tr>
<td>Node groups</td>
<td>Edit configuration data</td>
<td>All</td>
</tr>
<tr>
<td>Node groups</td>
<td>Set environment</td>
<td>All</td>
</tr>
<tr>
<td>Nodes</td>
<td>View node data from PuppetDB</td>
<td>-</td>
</tr>
<tr>
<td>Puppet agent</td>
<td>Run Puppet on agent nodes</td>
<td>-</td>
</tr>
<tr>
<td>Puppet environment</td>
<td>Deploy code</td>
<td>All</td>
</tr>
<tr>
<td>Puppet Server</td>
<td>Compile catalogs for remote nodes</td>
<td>-</td>
</tr>
<tr>
<td>Tasks</td>
<td>Run tasks</td>
<td>At minimum, $\text{cd4pe_jobs::run_cd4pe_job}$, which is required when running job hardware using a Puppet agent.</td>
</tr>
</tbody>
</table>

7. Once all permissions are 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, click **Add user**, and then commit your change.

9. Create a password for the Continuous Delivery user.
   a) Return to the **Users** page.
   b) Find and click the full name of your Continuous Delivery user, and then click **Generate password reset**.
   c) Follow the password reset link and create a password for the user. You'll use this password when adding your PE credentials to Continuous Delivery for PE.

Your Continuous Delivery user is now configured, has a password, and has the permissions needed to operate Continuous Delivery for PE.

**Add your Puppet Enterprise credentials**

Establishing an integration with your Puppet Enterprise (PE) instance allows Continuous Delivery for PE to work with PE tools, such as Code Manager and the orchestrator service, to deploy Puppet code changes to your nodes.

If necessary, you can add multiple PE instances to your Continuous Delivery for PE installation.

1. Make sure you are signed in to your individual Continuous Delivery for PE user account.

   **Important:** Do not perform these steps as the root user. Sign into your individual Continuous Delivery for PE user account before proceeding.

2. In the Continuous Delivery for PE web UI, click **Settings**.

3. Click **Puppet Enterprise**, and click **Add new credentials**.

4. In the **New Puppet Enterprise credentials** window, 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.

5. Enter the fully qualified domain name (FQDN) you use to access the PE console, such as \$\text{sample.pe.instance}\$. The FQDN must match the certname of your PE primary server or an alias included in the \$\text{dns\_alt\_names}\$ entry in your puppet.conf file.

© 2021 Puppet, Inc.
6. Select **Basic authorization** or **API token** and enter the required information.

**Important:** If SAML is enabled for your Continuous Delivery for PE installation, you must select **API token**.

- **Basic authorization:** Enter the username and password for your "Continuous Delivery" user. Continuous Delivery for PE uses this information to generate an API token for you. The username and password are not saved. Optionally, click **Change** to change the token's lifetime.
- **API token:** Use `puppet-access` or the RBAC v1 API to generate a PE access token for your "Continuous Delivery" user, as explained in **Token-based authentication**. Paste the token into the **API token** field.

**Tip:** To avoid unintended service interruptions, create an access token with a multi-year lifespan.

7. Click **Save changes**.

Continuous Delivery for PE uses the information you provide to access the primary SSL certificate generated during PE installation and to look up the endpoints for PuppetDB, Code Manager, orchestrator, and node classifier.

**Note:** If Continuous Delivery for PE cannot automatically locate the endpoints for your PE services, you are directed to a new page where you can manually enter these credentials. If this happens, go to **Manually configure a Puppet Enterprise integration**.

Your PE instance is now integrated with Continuous Delivery for PE. Click **Edit credentials** to view the integration information.

To enable impact analysis for this instance, see **Configure impact analysis**.

### Create environment node groups

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

Continuous Delivery for PE deploys changes to environment node groups. Setting up environment node groups allows you to define groups of nodes that you can choose to deploy changes to.

1. In the PE console, click **Node groups** (or **Classification** in PE versions prior to 2019.8.1).
2. If your node classification does not already have an **All Environments** node group, create one.

**Note:** The **All Environments** node group is created automatically in new installations of PE versions 2018.1.5 and newer. If you're running an older version or have upgraded your PE instance, you must manually create this node group.

Click **Add group...** and use these specifications:

- **Parent name:** All Nodes
- **Group name:** All Environments
- **Environment:** production
- **Environment group:** yes
- **Description:** Environment group parent and default

3. Open the **All Environments** node group and add a new rule with these specifications:

- **Fact:** name
- **Operator:** ~
- **Value:** .*

4. Edit the Agent-specified environment node group so that **All Environments** is its parent. The Agent-specified environment node 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 the Production environment node group's 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 these specifications:
      • Parent name: All Environments
      • Group name: <ENVIRONMENT_NAME>
      • Environment: <ENVIRONMENT>
      • Environment group: Yes
   d) Associate the relevant nodes with the environment group by creating rules or pinning nodes.
      Follow these best practices for associating nodes with environment node groups:
      • Use the `pp_environment` trusted fact, or a similar custom fact, to define which environment each node belongs to. Write a rule in each environment group that uses `pp_environment` or your custom fact to match nodes.
      • See if you can use other facts or trusted facts to create rules that match nodes to one, and only one, environment group.
      • If you can't use trusted facts, custom facts, or other facts to determine node environments, use pinning. Pin each node to only one environment group.
   e) Specify the Git branch corresponding to the environment.

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.
   a) Use these specifications for each child environment group:
      • 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
   b) After setting up the child environment node group, give it the rule `(agent_specified_environment ~ .+)`.
   c) Do not pin any nodes to child environment node groups.

The resulting environment node groups hierarchy is similar to:
• All Nodes
  • All Environments
    • Agent-specified environment
    • Production environment
      • Production one-time run exception (if you chose to configure this)
    • Staging environment
      • Staging one-time run exception (if you chose to configure this)
    • Test environment
      • Test one-time run exception (if you chose to configure this)
After configuring environment node groups, you can deploy new code to your nodes.

**Configure impact analysis**

Impact analysis is a Continuous Delivery for Puppet Enterprise (PE) tool that shows you the potential impact that new Puppet code can have on your PE-managed infrastructure, without actually merging the new code. When you add impact analysis to a control repo’s pipeline, Continuous Delivery for PE automatically generates a report on every proposed code change to that control repo.

**Configuration process and preparation**

To configure impact analysis, you must set the destination server for impact analysis tasks, install the `puppetlabs-cd4pe` module, and make a classification update.

Before configuring impact analysis, you must:

- Install Continuous Delivery for PE
- Integrate with your source control system
- Integrate with Puppet Enterprise

**Set the impact analysis destination**

When you integrate a new Puppet Enterprise (PE) instance with Continuous Delivery for PE, impact analysis tasks are automatically configured to run on the primary server. In many cases, it’s better to change the impact analysis task destination to a different server, such as a compiler or load balancer, to preserve the primary server’s compile capacity.

**Before you begin**

Determine a place in your PE infrastructure to direct impact analysis tasks:

- **Best:** A load balancer managing a pool of compilers.
- **Good:** An individual compiler dedicated to impact analysis tasks.
- **For small installations only:** The PE primary server.

**Note:** The primary server is the default configuration for running impact analysis tasks, but it is best suited to PE installations with limited node counts using the standard installation architecture. Impact analysis tasks can easily overrun the primary server’s limited compile capacity in a larger distributed architecture.
If you choose to run impact analysis tasks on the primary server, skip to Install modules on page 105.

To run impact analysis tasks on a load balancer or compiler:

1. In the Continuous Delivery for PE web UI, navigate to Settings > Puppet Enterprise and locate the PE instance you are configuring.
2. Click Edit credentials and expand the Impact analysis credentials section of the page.
3. In the Puppet Server service field, enter the hostname of the compiler or load balancer at :8140 (for example, loadbalancer.example.com:8140).
4. Optional: Set the environments path and adjust the maximum number of catalogs that Continuous Delivery for PE is able to simultaneously compile, as explained in Optional impact analysis settings.
5. Click Save Changes.

Next, Install modules on page 105.

Optional impact analysis settings

To adjust these optional settings, navigate to your Puppet Enterprise (PE) instance credentials in the Settings area and expand the Impact analysis credentials section.

Environments path

CAUTION: Before setting the environments path in your impact analysis credentials, read this section and decide if it is necessary.

Adding the environments path for your PE instance to your impact analysis credentials allows Continuous Delivery for PE to perform exact file matching. This can improve the performance of impact analysis tasks in some cases. However, whether you can use the environments path depends on your PE instance configuration.

• Add the environments path if your Puppet environments path is a single path at a known location, such as /etc/puppetlabs/code/environments.
• DO NOT add the environments path if your PE instance uses lockless code deploys to write code to versioned directories. The environments path setting is not compatible with lockless code deploys.

Concurrent catalog compilations

This setting sets the maximum number of catalog compilations Continuous Delivery for PE is permitted to perform simultaneously. By default, Continuous Delivery for PE performs up to 10 concurrent catalog compilations.

Tune this number to fit the needs and limitations of your installation. A lower number preserves processing capacity, while a higher number reduces the time Continuous Delivery for PE spends completing each impact analysis task.

Install modules

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

The puppetlabs-cd4pe module requires six dependent modules and a module to manage job hardware. The following table lists the required modules, required versions for each module, and links to the Forge page for each module.

<table>
<thead>
<tr>
<th>Module</th>
<th>Required version</th>
</tr>
</thead>
<tbody>
<tr>
<td>puppetlabs-cd4pe</td>
<td>3.1.0 or later in the 3.x series</td>
</tr>
<tr>
<td>puppetlabs-stdlib</td>
<td>4.19.0 or later</td>
</tr>
<tr>
<td>puppetlabs-puppet_authorization</td>
<td>0.5.0 or later</td>
</tr>
<tr>
<td>puppetlabs-hocon</td>
<td>0.9.3 or later</td>
</tr>
<tr>
<td>puppetlabs-concat</td>
<td>2.1.0 or later</td>
</tr>
</tbody>
</table>

© 2021 Puppet, Inc.
### Module Requirements

<table>
<thead>
<tr>
<th>Module</th>
<th>Required version</th>
</tr>
</thead>
<tbody>
<tr>
<td>puppetlabs-apt</td>
<td>4.4.1 or later</td>
</tr>
<tr>
<td>puppetlabs-translate</td>
<td>1.1.0 or later</td>
</tr>
<tr>
<td>puppetlabs-cd4pe_jobs</td>
<td>Minimum: 1.4.0 or later</td>
</tr>
<tr>
<td></td>
<td>1.6.0 or later is required to use secrets with jobs</td>
</tr>
</tbody>
</table>

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

   A sample Puppetfile entry:

   ```
   mod 'puppetlabs-cd4pe', '3.2.1'
   # Requirements for cd4pe
   mod 'puppetlabs-stdlib', '8.1.0'
   mod 'puppetlabs-puppet_authorization', '0.5.1'
   mod 'puppetlabs-hocon', '1.1.0'
   mod 'puppetlabs-concat', '7.1.1'
   mod 'puppetlabs-apt', '8.3.0'
   mod 'puppetlabs-translate', '2.2.0'
   mod 'puppetlabs-cd4pe_jobs', '1.6.0'
   ```

2. If you configured Continuous Delivery for PE to automatically deploy code changes to the relevant environments, Continuous Delivery for PE deploys the updated code for you. If you have not configured automatic code deployments, run `puppet code deploy <ENVIRONMENT>` to deploy the updated code to the relevant environments.

Next, Update classification on page 106.

### Update classification

Once `puppetlabs-cdpe` and its dependencies are deployed, you must update classification of your nodes.

1. In the PE console, click Node groups (or Classification in PE versions prior to 2019.8.1) and open the PE Infrastructure group.
2. Select the PE Master group and click Classes (or Configuration in PE versions prior to 2019.8.1).
3. In the Add new class field, select `cd4pe::impact_analysis` and click Add class, then commit your change.

   If you don't find `cd4pe::impact_analysis` in the class list, click Refresh to update class definitions.
4. Run Puppet on the nodes in the PE Master group.

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

5. Optional: Set the maximum number of concurrent node catalog compiles allowed for each workspace. By default, 10 concurrent catalog compilations are allowed.

   Adjust this setting to manage the catalog compilation load placed on a PE instance by Continuous Delivery for PE impact analysis report generation.

   a) In the Continuous Delivery for PE web UI, click Settings.

   b) Click Puppet Enterprise, locate the PE instance you’ve configured to use impact analysis, and click Edit credentials.

   c) Locate the Impact analysis credentials section. In the Max concurrent catalog compiles field, set the number of catalog compilations that are allowed to run simultaneously on this PE instance.

   d) Click Save changes.

To get started using impact analysis, see Analyzing the impact of code changes.
Integrate with source control

Integrate your source control system with Continuous Delivery for Puppet Enterprise (PE) by following the instructions for your source control provider.

Status notification prefixes for source control

Once you've integrated your Continuous Delivery for PE installation with your source control provider, Continuous Delivery for PE sends information about the outcome of each stage of each pipeline run to your source control provider.

When reporting to your source control provider, Continuous Delivery for PE uses the following default format to label each pipeline stage: `cd-pe/stage-<pipeline stage number>`

This labeling system is adequate if you connect a control repo or module repo to one (and only one) workspace. But if you connect multiple workspaces to a certain control repo or module repo, your source control system might receive identical notifications from multiple workspaces about multiple pipelines. The source control system can't differentiate between these identical notifications when performing automated testing.

To prevent this issue, you can add a status notification prefix to all communications Continuous Delivery for PE sends from your workspace to your source control provider. With the status notification prefix, your source control system can differentiate between, and accurately act on, pipeline status notifications coming from multiple workspaces to the same control repo or module repo.

To add a status notification prefix:

1. In the Continuous Delivery for PE web UI, click Settings > Source control.
2. In the Status notification prefix section, click Edit prefix.
3. Enter a prefix, such as your workspace's name. Click Save.

After saving your prefix, the example code updates to show the prefixed pipeline status labels that this workspace sends to your source control provider.

Integrate with Azure DevOps Services

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. You must create an Azure DevOps Services OAuth application to integrate your Azure DevOps Services instance with Continuous Delivery for PE and start using these tools.

Before you begin

Important: Continuous Delivery for PE is only compatible the Azure DevOps cloud offering. Continuous Delivery for PE is not compatible with the hosted version, Azure DevOps Server.

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

1. Sign into Continuous Delivery for PE as the root user, and click Settings > Integrations. The authorization callback URL required to create your OAuth app is shown in the root console.
3. Enter your company name.
4. In the Application Information section, enter a name for your OAuth application, such as `CD for PE`.
5. In the Application website field, enter the base URL for your Continuous Delivery for PE instance.
6. In the Authorization callback URL field, enter the authorization callback URL printed in the Continuous Delivery for PE root console.
7. In the Authorized scopes section, select Code (read and write).
8. Click **Create Application**. When you are redirected to the page showing the new application's setting, stay on this page.

**Important:** Don't close this page. You need the application settings information in the next step.

9. Return to the Continuous Delivery for PE root console. On the **Integrations** page, enter the application ID and client secret for your Azure DevOps Services OAuth application, and click **Add**.

Once you have an Azure DevOps Services OAuth application for your organization, each workspace must be authenticated with the application in order to integrate your Continuous Delivery for PE instance with Azure DevOps Services. To do this, you must grant code read and write permissions and add a public SSH key, which allows cloning of modules and control repos during automated tasks.

**Important:**

If your organization uses Azure DevOps Services branch permissions to limit user access to Git branches, review the permissions granted to Continuous Delivery for PE users. Make sure these users can force push to the relevant control repos and module repos.

Azure DevOps Services only supports cloning over SSH. HTTP(S) cloning is not supported. You must enable SSL on Continuous Delivery for PE to use Azure DevOps Services.

---

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

After integrating with your source control provider, see **Add repositories** on page 115.

**Integrate with Bitbucket Cloud**

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. You must create a Bitbucket Cloud OAuth application to integrate your Bitbucket Cloud instance with Continuous Delivery for PE and start using these tools.

**Before you begin**

An administrator on your team must create a Bitbucket Cloud OAuth consumer for Continuous Delivery for PE.

1. Sign into Continuous Delivery for PE as the root user, and click **Settings > Integrations**.
2. In your organization's Bitbucket Cloud account, create an OAuth consumer. Instructions to **Create a consumer** are in the Bitbucket Cloud documentation. The authorization callback URL required to create your OAuth consumer is shown in the root console.
3. Give the OAuth consumer these permissions:

<table>
<thead>
<tr>
<th>Category</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>Email, Read</td>
</tr>
<tr>
<td>Workspace membership</td>
<td>Read</td>
</tr>
<tr>
<td>Category</td>
<td>Permissions</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Repositories</td>
<td>Read, Write</td>
</tr>
<tr>
<td>Pull requests</td>
<td>Read, Write</td>
</tr>
<tr>
<td>Webhooks</td>
<td>Read and write</td>
</tr>
</tbody>
</table>

4. After creating the OAuth application, 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 you have a Bitbucket Cloud OAuth application for your organization, each workspace must be authenticated with the application in order to integrate your Continuous Delivery for PE instance with Bitbucket Cloud.

**Important:**
If your organization uses Bitbucket Cloud branch permissions to limit user access to Git branches, review the permissions granted to Continuous Delivery for PE users. Make sure these users have write access and the ability to rewrite history on the relevant control repos and module repos.

Bitbucket Cloud only supports cloning over HTTP(S). Bitbucket Cloud does not support SSH cloning, and it does not support pull requests from forks. Jobs run on pull requests from forks will fail.

1. In the Continuous Delivery for PE web UI, click Settings.
2. Click Source control > Bitbucket Cloud.
3. Click Add credentials to give Continuous Delivery for PE permission to read and write code to your Bitbucket Cloud account.
4. Click Add credentials.
   You're redirected to Bitbucket Cloud to authorize the OAuth application set up by your workplace administrator.
5. Click Grant access to allow Continuous Delivery for PE to access your Bitbucket Cloud account.

After integrating with your source control provider, Add repositories on page 115.

**Integrate with Bitbucket Server**
Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. Integrate your Bitbucket Server instance with Continuous Delivery for PE to start using these tools.

**Important:**
If your organization uses Bitbucket Server branch permissions to limit user access to Git branches, review the permissions granted to Continuous Delivery for PE users. Create an exemption rule to ensure these users can force push to the relevant control repos and module repos.

Bitbucket Server only supports cloning over SSH. Bitbucket Server does not support HTTP(S) cloning, and it does not support pull requests from forks. Jobs run on pull requests from forks will fail.

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

1. In the Continuous Delivery for PE web UI, click Settings.
2. Click Source control > 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 want to connect to Continuous Delivery for PE.
5. In the **SSH port** field, enter the port number on which your Bitbucket Server listens for SSH requests. To locate this port number:
   a) In the Bitbucket Server web UI, click **Administration** (the gear icon) and then click **Server settings**.
   b) Locate the SSH port in the **SSH access** section of the **Server settings** page.

6. If your Bitbucket Server's SSH base URL is different from the host URL, enter the SSH base URL. To view your SSH base URL:
   a) In the Bitbucket Server web UI, click **Administration** (the gear icon) and then click **Server settings**.
   b) Locate the SSH base URL in the **SSH access** section of the **Server settings** page.

7. Enter the SSH user for clones if it is something other than `git`.

8. Click **Add credentials**.

After integrating with your source control provider, [Add repositories](page 115).

### Integrate with GitHub

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. You must create a GitHub OAuth application to integrate your GitHub instance with Continuous Delivery for PE and start using these tools.

#### Before you begin

An administrator on your team must create a GitHub OAuth application for Continuous Delivery for PE.

1. Sign into Continuous Delivery for PE as the root user, and click **Settings > Integrations**.
2. In your organization's GitHub account, create an OAuth application. Instructions for [Creating an OAuth App](GitHub documentation) are in the GitHub documentation. In the **Homepage URL** field, enter the base URL for your Continuous Delivery for PE instance (http://<CD4PE-HOST-SERVER>). The **Authorization callback URL** is shown in the Continuous Delivery for PE root console.
3. After creating your OAuth application, note the Client ID and Client Secret shown on the application's page in the GitHub UI.
4. Return to the Continuous Delivery for PE root console. On the **Integrations** page, enter the client ID and secret for your GitHub OAuth application, and click **Add**.

Once you have a GitHub OAuth application for your organization, each workspace must be authenticated with the application in order to integrate your Continuous Delivery for PE instance with GitHub.

#### Important:

If your organization uses protected branches on GitHub, make sure you allow force pushing to protected branches, or that you use the GitHub Administrator user to connect control repos or module repos to Continuous Delivery for PE. GitHub only supports cloning over HTTP(S). SSH cloning is not supported.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. Click **Source control > GitHub**.
3. Click **Add credentials**.
   You're redirected to GitHub to authorize the OAuth application set up by your team's administrator.
4. Click **Grant access** to allow Continuous Delivery for PE to access your GitHub account.

After integrating with your source control provider, [Add repositories](page 115).

### Integrate with GitHub Enterprise

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. Integrate your GitHub Enterprise instance with Continuous Delivery for PE to start using these tools.

#### Important:
If your organization uses protected branches on GitHub Enterprise, make sure you allow force pushing to protected branches, or that you use the GitHub Enterprise Administrator user to connect control repos or module repos to Continuous Delivery for PE.

GitHub Enterprise only supports cloning over HTTP(S). SSH cloning is not supported.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. Click **Source control > 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 **View your profile and more** (profile photo icon) and then click **Settings**.
   b) Click **Developer settings > Personal access tokens > Generate new token**.
   c) Enter a token description, such as `CD for PE`.
   d) Select the `repo, read:org, and user:email` scopes.
   e) Click **Generate token**.
   f) Copy the personal access token created by GitHub Enterprise.
5. In the Continuous Delivery for PE web UI, enter the GitHub Enterprise token in the **Token** field.
6. Based on your GitHub Enterprise configuration, select either **This instance uses a standard CA certificate** or **This instance uses a custom CA certificate**. If you're using a custom certificate, paste the full certificate in the **Custom CA certificate** field.
7. Click **Add credentials**.

After integrating with your source control provider, see Add repositories on page 115.

Integrate with GitLab

Continuous Delivery for Puppet Enterprise (PE) works with your existing source control system to track changes to your Puppet code and manage code deployments to your nodes. Integrate your GitLab instance with Continuous Delivery for PE to start using these tools.

**Important:** If your organization uses protected branches on GitLab, make sure the GitLab user account connected to Continuous Delivery for PE is assigned to a GitLab role with “allow” rules that enable the user to push to the protected branch.

GitLab supports cloning over both SSH and HTTP(S). You set the cloning protocol for each Continuous Delivery for PE workspace.

1. In the Continuous Delivery for PE web UI, click **Settings**.
2. Click **Source control > GitLab**.
3. In the **Host** field, enter the resolvable HTTP or HTTPS URL for your GitLab instance, such as `http://gitlab.example.com`.
   The URL must be formatted as a resolvable HTTP or HTTPS address even if you are using an SSH connection.
4. Create a token allowing Continuous Delivery for PE to access your GitLab instance.
   a) In the GitLab web UI, navigate to your user settings and click **Access tokens**.
   b) Enter a name for the application, such as `CD for PE`, and set an expiration date for the token.
   c) Select the `api` and `read_user` scopes.
   d) Click **Create personal access token**.
   e) Copy the personal access token created by GitLab.
5. In the Continuous Delivery for PE web UI, enter the GitLab token in the **Token** field.
6. Select whether your workspace clones GitLab repositories via SSH or HTTP(S).

   a) For SSH:
      
      • **Optional**: Add the SSH user's credentials in the **SSH user** field.
      
      • **Optional**: In the **SSH port** field, specify the port on which your GitLab server listens for SSH requests. The default port number is 22.

   b) For HTTP(S):
      
      • If you're using a custom certificate, paste the full certificate into the **Custom CA certificate** field.

7. Click **Add credentials**.

After integrating with your source control provider, see [Add repositories on page 115](#).

### Update webhooks

If you change the location of your Continuous Delivery for Puppet Enterprise (PE) installation or change the hostname when migrating your Puppet Application Manager (PAM) installation, you must update the webhooks that connect Continuous Delivery for PE to your source control system.

1. In the Continuous Delivery for PE root console, click **Settings > Webhooks**.

2. Enter the backend service endpoint:
   
   • If you changed your Continuous Delivery for PE installation location, enter the backend service endpoint for your previous Continuous Delivery for PE installation.

   • If you changed the hostname when migrating your PAM installation, run `kubectl describe pod <CD4PE_POD> | grep BACKEND` to get the backend service endpoint.

3. Click **Update webhooks**.

Continuous Delivery for PE updates all webhooks according to your configuration.

### Configure job hardware

The servers Continuous Delivery for Puppet Enterprise (PE) uses to run tests on your Puppet code are known as job hardware. You must configure job hardware before you can run tests or use pipelines.

- **Job hardware requirements** on page 112

  System requirements for Continuous Delivery for PE job hardware depend on the size of your Continuous Delivery for PE installation, the type of jobs you run, and how frequently you run them.

- **Configure job hardware** on page 113

  The servers Continuous Delivery for Puppet Enterprise (PE) uses to run tests on your Puppet code are known as job hardware. You must configure job hardware before you can run tests or use pipelines. Any node with a Puppet agent installed on it can be job hardware.

- **Add job hardware capabilities** on page 114

  Job hardware servers are organized by capabilities in Continuous Delivery for PE. A capability is a tag that indicates what type of jobs can run on that job hardware server. If you're managing a lot of job hardware servers, use capabilities to distribute the testing load.

- **Configure global shared job hardware** on page 114

  All workspaces in your Continuous Delivery for Puppet Enterprise (PE) installation can use global shared job hardware. The root user or a super user must set up these special job hardware servers in the root console.

### Job hardware requirements

System requirements for Continuous Delivery for PE job hardware depend on the size of your Continuous Delivery for PE installation, the type of jobs you run, and how frequently you run them.

The size of the load placed on a job hardware server determines how robust the server's resources must be. Due to the number of variables involved in estimating this (such as the number of jobs running concurrently and the languages the jobs are written in), it's impossible to provide universal system requirements for job hardware.
Instead, we've created a sizing guide based on the estimated number of concurrent jobs you expect your job hardware server to handle regularly. While this guide represents our best estimates and understanding, it's only a starting point. Through testing and experience, you can fine-tune your job hardware and determine the optimum resource configuration for your Continuous Delivery for PE installation.

<table>
<thead>
<tr>
<th>Estimated concurrent job load</th>
<th>Memory</th>
<th>Disk storage</th>
<th>CPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 - 4 concurrent spec tests</td>
<td>4 GB</td>
<td>100 GB</td>
<td>2</td>
</tr>
<tr>
<td>4 - 8 concurrent spec tests</td>
<td>8 GB</td>
<td>100 GB</td>
<td>4</td>
</tr>
<tr>
<td>6 - 12 concurrent spec tests</td>
<td>8 GB</td>
<td>100 GB</td>
<td>6</td>
</tr>
</tbody>
</table>

When setting up your job hardware, keep in mind:

- Disk storage requirements are minimal and don't increase with added load. After a job run is complete, the job's log is passed to the object storage, and all data related to the job run is erased from the job hardware.
- You can run more jobs concurrently without increasing CPUs, but the jobs run slower.

**Job hardware requirements for Docker-based jobs**

To run Docker-based jobs, your job hardware must have a modern version of Docker CE or Docker EE installed. The puppetlabs/docker module (from the Forge) is our recommended way to install Docker and keep it updated.

Job hardware used for Docker-based jobs also requires internet access. If you're working in an air-gapped environment, set up an internal Docker registry, as explained in the Docker documentation.

**Configure job hardware**

The servers Continuous Delivery for Puppet Enterprise (PE) uses to run tests on your Puppet code are known as job hardware. You must configure job hardware before you can run tests or use pipelines. Any node with a Puppet agent installed on it can be job hardware.

**Before you begin**

Review the Job hardware requirements sizing guide.

**Important:** Only the administrator of a workspace or a super user can configure job hardware for a workspace.

1. Install a Puppet agent on each node you want to use as job hardware. The PE documentation provides instructions for Installing agents.

2. Make sure your Continuous Delivery user role in PE has permission to run the cd4pe_jobs::run_cd4pe_job task.

3. Install the puppetlabs-cd4pe_jobs module from the Forge. This module is required to run Continuous Delivery for PE jobs on your nodes.
   a) Add the puppetlabs-cd4pe_jobs module to:
      - The Puppetfile for the production environment on the PE primary server that manages the agent nodes you've selected as job hardware.
      - The Puppetfile on the main branch of the control repo associated with the PE primary server managing the agent nodes you've selected as job hardware.

      A sample Puppetfile entry:

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

   b) Run puppet code deploy production --wait to deploy the updated code to the production environment.

4. Install and configure Docker on each job hardware node. Docker is required for Docker-based pre-built jobs included in Continuous Delivery for PE. Instructions are available in the Docker documentation.
5. Assign nodes to the Docker hardware capability (which Continuous Delivery for PE automatically creates for you) to tell Continuous Delivery for PE the nodes can be used as job hardware for Docker-based jobs.
   a) In the Continuous Delivery for PE web UI, click Hardware.
   b) Locate the Docker capability and click Edit.
   c) Select the PE instance that manages the nodes you've selected as job hardware, and then select the job hardware nodes. Selected nodes appear in the Hardware with this capability list.
   d) After selecting all the job hardware nodes, click Save.

6. Optional: Create additional capabilities or add job hardware nodes to other capabilities, as explained in Add job hardware capabilities. If you're managing a lot of job hardware nodes, additional capabilities help distribute the testing load and ensure that each job runs on a hardware with the correct characteristics.

Your job hardware is configured and ready for use.

Add job hardware capabilities

Job hardware servers are organized by capabilities in Continuous Delivery for PE. A capability is a tag that indicates what type of jobs can run on that job hardware server. If you're managing a lot of job hardware servers, use capabilities to distribute the testing load.

Capabilities organize your job hardware servers and ensure that jobs run on hardware with the right characteristics. Continuous Delivery for PE automatically creates a Docker capability for you, and you can create additional capabilities according to your needs.

For example, assume you want to create a new job that requires Python 3. To ensure Continuous Delivery for PE can run this job, you need to create a python-3 capability in Continuous Delivery for PE and assign to that capability all job hardware servers that have Python 3 installed and configured. Then, when you create the job that uses Python 3, specify that the job requires job hardware with the python-3 capability. When you run this job, Continuous Delivery for PE automatically locates a job hardware server with the python-3 capability and uses that job hardware server to run the job.

In addition to software requirements, useful capabilities include:

- Operating systems
- Organizational designations, such as dev-team-philly-testing
- Puppet testing resources, such as onceover or Puppet Development Kit

**Important:** Only the administrator of a workspace, the root user, and super users can add hardware capabilities to a workspace.

1. In the Continuous Delivery for PE web UI, click Hardware.
2. Click Add capability and name your new capability.
3. To assign job hardware servers to the capability, select the PE instance that manages the node you've selected as job hardware, and then select the nodes you want to assign to the capability. Selected nodes appear in the Hardware with this capability list.
4. After selecting all the nodes you want to assign to this capability, click Save.

Your job hardware is now assigned to a capability. To add capabilities to jobs in your workspace, click Jobs > Edit job, and select options from the list of capabilities.

Configure global shared job hardware

All workspaces in your Continuous Delivery for Puppet Enterprise (PE) installation can use global shared job hardware. The root user or a super user must set up these special job hardware servers in the root console.

**Before you begin**
Review the Job hardware requirements sizing guide.

1. Install a Puppet agent on each node you want to use as global shared job hardware. The PE documentation provides instructions for Installing agents.
2. Make sure your Continuous Delivery user role in PE has permission to run the `cd4pe_jobs::run_cd4pe_job` task.

3. Install the `puppetlabs-cd4pe_jobs` module from the Forge. This module is required to run Continuous Delivery for PE jobs on your nodes.
   a) Add the `puppetlabs-cd4pe_jobs` module to:
      - The Puppetfile for the production environment on the PE primary server that manages the agent nodes you’ve selected as job hardware.
      - The Puppetfile on the main branch of the control repo associated with the PE primary server managing the agent nodes you’ve selected as job hardware.
   A sample Puppetfile entry:
   ```
   mod 'puppetlabs-cd4pe_jobs', '1.6.0'
   ```
   b) Run `puppet code deploy production --wait` to deploy the updated code to the production environment.

4. Install and configure Docker on your selected global shared job hardware nodes. Instructions are available in the Docker documentation.

5. In the Continuous Delivery for PE root console, click Hardware.

6. You must assign all global shared job hardware servers to the Docker capability, which Continuous Delivery for PE automatically creates for you. Locate the Docker capability and click Edit.

7. Assign all your global shared job hardware servers to the Docker capability. Select the PE instance that manages the nodes you’ve selected as job hardware, and then select the global shared job hardware nodes. Selected nodes appear in the Hardware with this capability list.

8. After selecting all the global shared job hardware nodes, click Save.

9. Optional: Create additional capabilities for your global job hardware servers, as explained in Add job hardware capabilities.

Your global shared job hardware is configured. Users in all workspaces now see a Run on shared hardware option when creating or editing a job. Jobs configured to run on your global shared job hardware have the Docker capability automatically selected, and these jobs run on the global shared job hardware assigned to the Docker capability.

CAUTION: If your installation specifies a Docker image in Puppet Application Manager (PAM) (this is common in offline environments), this image takes precedence over the image set in the Continuous Delivery for PE root console (which is the one shown when creating or editing a job).

### Add repositories

You must tell Continuous Delivery for Puppet Enterprise (PE) which repositories in your source control system are your control repos and module repos. This tells Continuous Delivery for PE where to listen for code changes.

**Before you begin**

Integrate with source control on page 107 and learn about Working with Git branches in Continuous Delivery for PE on page 139 and Understanding the Continuous Delivery for PE workflow on page 141.

Control Repos in Continuous Delivery for PE track changes made on active development branches of your source control system. Module repos, referred to as Modules in the Continuous Delivery for PE web UI, contain information about Puppet modules. When adding repos to Continuous Delivery for PE, it's important to connect the repo's main Git branch.

Adding a repo in Continuous Delivery for PE configures a webhook to the repository in your source control system. The webhook reports new commit activity on the repository to Continuous Delivery for PE, enabling you to track code changes and take action (such as triggering pipelines or running individual tests on the new code).
1. In the Continuous Delivery for PE web UI, click **Control repos > Add control repo.** To add a module repo, click **Modules > Add module.**
2. Select the source control provider and repository.
3. The repo's main branch is automatically selected, if one exists. If the repo does not have a main branch, select **Create a main branch from an existing branch** and select your active development branch as the basis for the new main branch. Continuous Delivery for PE creates a main branch for you based on the existing branch you choose.

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

4. Optional: Edit the repo's **Display name** as it appears in the Continuous Delivery for PE web UI. The name can contain only numbers, letters, dashes, and underscores.
5. Click **Add control repo** (or **Add module**).
6. If you added a control repo, make sure you add the puppetlabs-cd4pe_jobs module to the Puppetfile on the control repo's main branch, as explained in **Configure job hardware** on page 113.
7. Repeat to add more repos to Continuous Delivery for PE.

   Construct pipelines to test new code committed to your repos.

### Configure SAML

Continuous Delivery for Puppet Enterprise (PE) supports the use of Security Assertion Markup Language (SAML) authentication from a SAML identity provider (IDP). Once you configure your SAML IDP to integrate with Continuous Delivery for PE, you can use your chosen single sign-on tool to authenticate users to Continuous Delivery for PE.

**Before you begin**

Your enterprise SAML team must configure your organization’s SAML IDP to communicate with Continuous Delivery for PE. Provide the SAML team with the Continuous Delivery for PE SAML redirect URL for your installation: `<YOUR CD4PE WEB UI ENDPOINT>/saml-auth`. The SAML team uses this to register Continuous Delivery for PE as an application with permissions to interact with the IDP.

Get this information from your enterprise SAML team:

- The IDP-initiated SSO URL that Continuous Delivery for PE needs to direct user authentication requests.
- The IDP public signing certificate for Continuous Delivery for PE.
- The SAML attribute names that come back in the SAML assertion for these fields: first name, last name, email address, and username. (Attribute mapping is explained in step 4, below.)

A super user or the root user must perform this task.

1. Log into the root console by signing in as the root user or by selecting **Root console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar.
2. Click **Settings** and click the **Single sign on** tab if you are not already on it.
3. Select **SAML**.
4. Enter the required configuration information as per the instructions below.

**IdP Initiated SSO URL**

The unique URL created by the SAML IDP used by your organization that acts as a single sign-on (SSO) gateway for Continuous Delivery for PE. Your enterprise SAML team provides this URL.

**Public Signing Certificate**

The SAML IDP public signing certificate verifies SAML assertions from the IDP. Your enterprise SAML team provides this certificate. Paste the entirety of the certificate, including the header and footer, into this field.

**Attribute Mapping**

The SAML assertion sends four attributes to Continuous Delivery for PE. The Attribute Mapping matches attribute keys from the SAML IDP assertion to user accounts created by Continuous Delivery for PE.

- **First Name**: The SAML attribute key for the user's first name.
- **Last Name**: The SAML attribute key for the user's last name.
- **Email**: The SAML attribute key for the user's email address. This is the unique user identifier, so each user's email address must be unique.
- **Username**: The SAML attribute key for the user's username in Continuous Delivery for PE. Each username must be unique.

5. Click **Run Configuration Test** to send a sample authentication query to your SAML IDP.

6. If the configuration test is successful, you're ready to enable SAML authentication for your Continuous Delivery for PE instance, enable the SAML configuration switch and click **Save Configuration**.

---

**Configure LDAP**

Continuous Delivery for Puppet Enterprise (PE) supports use of the Lightweight Directory Access Protocol (LDAP) for managing user authentication. After configuring LDAP, use group mapping to associate your existing LDAP groups with role-based access control (RBAC) groups in Continuous Delivery for PE.

For organizational or failover protection purposes, you can add multiple LDAP configurations, each specifying a separate LDAP server, to your Continuous Delivery for PE instance. Continuous Delivery for PE uses the LDAP configurations you set up to search LDAP users in a specified order. Once a user is found, the search ends and that LDAP configuration is used to perform the login operation.

**Note**: If your LDAP server has a search result limitation below 500, see **Configuring LDAP server search result limits**.

**Create a new LDAP configuration**

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

A super user or the root user must perform this task.

1. Log into the root console by signing in as the root user or by selecting **Root console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar.

2. Click **Settings** and click the **Single sign on** tab if you are not already on it.

3. Select **LDAP** and click **Add LDAP configuration**.

© 2021 Puppet, Inc.
4. In the **New LDAP configuration** window, enter the configuration information as per the instructions below.

**Name**
A friendly identifier for this LDAP configuration. You can't change this, so choose carefully.

**Endpoint URL**
The LDAP server's endpoint URL, including the LDAP scheme, hostname, and port. If these aren't included, the default scheme is `ldaps` and the default port is 636.

**Bind DN**
The distinguished name of the LDAP account that Continuous Delivery for PE binds as when performing LDAP operations. Usually, this is an admin account or a service account created specifically for Continuous Delivery for PE. If you create a new account, make sure it has permission to search for users and groups.

**Bind DN password**
The password associated with the bind DN account.

**Note:** You must enter this password each time you update the LDAP configuration. You don't have to enter the password when you disable the LDAP configuration.

**User base DN**
The LDAP base DN that informs Continuous Delivery for PE where users are located in the directory. Using a more specific DN results in better LDAP search performance.

**User attribute**
The LDAP user attribute that maps LDAP users to Continuous Delivery for PE usernames. Usually, this is the `mail` attribute, but you can use any attribute as long as the LDAP database doesn't contain any duplicate values for that attribute.

**Optional: User base filter**
A filter that Continuous Delivery for PE can use to restrict user search results. This is useful in edge cases where you want to include or exclude certain users.

**Group base DN**
The LDAP base DN that informs Continuous Delivery for PE where groups are located in the directory. Using a more specific DN results in better LDAP search performance.

**Note:** If users and groups are stored in the same location, the **Group base DN** is the same as the **User base DN**.

**Group user attribute**
The user attribute that group entries use to identify users. Usually, this is `dn`.

**Group member attribute**
The group attribute that maps to a group member. Usually, this is either `member` or `uniqueMember`.

**Group name attribute**
The group attribute that identifies the group name. Usually, this is `cn`.

**Optional: Group base filter**
A filter that Continuous Delivery for PE can use to restrict group search results. This is useful in edge cases where you want to include or exclude certain groups.

**User object class**
Specifies the value of the `objectClass` attribute that allows Continuous Delivery for PE to query user entries. Usually, this is either `user` or `person`.

**Group object class**
Specifies the value of the `objectClass` attribute that allows Continuous Delivery for PE to query group entries. Usually, this is either `group` or `groupOfUniqueNames`.

**Optional: Mail attribute**
The LDAP user attribute used to identify each member's email address. Defaults to `mail` if unset.

**Optional: User member attribute**
Specifies the user attribute that can be used to identify the membership of a group. If present, this is usually `memberOf`. © 2021 Puppet, Inc.
5. Optional: Enter the trusted server's CA certificate. If the LDAP server uses a certificate signed by a trusted CA, you do not need to enter a CA certificate here.

6. Select the **Priority** number for this LDAP configuration. If you have multiple LDAP configurations, this number indicates the order in which Continuous Delivery for PE searches for users in your LDAP configurations during login and when synchronizing groups.

7. Optional: Set the toggle to enable recursive LDAP queries for nested groups. This option is available only if your LDAP server’s implementation supports the ExtensibleMatch search filter.

8. Optional: Set the toggle to enable use of login filtering. When enabled, users who are not part of mapped LDAP groups are not allowed to log in to Continuous Delivery for PE.

9. Activate the **Enable LDAP** switch and click **Run configuration test** to check the connection between Continuous Delivery for PE and the LDAP server.

   **Note:** This configuration test checks the LDAP server connection; it does not test the other configuration options.

10. Click **Save configuration.** Your new LDAP configuration is shown on the **Single sign on** settings page, where you can edit or delete the configuration.

   **Important:** Once you enable an LDAP configuration, Continuous Delivery for PE automatically disables all local Continuous Delivery for PE accounts other than the root account, and it attempts to use LDAP authentication. If LDAP authentication fails, navigate to `<YOUR CD4PE WEB UI ENDPOINT>/root/login`, sign in as the root user, and adjust the LDAP settings.

---

**Create an LDAP group map**

After adding an LDAP configuration to Continuous Delivery for Puppet Enterprise (PE), use group maps to map your existing LDAP groups to Continuous Delivery for PE RBAC groups. This makes it possible to mirror LDAP group membership in Continuous Delivery for PE groups.

**Before you begin**

You must add at least one LDAP configuration to your Continuous Delivery for PE instance before you can create an LDAP group map.

1. Log in to the root console by navigating to `<YOUR CD4PE WEB UI ENDPOINT>/root/login` and signing in as the root user.

2. Click **Settings** and click the **Single sign on** tab if you are not already on it.

3. Click **LDAP > Manage groups > Add LDAP group mapping.**

4. From the **LDAP configuration name** list, select the LDAP configuration you want to create group mapping for.

5. From the **LDAP group name** list, select the group to use to perform the mapping.

   **Tip:** You can search for groups by name (partial matches and case-insensitive matches allowed) or distinguished name (case-insensitive matches allowed).

6. Select a Continuous Delivery for PE account associated with the RBAC group you want to map to the selected LDAP group.

7. From the list of available Continuous Delivery for PE RBAC groups, select the RBAC group you want to map to the selected LDAP group.

8. Click **Add group mapping.**

After setting up a group map, Continuous Delivery for PE synchronizes with your LDAP groups based on the mapping you created.

---

**Configure SMTP**

Configure SMTP for your Continuous Delivery for Puppet Enterprise (PE) installation so that users can receive email notifications from the software.
The root user or a super user must do this task.

1. Log into the root console by signing in as the root user or by selecting **Root console** from the workspaces menu at the top of the Continuous Delivery for PE navigation bar.

2. Click **Settings > SMTP**.

3. If your SMTP server requires authentication, enter the **SMTP username** and **SMTP password** in the relevant fields.

4. In the **Send emails from this address** field, enter the sender address you want to show on Continuous Delivery for PE emails.

   **Important:** If this field is not specified, Continuous Delivery for PE uses the root user’s email address as the sender/from address when it sends an email. In this case, the specified SMTP account must have permission to send emails from the root user’s email address.

5. Enter your **SMTP host** name and the **SMTP port** number in the relevant fields.

   **Note:** Contact your email server administrator if you need help determining the correct SMTP port.

6. If your server requires TLS authentication, enable the **Enable TLS** switch.

7. Click **Save settings**.

8. To test your new SMTP configuration, generate a password reset email:
   a) Log out of Continuous Delivery for PE.
   b) On the sign-in screen, click **Forgot your password?**.
   c) Enter the email address associated with your Continuous Delivery for PE account and click **Send reset instructions**.
   d) Check your inbox for the password reset email. You can ignore the password reset instructions and keep your current password; this test just ensures Continuous Delivery for PE can send emails.

After configuring SMTP, Continuous Delivery for PE sends these emails:

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

---

### Set up external PostgreSQL

To use an external PostgreSQL database with Continuous Delivery for Puppet Enterprise (PE), set up the connection on the **Config** page in Puppet Application Manager (PAM).

On the **Config** page in PAM, click **Set up external PostgreSQL** and complete these fields:

- **PostgreSQL endpoint**: The hostname and port (in **hostname:port** format) used to connect to PostgreSQL.
- **CD4PE PostgreSQL username**: The username for a user authorized to connect to the CD4PE PostgreSQL database.
- **CD4PE PostgreSQL password**: The password for the CD4PE PostgreSQL username.
- **CD4PE PostgreSQL database**: The database to use with CD4PE. Defaults to **cd4pe** if unspecified.

Select the **PostgreSQL SSL mode** to use when connecting to the databases during preflight checks.

Click **Save config**.

---

### Advanced Continuous Delivery for PE configuration

Advanced configuration settings for Continuous Delivery for PE help you fine-tune aspects of the software that can impact runtime and operation speed.
Adjusting available memory

Adjust the amount of memory available to Continuous Delivery for Puppet Enterprise (PE) based on repository size and workflow complexity.

In some cases, the amount of memory allocated to the Continuous Delivery for PE application by default isn't sufficient. Memory consumption scales with repository size and the number of jobs running concurrently.

To adjust the amount of memory allocated to Continuous Delivery for PE:

1. Navigate to the Config page in Puppet Application Manager (PAM).
2. In the Advanced configuration and tuning section, adjust the Memory available for CD4PE (in MiB) setting.

Adjusting the timeout period for snapshots

When creating a snapshot, Puppet Application Manager (PAM) makes backup copies of various Continuous Delivery for Puppet Enterprise (PE) components. You can change the amount of time PAM spends making each component's backup copy.

You can learn more about creating and using snapshots in the PAM documentation.

To configure timeout periods, navigate to the Config page in PAM, and edit these settings in the Advanced configuration and tuning section:

- Object store backup timeout: The amount of time a running snapshot spends attempting to backup the Continuous Delivery for PE object store. The default is 30 minutes.
- Continuous Delivery for PE database backup timeout: The amount of time a running snapshot spends attempting to backup the Continuous Delivery for PE database. The default is 10 minutes.
- Nodes database backup timeout: The amount of time a running snapshot spends attempting to backup the database for the Nodes page. The default is 10 minutes.

Improving job performance by caching Git repositories

If you have large Git repositories, you can enable Git repository caching to improve job performance. Repository caching is disabled by default.

Navigate to the Config page in Puppet Application Manager (PAM), and enable Git Repo caching in the Advanced configuration and tuning section.

Including the .git directory in cached repositories

The .git directory is automatically omitted when copying cached Git repositories to job hardware. This means that the job cannot perform Git actions on the code. If needed, you can adjust this setting so that the .git directory is included in the cached repository.

To include the .git directory in copies of cached Git repositories sent to job hardware, navigate to the Config page in PAM, and enable Include Git history for jobs in the Advanced configuration and tuning section.

Adjusting the timeout period for jobs

In some circumstances, such as when working with large Git repositories, you may need to adjust the length of the job timeout period. Use the settings in this section to customize the job timeout period.

To configure timeout periods, navigate to the Config page in PAM, and edit these settings in the Advanced configuration and tuning section:

- Repo cache retrieval timeout (minutes): Sets the timeout period for a thread attempting to access a cached Git repository. The default is 28 minutes.

  Note: This is only available if you enabled Git repository caching.

- Job HTTP read timeout (minutes): Sets the timeout period for a job connecting to an endpoint. The default is 29 minutes.
• **Job global timeout (minutes)**: Sets the global default timeout period for jobs. Once this time elapses, the job fails and a timeout message is printed to the log. The default is 30 minutes.

• **Bolt PCP read timeout (seconds)**: Sets the Bolt PCP read timeout period. The default is 60 seconds.

  **Note:** Jobs cannot proceed while file sync is running. If a file sync operation is not completed before the Bolt PCP read timeout period elapses, the job fails. Increase the Bolt PCP read timeout period to prevent these job failures.

**Important:** To ensure useful error messages are printed to the logs, make sure that the value of **Job global timeout** is larger than the value of **Job HTTP read timeout**, and that both are larger than the value of **Repo cache retrieval timeout**.

---

**Configuring LDAP server search result limits**

By default, Continuous Delivery for Puppet Enterprise (PE) requests 500 search results at a time from a connected LDAP server. If your LDAP server's search result limitation is below 500, you can configure Continuous Delivery for PE to match the LDAP server's search result threshold.

Navigate to the **Config** page in Puppet Application Manager (PAM) and set the **LDAP group search size limit** in the **Advanced configuration and tuning** section.

---

**Adjusting HTTP timeout periods**

If necessary, you can adjust the timeout periods for various HTTP operations.

To configure HTTP timeout periods, navigate to the **Config** page in PAM, and edit these settings in the **Advanced configuration and tuning** section:

• **Global HTTP connection timeout (seconds)**: Sets the timeout period for all external HTTP connections. The default is 120 seconds.

• **Global HTTP read timeout (seconds)**: Sets the timeout period for all external HTTP read actions. The default is 120 seconds.

  **Note:** This value is also used as the value for the **CD4PE_MODULE_DEPLOY_READ_TIMEOUT** environment variable in deployment tasks.

• **Global HTTP write timeout (seconds)**: Sets the timeout period for all external HTTP write actions. The default is 120 seconds.

• **Global HTTP request timeout (seconds)**: Sets the total amount of time an external HTTP request remains open. The default is 300 seconds.

**Configuring login attempt limits**

By default, Continuous Delivery for PE limits the number of unsuccessful login attempts users can make within a certain timeframe. If the user exceeds the allowed number of unsuccessful login attempts within that timeframe, their account is temporarily locked. You can customize login attempt limits for your installation.

To configure login attempt settings, navigate to the **Config** page in PAM, and edit these settings in the **Advanced configuration and tuning** section:

• **Max login attempts before lockout**: The number of unsuccessful login attempts a user can make before the account is locked. The default is 10 attempts.

• **Time period (minutes) to look at for failed logins**: The amount of time that must elapse before the failed login count resets. The default is 15 minutes.

• **Time period (minutes) to lock an account**: How long user accounts are locked after exceeding the number of unsuccessful login attempts within the failed login timeframe. The default is 120 minutes.
Managing teams and team members

• Managing workspaces on page 123
Use workspaces to share access to key Continuous Delivery for Puppet Enterprise (PE) resources, such as control repos, modules, and jobs, with the other members of your team. After setting up a workspace, add your team members to that workspace and give them the permissions needed to do their work.
• Managing access on page 124
The user access and management tools in Continuous Delivery for Puppet Enterprise (PE) ensure that all users, from read-only to super users, have the access and permissions appropriate to their role on the team.

Managing workspaces
Use workspaces to share access to key Continuous Delivery for Puppet Enterprise (PE) resources, such as control repos, modules, and jobs, with the other members of your team. After setting up a workspace, add your team members to that workspace and give them the permissions needed to do their work.

Best practices when creating workspaces
These suggestions and best practices are intended to help you and your organization understand Continuous Delivery for Puppet Enterprise (PE) workspaces and use them effectively.

Workspaces support teams who write and deploy Puppet code to nodes managed by PE. Whether your organization has one team writing and deploying all Puppet code, or multiple teams writing and testing Puppet code with a central deployment team pushing changes to production, workspaces help ensure each team member has the Continuous Delivery for PE resources they need.

Single team/repository recommendations
If your organization has one team writing, testing, and deploying all Puppet code or stores Puppet code in one source control repository, we recommend:
• Using one workspace for the whole team.
• Applying permissions carefully so each team member only has access to what they need.

Multiple teams/repositories recommendations
If your organization has multiple teams writing and testing Puppet code, multiple source control repositories for Puppet code, and/or a deployment team responsible for pushing Puppet code changes to production, we recommend:
• Having separate workspaces for each writing and testing team; ideally, one workspace for each control repository.
• Creating a separate workspace for the deployment team, and allow these team members to access all other workspaces.
• Applying permissions carefully so each team member only has access to what they need.

Set up a workspace
These steps explain how to create and add users to a workspace. If you need multiple workspaces, repeat this process as many times as you need.

Before you begin
Make sure all team members have created their individual Continuous Delivery for Puppet Enterprise (PE) accounts.

Note: Tell your team members to log out when they reach the Choose a workspace screen after creating their account.
1. Log in to the Continuous Delivery for PE web UI, and select **Manage workspaces** from the workspaces drop down menu at top of the navigation bar.

2. Click **Add new workspace** and give the workspace a descriptive name (If necessary, you can change the name later on the **Settings** page).

   After creating a workspace, you are automatically set as the workspace owner. Other Continuous Delivery for PE super users can access your workspace by navigating to the root console and clicking the name of your workspace in the **Workspaces** tab.

3. Switch to your new workspace by clicking its name in the **My workspaces** list or by selecting it from the workspaces drop down menu.

4. Follow the new workspace prompts at the top of the page to set up the workspace's required resources:
   a) If necessary, [Integrate with Puppet Enterprise](#) on page 100.
   b) [Integrate with source control](#) on page 107.
   c) [Configure job hardware](#) on page 113 specifically for this workspace or use global shared job hardware (If you haven't already done so, [Configure global shared job hardware](#) on page 114).

5. Click **Control repos** > **Add control repo**, and add the control repo where the team stores Puppet code.

6. Click **Settings > Users**, and add each team member as a user of this workspace.

   **Note:** If a team member is missing, they have not created their Continuous Delivery for PE account yet.

7. Click **Groups** and create one or more groups. Assigning appropriate permissions and users to each group. Make sure every team member is assigned to at least one group.

   If you add a user to a workspace without adding them to a group, they get a 403 error when signing in to Continuous Delivery for PE.

8. Invite the team members to sign in to Continuous Delivery for PE. They'll now see the new team workspace in the workspaces drop down menu on the Continuous Delivery for PE web UI.

   According to their group permissions, team members can view and interact with the resources you added to the workspace.

   To delete workspaces, navigate to **Settings > Workspace > Delete workspace**. Only super users or the workspace owner can do this.

### Transfer workspace ownership

When a workspace owner changes teams, leaves your organization, or is otherwise no longer the right person to manage a workspace, you can reassign workspace ownership to a different Continuous Delivery for PE user.

To transfer workspace ownership, you must have root or super user permissions allowing you to access the root console and perform this action.

1. Log into the root console by signing in as the root user or by selecting **Root console** from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.

2. Click **Workspaces**, locate the workspace you need to reassign, and click **Reassign workspace**.

3. Select the new owner's username and click **Save changes**.

   After transferring ownership, the original workspace owner is automatically removed from the workspace. The new owner can add the former owner as a workspace user by navigating to **Settings > Users**.

### Managing access

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

**Note:** [Configuring login attempt limits](#) explains how to customize failed login controls for your installation.
• Create a user account on page 125
After an administrator installs Continuous Delivery for Puppet Enterprise (PE), each person who needs to use the software must create an individual user account.
• Adding users and creating groups on page 125
Continuous Delivery for Puppet Enterprise (PE) facilitates collaboration between team members. You can share resources by adding users to your workspace, and you can set up groups to manage permissions granted to each team member.
• Permissions reference on page 127
This table lists Continuous Delivery for Puppet Enterprise (PE) group permissions and a short explanation of each permission's scope.
• Using the root console on page 129
Super users and the root user can access the Continuous Delivery for Puppet Enterprise (PE) root console to manage users' account credentials, change super user permissions, configure single sign-on, access all workspaces associated with the installation, transfer workspace ownership, and update the installation's settings.

Create a user account
After an administrator installs Continuous Delivery for Puppet Enterprise (PE), each person who needs to use the software must create an individual user account.

**Before you begin**
Obtain the Continuous Delivery for PE web UI endpoint from the administrator who installed the software.

1. Point your browser to the Continuous Delivery for PE web UI endpoint.
2. On the login page, click **Create an account**.
3. Fill in the registration form and create a username and password.
4. Click **Sign up**.

You now have a Continuous Delivery for PE account. You can view your account credentials by clicking your username at the bottom of the navigation bar in the Continuous Delivery for PE web UI.

Adding users and creating groups
Continuous Delivery for Puppet Enterprise (PE) facilitates collaboration between team members. You can share resources by adding users to your workspace, and you can set up groups to manage permissions granted to each team member.

**Add a user to a workspace**
Adding users to your workspace allows you to collaborate on projects and share account resources such as jobs and pipelines. You can add users in the **Settings** area of the Continuous Delivery for PE web UI.

**Before you begin**
Make sure each user you wish to add has signed up for a Continuous Delivery for PE account. You'll need their selected username or the email address they used when signing up.

1. In the Continuous Delivery for PE web UI, click **Settings** then click **Users**.
2. Click **Add a user**.
3. On the **Add users to workspace** page, search for the username or email address of the user you wish to add to your workspace.
4. Select the user you wish to add to the workspace.
5. **For version 4.4.0 and newer versions**: Assign the user to one or more permission groups.

**Note:** Beginning in Continuous Delivery for PE version 4.4.0, newly created workspaces include three built-in user groups: Administrators, Operators, and Viewers. For details on the permissions included in each built-in group, see the Permissions reference.
6. Click **Add users to workspace**. Repeat these steps to add additional users.

7. When you’re finished adding users, click **Done** to return to the **Users** screen, where you’ll see the full list of users with access to your Continuous Delivery for PE workspace.

**Create a group and set group permissions**

Groups to organize users by permission level, job focus, geography, or any other criteria you choose. Permissions define the tasks that group members can perform in Continuous Delivery for Puppet Enterprise (PE), such as adding control repos, editing jobs, and deploying code changes. Assign permissions to groups based on each group’s functions and needs.

1. In the Continuous Delivery for PE web UI, click **Settings**.

2. On the **Groups** tab, click **Create new group**.

3. Enter a name and description for the new group.

   **Tip:** Choose a name that describes the group’s purpose, role, or permission level, such as Read-only users or Module developers.

   **CAUTION:** You can't edit group names or descriptions after this point. If you need to change the name or description, you must delete the group and recreate it.

4. Select permissions to assign to this group. For explanations of each permission’s scope, see the Permissions reference on page 127. Group permissions are additive; therefore, if a user belongs to multiple groups, that user can perform all actions allowed by all permissions from all of their assigned groups.

   To assign permissions on a certain subset of the control repos or modules in your workspace:

   a) In the permissions section, click **Edit permissions** next to **Control Repos subset** or **Modules subset**.

   b) Select the repos or modules to include in the subset and click **Save selection**.

   c) Select permissions to give the user group. You can allow list (view), edit, and delete permissions for all control repos/modules in the workspace or only on the selected subset.

   If you create a subset and do not assign any permissions on the subset, the subset is discarded when you save your permission selections.

5. Click **Save and add users**.

6. On the **Add users** page, select workspace members to add to this group, and click **Add users to group**.

7. Click **Done**. Continuous Delivery for PE creates your user group and redirects you to the group page.

   Review the group members and permissions on the group page, and make any changes, if needed.

   To delete a group, first remove all members from the group, and then click **Delete group**.

**Remove users from workspaces**

You can remove users from Continuous Delivery for PE workspaces in the **Settings** area of the Continuous Delivery for PE web UI.

**Note:** Removing a user from a Continuous Delivery for PE workspace automatically removes the user from all groups in that workspace.

1. Log in to the Continuous Delivery for PE web UI and make sure you are viewing the workspace you want to remove the user from.

2. Click **Settings > Users**.

3. Locate the user you want to remove from the workspace.

4. Click **Delete user** and confirm your decision.

5. Repeat these steps to remove additional users from the workspace.
## Permissions reference

This table lists Continuous Delivery for Puppet Enterprise (PE) group permissions and a short explanation of each permission’s scope.

<table>
<thead>
<tr>
<th>Type</th>
<th>Permission</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control repos</td>
<td>Create</td>
<td>The ability to create new control repos in Continuous Delivery for PE.</td>
</tr>
<tr>
<td>Control repos</td>
<td>Delete</td>
<td>The ability to delete existing control repos.</td>
</tr>
<tr>
<td>Control repos</td>
<td>Edit</td>
<td>The ability to edit existing control repos and deploy code.</td>
</tr>
<tr>
<td>Control repos</td>
<td>List</td>
<td>The ability to view existing control repos.</td>
</tr>
<tr>
<td>Control repos</td>
<td>Promote</td>
<td>The ability to promote changes in control repo pipelines.</td>
</tr>
<tr>
<td>Modules</td>
<td>Create</td>
<td>The ability to create new module repos in Continuous Delivery for PE.</td>
</tr>
<tr>
<td>Modules</td>
<td>Delete</td>
<td>The ability to delete existing module repos.</td>
</tr>
<tr>
<td>Modules</td>
<td>Edit</td>
<td>The ability to edit existing module repos.</td>
</tr>
<tr>
<td>Modules</td>
<td>List</td>
<td>The ability to view existing module repos.</td>
</tr>
<tr>
<td>Modules</td>
<td>Promote</td>
<td>The ability to promote changes in module pipelines.</td>
</tr>
<tr>
<td>Jobs</td>
<td>Create</td>
<td>The ability to create new jobs in Continuous Delivery for PE.</td>
</tr>
<tr>
<td>Jobs</td>
<td>Delete</td>
<td>The ability to delete existing jobs.</td>
</tr>
<tr>
<td>Jobs</td>
<td>Edit</td>
<td>The ability to edit existing jobs.</td>
</tr>
<tr>
<td>Jobs</td>
<td>List</td>
<td>The ability to view all existing jobs and job hardware capabilities.</td>
</tr>
<tr>
<td>Jobs</td>
<td>Run</td>
<td>The ability to run all existing jobs.</td>
</tr>
<tr>
<td>Users</td>
<td>Edit</td>
<td>The ability to add and remove users from Continuous Delivery for PE workspaces.</td>
</tr>
<tr>
<td>Groups</td>
<td>Create</td>
<td>The ability to create new user groups.</td>
</tr>
<tr>
<td>Groups</td>
<td>Delete</td>
<td>The ability to delete existing user groups.</td>
</tr>
<tr>
<td>Groups</td>
<td>Edit</td>
<td>The ability to edit existing user groups by adding or removing members and user permissions.</td>
</tr>
<tr>
<td>Groups</td>
<td>List</td>
<td>The ability to view all existing user groups and their permissions.</td>
</tr>
<tr>
<td>Integrations</td>
<td>Connect</td>
<td>The ability to configure new source control or PE integrations.</td>
</tr>
</tbody>
</table>
### Built-in user group permissions reference

New workspaces in Continuous Delivery for PE versions 4.4.0 and newer include three built-in user groups. These tables list the permissions granted to each built-in group.

#### Administrators group permissions

<table>
<thead>
<tr>
<th>Type</th>
<th>Permissions granted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control repos</td>
<td>Create, list, edit, promote, delete</td>
</tr>
<tr>
<td>Modules</td>
<td>Create, list, edit, promote</td>
</tr>
<tr>
<td>Jobs</td>
<td>Create, list, edit, run, delete</td>
</tr>
<tr>
<td>Users</td>
<td>None</td>
</tr>
<tr>
<td>Groups</td>
<td>Create, list, edit, delete</td>
</tr>
<tr>
<td>Integrations</td>
<td>Connect, disconnect</td>
</tr>
<tr>
<td>Inventory</td>
<td>List</td>
</tr>
</tbody>
</table>

#### Operators group permissions

<table>
<thead>
<tr>
<th>Type</th>
<th>Permissions granted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control repos</td>
<td>Create, list, promote</td>
</tr>
<tr>
<td>Modules</td>
<td>Create, list, promote</td>
</tr>
<tr>
<td>Jobs</td>
<td>Create, list</td>
</tr>
<tr>
<td>Users</td>
<td>None</td>
</tr>
<tr>
<td>Groups</td>
<td>None</td>
</tr>
<tr>
<td>Integrations</td>
<td>None</td>
</tr>
<tr>
<td>Inventory</td>
<td>None</td>
</tr>
</tbody>
</table>

#### Viewers group permissions

<table>
<thead>
<tr>
<th>Type</th>
<th>Permissions granted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control repos</td>
<td>List</td>
</tr>
<tr>
<td>Modules</td>
<td>List</td>
</tr>
<tr>
<td>Jobs</td>
<td>List</td>
</tr>
<tr>
<td>Users</td>
<td>None</td>
</tr>
<tr>
<td>Groups</td>
<td>None</td>
</tr>
<tr>
<td>Integrations</td>
<td>None</td>
</tr>
<tr>
<td>Inventory</td>
<td>None</td>
</tr>
</tbody>
</table>

© 2021 Puppet, Inc.
Using the root console

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

To access the root console as the root user, sign into Continuous Delivery for PE with the root account credentials established during installation. As a super user, select Root console from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.

To exit the root console, click any workspace name.

Designate super users

Any Continuous Delivery for Puppet Enterprise (PE) user except the root user can be designated as a super user. Super users can access the root console to reset other users' credentials, delete users, manage global shared job hardware, and designate the users allowed to manually approve deployments to protected Puppet environments.

To designate super users, you must have root or super user permissions to access the root console.

1. Log into the root console by signing in as the root user or by selecting Root console from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.
2. Click Accounts.
3. Locate the user's name in the list, enable the Super User switch, and confirm the change.
4. Repeat to designate additional super users.

Change a user's password

Users can reset their own passwords at any time by clicking Forgot your password? on the Continuous Delivery for Puppet Enterprise (PE) login screen. If you need to update the root user's password, change a user's password on their behalf, or revoke a user's access to Continuous Delivery for PE, use the root console.

To perform these actions, you must have root or super user permissions to access the root console.

1. Log into the root console by signing in as the root user or by selecting Root console from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.
2. Click Accounts.
3. Locate the user whose password you want to reset and click User settings.
4. Click Change password, enter and confirm the new password, and click Change password.

Reset a user's email address

Users can change their own email address at any time by clicking Change email on the Profile page. If you need to change a user's email address on their behalf, use the root console.

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

1. Log into the root console by signing in as the root user or by selecting Root console from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.
2. Click Accounts.
3. Locate the user whose email address you want to change and click User settings.
4. Enter the user's new email address and click Change email.
Delete a user
Use the root console to permanently delete a user from Continuous Delivery for Puppet Enterprise (PE).

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

1. Log into the root console signing in as the root user or by selecting Root console from the workspaces drop down menu on the Continuous Delivery for PE navigation bar.
2. Click Accounts.
3. Locate the user you want to delete and click Delete user.
4. Confirm that you want to permanently delete the user by entering the user's user name and clicking Yes, delete user.

The user is permanently deleted from your Continuous Delivery for PE installation.

Reviewing node inventory

The nodes from all Puppet Enterprise (PE) instances integrated with a workspace are shown on that workspace's Nodes page. This page gives you a federated view of all the nodes that your workspace deploys code to.

Customize your node table

The Nodes page lists basic information about every node from every Puppet Enterprise (PE) instance integrated with your workspace. You can customize your page view to display node facts relevant to your work.

By default, the node table shows this information about each node:

- Node name.
- Time since the node's most recent PE report was generated.
- Node run status for the most recent Puppet run.
- The PE server the node is associated with.
- The value of the ipaddress fact.
- The value of the operating system fact.

You can add additional columns to the node table to show other fact values (including structured fact values), and you can remove irrelevant columns.

1. In the Continuous Delivery for PE web UI, click Nodes.
2. Click Add/Remove columns to open the column selector.
3. To add a column, search for a fact value and click Add. The fact value appears in the list of columns.
4. Use the check marks next to the columns list to select the columns you want to show. Unchecked columns are hidden.
5. Click Apply to apply your changes to the table.

Create filters to focus on specific node sets

Build basic or compound (multi-element) filters to help you answer complex queries about your nodes. By combining multiple filters with the AND and OR logical operators, you can target specific node sets.

Important:
Due to a processing limitation, you can apply filters to a specific Puppet Enterprise (PE) primary server or to all integrated PE primary servers. Attempting to add a filter to two or more specific primary servers triggers a **Cannot add this filter warning.**

For example, the following filter is valid because it specifies only one primary server and filters the nodes on that server by their operating system.

```
PE primary server = my-server AND Operating system = purpleOS
```

The following filter is invalid:

```
PE primary server = my-server OR Operating system = purpleOS
```

While the `PE primary server` parameter only specifies one server, using the `OR` operator ignores the first filter and attempts to apply the `Operating system` filter to all primary servers.

1. In the Continuous Delivery for Puppet Enterprise (PE) web UI, click **Nodes**.
2. Click **Create filter**, select a filter category, define the filter values:

<table>
<thead>
<tr>
<th>Category</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE primary server</td>
<td>Select from a list of integrated PE instances</td>
</tr>
<tr>
<td>Change status for the most recent Puppet run</td>
<td>Select from a list of statuses</td>
</tr>
<tr>
<td>No-op status for the most recent Puppet run</td>
<td>Noop or Enforced</td>
</tr>
<tr>
<td>Operating system</td>
<td>An operator and a case-sensitive value</td>
</tr>
<tr>
<td>Node group</td>
<td>Select a PE primary server and select from available node groups</td>
</tr>
<tr>
<td>Facter fact value</td>
<td>A fact name, an operator, and a case-sensitive value</td>
</tr>
</tbody>
</table>

3. Click **Add**, and the filter appears on the **Nodes** page. Then:
   - **If you only need one filter**: Go to step 4.
   - **If you want to add another basic filter**: Click **Add new filter**, configure the filter, click **Add**, and then click the logical operator between the filters (**AND** or **OR**) to specify whether nodes must match one or both filters.

4. **If you want to build a compound filter**: Click **Create compound filter** on your basic filter, configure the filter, click **Add**, and then click the logical operator (**AND** or **OR**) to select whether nodes must match one or both filters. Compound filters are like parenthesis in your query.

   **Note:** You cannot mix logical operators within a single compound filter. To create queries with more complex logic, use a combination of basic and compound filters.

4. Click **Apply filters** to apply the filters to your node table.

   To modify the filters, click **Edit filters**.

   To remove filters, click **Edit filters** (if you're not already in edit mode), and click **Remove filter** (the small blue x icon to the right of each individual filter).

---

**Build a fact chart**

Fact charts are visual representations of the distribution of Facter fact values across all nodes managed by the Puppet Enterprise (PE) instances you’ve integrated with Continuous Delivery for PE. You can add custom fact charts to any saved view for quick reference.
Select the Facter facts that are important to your organization and visualize the distribution of unique values for a given fact in a fact chart. For example, you can create a fact chart showing how many Windows Server versions you currently have in your inventory and how many servers are on each unique version.

You can create a fact chart for any Facter fact, including custom facts and structured facts.

1. In the Continuous Delivery for PE web UI, click **Nodes**.
2. Click **Add fact chart** and enter the name of the fact you want to visualize.
3. Select whether to exclude nodes that do not use the specified fact from the fact chart.
   - If the fact chart includes all nodes, the nodes without the specified fact are shown in the **Fact absent** section of the resulting fact chart.
4. Click **Save**. The new fact chart appears in the fact chart section of the page.
5. To edit or remove fact charts:
   a) Click **More actions** on the fact chart you want to edit or remove.
   b) Click **Edit** or **Remove**.

In addition to the legend on each fact chart, you can hover over the fact charts to see details about the number and percentage of nodes with the specified fact value.

**Save custom views** on page 132 to save fact charts for future reference and share them with your colleagues.

---

### Save custom views

After customizing the nodes table and creating fact charts, you can save your customized view for later reference. Saved views are automatically shared with all members of your workspace, and each member can save their favorite views for quick access.

1. Customize the **Nodes** page:
   - **Customize your node table** on page 130
   - **Create filters to focus on specific node sets** on page 130
   - **Build a fact chart** on page 131
2. Click **Save > Save as new** to create a new saved view.
3. Enter a name and optional description for the view.
   - Saved views are added to the library of saved views created by everyone in your workspace. All workspace members can see and use these saved views.
4. Optional: Designate this saved view as one of your favorites.
   - Marking a saved view as a favorite prioritizes that view on the **Saved views** page and when you Select a different view from the **Nodes** page. Each workspace member has their own list of favorite views.
5. Click **Save**.
6. You're redirected to the **Saved views** page where you can access all saved views for this workspace. In the **Actions** column you can:
   - Use the **Star** icon to add or remove views from your favorites.
   - Use the **Trash** icon to delete a view you created.

   **Remember:** Because all workspace members share all saved views, make sure no one else is using a view before you delete it.

7. Click a view's name open the **Nodes** page with that view applied.
   - From the **Nodes** page, you can click **Select a different view** to quickly switch views.

© 2021 Puppet, Inc.
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 want, including module validation, linting, and more.

**Important:** Before adding jobs to pipelines, make sure you Configure job hardware on page 112 and install any additional software, such as Docker or Puppet Development Kit (PDK).

What is a job?

In Continuous Delivery for Puppet Enterprise (PE), jobs test your Puppet code. You create and store jobs in the web UI, where you can run them on demand and add them to automated pipelines to run tests every time new code is committed to a repository.

Jobs are fully customizable and can be written to test any aspect of your code. Jobs can leverage Puppet Development Kit (PDK) and other testing tools created and maintained by the Puppet community, such as:

- rspec-puppet
- onceover
- puppet-lint
- rspec-puppet-facts

Where do jobs run?

Jobs run on designated nodes called job hardware. There are two job hardware types:

- **Global shared job hardware:** These are job hardware servers configured in the root console and available to all workspaces in your Continuous Delivery for PE installation.
- **Workspace hardware:** These are job hardware servers configured in a specific workspace and available to members of that workspace, but not available to other workspaces.

For more information about setting up job hardware, go to Configure job hardware.

**Important:** Be aware of any prerequisites or dependencies needed to run a job, and ensure your job hardware has the necessary software, operating systems, and other resources installed before attempting to run the job.

Global shared job hardware uses a shared Docker image set in the root console. When configuring a job to run on workspace hardware, you can utilize this shared image or use environment variables in the job to overwrite the default image. However:

**Important:** If your installation specifies a Docker image in Puppet Application Manager (PAM) (this is common in offline environments), this image takes precedence over the image set in the root console (which is the one shown when configuring jobs).

Related information

- **Improving job performance by caching Git repositories** on page 121
  If you have large Git repositories, you can enable Git repository caching to improve job performance. Repository caching is disabled by default.
- **Adjusting the timeout period for jobs** on page 121
  In some circumstances, such as when working with large Git repositories, you may need to adjust the length of the job timeout period. Use the settings in this section to customize the job timeout period.
Add secrets to jobs

You can add secrets, such as tokens and certificates, to Continuous Delivery for Puppet Enterprise (PE) jobs. Jobs can use secrets when they run.

**Before you begin**

You must create a job before you can add a secret. This could be one of the Pre-built jobs or a custom job, as demonstrated in the Sample non-Docker-based module jobs on page 136 and Sample non-Docker-based control repo jobs on page 137.

**Important:** Jobs with secrets require puppetlabs-cd4pe_jobs module version 1.6.0 or higher. You install this module when you Configure job hardware on page 112.

Secrets are limited to the job where you create them. If a secret is applicable to multiple jobs, you must add that secret to each relevant job.

When you run jobs that use secrets, you can see where jobs use secrets in the jobs' logs. Sensitive values are redacted.

1. In the Continuous Delivery for PE web UI, click Jobs.
2. Locate the job you want to add a secret to and click Edit.
3. Click Add secret.
4. Enter a Name and (optional) Description for the secret.
   
   Secret names must be unique within each workspace and can only contain letters, numbers, and underscores. If you use a dash or space in a secret name, it is automatically converted to an underscore.

   **Important:** Once you save a secret, you can't change its name. If you need to change a secret's name, you must delete and recreate it.

5. Select the Secret Type, complete the remaining fields according to the secret type, and click Save.

   Once you save the secret, you can't see the information you put in these sensitive fields when you edit the secret.

6. Continuous Delivery for PE stores secrets at environment variables. After saving a secret, Continuous Delivery for PE shows you the environment variable you can use to reference the secret in the code in the Job commands section.

   **Example of partial job command code without a secret variable:**
   ```bash
   curl -H "Authorization: Bearer <TOKEN>" -k -s
   ```

   **Example of partial job command code with a secret variable:**
   ```bash
   curl -H "Authorization: Bearer $CD4PE_SECRET_<SECRET_NAME>" -k -s
   ```

   Secrets with multiple values, such as username and password combinations, generate multiple environment variables. Make sure you use the correct environment variables at the correct locations in your code. You can click the secret's name on the Edit job page to see the secret's environment variable(s) at any time.

7. Click Save changes when you are done adding secrets to the job and customizing the Job commands code.

**Pre-built job reference**

Use these job templates to set up Docker-based jobs for testing control repos and modules. These jobs run inside a Docker container and require job hardware that has a Docker capability.

**Validate the syntax of a control repo's Puppetfile**

To add this job to your Continuous Delivery for Puppet Enterprise (PE) instance:
1. In the Continuous Delivery for PE web UI, click Jobs > New job.
2. Complete the fields:
   a. Name: control-repo-puppetfile-syntax-validate
   b. Description: Validate that a control repo’s Puppetfile is syntactically correct
   c. Job commands: rake -f /Rakefile r10k:syntax
   d. Where can this job run?: Select Run on <WORKSPACE_NAME> hardware
   e. Hardware capabilities: Select Docker and click Apply
   f. Docker configuration: Enable the Run this job in a Docker container switch and select Default image (puppet/puppet-dev-tools)
3. Click Save job.

Configure job hardware on page 113 and ensure Docker is installed before running this job.

Validate the syntax of a control repo’s Puppet templates

To add this job to your Continuous Delivery for PE instance:
1. In the Continuous Delivery for PE web UI, click Jobs > New job.
2. Complete the fields:
   a. Name: control-repo-template-syntax-validate
   b. Description: Validate that a control repo’s Puppet templates are syntactically correct.
   c. Job commands: rake -f /Rakefile syntax:templates
   d. Where can this job run?: Select Run on <WORKSPACE_NAME> hardware
   e. Hardware capabilities: Select Docker and click Apply
   f. Docker configuration: Enable the Run this job in a Docker container switch and select Default image (puppet/puppet-dev-tools)
3. Click Save job.

Configure job hardware on page 113 and ensure Docker is installed before running this job.

Validate the syntax of a control repo’s Hiera data

To add this job to your Continuous Delivery for PE instance:
1. In the Continuous Delivery for PE web UI, click Jobs > New job.
2. Complete the fields:
   a. Name: control-repo-hiera-syntax-validate
   b. Description: Validate that a control repo’s Hiera data is syntactically correct.
   c. Job commands: rake -f /Rakefile syntax:hiera
   d. Where can this job run?: Select Run on <WORKSPACE_NAME> hardware
   e. Hardware capabilities: Select Docker and click Apply
   f. Docker configuration: Enable the Run this job in a Docker container switch and select Default image (puppet/puppet-dev-tools)
3. Click Save job.

Configure job hardware on page 113 and ensure Docker is installed before running this job.

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

To add this job to your Continuous Delivery for PE instance:
1. In the Continuous Delivery for PE web UI, click Jobs > New job.
2. Complete the fields:
   a. Name: control-repo-manifest-validate
   b. Description: Validate that a control repo’s Puppet manifest code is syntactically correct
   c. Job commands: rake -f /Rakefile syntax:manifests
   d. Where can this job run?: Select Run on <WORKSPACE_NAME> hardware
   e. Hardware capabilities: Select Docker and click Apply
   f. Docker configuration: Enable the Run this job in a Docker container switch and select Default image (puppet/puppet-dev-tools)

3. Click Save job.

Configure job hardware on page 113 and ensure Docker is installed before running this job.

Validate the syntax of a module’s Puppet manifest code

To add this job to your Continuous Delivery for PE instance:
1. In the Continuous Delivery for PE web UI, click Jobs > New job.
2. Complete the fields:
   a. Name: module-pdk-validate
   b. Description: Validate that a module’s Puppet manifest code is syntactically correct
   c. Job commands: pdk validate --parallel
   d. Where can this job run?: Select Run on <WORKSPACE_NAME> hardware
   e. Hardware capabilities: Select Docker and click Apply
   f. Docker configuration: Enable the Run this job in a Docker container switch and select Default image (puppet/puppet-dev-tools)

3. Click Save job.

Configure job hardware on page 113 and ensure Docker is installed before running this job.

Run rspec-puppet unit tests on a module

To add this job to your Continuous Delivery for PE instance:
1. In the Continuous Delivery for PE web UI, click Jobs > New job.
2. Complete the fields:
   a. Name: module-rspec-puppet
   b. Description: Run rspec-puppet unit tests on a module
   c. Job commands: pdk test unit
   d. Where can this job run?: Select Run on <WORKSPACE_NAME> hardware
   e. Hardware capabilities: Select Docker and click Apply
   f. Docker configuration: Enable the Run this job in a Docker container switch and select Default image (puppet/puppet-dev-tools)

3. Click Save job.

Configure job hardware on page 113 and ensure Docker is installed before running this job.

Sample non-Docker-based module jobs

You can use these Continuous Delivery for Puppet Enterprise (PE) job templates as-is or customize them to your deployment's needs. To add jobs, go to Jobs > New job in the Continuous Delivery for PE web UI.

Note: When configuring jobs that are not Docker-based, you can use the $REPO_DIR environment variable to reference the directory containing the relevant module repo.
Puppet Development Kit validation tests

These jobs validate module code against Puppet Development Kit (PDK). Use the configuration for your operating system.

For Linux:
- **Job name**: module-pdk-validate-linux
- **Description**: Validate via PDK
- **Job commands**: `pdk validate`
- **Hardware capabilities**: Linux

For Windows:
- **Job name**: module-pdk-validate-windows
- **Description**: Validate via PDK
- **Job commands**: `powershell.exe -c "pdk validate"`
- **Hardware capabilities**: Windows

To use these jobs you must Install PDK and install `puppet-agent` on the job hardware.

Puppet Development Kit rspec-puppet tests

These jobs run unit tests on your module code with rspec-puppet. Use the configuration for your operating system.

For Linux:
- **Job name**: module-pdk-test-unit-linux
- **Description**: Run unit tests via PDK
- **Job commands**: `pdk test unit`
- **Hardware capabilities**: Linux

For Windows:
- **Job name**: module-pdk-test-unit-windows
- **Description**: Run unit tests via PDK
- **Job commands**: `powershell.exe -c "pdk test unit"`
- **Hardware capabilities**: Windows

To use these jobs you must Install PDK and install `puppet-agent` on the job hardware.

Sample non-Docker-based control repo jobs

You can use these Continuous Delivery for Puppet Enterprise (PE) job templates as-is or customize them to your deployment's needs. To add jobs, go to Jobs > New job in the Continuous Delivery for PE web UI.

- **Note**: When configuring jobs that are not Docker-based, you can use the $REPO_DIR environment variable to reference the directory containing the relevant module repo.

- **Note**: You can Add secrets to jobs on page 134 after you create them.

Syntax validation

This job validates the syntax of everything in your control repo.
To use this job you must use a *nix host and install puppet-agent on the job hardware.

- **Job name:** control-repo-validate-linux
- **Description:** Validate syntax
- **Hardware capabilities:** Linux
- **Job commands:**

```bash
#!/bin/bash
shopt -s globstar nullglob
green="$\$(tput setaf 2)"
red="$\$(tput setaf 1)"
reset="$\$(tput sgr0)"

for f in **/**pp; do
    [[ $f =~ plans/ ]] && continue
    if puppet parser validate "$f"; then
        echo "$\{green\}SUCCESS: $f\$\{reset\}"
    else
        echo "$\{red\}FAILED: $f\$\{reset\}"
        failures+=("$f")
    fi
done

if (( ${#failures[@]} > 0 )); then
    echo "$\{red\}Syntax validation on the Control Repo has failed in the following manifests:"
    echo -e \"\t\${failures[@]}\$\{reset\}\"
    exit 1
else
    echo "$\{green\}Syntax validation on the Control Repo has succeeded.\$\{reset\}"
fi
```

**Puppet linter**

This job checks the Puppet code in your control repo for programming and stylistic errors.

To use this job you must use a *nix host and install puppet-agent on the job hardware.

- **Job name:** control-repo-lint-linux
- **Description:** Lint Puppet code
- **Hardware capabilities:** Linux
- **Job commands:**

```bash
#!/bin/bash
shopt -s globstar nullglob
green="$\$(tput setaf 2)"
red="$\$(tput setaf 1)"
reset="$\$(tput sgr0)"

LINT_OPTS=("--fail-on-warnings" "--no-documentation-check" "--no-140chars-check" "--no-autoloader_layout-check" "--no-class_inherits_from_params_class-check")

sudo /opt/puppetlabs/puppet/bin/gem install puppet-lint || {
    echo "$\{red\}Failed to install puppet-lint gem\"*
    exit 2
}
```
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:
	 ${failures[@]}${reset}"
    exit 1
else
    echo "${green}Puppet-lint validation on the Control Repo has succeeded.
${reset}"
fi

Key concepts

• Working with Git branches in Continuous Delivery for PE on page 139
Continuous Delivery for Puppet Enterprise (PE) watches for Puppet code changes you make in Git and helps you deploy those changes to your environment node groups. You and Continuous Delivery for PE work together, so it's important to understand what you need to do in Git, and what Git work Continuous Delivery for PE handles for you.

• Understanding the Continuous Delivery for PE workflow on page 141
Continuous Delivery for Puppet Enterprise (PE) helps you to deliver changes to your organization's infrastructure-as-code. Use this workflow to develop and deploy changes with Continuous Delivery for PE.

• Using the feature branch workflow on page 143
Use the feature branch workflow to safely and efficiently move new code from an isolated development environment, through initial testing and validation, and then into your organization's process for staging, review, and promotion to production. Adapt the steps of this Continuous Delivery for Puppet Enterprise (PE) workflow as needed for your team's requirements and best practices.

Working with Git branches in Continuous Delivery for PE

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

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

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

You must create new environment branches when you first set up environment node groups. Environment branches are long-lived Git branches that control which code version is available in corresponding Puppet environments. Continuous Delivery for PE owns environment branches after you create them; it deploys changes to them and keeps them updated. You and your team don't change these branches directly because manual changes to environment branches are overwritten when an automated deployment occurs.
While you focus on committing to the main branch and merging pull requests, Continuous Delivery for PE works in your environment branches on your behalf. It makes sure your commits or changes are tested, deployed, and promoted to your environment node groups in accordance with your control repo pipelines and your manual deployment directives.

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

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

## Organizing Puppet content in your source control repository

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

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

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

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

### Control repos

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

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

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

### Puppet module repos

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

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

### Other non-module repos

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

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

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

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

Tip: Before you begin, review Working with Git branches in Continuous Delivery for PE on page 139.

Workflow phase 1: Developing changes

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

This is the general change development process:

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

Important: Once merged into the main branch, your change is finished. However, a finished change is not the same as a deployed change. Finished changes are ready to be tested and deployed using the deployment process described in Workflow phase 2: Deploying changes on page 141.

Note on continuous integration

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

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

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

Workflow phase 2: Deploying changes

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

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

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

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

© 2021 Puppet, Inc.
Environments

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

When defining environment node groups:

- Each node can only belong to a single environment node group.
- You can name the environment node groups anything that makes sense to your organization.
- Each environment node group must be assigned a unique Puppet environment value. Functionally, this relates to how Continuous Delivery for PE performs bookkeeping of the environment's content in Git (using the bookkeeping branches mentioned in Working with Git branches in Continuous Delivery for PE on page 139). Unique values ensure simple bookkeeping and that deployments do not affect nodes they are not supposed to.

Deploying changes from control repos

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

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

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

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

Deploying changes from module repos

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

Pipelines, or the path to production

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

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

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

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

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

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

Workflow phase 1: Develop and test code changes

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

Before you begin

Create a regex branch pipeline on page 149 for the control repo or module repo you’re working on. This pipeline can be as simple or complex as you need it to be, but at minimum it must:

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

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

This is the same pipeline formatted as code:

```
pipelines:
```
In your source control system, navigate to the control repo or module repo you want to update.

2. Create a feature branch by create a new branch based on the main branch and naming it `feature_<BRANCHNAME>`. The `<BRANCHNAME>` can be a ticket number, fix description, or feature name.

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

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

3. On your feature branch, make your code changes.

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

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

**How does a feature branch deployment work?** Using Code Manager, Continuous Delivery for PE deploys your committed code to a Puppet environment with the same name as the branch that triggered the pipeline. If an environment with this name doesn't already exist (as is most likely the case), Continuous Delivery for PE creates it.
5. When the feature branch deployment is complete, run Puppet on a test node, specifying the special environment created by Continuous Delivery for PE. This Puppet run applies your changes to the test node so you can review them. You can run Puppet from the PE console or the command line:

a) From the PE console, navigate to the Run Puppet page and create a job with the following settings, and click Run job:

- Job description: Enter a description for your test run.
- Environment: Select Select an environment for nodes to run in and choose the environment that matches your feature branch name.
- Schedule: Now.
- Run Mode: Leave these options unchecked.
- Inventory: Select Node list and add the name of your test node.

b) From the command line, run:

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

Tip: Running Puppet on demand from the CLI provides more information on the `puppet job run` command.

6. When the Puppet run is complete, navigate to the test node and review the changes. If further work is needed, repeat steps 3 through 5 until your code is ready for deployment to production. When you're satisfied with your new code, move on to Workflow phase 2: Review and merge to production on page 145.

**Workflow phase 2: Review and merge to production**

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

**Before you begin**

Complete Workflow phase 1: Develop and test code changes on page 143 and make sure your main branch pipeline:

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

1. In your source control system, create a pull request from your feature_<BRANCHNAME> branch to the main branch.

2. The pull request triggers your main branch pipeline to test your PR code against the jobs and impact analysis report tasks included in the pipeline up to the PR gate. You can monitor the pipeline's progress in the Continuous Delivery for PE web UI.

   You can also see whether each pipeline stage succeeded or failed on the pull request page in your source control system.

3. Your team's stakeholders can review the results of the jobs and impact analysis reports generated by the pipeline. If the PR is approved, merge it into the main branch.

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

Merging a PR creates a commit to the main branch and automatically re-triggers the main branch pipeline. The pipeline starts over from the top, re-running all the jobs and impact analysis tasks to ensure the newly updated main branch code is fully tested and vetted. However, because a commit triggered this pipeline run, the pipeline proceeds past the PR gate and performs any tasks included in the pipeline after the PR gate, such as additional jobs or deployment to a staging environment.
4. Once the code is deployed to a staging environment, perform additional testing by running Puppet against test or staging nodes. Make sure to specify the staging environment. The scope of this testing is determined by your team or company best practices.

5. When testing is complete and the code changes are approved, promote the code to the production environment.

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

Deployments to protected environments require review and approval from a member of a designated approval group before proceeding. These deployments remain in a pending state until an approval decision is provided. Require approval for deployments to protected Puppet environments on page 176 explains how to set up approval groups.

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

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

**Constructing pipelines**

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

**Stages and tasks**

Pipelines in Continuous Delivery for PE are made up of stages and tasks. Tasks include deployments, impact analyses, and jobs to test code. Stages are groups of tasks. Stages allow you to break pipelines into a series of sequential task sets. This allows you to have logical control in your pipeline, such as:

- Start task B only if all tasks in stage A succeed.
- If a task in stage A fails, stop the pipeline and report the error.

You can configure pipelines to require manual approval to proceed to the next stage or advance automatically based on your defined conditions.

**Building and managing pipelines: web UI or code**

You can build and manage pipelines with the Continuous Delivery for PE web UI or a YAML file. Using the web UI is the simplest method for building and managing pipelines, and it is easier to make iterative changes with the web UI controls. Using a YAML file to build pipelines as code is more complex but ideal if you:

- Need a record of changes to your pipeline over time
- Want to avoid manually creating similar pipelines for many control repos or modules.
- Make changes to your Puppet code that require new pipeline definitions and you want to commit changes to your pipelines-as-code at the same time.

**Important:** You can't mix these methods within a single control repo or module. Web UI controls are disabled for pipelines that are managed as code. The following image compares pipelines in the web UI. The pipeline managed in the web UI has controls available, such as deleting the
pipeline, deleting tasks, and adding stages. The pipeline managed as code does not have these

**Managed in the web UI**

**Pipelines**

- **master**

**Pipeline trigger: Commit**

**Code Validation stage**

- **Job - control-repo-puppetfile-syntax-validate**
- **Job - control-repo-template-syntax-validate**

- **Auto promote**

**Deployment stage**

- **Policy:** deployments::rolling
- **Instance:** cdpe-delivery
- **Environment:** production

**+ Add stage**
Constructing pipelines in the web UI on page 148
Build and manage pipelines for your control repo or module in the Continuous Delivery for Puppet Enterprise (PE) web UI using the tools built into the interface. The steps on this page walk through the process of setting up and using a basic pipeline.

Constructing pipelines from code on page 151
If you manage your pipelines with code, rather than in the web UI, you can maintain a record of pipeline changes over time. When you choose to manage pipelines with code, a .cd4pe.yaml file, containing the pipelines' definitions for a control repo or module, is stored in your source control system alongside the Puppet code for that control repo or module.

Constructing pipelines in the web UI

Build and manage pipelines for your control repo or module in the Continuous Delivery for Puppet Enterprise (PE) web UI using the tools built into the interface. The steps on this page walk through the process of setting up and using a basic pipeline.

Create a pipeline

Set up a pipeline to enable automatic testing of new code by adding stages and jobs in the Continuous Delivery for Puppet Enterprise (PE) web UI.

Every pipeline must have at least one stage.

1. In the Continuous Delivery for PE web UI, click Control repos, and click the name of the control repo you want to create a pipeline for.

   Continuous Delivery for PE automatically creates a pipeline for the branch you selected when you set up the control repo (the main branch). The branch name is at the top of the Pipelines section. You can set up pipelines for other branches in the repo if you want to automatically test code changes that Continuous Delivery for PE makes on your behalf. To create a pipeline for a different branch, click Add pipeline. If you don't want a pipeline, click Delete pipeline.

2. You can also go to Modules to create pipelines for module repos.

3. Click Add stage and enter a Stage name.

4. Select Jobs from the Select item dropdown menu, select a job from the Select jobs list, click Add stage to add the job to the pipeline, and click Done.

   a) To add another job to the same stage, click More actions, select Add item to stage, and repeat step 3.

   b) To add another stage to your pipeline, repeat steps 2 and 3.

   Note: All steps (jobs, deployments, and so on) in a stage run concurrently. If you need certain steps to occur after others, use stages to separate them.

   To reorganize, delete, or rename stages, click More actions.
5. Once you have a pipeline with two stages, an promotion step is automatically added between the stages. Select Auto promote and choose a promotion condition from the dropdown menu. The promotion condition tells the pipeline how to decide if it can continue to the next stage. The promotion conditions, from most to least restrictive, are:

- **All succeeded**: The pipeline continues to the next stage only if all tasks in the current stage finish with "succeeded" or "done" status.
- **All completed**: The pipeline continues to the next stage if all tasks in the current stage finish with "succeeded," "done," or "failed" status. The pipeline stops if any task in the current stage was canceled.
- **Any succeeded**: The pipeline continues to the next stage if at least one task in the current stage finishes with a "succeeded" or "done" status.
- **Any completed**: The pipeline continues to the next stage if at least one task in the current stage finishes with a "succeeded," "done," or "failed" status. The pipeline stops if all tasks in the current stage were canceled.

6. Optional: If you want the pipeline to run on both commits and pull requests, click Manage pipelines to enable pull requests as a pipeline trigger.

This pipeline runs on one branch in one control/module repo.

**Related information**

Testing Puppet code with jobs on page 133

Jobs are fully customizable tests for your Puppet code. You can create a job that runs any sort of test you want, including module validation, linting, and more.

**Create a regex branch pipeline**

Most pipelines are linked to a single branch in your control repo or module repo. A regex branch pipeline can recognize and act on changes to any branch in your control/module repo that has a name matching the regular expression you set.

A regex branch pipeline lets you use a single pipeline for all feature branches you and your team create (and destroy) in the process of developing new code. Regex branch pipelines use naming conventions to detect branches to run the pipeline against. You define the naming convention when you create the pipeline. When you commit to (or create a pull request for) a feature branch that has a name matching the naming convention, Continuous Delivery for Puppet Enterprise (PE) automatically runs the regex pipeline against that branch.

1. In the Continuous Delivery for PE web UI, click Control Repos or Modules and select the control repo or module repo where you want to create the regex pipeline.

2. 
   Click Add pipeline.

3. In the Add pipeline window, select Branch regex.
   Each control repo or module repo can have only one regex branch pipeline. If the Branch regex option is not available, then a regex branch pipeline already exists for this repo.

4. In the Configure regex field, enter the naming convention, as a regular expression, that you want the regex branch pipeline to use to detect feature branches (and initiate pipeline runs against those branches).
   The default regular expression is feature_. *.
   If you use the default regex, your feature branch names must follow the feature_<BRANCHNAME> naming convention.
   For example, a branch named feature_1234 triggers the pipeline, but a branch named 1234_feature does not.

   **Tip:** All team members must follow this naming convention when creating feature branches if they want their changes to trigger the regex branch pipeline. We strongly recommend using the default regular expression or one that's similarly simple and memorable.

5. Click Add pipeline. The regex branch pipeline is created.

6. Click Manage pipelines to select whether the regex branch pipeline is triggered by commits, pull requests, or both.

7. Build out your pipeline by adding stages, jobs, impact analysis tasks, deployments, and promotion conditions.

This pipeline runs on any branch that matches the defined naming convention in the associated control/module repo.

© 2021 Puppet, Inc.
Test code automatically with a pipeline

After creating a pipeline for your control repo's main branch, when you commit new code to the main branch, the pipeline tests the code (by running the jobs you selected) and reports any errors or failures.

**Before you begin**

Create a pipeline on page 148

1. To trigger the pipeline, make a trivial change, such as adding a line to a README file, in your control repo's main branch, and commit your change.

2. In the Continuous Delivery for PE web UI, click Control repos and click the name of the control repo you just committed to.

3. The Events area shows the push event initiated by your commit and an entry for each job in the pipeline. Job statuses appear here and update to report success or failure when a job run is complete.

   Each event summary includes a link to the commit in the control repo, and each job event includes a job number. Click the job number to go to the Job details page, where you can see the job's log for its run and review error messages.

Test pull requests automatically

If you enable pull requests as a pipeline trigger, your pipeline automatically tests new code when a pull request (PR) is opened against the relevant branch. Add PR gates to pipelines to test new code but stops the pipeline before it can advance to further tests or deployment.

**Before you begin**

Create a pipeline on page 148 that has pull requests as a pipeline trigger.

A PR gate stops the pipeline at a certain point if the pipeline was triggered by a pull request. With a PR gate, you can automatically run acceptance tests on new pull requests while manually managing additional testing or deployment based on the results of the initial automated tests.

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

1. To add a PR gate to your pipeline, navigate to the pipeline in the web UI.

2. In your pipeline's first stage, click More actions, select Add item to stage, and select Pull request gate.

3. Click Add PR gate, and click Done if the PR gate was added successfully.

   **Important:** A pipeline can only have one PR gate. To move a PR gate, delete the existing gate and create a new one in the desired stage.

4. To test PR gate, switch to a branch in your control repo other than the main branch. Make a trivial change (such as adding a new line to the README file) in your control repo, commit your change, and open a pull request against the main branch for this commit.

   **Remember:** You must make the pull request against the branch associated with the pipeline.

5. In the Continuous Delivery for PE web UI, click Control repos, then click the name of the control repo you just created the pull request against.

6. The Events area shows the pull request event initiated by your commit and entries for jobs in your pipeline before the PR gate. Notice that jobs in pipeline stages beyond the PR gate have not been triggered.

Deploy code automatically with a pipeline

You can use a control repo pipeline to automatically deploy new code to a specified set of nodes every time a commit is made.
Before you begin

Create a pipeline on page 148

1. In the Continuous Delivery for PE web UI, click Control repos, and click the name of the control repo you've created a pipeline for.
2. Click Add stage.
3. In the Add new stage window, select Deployment from the Select item dropdown menu.
4. Select your Puppet Enterprise instance and the node group you want to deploy changes to when this pipeline runs.
5. Select a deployment policy. If this is your first pipeline and you're just testing the process, select Direct deployment policy.

Built-in deployment policies on page 169 explains the four deployment policies and how they work.
6. Optional: Set termination conditions for this pipeline's deployments, and choose the number of nodes that can fail before the deployment is stopped.
7. Click Add Stage and click Done. Your new stage, containing the deployment details, is added to the pipeline.
8. To test the deployment, make a trivial change (such as adding a new line to the README file) on the main branch of your control repo and commit your change.
9. In the Continuous Delivery for PE web UI, click Control repos and click the name of the control repo you just committed to.
10. The Events area shows the push event initiated by your commit and a deployment event. The deployment status appears here and updates to report success or failure when the deployment is complete.

Each event summary includes a link to the commit in the control repo, and deployment events include a deployment number. Click the deployment number to go to the deployment details page.
11. You can run the same deployment again from the web UI by retriggering the webhook. In the Events area, locate the push event you want to rerun and click View webhook.

The webhook's request and response data is shown in the Webhook data pane. This can be useful for troubleshooting.
12. Click Redeliver webhook and confirm your action. The Events area shows the new event triggered by the redelivery of the webhook.

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

Constructing pipelines from code

If you manage your pipelines with code, rather than in the web UI, you can maintain a record of pipeline changes over time. When you choose to manage pipelines with code, a .cd4pe.yaml file, containing the pipelines' definitions for a control repo or module, is stored in your source control system alongside the Puppet code for that control repo or module.

• Configuring your pipelines for management with code on page 151
To manage pipelines-as-code, you need a .cd4pe.yaml file containing the pipeline definitions. Use these steps to convert existing pipelines to code, update pipelines-as-code, or create new pipelines for new repos.

• Structuring a .cd4pe.yaml file on page 154
When managing your pipelines with code, pipeline definitions are expressed in a structured format and stored in a .cd4pe.yaml file that is stored in your control repo or module repo.

Configuring your pipelines for management with code

To manage pipelines-as-code, you need a .cd4pe.yaml file containing the pipeline definitions. Use these steps to convert existing pipelines to code, update pipelines-as-code, or create new pipelines for new repos.

Choose the procedure appropriate for your circumstances:

• If you created pipelines for a control repo or module in the Continuous Delivery for Puppet Enterprise (PE) web UI and you now want to manage those pipelines with code, you can Convert your existing pipelines to code on
Continuous Delivery for PE generates a `.cd4pe.yaml` file based on the repo's existing pipelines. You can then **Update your pipelines with a new `.cd4pe.yaml` file** on page 152 when you need to make changes to the pipelines.

- If you've created a `.cd4pe.yaml` file you want to use to replace existing pipelines (that you created in the web UI or with another `.cd4pe.yaml` file), you need to **Update your pipelines with a new `.cd4pe.yaml` file** on page 152.
- If you have created your own `.cd4pe.yaml` file for a new repo that you haven't yet added to Continuous Delivery for PE, you can **Create new pipelines using a `.cd4pe.yaml` file**.

**Note:** This process only applies to new repos you haven't yet added to Continuous Delivery for PE. If you already added a repo to Continuous Delivery for PE, you need to **Convert your existing pipelines to code** on page 152 or **Update your pipelines with a new `.cd4pe.yaml` file** on page 152.

- If you want to re-enable the pipeline controls in the web UI, you can **Stop managing your pipelines with code** on page 153.

### Convert your existing pipelines to code

If you've created pipelines using the Continuous Delivery for Puppet Enterprise (PE) web UI, and now want to manage those pipelines with code, Continuous Delivery for PE can render your existing pipelines in YAML format.

#### Before you begin

Add a control repo or module to Continuous Delivery for PE, configure job hardware, and set up a pipeline in the web UI:

- **Add repositories** on page 115
- **Configure job hardware** on page 112
- **Constructing pipelines in the web UI** on page 148

1. In the Continuous Delivery for PE web UI, navigate to the control or module repo that has pipelines you want to manage with code.
2. At the top of the **Pipelines** section, click **Manage pipelines > Manage as code**.
3. Continuous Delivery for PE displays your pipelines in YAML format. Copy the YAML code by clicking **Copy to clipboard** at the bottom of the code block.
4. Create a new `.yaml` file named `.cd4pe` and paste the YAML code into it.

   **Important:** The complete file name must be `.cd4pe.yaml`. Make sure you do not miss the leading period.

5. Save the `.cd4pe.yaml` file to the repo's root directory (on any branch) and commit the change in your source control system.
6. Go to the control or module repo in the Continuous Delivery for PE web UI, click **Manage pipelines > Manage as code**, and, in the **Select branch** area, select the branch where you saved the `.cd4pe.yaml` file.
7. Click **Save settings**.

Continuous Delivery for PE reads the contents of the `.cd4pe.yaml` file before every pipeline run, and it uses the YAML code to render the pipelines' definitions in the web UI. Because you're now managing this repo's pipelines with code, the pipeline controls in the web UI are disabled for this repo.

If you need to change this repo's pipelines, **Update your pipelines with a new `.cd4pe.yaml` file** on page 152. If you want to go back to managing this repo's pipelines with the web UI, you can **Stop managing your pipelines with code** on page 153.

### Update your pipelines with a new `.cd4pe.yaml` file

You can use a `.cd4pe.yaml` file to replace pipelines you created in the Continuous Delivery for Puppet Enterprise (PE) web UI or to update pipelines you're already managing with code.
Before you begin
You need a properly formatted .cd4pe.yaml file containing the definitions of the pipelines you want to create or update. To learn about .cd4pe.yaml file syntax, go to Structuring a .cd4pe.yaml file. If you were previously managing pipelines in the web UI, you can Convert your existing pipelines to code on page 152 and then update the resulting .cd4pe.yaml file, rather than creating a completely new file.

1. Add the new or updated .cd4pe.yaml file to the root directory of your control repo or module repo, and commit the file to your source control system.

   If you are updating an existing .cd4pe.yaml file, you can replace the previous file. If you save the updated file on a different branch, you need to change the branch in the Continuous Delivery for PE web UI.

   If the repo does not already have a .cd4pe.yaml file, you can save the file to the root directory on any branch in the repo.

2. Go to the control or module repo in the Continuous Delivery for PE web UI, click Manage pipelines > Manage as code, and, in the Select branch area, select the branch where you saved the .cd4pe.yaml file.

3. Click Save settings.

   Continuous Delivery for PE reads the contents of the .cd4pe.yaml file before every pipeline run, and it uses the YAML code to render the pipelines' definitions in the web UI. Because you're now managing this repo's pipelines with code, the pipeline controls in the web UI are disabled for this repo.

   Repeat this process when you need to change this repo's pipelines. If you want to switch to managing this repo's pipelines with the web UI, you can Stop managing your pipelines with code on page 153.

Create new pipelines using a .cd4pe.yaml file
You can create pipelines-as-code for a new control repos or module repos that you haven't yet added to Continuous Delivery for Puppet Enterprise (PE). If a .cd4pe.yaml file exists in the repo when you first add it to Continuous Delivery for PE, the software detects the file and asks if you want to use it to build and manage your pipelines.

Before you begin
You need a properly formatted .cd4pe.yaml file containing the definitions of the pipelines you want to create. To learn about .cd4pe.yaml file syntax, go to Structuring a .cd4pe.yaml file.

Note: This process only applies to new repos you haven't yet added to Continuous Delivery for PE. If you already added a repo to Continuous Delivery for PE, you need to Convert your existing pipelines to code on page 152 or Update your pipelines with a new .cd4pe.yaml file on page 152.

1. Add a .cd4pe.yaml file to the root directory of the main branch of your control repo or module repo. Commit the file to your source control system.

2. Add the control repo or module to Continuous Delivery for PE.

   For instructions, go to Add repositories on page 115.

3. After adding the repo, Continuous Delivery for PE detects the .cd4pe.yaml file and asks if you want to use it to build and manage your pipelines. Click Confirm.

   Continuous Delivery for PE reads the contents of the .cd4pe.yaml file before every pipeline run, and it uses the YAML code to render the pipelines' definitions in the web UI. Because you're managing this repo's pipelines with code, the pipeline controls in the web UI are disabled for this repo.

   If you need to change this repo's pipelines, Update your pipelines with a new .cd4pe.yaml file on page 152. If you want to start managing this repo's pipelines with the web UI, you can Stop managing your pipelines with code on page 153.

Stop managing your pipelines with code
If you want to use the web UI, instead of a .cd4pe.yaml file, to manage your pipelines, you can quickly make this switch in the Continuous Delivery for Puppet Enterprise (PE) web UI.

1. In the Continuous Delivery for PE web UI, navigate to the control repo or module that has pipelines you are currently managing with a .cd4pe.yaml file.
2. At the top of the **Pipelines** section, click **Manage pipelines > Manage in the web UI**, and make any necessary adjustments to the pipeline settings.

3. Click **Save settings**.

The pipeline controls in the web UI are enabled. Continuous Delivery for PE ignores your `.cd4pe.yaml` file and you must make all pipeline changes in the web UI.

**Tip:** To avoid confusion, remove the `.cd4pe.yaml` file from the repo's root directory in source control.

If you want to go back to managing pipelines as code, you can **Convert your existing pipelines to code** on page 152.

### Structuring a `.cd4pe.yaml` file

When managing your pipelines with code, pipeline definitions are expressed in a structured format and stored in a `.cd4pe.yaml` file that is stored in your control repo or module repo.

Your `.cd4pe.yaml` file must use this structure:

- `spec_version`: The version of the pipelines-as-code file specification you've used to write this YAML file. Currently, only one specification version, `v1`, is available.
- `config`: Pipeline configuration settings for the control repo or module repo as a whole.
- `pipelines`: The names of pipelines you're creating, either the branch name (for fixed-branch pipelines) or the naming convention (for regex branch pipelines). Nested under each named pipeline are:
  - `triggers`: What source control activities the pipeline listens for.
  - `stages`: Parts of the pipeline. Each stage includes:
    - `name`: The stage's name.
    - `auto_promote`: The promotion condition that determines if and how the pipeline proceeds to the next stage after completing steps in the current stage.
    - `steps`: Details about the exact work the pipeline performs in each stage, such as jobs and deployments. Each step must specify a `type`. Additional specifications depend on the type.

Go to [`.cd4pe.yaml` file syntax](#) on page 157 for details about specific parameters and definitions. Go to [Validating your `.cd4pe.yaml` file](#) on page 157 to learn how to check if your file is well formed.

Here's an example of a complete `.cd4pe.yaml` file for a control repo. This file describes a regex branch pipeline and a main branch pipeline. Notice the **regex branch pipeline** is identified by the feature branch naming convention, `/*`

```yaml
spec_version: v1
config:
  enable_pull_requests_from_forks: false
  deployment_policy_branch: production
  enable_pe_plans: true
pipelines:
  /feature_*/:
    triggers:
      - pull_request
      - commit
    stages:
      - name: "Lint/Parser validation"
        auto_promote: all_succeeded
        steps:
          - type: job
            name: control-repo-puppetfile-syntax-validate
          - type: job
            name: control-repo-template-syntax-validate
          - name: "Deploy feature environment"
            steps:
              - type: deployment
```

© 2021 Puppet, Inc.
Using a `.cd4pe.yaml` file with a module

A `.cd4pe.yaml` file for a module is almost identical to one for a control repo, with these exceptions:

- If the module pipeline includes an impact analysis step, you must include a pointer to the control repo used in the deployment associated with the impact analysis task.
• If the .cd4pe.yaml file includes a regex branch pipeline using the Feature branch deployment policy on page 173, the deployment step must include the name of the control repo associated with the module and the name of the control repo branch on which to base the feature branches Continuous Delivery for PE creates.

Here's an example of a complete .cd4pe.yaml file for a module with an impact analysis step. Notice, at the end of the pipeline, the target parameter for the type: deployment step includes a control_repo pointer for the impact analysis task.

```yaml
spec_version: v1
config:
  enable_pull_requests_from_forks: false
pipelines:
  staging:
    triggers:
      - commit
    stages:
      - name: "Impact analysis"
        auto_promote: false
        steps:
          - type: impact_analysis
            deployments:
              - "Deployment to staging-rustler-1"
              concurrent_compilations: 10
              all_deployments: false
        - name: "Deployment to staging"
          steps:
            - type: deployment
              name: "Deployment to staging-rustler-1"
              policy:
                name: eventual_consistency
                timeout: 3600000
                concurrent_compilations: 0
                all_deployments: false
              pe_server: pe-github
              target:
                type: node_group
                node_group_id: 250fd263-8456-4114-b559-9c6d9fa27748
              control_repo: cdpe-code-rustler-app-2020
```

Here's an example of a module's regex branch pipeline using the feature branch deployment policy. The pipeline's name is the feature branch naming convention. At the end of the pipeline, the control_repo and base_feature_branch parameters in the type: deployment step are required for this type of pipeline. These parameters specify which control repo is associated with this module and which branch of that control repo is used to create feature branches when this pipeline is triggered.

```yaml
spec_version: v1
config:
  enable_pull_requests_from_forks: false
pipelines:
  /feature_*/:
    triggers:
      - commit
    stages:
      - name: "Validation stage"
        auto_promote: any_completed
        steps:
          - type: job
            name: "module-pdk-validate"
      - name: "Feature branch deployment stage"
        auto_promote: false
        steps:
          - type: deployment
```

© 2021 Puppet, Inc.
Validating your `.cd4pe.yaml` file

Every time you commit a change to your `.cd4pe.yaml` file in your control repo or module repo, Continuous Delivery for PE uses the YAML code to render the pipelines' definitions in the web UI. To ensure that you commit only well-formed YAML code, use the validation tool in the Continuous Delivery for PE web UI before committing.

1. In the Continuous Delivery for PE web UI, navigate to the control repo or module whose pipelines you're managing with code.
2. At the top of the Pipelines section, click Manage pipelines.
3. Select Manage as code. Continuous Delivery for PE displays your current pipelines in YAML format.
4. Update the code in the window with the changes you wish to make. Alternatively, you can delete the contents of the window and paste in code you've written elsewhere.
5. To check the syntax of your code, click Validate.
   - If your syntax is invalid, an error message about the location of the issue appears.
   - If your syntax is valid, Copy to clipboard is activated.
6. Once your changes are complete and successfully validated, copy them into the `.cd4pe.yaml` file in your source control or module and commit your change.

`.cd4pe.yaml` file syntax

Your `.cd4pe.yaml` file must be formatted properly.

Spec version

Every `.cd4pe.yaml` file begins with a spec_version declaration expressed as `v<VERSION NUMBER>`. This sets the version of the Continuous Delivery for Puppet Enterprise (PE) pipelines-as-code specification used to write the file.

As there is only one version of the Continuous Delivery for PE pipelines-as-code specification, your file must begin with:

```yaml
---
spec_version: v1
```

Config

The config section of your `.cd4pe.yaml` file provides global pipeline configuration settings for all the pipelines for the control repo or module where the YAML file is housed.

<table>
<thead>
<tr>
<th>config key</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_pull_requests</td>
<td>States whether pull requests from forks of the control repo or module can trigger its pipelines.</td>
<td>Accepts a Boolean.</td>
<td>false</td>
</tr>
</tbody>
</table>
## Continuous Delivery

### Constructing Pipelines

<table>
<thead>
<tr>
<th>config key</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployment_policy_branch</td>
<td>Optional. The name of the branch where custom deployment policies are kept. If custom deployment policies are found on the specified branch, they can be used when building pipelines.</td>
<td>Accepts a string.</td>
<td>-</td>
</tr>
<tr>
<td>enable_pe_plans</td>
<td>Optional. States whether to allow the inclusion of Bolt tasks in custom deployment policies used in pipelines for this control repo or module.</td>
<td>Accepts a Boolean.</td>
<td>true</td>
</tr>
</tbody>
</table>

Here's a sample `config` section for a control repo that allows pull requests from forks and serves custom deployment policies from the `production` branch:

```yaml
config:
  enable_pull_requests_from_forks: true
  deployment_policy_branch: production
  enable_pe_plans: true
```

### Pipelines

The `pipelines` section names each pipeline you're creating for a control repo or module. Its value is key/value pairs where each key corresponds to a pipeline. The key must have either the same name as a branch or a regular expression (for regex branch pipelines).

**Note:** Only one regular expression (regex) entry is allowed for each control repo or module.

To create two pipelines (one for a regex branch and one for the main branch), format the `pipelines` section as follows:

```yaml
pipelines:
  /feature_.*/:  # Use a regex for a branch
    triggers:
      - pull_request
      - commit
    stages:
      - name: "Lint/Parser validation"
        auto_promote: all_succeeded
        steps:
          - type: job
            name: control-repo-puppetfile-syntax-validate

main:
  triggers:
    - pull_request
    - commit
  stages:
    - name: "Deploy to production"
      auto_promote: all_succeeded
      steps:
        - type: deployment
          policy: direct_merge
          name: "Direct merge to Production"
```

© 2021 Puppet, Inc.
Triggers

You must set the triggers section for each pipeline you create. This section specifies the events in your source control system that start the pipeline.

The only allowed values are pull_request and commit. You can set either or both of these values. If no value is set, the pipeline does not run unless triggered manually.

Stages

Each pipeline must have a stages section.

The stages section accepts an array where each item is a stage in the pipeline. Each stage has a name, instructions about when and whether to auto-promote to the next stage in the pipeline, and a list of steps to be executed in parallel.

<table>
<thead>
<tr>
<th>stages key</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name you're giving the pipeline stage.</td>
<td>Accepts a string in quotes.</td>
<td>Stage &lt;STAGE NUMBER&gt;</td>
</tr>
</tbody>
</table>
### Stages

A stage is a collection of steps. A stage can have multiple steps, and each step defines a particular job to be performed in the pipeline. Steps run concurrently, so the order of steps in a stage doesn't matter. If you need certain steps to occur after others, use stages to separate them.

Every step is defined using a hash of key/value pairs. The `type` key is always required. Requirements for other keys depend on the `type`.

The `type` key accepts the following values:

<table>
<thead>
<tr>
<th>type values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>job</td>
<td>Use to specify a job available in the workspace where this pipeline exists. Click <strong>Jobs</strong> in the web UI to review available jobs and create jobs.</td>
</tr>
<tr>
<td><strong>type values</strong></td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pull_request_gate</td>
<td>A conditional checkpoint that ends the pipeline execution at the pull request gate if the pipeline was triggered by a pull request. If the pipeline was triggered by a commit, the pipeline continues beyond the pull request gate.</td>
</tr>
<tr>
<td>impact_analysis</td>
<td>A report assessing the code change in the pipeline for potential impact to nodes and configurations in deployments defined later in the pipeline.</td>
</tr>
<tr>
<td>deployment</td>
<td>A Puppet code deployment.</td>
</tr>
</tbody>
</table>

Additional keys for each value are described below.

For **type**: job steps, you must specify the job name as listed on the workspace's Jobs page in the web UI. For example, to add a job called control-repo-puppetfile-syntax-validate to a pipeline, format the type: job entry as follows:

```yaml
steps:
  - type: job
    name: control-repo-puppetfile-syntax-validate
```

No additional keys are needed or available for type: pull_request_gate steps. To add a pull request gate to your pipeline, add this code to the steps:

```yaml
steps:
  - type: pull_request_gate
```

For **type**: impact_analysis steps, your pipeline must also include a type: deployment step, and you must set the control repo key on the deployment associated with the impact analysis step (for more information, refer to the additional details about type: deployment and the target key at the end of this page). Additionally, use these keys for the type: impact_analysis step:

<table>
<thead>
<tr>
<th><strong>impact_analysis key</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>concurrent_compilations</td>
<td>How many catalog compilations to perform at the same time.</td>
</tr>
</tbody>
</table>

**Tip:** If your compilers are hitting capacity when performing an impact analysis, lower this number. However, lowering this number increases the impact analysis run time.
<table>
<thead>
<tr>
<th>impact_analysis key</th>
<th>Description</th>
<th>Required?</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployments</td>
<td>A list of the deployments for which to assess the potential impact. Accepts an array of strings where each item is the name of a deployment in your pipeline definition.</td>
<td>Yes, either deployments or all_deployments</td>
<td>-</td>
</tr>
<tr>
<td>all_deployments</td>
<td>Whether to assess the impact to all deployments in the pipeline. Accepts a Boolean value.</td>
<td>Yes, either deployments or all_deployments</td>
<td>false</td>
</tr>
<tr>
<td>puppetdb_connection_timeout_sec</td>
<td>The timeout period (in seconds) for requests to PuppetDB during an impact analysis task.</td>
<td>No</td>
<td>120</td>
</tr>
</tbody>
</table>

The following example illustrates an impact analysis step that runs 10 compilations at a time on two specific deployments, and a second impact analysis step that runs on all deployments in your pipeline and times out PuppetDB requests after three minutes:

```yaml
steps:
  - type: impact_analysis
    concurrent_compilations: 10
    deployments:
      - "Direct merge to Production"
      - "My PCI deployment"
  - type: impact_analysis
    all_deployments: true
    puppetdb_connection_timeout_sec: 180
```

Use these keys for type: deployment steps:
<table>
<thead>
<tr>
<th>deployment key</th>
<th>Description</th>
<th>Value</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>policy</td>
<td>The deployment policy to be used.</td>
<td>For built-in deployment policies, accepts the policy name as a string. For custom deployment policies, accepts a key-value pair defining the policy to be used.</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>CAUTION: Do not include sensitive parameters in custom deployment policies used in pipelines managed with code.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>The name you're giving the deployment.</td>
<td>Accepts a string in quotes.</td>
<td>yes</td>
</tr>
<tr>
<td>target</td>
<td>The infrastructure (node group) the deployment targets.</td>
<td>Accepts key/value pairs and requires a type: node_group key. In a module pipeline that contains an impact analysis task, this also requires a control_repo: key. More information about this key is provided below.</td>
<td>yes, except for regex branches</td>
</tr>
<tr>
<td>pe_server</td>
<td>The name in Continuous Delivery for PE of the Puppet Enterprise instance the deployment targets.</td>
<td>Accepts a string.</td>
<td>yes</td>
</tr>
<tr>
<td>parameters</td>
<td>Any relevant parameters defined by the deployment policy. To find out which parameters the policy takes, see Built-in deployment policies on page 169 or refer to the custom deployment policy’s documentation.</td>
<td>Accepts key/value pairs.</td>
<td>no</td>
</tr>
</tbody>
</table>

Tip: Find the name of your PE instance by going to Settings > Puppet Enterprise in the web UI.
<table>
<thead>
<tr>
<th>deployment key</th>
<th>Description</th>
<th>Value</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>control_repo</td>
<td>The name of the control repo a module is associated with.</td>
<td>Accepts a string.</td>
<td>only for module regex branch pipelines using the feature branch deployment policy</td>
</tr>
<tr>
<td>base_feature_branch</td>
<td>The branch in the control_repo that you want Continuous Delivery for PE to use to create feature branches.</td>
<td>Accepts a string.</td>
<td>only for module regex branch pipelines using the feature branch deployment policy</td>
</tr>
</tbody>
</table>

To add a deployment using a built-in deployment policy to your pipeline, format the `type: deployment` entry as follows:

```
steps:
  - type: deployment
    policy: cd4pe_deployments::direct_merge
    name: "Direct merge to Production"
    target:
      type: node_group
      node_group_id: fcda068f-e499-44ef-81f2-255fc487b2e2
      pe_server: cdpe-delivery
    parameters:
      stop_if_nodes_fail: 10
      noop: false
      timeout: 60
```

To add a deployment using a custom deployment policy to your pipeline, format the `type: deployment` entry as follows:

```
steps:
  - type: deployment
    policy:
      name: deployments::custom_policy1
      source: name-of-control-repo
      name: "Direct merge to Production"
    target:
      type: node_group
      node_group_id: fcda068f-e499-44ef-81f2-255fc487b2e2
      pe_server: cdpe-delivery
    parameters:
      policy_parameter1: 10
      policy_parameeter2: false
```

To add a deployment to a module for a regex branch pipeline that uses the feature branch deployment policy, format the `type: deployment` entry as follows:

```
steps:
  - type: deployment
    policy:
      name: "cd4pe_deployments::feature_branch"
      name: "Deployment to hot-dog-prod-2"
    target:
      type: node_group
    pe_server: "hot-dog-prod-2"
    control_repo: "cc-hot-dog-app"
    base_feature_branch: "production"
```

© 2021 Puppet, Inc.
The format of the target key depends on other characteristics of the pipeline, however; the target key is never used for deployments from regex branch pipelines.

When used for a module pipeline that does not include impact analysis or any control repo pipeline, the target key must include the type and node_group_id, as illustrated in the following example:

```yaml
target:
  type: node_group
  node_group_id: fcda068f-e499-44ef-81f2-255fc487b2e2
```

Presently, node_group is the only available type. To locate the node_group_id:

1. In the PE console, click Node groups (or Classification in PE versions prior to 2019.8.1).
2. Locate your target node group and click its name. A page with information about the node group opens.
3. In the page URL, locate and copy the alphanumeric string that follows /groups/. This is your node group ID.

When used for a module pipeline that includes an impact analysis step, the target key must also include the control_repo (in addition to the type and node_group_id) as illustrated in the following example:

```yaml
target:
  type: node_group
  node_group_id: fcda068f-e499-44ef-81f2-255fc487b2e2
  control_repo: cdpe-2018-1-pe-master-1-control
```

The control_repo value is the name of the control repo used in the deployment associated with the impact analysis task.

## Analyzing 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 on demand for any commit to a control repo or module, or add impact analysis to a pipeline to automatically generate a report on all proposed code changes.

- **Generating impact analysis reports** on page 165
  Impact analysis reports show you the effect and risk of proposed code changes, allowing for quick review of lower-risk changes while enabling additional scrutiny for higher-risk changes. Add an impact analysis task to each control repo and module repo pipeline to generate an impact analysis report for every change submitted to your repos.

- **Limitations of impact analysis** on page 168
  Some code changes may impact your infrastructure beyond what an impact analysis report displays. The technical limitations of impact analysis reports are outlined here.

## Generating impact analysis reports

Impact analysis reports show you the effect and risk of proposed code changes, allowing for quick review of lower-risk changes while enabling additional scrutiny for higher-risk changes. Add an impact analysis task to each control repo and module repo pipeline to generate an impact analysis report for every change submitted to your repos.

**Note:** When impact analysis is enabled, catalog compiles are generated every time an impact analysis task detects that a node may be impacted by a change to your Puppet code. These additional compiles increase the performance load placed on your PE primary server.
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.

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. Optional: Set the catalog compilation batch size. By default, Continuous Delivery for PE compiles 10 catalogs at a time when performing an impact analysis task.
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.

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. Optional: Set the catalog compilation batch size. By default, Continuous Delivery for PE compiles 10 catalogs at a time when performing an impact analysis task.
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.

**Generate an impact analysis report on demand**

You can create a new impact analysis report for any code change committed to a control repo or module in Continuous Delivery for PE without triggering a pipeline.

**Before you begin**

1. **Configure impact analysis.**

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.

1. In the Continuous Delivery for PE web UI, navigate to the control repo or module you wish to generate an impact analysis report for.
2. Click Manual actions and select New impact analysis.
3. Select the branch where the code you want to analyze is located, then select the specific commit that contains the new code.
4. Select the PE instance that manages the nodes you want to analyze, then select the specific node group.
   - For impact analysis on a control repo: If relevant, select an environment prefix.
   - For impact analysis on a module: Select the control repo where the module's environment code is deployed.
5. Finally, set the number of node catalogs that should be compiled concurrently.
6. Click **Analyze.** A new impact analysis report based on your selections is generated for you. Click **View impact analysis** to be redirected to the report.

**Note:** Depending on the size of your node group, it may take several minutes for the impact analysis report to be created.
If you want to download the impact analysis report's data, click Export on the report page to generate a CSV file.

Limitations of impact analysis

Some code changes may impact your infrastructure beyond what an impact analysis report displays. The technical limitations of impact analysis reports are outlined here.

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 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.
- **Changes to Embedded Puppet templates.** The impact of changes made to Embedded Puppet (.epp) or Embedded Ruby (.erb) template files alone cannot be analyzed.

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

Deploying Puppet code

- **Built-in deployment policies** on page 169
  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.
- **Creating custom deployment policies** on page 173
  If the built-in deployment policies included in Continuous Delivery for PE don't align with the way your organization works with Puppet Enterprise, you can write a custom deployment policy tailored to your needs.
- **Deploy code manually** on page 174
  Use the manual deployment workflow to push a code change to a specified group of nodes on demand.
- **Deploy module code** on page 175
  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 176

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.

## Built-in 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.

<table>
<thead>
<tr>
<th>Deployment policy</th>
<th>Best for...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct deployment policy</td>
<td>• Small or trivial changes</td>
</tr>
<tr>
<td></td>
<td>• Changes that you have a high level of confidence will not cause issues when deployed</td>
</tr>
<tr>
<td>Temporary branch policy</td>
<td>• Fully tested changes</td>
</tr>
<tr>
<td></td>
<td>• Changes you'd like to have validated through deployment</td>
</tr>
<tr>
<td></td>
<td>• Workflows that don't require extensive logging of historic data</td>
</tr>
<tr>
<td>Eventual consistency policy</td>
<td>• Fully tested changes</td>
</tr>
<tr>
<td></td>
<td>• Large PE installations with regularly scheduled Puppet runs</td>
</tr>
<tr>
<td></td>
<td>• Situations in which running additional orchestrator jobs is undesirable</td>
</tr>
<tr>
<td>Feature branch policy</td>
<td>• Changes made on a feature branch that haven't been merged</td>
</tr>
</tbody>
</table>

### 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.

**Direct deployment policy parameters**

The following optional parameters are available to customize the behavior of your direct deployment. Set these parameters in the web UI when creating a new deployment using the direct deployment policy, or add them to the your `.cd4pe.yaml` file as part of a `deployment` step type declaration.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required?</th>
<th>Default value</th>
<th>Value type</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_node_failure</td>
<td>The maximum number of nodes that can fail before the deployment is stopped and reported as a failure.</td>
<td>no</td>
<td>-</td>
<td>integer</td>
</tr>
<tr>
<td>noop</td>
<td>Whether to perform the Puppet runs in no-op mode.</td>
<td>no</td>
<td>false</td>
<td>Boolean</td>
</tr>
<tr>
<td>fail_if_no_nodes</td>
<td>Whether to stop the deployment and report a failure if the selected node group does not contain any nodes.</td>
<td>no</td>
<td>false</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

**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.

Temporary branch deployment policy parameters

The following optional parameters are available to customize the behavior of your temporary branch deployment. Set these parameters in the web UI when creating a new deployment using the temporary branch deployment policy, or add them to your .cd4pe.yaml file as part of a deployment step type declaration.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required?</th>
<th>Default value</th>
<th>Value type</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch_delay</td>
<td>How many seconds to wait between each deployment batch.</td>
<td>no</td>
<td>60</td>
<td>integer</td>
</tr>
<tr>
<td>batch_size</td>
<td>How many nodes to include in a deployment batch.</td>
<td>no</td>
<td>10</td>
<td>integer</td>
</tr>
<tr>
<td>max_node_failure</td>
<td>The maximum number of nodes that can fail before the deployment is stopped and reported as a failure.</td>
<td>no</td>
<td>-</td>
<td>integer</td>
</tr>
<tr>
<td>noop</td>
<td>Whether to perform the Puppet runs in no-op mode.</td>
<td>no</td>
<td>false</td>
<td>Boolean</td>
</tr>
<tr>
<td>fail_if_no_nodes</td>
<td>Whether the deployment should fail if there are no nodes in the target node group or environment.</td>
<td>no</td>
<td>true</td>
<td>Boolean</td>
</tr>
</tbody>
</table>
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.

Eventual consistency deployment policy parameters

The eventual consistency deployment policy does not have any optional or required policy parameters.

Feature branch deployment policy

The feature branch deployment policy is used only in regex branch pipelines. Code changes you've made in a feature branch are deployed to a Puppet environment with the same name as the feature branch.

When you kick off a deployment to a regex branch pipeline using the feature branch deployment policy, Continuous Delivery for PE performs the following steps:

**Step 1:** puppet-code deploy

Using Code Manager, Continuous Delivery for PE deploys the commit in the pipeline to a Puppet environment with the same name as the branch that triggered the pipeline. If an environment with this name doesn't already exist, Continuous Delivery for PE creates it during the deployment.

Feature branch deployment policy parameters

The feature branch deployment policy does not have any optional or required policy parameters.

Creating custom deployment policies

If the built-in deployment policies included in Continuous Delivery for PE don't align with the way your organization works with Puppet Enterprise, you can write a custom deployment policy tailored to your needs.

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

In a custom deployment policy, you declare the steps that need to happen in order to deploy your Puppet code changes. Because custom deployment policies leverage Bolt plans, you can configure your policies to automate the processes that make up your current deployment workflow. This means your custom policies can include:

- Git processes in your source control provider of choice
- Changes to your node groups with Puppet orchestrator
- Bolt tasks and plans
- Automatic notifications via the third-party services you use daily, such as Slack and ServiceNow

The `puppetlabs-cd4pe_deployments` module is built into Continuous Delivery for PE, so you don't need to download or install anything in order to use custom deployment policies. To see the full list of what's available for inclusion in your custom deployment policy, see the documentation for the `puppetlabs-cd4pe_deployments` module.

© 2021 Puppet, Inc.
If you need some inspiration, see how the deployment policies built into Continuous Delivery for PE look when written in custom deployment policy format.

**Custom deployment policy prerequisites and cautions**

- In order for your custom deployment policies to be available for use in the web UI, you must have webhooks enabled between Continuous Delivery for PE and your source control system.
- Do not include sensitive parameters in your custom deployment policy if you use the custom deployment policy in a pipeline managed with code.

**Add a custom deployment policy to your control repo**

Each custom deployment policy file you create must be stored in a module named `deployments` in the control repo that will utilize the policy.

**Before you begin**

Decide which branch in your control repo will house your custom deployment policies. By default, Continuous Delivery for PE looks for custom deployment policies on the `production` branch.

If your control repo doesn't have a `production` branch, or if you prefer to store custom deployment policies on a different branch, tell Continuous Delivery for PE where to look for your custom deployment policies by using the `deployment_policy_branch` setting of the `config` section of `.cd4pe.yaml` file. For details, see Structuring a `.cd4pe.yaml` file.

**Important:** If you don't store your custom deployment policies on the `production` branch, you must manage your control repo's pipelines with a `.cd4pe.yaml` file in order to use them.

1. In your control repo, on the `production` branch (or the branch you've set as the `deployment_policy_branch` in your `.cd4pe.yaml` file), create a new Puppet module named `deployments`.
   In order for Continuous Delivery for PE to find and run your custom deployment policies correctly, this module must live in the `/modules`, `/site`, or `/site-modules` directory of your control repo.
2. Add your custom deployment policy file to the `/plans` directory in the `deployments` module.

**Deploy code manually**

Use the manual deployment workflow to push a code change to a 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 **Manual actions** and select **New deployment**.
3. Select the branch on which you made the change you're going to deploy, the commit you want to deploy, and your Puppet Enterprise instance.
   **Note:** If you're creating an on-demand deployment for a regex branch pipeline, the commit at the HEAD of your feature branch is selected for you.
4. Select the Puppet environment you wish to deploy the change to.
5. Select a deployment policy.
   See Deployment policies to learn more about the deployment policies and how they work.
6. Optional: 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 => 'main'
   ```

   To learn more about the `:branch => :control_branch` option, see Declare content from a relative control repo branch in the Code Manager documentation.

2. If you are running Puppet Enterprise version 2019.0.x or any earlier version:
   a) Add the following to the top of your Puppetfile:

   ```
   def default_branch(default)
     begin
       match = /(.+)_cdpe\d+$/.
     match(@librarian.environment.name)
     match ? match[1]:default
     rescue
       default
     end
     end
   
   def default_branch=default_branch
   ```

   b) Add the following `:branch` and `:default_branch` entries to the module entry in your Puppetfile:

   ```
   :branch => :control_branch, :default_branch => default_branch('main')
   ```

3. In the Continuous Delivery for PE web UI, click **Modules** and select the module you wish to deploy.
4. If you haven't already done so, create a pipeline for your module.
5. Click **Add stage**. Select **Deployment**.
6. Select your PE instance and choose the Puppet environment the module code will be deployed to.
7. 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 new group**.
   c) Enter a name (such as **Approval**) and description for your new group, then click **Save**.
   d) In each permissions category, select the permissions you wish to assign to the approval group and click **Save and add users**.

   **Important:** At a minimum, the approval group must have the **List** permission for **Control repos** in order to view and approve or deny deployments.

   e) On the **Add users** page, add the individuals with the authority to approve or deny deployments to protected environments.
      - Search for users by username or email address.

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.

---

**Support and troubleshooting**

This section includes a guide to troubleshooting issues with Continuous Delivery for PE as well as information on working with the Puppet Support team.

- [Troubleshooting Continuous Delivery for PE](#) on page 177

Use this guide to troubleshoot issues with your Continuous Delivery for Puppet Enterprise (PE) installation.
Troubleshooting Continuous Delivery for PE

Use this guide to troubleshoot issues with your Continuous Delivery for Puppet Enterprise (PE) installation.

Note: For PE instances with a replica configured for disaster recovery. In the event of a partial failover, Continuous Delivery for PE is not available. Learn more at What happens during failovers in the PE documentation. To restore Continuous Delivery for PE functionality, you must promote the replica to primary server.

Look up a source control webhook

Continuous Delivery for PE creates a webhook and attempts to automatically deliver it to your source control system when you add a new control repo or module to your workspace. You can look up this webhook if you ever need to manually add or re-add it to your source control repository.

1. In the Continuous Delivery for PE web UI, click Control Repos or Modules and select the control repo or module whose webhook you want to view.
2. In the Pipelines area of the screen, click Manage pipelines.
3. If necessary, select Manage in the web UI.

   Note: If you're managing the pipelines for this control repo or module as code, don't worry. After copying the webhook, do not save any changes and switch back to Manage as code before closing the window. Your pipeline code will not be affected.

4. In the Automation webhook area, copy the full webhook address printed in the black box. This is the unique webhook created to connect this control repo or module in Continuous Delivery for PE with its corresponding repository in your source control system.

Add the webhook to the corresponding repository in your source control system, following the source control system's documentation. In most cases, webhooks are managed in the repository's settings area.

Manually configure a Puppet Enterprise integration

When you add credentials for a PE instance, Continuous Delivery for PE attempts to look up the endpoints for PuppetDB, Code Manager, orchestrator, and node classifier, and to access the primary SSL certificate generated during PE installation. If this information can't be located, such as in cases where your PE instance uses customized service ports, you are asked to enter it manually.

This task assumes you have completed the steps in Add your Puppet Enterprise credentials and have been redirected to the manual configuration page.

1. In the Name field, enter a unique friendly name for your 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.

2. In the API token field, paste a PE access token for your "Continuous Delivery" user. Generate this token using puppet-access or the RBAC v1 API.
   
   For instructions on generating an access token, see Token-based authentication.
3. In the four service fields, enter the endpoints for your PuppetDB, Code Manager, orchestrator, and node classifier. You can locate these endpoints with the PE console.
   a) In the PE console, go to **Overview** (or **Status** in newer PE versions) and click **Puppet Services status**.
   b) Copy the endpoints from the Puppet Services status monitor and paste them into the appropriate fields in Continuous Delivery for PE, omitting the https:// prefix for each endpoint, as shown in the sample below.

<table>
<thead>
<tr>
<th>Puppet Services status format</th>
<th>Continuous Delivery for PE format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PuppetDB Service</td>
<td>sample.host.puppet:8081</td>
</tr>
<tr>
<td>Code Manager Service</td>
<td>sample.host.puppet:8140</td>
</tr>
<tr>
<td>Orchestrator Service</td>
<td>sample.host.puppet:8143</td>
</tr>
<tr>
<td>Classifier Service</td>
<td>sample.host.puppet:4433</td>
</tr>
</tbody>
</table>

   Important: Use port 8170 for Code Manager in Continuous Delivery for PE.

4. Locate the master SSL certificate generated by PE during installation by running:

   ```
curl https://<HOSTNAME>:8140/puppet-ca/v1/certificate/ca --insecure
   
```

5. Copy the certificate in its entirety (including the header and footer) and paste it into the **CA certificate** field.

6. Click **Save Changes**.

7. Optional: Once the main PE integration is set up, manually configure impact analysis.
   a) Generate a dedicated PE certificate to authenticate with the endpoints used for impact analysis. On the node hosting your PE certificate authority service (typically the primary server), run the following, replacing `<CERTIFICATE_NAME>` with the name you want to give to this certificate, such as `cd4pe-impact-analysis`:

   ```
puppetserver ca generate --certname <CERTIFICATE_NAME>
   
```

   b) Complete the steps in **Configure impact analysis** to install the relevant modules and update classification.
   c) In the Continuous Delivery for PE web UI, navigate to **Settings > Puppet Enterprise** and click **Edit credentials** to reopen the configuration pane.
   d) Follow the instructions in step 3 above to locate the endpoint for Puppet Server. Enter it in the **Puppet Server service** field, omitting the https:// prefix.

   Important: To run impact analysis tasks on a compiler or load balancer instead of the primary server (strongly recommended for PE installations that use these components as part of their architecture) enter the hostname of the compiler or load balancer at :8140 (for example, loadbalancer.example.com:8140) in the **Puppet Server service** field.

   e) 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
   
```

   f) 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.

   g) Click **Save Changes**.

© 2021 Puppet, Inc.
**Restart Continuous Delivery for PE**

Continuous Delivery for PE is run in a managed Kubernetes cluster, so restarting the pod is an appropriate first step when troubleshooting.

To restart the pod, run:

```
kubectl rollout restart deployment cd4pe
```

For more on the `kubectl rollout` command, see the Kubernetes documentation.

**Logs**

Because Continuous Delivery for PE is run in a managed Kubernetes cluster, the software’s logs are accessed with the `kubectl logs` command.

To access the Continuous Delivery for PE logs, run:

```
kubectl logs deployment/cd4pe
```

To access your installation’s PostgreSQL logs, run:

```
kubectl logs statefulset/postgres
```

**Trace-level logging**

Enable or disable trace-level logging by setting Enable trace logging in the Advanced configuration and tuning section of the Config page on Puppet Application Manager.

**PE component errors in logs**

The logs include errors for both Continuous Delivery for PE and for the numerous PE components used by the software. It’s important to realize that an error in the Continuous Delivery for PE log may actually indicate an issue with Code Manager, r10k, or another component.

For example, this log output indicates a Code Manager issue:

```
Module Deployment failed for PEModuleDeploymentEnvironment[nodeGroupBranch=cd4pe_lab,
nodeGroupId=a923c759-3aa3-43ce-968a-f1352691ca02, nodeGroupName=Lab
environment,
peCredentialsId=PuppetEnterpriseCredentialsId[domain=d3, name=lab-MoM],
pipelineDestinationId=null, targetControlRepo=null,
 targetControlRepoBranchName=null,
targetControlRepoHost=null, targetControlRepoId=null].
Puppet Code Deploy failure: Errors while deploying environment
'cd4pe_lab' (exit code: 1):
ERROR -> Unable to determine current branches for Git source 'puppet' (/etc/puppetlabs/code-staging/environments)
Original exception: malformed URL 'ssh://git@bitbucket.org:mycompany/control_lab.git'
at /opt/puppetlabs/server/data/code-manager/worker-caches/deploy-pool-3/
sshe---git@bitbucket.org:mycompany-control_lab.git
```

For help resolving issues with PE components, see the Puppet Enterprise troubleshooting documentation.

**Error IDs in web UI error messages**

Occasionally, error messages shown in the Continuous Delivery for PE web UI include an error ID and instructions to contact the site administrator. For example:
Remove 'docker' capability?

Are you sure you want to remove this capability?

Please contact the site administrator for support along with error-id=[md5:35910b8341bf229e7040af2a1ce4bfe7 2020-02-06 19:33 1e6gebel26ada0x1k8uxtm244v7]

Errors of this kind are shown without additional detail for security purposes. Users with root access to the Continuous Delivery for PE host system can search the log for the error ID to learn more.

Duplicate job logs after reinstall

A job's log is housed in object storage after the job is complete. If you reinstall Continuous Delivery for PE but reuse the same object storage without clearing it, you might notice logs for multiple jobs with the same job number, or job logs already present when a new job has just started.

To remove duplicate job logs and prevent their creation, make sure you clear both the object storage and the database when reinstalling Continuous Delivery for PE.

Name resolution

Continuous Delivery for PE uses CoreDNS for name resolution. Many can't reach and timeout errors reported in the logs are actually DNS lookup failures.

To add DNS entries to CoreDNS, run:

```
kubectl -n kube-system edit configmaps coredns
```

The system will open the configmap file in an editor. Add a hosts stanza directly after the kubernetes stanza using the following format:

```
hosts /etc/hosts <DOMAIN> {
    <IP ADDRESS> <HOSTNAME> [ALIASES]
    fallthrough
}
```

Here's an example edited configmap with a hosts stanza added:

```
apiVersion: v1
data:
  Corefile: |
  .:53 {
    errors
    health {
      lameduck 5s
    }
    ready
    kubernetes cluster.local in-addr.arpa ip6.arpa {
      pods insecure
      fallthrough in-addr.arpa ip6.arpa
      ttl 30
    }
    hosts /etc/hosts puppetdebug.vlan {
      10.234.4.29 pe-201922-master.puppetdebug.vlan pe-201922-master
      fallthrough
    }
    prometheus :9153
    forward . /etc/resolv.conf
```

© 2021 Puppet, Inc.
Looking up information about Continuous Delivery for PE

Use `kubectl` commands to access information about your Continuous Delivery for PE installation.

**Look up the environment variables in use**

To list the environment variables in use on your installation, run:

```
kubectl describe deployments.apps cd4pe
```

**Note:** For information on tuning your Continuous Delivery for PE instance by setting environment variables, including adjusting HTTP and job timeout periods, changing the size of LDAP server searches, or enabling Git repository caching, see Advanced Continuous Delivery for PE configuration.

**Look up Continuous Delivery for PE version**

To print the version of Continuous Delivery for PE running in your installation, run:

```
kubectl get deployment cd4pe -o jsonpath='{.spec.template.spec.containers[0].image}' ; printf "\n"
```

**Drain a node**

Drain the impacted nodes when performing maintenance on your Kubernetes cluster such as upgrading system packages or rebooting the system.

To drain a node in order to perform system maintenance, run:

```
/opt/ekco/shutdown.sh
```

**Getting support**

The Support team at Puppet provides support for all features and capabilities included in Continuous Delivery for PE.

Report your support issues through our confidential Support Portal. When you purchase Continuous Delivery for PE, you receive an account and log-on for the portal, which includes access to the Support Knowledge Base.

**Supported installation methods**

The Support team can assist with the Continuous Delivery for PE installation process using the documented installation methods for the 4.x series:

- Install Continuous Delivery for PE
- Install Continuous Delivery for PE in an offline environment
The Puppet Support team only supports the approved installation methods listed on this page, and cannot assist you with Kubernetes configuration or container runtime issues unrelated to Continuous Delivery for PE.

Application support

The Puppet Support team supports all Continuous Delivery for PE features and capabilities, regardless of where the application is running. However, if a feature is not working correctly due to your runtime environment, the Support team reserves the right to reject the support request.

Creating a support bundle

When seeking support, you might be asked to generate and provide a support bundle. This bundle collects a large amount of logs, system information and Continuous Delivery for PE diagnostics.

To create a support bundle:

1. In Puppet Application Manager, click Troubleshoot > Generate a support bundle.
2. Select a method for generating the support bundle:
   - Generate the bundle automatically. Click Analyze Continuous Delivery for Puppet Enterprise, and Puppet Application Manager generates the bundle for you and uploads it to the Troubleshoot page.
   - Generate the bundle manually. Click the prompt to generate a custom command for your installation, then run the command on your cluster. Follow the prompts to upload the bundle to Puppet Application Manager.
3. Review the collected data before forwarding it to Puppet, as it may contain sensitive information that you wish to redact.
4. Return to the Troubleshoot page, download the newly created support bundle, and send it to your Puppet Support contact.

Airgap bundles

This page lists available 4.x series .airgap bundles, which are used when installing Continuous Delivery for PE in an offline environment.

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Airgap bundle</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.11.5</td>
<td>22 February 2022</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.11.5.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.11.5.airgap</a></td>
</tr>
<tr>
<td>Version</td>
<td>Release date</td>
<td>Airgap bundle</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.10.5.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.10.4.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.10.3.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.10.2.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.10.1.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.10.0.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.9.0.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.8.2.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.8.1.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.8.0.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.7.2.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.7.1.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.7.0.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.6.1.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.6.0.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.5.2.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.5.1.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.5.0.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.4.2.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.4.1.airgap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cd4pe-4.4.0.airgap</td>
</tr>
<tr>
<td>Version</td>
<td>Release date</td>
<td>Airgap bundle</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>4.3.3</td>
<td>23 February 2021</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.3.3.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.3.3.airgap</a></td>
</tr>
<tr>
<td>4.3.2</td>
<td>3 February 2021</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.3.2.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.3.2.airgap</a></td>
</tr>
<tr>
<td>4.3.1</td>
<td>26 January 2021</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.3.1.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.3.1.airgap</a></td>
</tr>
<tr>
<td>4.2.4</td>
<td>17 December 2020</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.4.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.4.airgap</a></td>
</tr>
<tr>
<td>4.2.3</td>
<td>17 November 2020</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.3.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.3.airgap</a></td>
</tr>
<tr>
<td>4.2.2</td>
<td>12 November 2020</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.2.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.2.airgap</a></td>
</tr>
<tr>
<td>4.2.0</td>
<td>3 November 2020</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.0.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.2.0.airgap</a></td>
</tr>
<tr>
<td>4.0.0</td>
<td>25 August 2020</td>
<td><a href="https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.0.0.airgap">https://cd4pe-builds.s3.amazonaws.com/stable/cd4pe-4.0.0.airgap</a></td>
</tr>
</tbody>
</table>