Comply

Contents

Welcome to Puppet Comply

Puppet Comply is a tool that assesses the infrastructure you manage with Puppet Enterprise against CIS Benchmarks — the best practices for securely configuring systems from the Center for Internet Security (CIS).

Using Comply, you can:

- Run scans to check the compliance of your infrastructure against CIS Benchmarks.
- Set your desired compliance — a default benchmark and profile that you want your scans to be measured against.
- Customize profiles to specify which rules you want visible in scan reports.
- Identify the cause and source of compliance failures, and determine what configuration changes must be made to which systems.

Comply uses Puppet Enterprise (PE) to retrieve node and fact information. Once you have installed Comply, you must configure it to integrate with PE.

If this is your first time using Comply, try out our Beginner's guide to Comply.

Puppet Comply docs links	Other useful places
Learn the basics: Comply overview Comply terminology Beginner's guide to Comply Release notes **Install and configure Comply:** System requirements Install Puppet Application Manager Set up Comply **Run and manage CIS scans:** Run a CIS scan Set desired compliance Create a custom profile on page 77 View scan results	**Comply videos:** Comply introduction and demo **Docs for related Puppet products:** Puppet Enterprise Puppet Forge **Get help:** Troubleshooting Support portal

Comply terminology

Key terms to be familiar with when using Puppet Comply.

CIS Benchmarks

Developed by the Center for Internet Security (CIS), *CIS Benchmarks* are internationally recognized standards and best practices for securely configuring systems. For more information, see CIS Benchmarks.

CIS assessor

Comply integrates with the *CIS assessor* (CIS-CAT PRO), the scanner tool that assesses CIS benchmarks. As part of the Comply configuration process, Puppet Enterprise (PE) installs the CIS assessor on your target nodes. For more information on the assessor, see CIS-CAT Pro.

Profiles

CIS Benchmarks include different levels of security settings, called *profiles*. The *Level 1* profiles are the base recommendation for every system, and the *Level 2* profiles are intended for environments requiring greater security. Comply can scan for either profile.

Rules

Each profile contains multiple *rules* that define specific elements of system configuration.

Custom profiles

A *custom profile* is a benchmark profile that you customize to fit your organization's internally defined standards, by specifying which rules you want visible in scan reports. Once you create a custom profile, it appears as an option in Comply when selecting a benchmark and profile.

Desired compliance

Desired compliance is the benchmark and profile that you assign to a node. It becomes the default scan for that node.

For a full list of Puppet terminology, see the Puppet Glossary.

Comply overview

Welcome to the Puppet Comply!

This overview is intended for new users of Comply. We go over what Comply is, how it works, and show a demo of the 1.0.0 release. Before you begin, we recommend familiarizing yourself with our terminology.

What is Comply and how does it work?

Comply is a tool that expands the compliance capabilities of Puppet Enterprise (PE), by integrating with the *CIS assessor* to scan your infrastructure against the latest *CIS Benchmarks*. Comply connects to your PE environment and gathers information about your PE managed nodes, including operating system facts and classification node groups. It uses this information to suggest appropriate scans.

You can choose to run ad-hoc scans or *desired compliance* scans — a default CIS benchmark and profile scan that you assign to a node. Comply can automate desired compliance for you based on the information it gathers about your nodes from PE, or you can manually choose your desired compliance from a list of benchmarks and profiles. You can also create *custom profiles* to fit internally defined standards, by specifying which rules you want visible in scan reports. Most of the time, you only need to set your desired compliance once.

The scans are run as a *task* in PE. Scan results populate in the Comply **Compliance dashboard**, where you can see the number of nodes scanned and their compliance breakdown. In each node listed, there is a further breakdown of rule information which tells you why that rule is important, and steps you can take to fix the rule if it is failing the scans.

To see Comply in action, watch the demo below, or go through the steps yourself in our getting started guide.

For a full list of features, see the release notes.

Comply demo

The following demo walks you through the key features of the Comply 1.0.0 release:

Supported CIS benchmarks

Comply supports the following CIS operating system benchmarks.

Operating system	Supported versions
Amazon Linux	2
CentOS	6, 7, 8
Debian	8, 9, 10
Oracle Linux	6, 7, 8
Red Hat Enterprise Linux (RHEL)	6, 7, 7 STIG, 8
SUSE Linux Enterprise Server (SLES)	12, 15
Ubuntu	16.04, 18.04, 20.04, 20.04 STIG
Windows	2010 H21, 2012 R2, 2016, 2016 STIG, 2019, 2019 STIG
Mac OS X	10.14, 10.15, 11.0

Note: If you use Ubuntu 16.04, you need Curl with mutual TLS support to upgrade and run scans.

Comply release notes

These are the new features, enhancements, and resolved issues for the Puppet Comply 2.x release series.

Comply 2.1.0

Released 7 October 2021.

New in this release:

- **Scan Reports**. The Comply UI has a new **Scan Reports** page that provides a report on rules passed/failed and node compliance from the most recent CIS scan. For more information, see CIS scan reports on page 78.
- **CIS-CAT Pro Assessor v4.9.0.** Comply 2.1.0 includes the latest version of the CIS-CAT assessor and its associated benchmark:
  - CentOS Linux 7 v3.1.2
- **Scanner upgrades**. Scanner upgrade in Comply is not forced but optional to allow better management of PE jobs.

Note: By default in Comply 2.1.0, assessor upgrade does not happen automatically when you upgrade Comply. Assessor upgrade takes place when you instigate a Puppet Enterprise (PE) Puppet run job after Comply is upgraded. For more information, see Upgrade from Comply 2.0.0 to 2.1.0 on page 73

Resolved in this release:

- **Desired compliance upgrades**. Fixed an issue where Windows 10 nodes lost their desired compliance after upgrade to Compliance 2.x
- **Upgrade statistics**. Resolved an issue where statistics were overwritten when multiple upgrades take place.
- **Service start up**. Updated Comply so that it now starts when IPv6 is disabled.
- **Preflight failure**. Fixed an issue where preflight checks failed during install when trailing newline returns were present in certificates.
- **Scan wizard**. The Comply scan wizard was updated to correct an issue where the **environment name** field did not revert to the previous saved value if the scan set up was cancelled.

Comply 2.0.0

Released August 2021.

New in this release:

- **CIS-CAT Pro Assessor v4.8.2.** Comply 2.0.0 includes the latest version of the CIS-CAT assessor and its associated benchmarks:
  - Apple macOS 10.14 v1.4.0
  - Apple macOS 10.15 v1.4.0
  - Apple macOS 11.0 v1.2.0
  - CentOS Linux 7 v3.1.1
  - CentOS Linux 8 v1.0.1
  - Debian Linux 8 v2.0.2
  - Microsoft Windows Server 2019 v1.2.1
  - Microsoft Windows Server 2019 STIG v1.0.1
  - Microsoft Windows 10 20H2 v1.10.1
  - Oracle Linux 7 v3.1.1
  - Oracle Linux 8 v1.0.1
  - Red Hat Linux 7 v3.1.1
  - Red Hat Linux 8 v1.0.1
  - Amazon Linux 2 v2.0.0
  - Microsoft Windows 10 21H1 v1.11.0
  - Microsoft Windows Server 2016 v1.3.0
  - Ubuntu Linux 20.04 LTS STIG v1.0.0
- **Automatic upgrades of the CIS-CAT assessor.** Every time you upgrade your Comply application, the assessor automatically upgrades to the latest version. This update also includes the following changes to how you interact with Comply:
  - You can only run a desired compliance scan against nodes with the latest version of the assessor.
  - You can only run a custom scan against benchmarks with the latest version of the assessor.
  - On the node inventory screen, nodes without the latest assessor are highlighted red to indicate that they need upgrading.
  - You can no longer set a desired compliance benchmark against a node that does not have the latest version of the assessor.
  - When the assessor upgrades, custom profiles are automatically updated to use the new benchmarks and profiles, sending you a notification.
- **Assessor upgrades tab.** The **Assessor upgrades** tab on the **Activity feed** screen provides a summary of assessor upgrades, including the number of nodes that have passed or failed. Note that this only shows the status of your nodes after the upgrade, and does not update again, even if your nodes change to passing.
- **`comply` module Secure Sockets Layer (SSL).** This includes changes to how you install and upgrade the Comply module.

Resolved in this release:

- **Comply tries to install 7-zip on Windows.** The `comply` module no longer installs 7zip on Windows systems.
- **Windows Server Semi Annual Channel (SAC) builds are assigned the wrong CIS profile.** SAC builds are now assigned the correct Windows 2019 profile.

Security notice:

- **Vulnerability in 12.18.3-alpine image.** The release updates the alpine image to 15.13.0.
- **Vulnerability keycloak:15.0.0.** This release updates keycloak to version 15.0.0.
- **Vulnerability in dependencies.** This release upgrades NodeJS to version 14.17.1 and React to version 17.0.2.

For upgrade instructions, see Upgrade from Comply 2.0.0 to 2.1.0 on page 73.

Comply known issues

These are the known issues for the Comply 1.x and 2.x releases.

Scan report metrics bar node count not matched in Scan Report page Nodes tab table in Comply 2.1.0

If a error occurs after a scan report is sent from PE to Comply (owing, for example, to the Comply module being out-of-date on the node), the number of nodes appearing in the **Scan Report** page **Nodes** tab table can differ from the node count that appears in the **Scan report metrics bar**.

Running scans on CentOS 7 with Comply 1.0.4

The CentOS 7 benchmark in Comply 1.0.4 has been updated to version 3.1.0. If you have already installed Comply and set desired compliance for your CentOS 7 nodes, you need to run following command on your `comply-scarpy` pod to update the benchmark version from 3.0.0 to 3.1.0:

```
kubectl exec --stdin --tty -n <namespace> $(kubectl get pods -n dio-comply
 | grep comply-scarpy | awk '{print $1}') -- /bin/scarp upgrade-assessor --
assessor_version '4.6.0'
```

This ensures Comply uses the latest CISCAT benchmark and profiles.

Running scan tasks in Puppet Enterprise (PE)

Comply uses PE tasks to run compliance scans on nodes. While you can see the scan task in PE, we advise *against* running them from here as it can have unforeseen effects on both PE and Comply. Instead, run all CIS scans from Comply. You can view the results of the scan in both products.

Beginner’s guide to Comply

Welcome to the Beginner’s guide to Comply! As a new user, you'll need to perform some initial installation and configuration tasks, and then we'll show you how to use the core features of Comply.

You're just a few steps away from enforcing compliant configurations across your infrastructure. Before you begin, we recommend familiarizing yourself with our terminology and Comply overview on page 5.

Step 1: Install and configure Comply

Use the main documentation to install and configure Comply. If you've already completed these steps, proceed to step 2.

- Install Puppet Application Manager (PAM)
- Set up Comply

Related concepts

Install PAM on page 25
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.

Set up Comply on page 68
To start using Puppet Comply, you must complete the setup process, using both Puppet Application Manager (PAM) and Puppet Enterprise (PE).

Step 2: Set desired compliance

Desired compliance is the benchmark and profile that you to assign to a particular node. It is what is scanned on that node by default. Most of the time, you only need to set this once for your nodes.

Based on fact information from PE, Comply can automatically assign an appropriate benchmark for each operating system, along with a Level 1 profile, to nodes that have not been set. This is the quickest way to get up and running with desired compliance. To manually choose your own benchmark and profiles, see Manually set desired compliance.

1. In Comply, click **Nodes**.

   Comply lists the nodes that have been classified with the `comply` class. If you do not see any nodes, ensure you have classified your nodes correctly.
2. In the message box that appears in the top right corner, click **Apply suggested profiles**.

   Comply automatically assigns profiles to all the nodes that have not already been set on your *current* page. To apply the suggested profile to all the nodes in your inventory, you must do this on every page.

> **Tip:** If you want to customize your scans to fit your organization's internally defined standards, see Creating custom profiles, which shows you how to exclude rules in a profile.

The ## sign in the **Profile assigned** column tells you that the desired compliance is set. You can view the node's information, including its assigned benchmark and profile, by clicking on the node. If you want to change a node's desired compliance, use the drop-down menu and click **Update**.

Step 3: Run a CIS scan

You are now ready to run a scan.

1. In Comply, click **Scan**.
2. In the **Benchmark** drop-down, select **Desired Compliance**.

   This scans each node with the profiles you assigned in the previous step.
3. Click **Next** to review the PE credentials and environment you want the scan to run on.
4. Click **Next** to see the nodes selected for scanning.

   To only scan a subset of nodes, deselect any that you do not want to include.
5. Click **Scan** and then **Start**.

   You'll be taken to the **Activity Feed**, which lists each scan. Scans are run as a task in PE. To see the details of the job, click on the job ID to be taken to PE.

> **Tip:** You can also run a scan by clicking the **Scan nodes** button at the top right corner on several pages. This option uses the nodes listed on the page you are currently viewing.

6. In Comply, navigate to the **Compliance dashboard** to see the results of your scan.

   See Viewing scan results for a description of the scan data.

Congratulations! You've completed the Beginner’s guide to Comply. You're now familiar with the core features and know how to run CIS scans with Comply. To find out how you can enforce and automate CIS benchmarks on your failing nodes, see Enforce CIS benchmarks.

Related information

Enforce CIS benchmarks on page 80

Puppet Comply provides visibility into your compliance status, but it cannot fix your failing nodes. Instead, you can use Puppet’s Compliance Enforcement Modules (CEM).

Puppet Application Manager

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

> **Note:** Puppet Application Manager was previously called the platform admin console in the Comply 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). Comply runs on this Kubernetes cluster, and Puppet Application Manager manages the cluster for you.

In the Puppet Application Manager UI, you can configure Comply, 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 Comply?

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

- Troubleshooting PAM on page 63

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:

Puppet Application Manager docs links	Other useful places
Before you install	**Docs for related Puppet products**
Release notes	Continuous Delivery for PE
System requirements	Comply
Install Puppet Application Manager	**Get support**
PAM standalone online install on page 25	Support
PAM standalone offline install on page 28	Upgrade your support plan
PAM HA online install on page 31	**Share and contribute**
PAM HA offline install on page 37	Engage with the Puppet community
Upgrading, disaster recovery, and troubleshooting	Puppet Forge
Upgrading PAM on a Puppet-supported cluster on page 54	Open source projects from Puppet on GitHub
Backing up PAM using snapshots on page 58	
Troubleshooting PAM on page 63	

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.

6 October 2021 (Puppet Application Manager 1.52.1)

New in this release:

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

Resolved in this release:

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

25 August 2021 (Puppet Application Manager 1.49.0)

New in this release:

- **Component upgrades to address CVEs.** To address various CVEs, this version includes an upgrade of Kubernetes to 1.19.13, an upgrade of Project Contour to 1.18.0, and an upgrade of Velero to 1.6.2.
- **Goldpinger.** High availability architectures now include Goldpinger, which aids the debugging of network issues.

- **containerd upgrade.** This version includes an upgrade of containerd to version 1.4.6, and removes the need to use the `force-reapply-addons` option when upgrading.
- **Other component upgrades.** This version includes an upgrade of KOTS to version 1.49.0, an upgrade of ekco to 0.11.0, an upgrade of Prometheus to 0.49.0, and an upgrade of Rook to 1.5.12.

30 June 2021 (Puppet Application Manager 1.44.1)

New in this release:

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

Resolved in this release:

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

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

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

26 May 2021

New in this release:

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

Known issues in this release:

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

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

10 May 2021 (Puppet Application Manager 1.40.0)

New in this release:

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

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

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

  For more information on legacy upgrades, see PAM legacy upgrades on page 56.

15 April 2021 (Puppet Application Manager 1.38.0)

New in this release:

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

17 February 2021 (Puppet Application Manager 1.29.3)

New in this release:

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

3 February 2021 (Puppet Application Manager 1.29.2)

New in this release:

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

Resolved in this release:

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

7 December 2020

New in this release:

- **Support for Red Hat Enterprise Linux (RHEL) 8 and CentOS 8.** You can now run Puppet Application Manager on RHEL version 8 and CentOS version 8. To support this change, `containerd` is now used independently of Docker during the installation process.
- **Component upgrades.** This version includes an upgrade of Kubernetes to version 1.17.13.

Related information

Upgrading PAM on a Puppet-supported cluster on page 54
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 58
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.

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 58
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. It omits optional features like Prometheus and Grafana, and stores data directly on disk. 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 61.

HA architecture

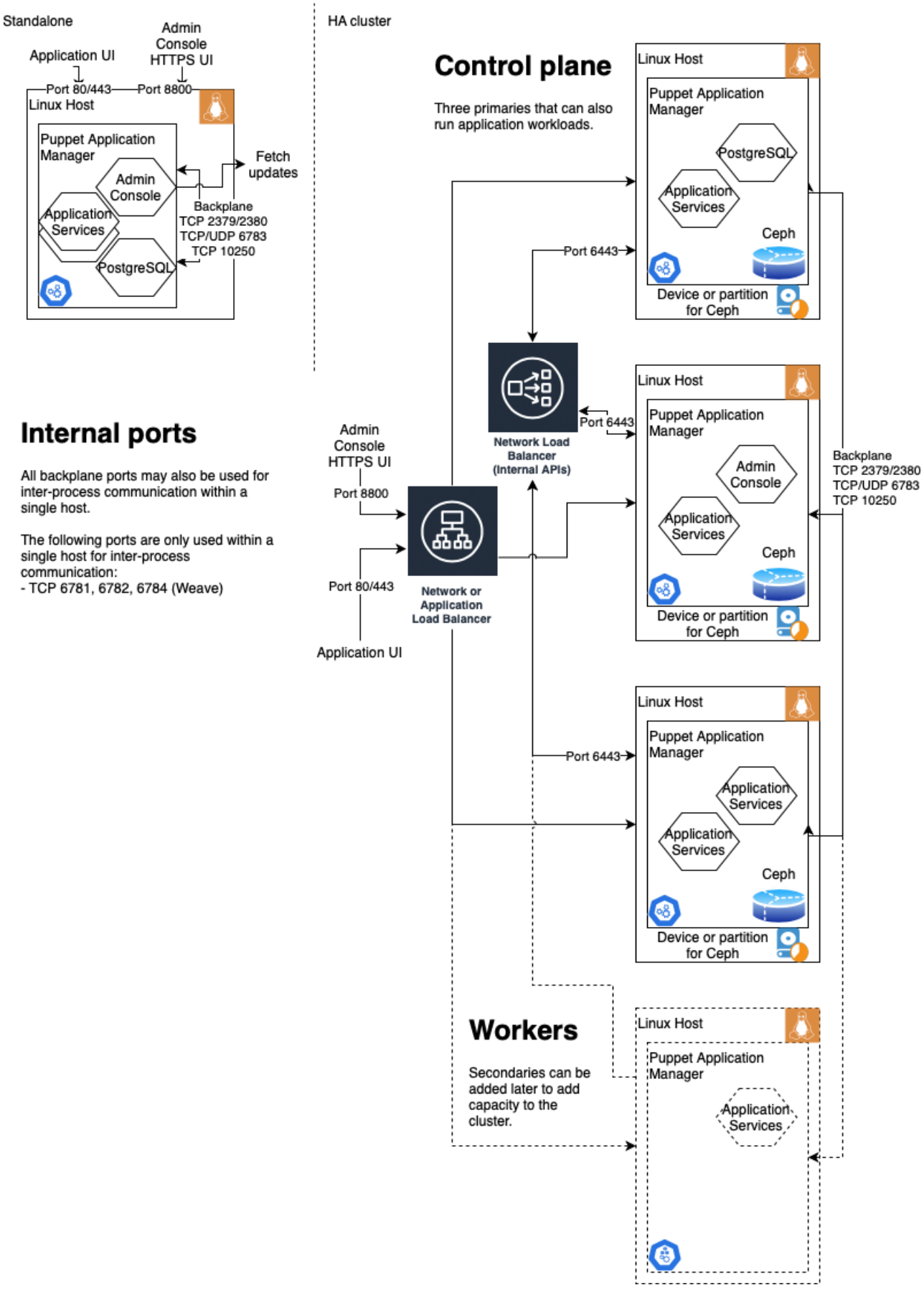
A high availability (HA) architecture provides high availability for scheduling application services during failure and uses Ceph for distributed storage in case of node failure. Individual applications may still experience some loss of availability (up to 10 minutes) if individual services do not have 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 65.

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 Puppet-supported cluster port requirements on page 21 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 54

Legacy Architecture

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 65.

For information on upgrading to the new architecture, see PAM legacy upgrades on page 56 and PAM offline legacy upgrades on page 56.

For information on migrating data from legacy architectures, see Disaster recovery or migration using a snapshot on page 61.

Related information

Reduce recovery time when a node fails on page 64
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 47
Use these instructions to install Puppet Application Manager and any Puppet applications on an existing Kubernetes cluster.

PAM legacy upgrades on page 56
Use this method to upgrade Puppet Application Manager (PAM) from a version installed before May 2021.

PAM offline legacy upgrades on page 56
Use this method to upgrade Puppet Application Manager (PAM) on offline nodes from a version installed before April 2021.

Troubleshooting PAM on page 63
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

Hardware requirements for customer-supported cluster deployments are dictated largely by your system capabilities. Make sure, however, that your Kubernetes cluster meets the minimum requirements:

- Kubernetes version 1.17 or newer.
- 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).

Puppet-supported cluster hardware requirements

While you can install PAM on a single standalone server, if you want to provide availability in the event of server failures, choose a high availability (HA) configuration with multiple servers.

For standalone implementations:

Memory	Storage	CPUs	Open ports
2 GB + application requirements	At least 100 GB for `/var/lib` and `/var/openebs`. This is primarily divided among: • 2 GB for `/var/lib/etcd` • 32 GB for `/var/lib/kubelet` • 40 GB for `/var/lib/containerd` • 20 GB for `/var/openebs` + additional application-specific storage.	2 + application requirements	TCP: 443, 2379,2380, 6443, 6783, 8800, 9001 (offline only), and 10250 UDP: 6783, 6784

For HA implementations, each server must meet the following minimum requirements. Secondaries must only be added after setting up three primaries:

Node type	Memory	Storage	CPUs	Open ports
Primary	7 GB + application requirements	At least 50 GB on an unformatted, unpartitioned storage device + additional application-specific storage At least 100 GB for `/var/lib.` This is primarily divided among: • 2 GB for `/var/lib/etcd` • 4 GB for `/var/lib/rook` (plus buffer) • 32 GB for `/var/lib/kubelet` • 40 GB for `/var/lib/containerd` Note: Ceph storage back-end 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`.	4 + application requirements	TCP: 443, 2379,2380, 6443, 6783, 8000, 8800, 9001 (offline only) and 10250 UDP: 6783, 6784

Node type	Memory	Storage	CPUs	Open ports
Secondary	1.5 GB + application requirements	At least 80 GB for `/var/lib`. This is primarily divided among: • 32 GB for `/var/lib/kubelet` • 40 GB for `/var/lib/containerd`	1 + application requirements	

Hardware requirements for individual applications

Hardware requirements for applications run on customer-supported Kubernetes clusters are the same as those on Puppet-supported clusters.

For standalone implementations, the application hardware requirements for the application are directly added to the cluster hardware. However, for HA implementations, CPU and memory requirements are spread across multiple servers, because the individual service requirements are covered by the multiple servers within the cluster. The estimates for an HA implementation include a buffer that accounts for the eventuality of a single server failure situation. The minimum requirements for secondary nodes are 4 CPUs and 8 GB of memory. Apply the application-specific storage requirements to primary nodes only.

Application	Memory	Storage	CPU
Continuous Delivery for PE with 1-2 concurrent pipelines	4 GB	30 GB	2 CPU
Continuous Delivery for PE with 3 or more concurrent pipelines and/or HA	8 GB	50 GB	3 CPU
Puppet Comply Standalone with low activity	4 GB	30 GB	4 CPU
Puppet Comply with frequent use and/or HA	6 GB	50 GB	6 CPU

Example: HA cluster with Continuous Delivery for PE

An HA cluster capable of running Continuous Delivery for PE requires 3 CPU and 8 GB memory in addition to the per-node baselines listed above; 4 CPUs and 7 GB of a primary server's memory are used for core services, while a secondary server uses 1 CPU and 1.5 GB of memory.

Create the cluster as follows:

- Three primaries with 4 CPUs and 8 GB memory which provides an excess 3 GB of memory for application workloads. Each primary must have 100 GB storage in an unformatted, unpartitioned storage device for Ceph (the cluster uses 50 GB of storage for core services and 50 GB for Continuous Delivery for PE), and 100 GB for `/var/lib`.
- One secondary with 4 CPUs and 8 GB memory, which provides an excess 3 CPUs and 6.5 GB of memory for application workloads. Each secondary must have 80 GB for `/var/lib`.

The four nodes provide a total of 3 CPUs and 9.5 GB of memory for application workloads, of which Continuous Delivery for PE uses 3 CPUs and 8 GB of memory.

Example: HA cluster with Puppet Comply

An HA cluster capable of running Puppet Comply requires 6 CPUs and 6 GB of memory in addition to per-node baselines.

Create the cluster as follows:

- Three primaries with 4 CPUs and 8 GB of memory, which provides an excess 3 GB of memory for application workloads. Each primary must have 100 GB storage in an unformatted, unpartitioned storage device Ceph and 100 GB of storage for `/var/lib`.
- Two secondaries with 4 CPUs and 8 GB of memory, which provides an excess 6 CPUs and 13 GB of memory for application workloads. Each secondary must have 80 GB of storage for `/var/lib`.

The five nodes provide a total of 6 CPUs and 16 GB of memory for application workloads, of which Puppet Comply uses 6 CPUs and 6 GB of memory.

Example: HA cluster with both

An HA cluster capable of running both Continuous Delivery for PE and Puppet Comply requires 9 CPUs and 14 GB of memory in addition to per-node baselines.

Create the cluster as follows:

- Three primaries with 4 CPUs and 8 GB of memory, which provides an excess 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 with 4 CPUs and 8 GB of memory which provides an excess 9 CPUs and 19.5 GB of memory for application workloads. Each secondary must have 80 GB of storage for `/var/lib`.

The six nodes provide a total of 9 CPUs and 22.5 GB of memory for application workloads, of which Continuous Delivery for PE uses 3 CPUs and 8 GB of memory and Puppet Comply uses 6 CPUs and 6 GB of memory.

Or, with larger nodes:

- Three primaries with 7 CPUs and 12 GB of memory, which provides an excess 9 CPUs and 15 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`.

Puppet-supported cluster port requirements

Puppet Application Manager uses the following ports:

Port	Protocol	Purpose	Source	Destination
Puppet application ports				
443	TCP	Web UI (Relies on Server Name Indication to route requests to the application)	Browser	Kubernetes host
8000	TCP	Webhook service (Can be configured to a different port number)	Source control	Kubernetes host

Port	Protocol	Purpose	Source	Destination
Platform ports				
2379	TCP	High availability (HA) communication	etcd on the Kubernetes host	etcd on the Kubernetes host
2380	TCP			
6443	TCP	Kubernetes API	Admin workstation	Kubernetes host
6783	TCP/UDP	Kubernetes networking - Weave	Kubernetes host	Kubernetes host
6784	UDP			
8800	TCP	Puppet Application Manager	Admin browser	Kubernetes host
9001 (Offline installations only)	TCP	Docker registry for offline installations	Job hardware	Kubernetes host
10250	TCP	Kubernetes cluster management	Kubernetes host	Kubernetes host

Remember: Applications that you install with Puppet Application Manager may require other ports to be open. For more information on application-specific port requirements, check the relevant application documentation.

Customer-supported cluster port requirements

Ensure that the ports listed in the following table are open:

Port	Protocol	Purpose	Source	Destination
443	TCP	Web UI	Browser	Puppet Application Manager
8000	TCP	Webhook service	Source control	Puppet Application Manager
8800	TCP	Puppet Application Manager	Admin browser	Kubernetes host

IP address range requirements

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

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 ports and web URLs for internal and outbound network traffic.

TCP Ports:

- 443
- 80

Web URLs:

Category	**URLs**
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

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.

Find more information in the `pam_firewall` README.

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
```

Antivirus and antimalware considerations

Antivirus and antimalware software can impact Puppet Application Manager 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` (Standalone)
- `/var/lib/rook` (HA)
- `/var/lib/docker`
- `/var/lib/kubelet`
- `/var/lib/containerd`

Supported operating systems

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

Operating system	Supported versions
CentOS	7.4, 7.5, 7.6, 7.7, 7.8, 7.9
	8.1, 8.2, 8.3
Red Hat Enterprise Linux (RHEL)	7.4, 7.5, 7.6, 7.7, 7.8, 7.9
	8.1, 8.2, 8.3
Ubuntu (General availability kernels)	18.04
	20.04

Supported browsers

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

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

Related information

Resolve IP address range conflicts on page 63
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.

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. Installing a different runtime from the OS vendor or another third-party onto a node hosting a Puppet-supported Puppet Application Manager cluster can lead to upgrade failures.

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 47.

Related information

Architecture overview on page 14

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

PAM standalone online install

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**.

   The server must meet the following minimum requirements:

Memory	Storage	CPUs	Open ports
2 GB + application requirements	At least 100 GB for `/var/lib` and `/var/openebs`. This is primarily divided among: • 2 GB for `/var/lib/etcd` • 32 GB for `/var/lib/kubelet` • 40 GB for `/var/lib/containerd` • 20 GB for `/var/openebs` + additional application-specific storage.	2 + application requirements	TCP: 443, 2379,2380, 6443, 6783, 8800, 9001 (offline only), and 10250 UDP: 6783, 6784

   Note: 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.

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

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

   If you want to keep Firewalld enabled:

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

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
 sysctl -p"]
    firewalldCmds:
      - ["--permanent", "--zone=trusted", "--add-interface=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
```

3. 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.
4. 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
```

   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 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 64
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 17
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 63
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 14
Puppet Application Manager runs on Kubernetes. Puppet provides several supported configurations for different use cases.

PAM standalone offline install

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**.

   The server must meet the following minimum requirements:

Memory	Storage	CPUs	Open ports
2 GB + application requirements	At least 100 GB for `/var/lib` and `/var/openebs`. This is primarily divided among: • 2 GB for `/var/lib/etcd` • 32 GB for `/var/lib/kubelet` • 40 GB for `/var/lib/containerd` • 20 GB for `/var/openebs` + additional application-specific storage.	2 + application requirements	TCP: 443, 2379,2380, 6443, 6783, 8800, 9001 (offline only), and 10250 UDP: 6783, 6784

   Note: 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.

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

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

   If you want to keep Firewalld enabled:

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

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
 sysctl -p"]
    firewalldCmds:
      - ["--permanent", "--zone=trusted", "--add-interface=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
```

3. 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.
4. Ensure that the nodes can resolve their own hostnames, through either local host mapping or a reachable DNS server.
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 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 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 64
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 17
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 63
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 14
Puppet Application Manager runs on Kubernetes. Puppet provides several supported configurations for different use cases.

PAM HA online install

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**.

   For HA implementations, each server must meet the following minimum requirements. Secondaries must only be added after setting up three primaries:

Node type	Memory	Storage	CPUs	Open ports
Primary	7 GB + application requirements	At least 50 GB on an unformatted, unpartitioned storage device + additional application-specific storage At least 100 GB for `/var/lib.` This is primarily divided among: • 2 GB for `/var/lib/etcd` • 4 GB for `/var/lib/rook` (plus buffer) • 32 GB for `/var/lib/kubelet` • 40 GB for `/var/lib/containerd` Note: Ceph storage back-end 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`.	4 + application requirements	TCP: 443, 2379,2380, 6443, 6783, 8000, 8800, 9001 (offline only) and 10250 UDP: 6783, 6784

Node type	Memory	Storage	CPUs	Open ports
Secondary	1.5 GB + application requirements	At least 80 GB for `/var/lib`. This is primarily divided among: • 32 GB for `/var/lib/kubelet` • 40 GB for `/var/lib/containerd`	1 + application requirements	

Note: 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.

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

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

   If you want to keep Firewalld enabled:

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

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
 sysctl -p"]
    firewalldCmds:
      - ["--permanent", "--zone=trusted", "--add-interface=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
```

3. 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

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.

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

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
```

> **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] # for standard
>  SCSI disks
> ```

   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 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 64
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 17
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 63
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 14
Puppet Application Manager runs on Kubernetes. Puppet provides several supported configurations for different use cases.

PAM HA offline install

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**.

   For HA implementations, each server must meet the following minimum requirements. Secondaries must only be added after setting up three primaries:

Node type	Memory	Storage	CPUs	Open ports
Primary	7 GB + application requirements	At least 50 GB on an unformatted, unpartitioned storage device + additional application-specific storage At least 100 GB for `/var/lib.` This is primarily divided among: • 2 GB for `/var/lib/etcd` • 4 GB for `/var/lib/rook` (plus buffer) • 32 GB for `/var/lib/kubelet` • 40 GB for `/var/lib/containerd` Note: Ceph storage back-end 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`.	4 + application requirements	TCP: 443, 2379,2380, 6443, 6783, 8000, 8800, 9001 (offline only) and 10250 UDP: 6783, 6784

Node type	Memory	Storage	CPUs	Open ports
Secondary	1.5 GB + application requirements	At least 80 GB for `/var/lib`. This is primarily divided among: • 32 GB for `/var/lib/kubelet` • 40 GB for `/var/lib/containerd`	1 + application requirements	

Note: 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.

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

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

   If you want to keep Firewalld enabled:

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

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  firewalldConfig:
    firewalld: enabled
    command: ["/bin/bash", "-c"]
    args: ["echo 'net.ipv4.ip_forward = 1' | tee -a /etc/sysctl.conf &&
 sysctl -p"]
    firewalldCmds:
      - ["--permanent", "--zone=trusted", "--add-interface=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
```

3. 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

4. Ensure that the nodes can resolve their own hostnames, through either local host mapping or a reachable DNS server.
5. Set all nodes used in your HA implementation to the UTC timezone.
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.

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] # for standard
>  SCSI disks
> ```

   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.

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 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 64
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 17

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 63
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 14
Puppet Application Manager runs on Kubernetes. Puppet provides several supported configurations for different use cases.

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 17.

1. Install Puppet Application Manager. For detailed instructions, see PAM HA online install on page 31.

2. Define the configuration values for your Puppet application installation, using Kubernetes YAML format.

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

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

Replace the values indicated:

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

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

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

3. Write your license file and the configuration values generated in step 1 to the following locations:
   - Write your license file to `./replicated_license.yaml`
   - Write your configuration values to `./replicated_config.yaml`
4. Add the Puppet application definition to Puppet Application Manager with the license file and configuration values, passing in the Puppet Application Manager password you set in step 4:

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

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

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

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

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

   When the deployment is complete, the application status changes to **Ready**.
7. Update your DNS or `/etc/hosts` file to include the hostname you chose during configuration.
8. Installation is now complete! Navigate to `https://<HOSTNAME>` and sign into Puppet application.

Related information

PAM HA online install on page 31
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 52
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 17.

1. Install Puppet Application Manager. For detailed instructions, see PAM HA offline install on page 37.

2. Define the configuration values for your Puppet application installation, using Kubernetes YAML format.

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

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

Replace the values indicated:

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

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

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

3. Write your license file and the configuration values generated in step 1 to the following locations:
   - Write your license file to `./replicated_license.yaml`
   - Write your configuration values to `./replicated_config.yaml`
4. Download the application bundle:

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

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

```
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
```

Related information

PAM HA offline install on page 37
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 52
If you installed a Puppet application following the automated offline installation instructions, run a script to upgrade to the latest version.

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

   Note: If you employ a different distribution, contact Puppet Support for more information on compatibility with Puppet Application Manager.
3. Make sure your Kubernetes cluster meets the minimum requirements:
   - Kubernetes version 1.17 or newer.
   - 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).

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

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

   The ingress must have a generous limit for total connection time. Setting the connection timeout to `infinity` in conjunction with an idle timeout is recommended.
4. If you are setting up Puppet Application Manager behind a proxy server, the installer supports proxies configured via `HTTP_PROXY`/`HTTPS_PROXY`/`NO_PROXY` environment variables.

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

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

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

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

2. If you are performing an offline install, ensure the required images are available in a local registry.
   a) Download the release assets matching the CLI version using the following command:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

   a. On the Puppet Application Manager **Dashboard** page, under **Config** > **Optional configuration**, select **View options for using a proxy or external load balancer**.
   b. Enter a new value for **Webhook service port**.

5. To configure your installation further, click **Config** On this tab, you can configure a public hostname, root user, and other settings. These are written as Kubernetes secrets in the deployment manifests. An Ingress is registered with the configured hostname. Take any additional steps needed to ensure you can use that hostname.
   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
   ```

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

Related information

Upgrade PAM on a customer-supported online cluster on page 57
Upgrading Puppet Application Manager (PAM) on a customer-supported online Kubernetes cluster is simple and can be done with a single command.

Upgrade PAM on a customer-supported offline cluster on page 58
Upgrading Puppet Application Manager (PAM) on a customer-supported offline Kubernetes cluster requires a few simple kubectl commands.

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
```

Working with Puppet applications

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

Install applications via the PAM UI

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

Important: Ensure you are using the following Puppet application versions if you want to add more than one Puppet application via the Puppet Application Manager UI:

Application	Version
Continuous Delivery for Puppet Enterprise	4.6.0 or later
Comply	1.0.4 or later

For information on installing Puppet applications via the command line, see Automate PAM and Puppet application online installations on page 43 and Automate PAM and Puppet application offline installations on page 45.

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

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

4. Click **Go to new version**, which redirects you to the **Version history** tab. The newly created version is shown in the **All versions** section of the page.
5. Monitor the new version's preflight checks. The **Running Checks** indicator is shown on the screen while your system is checked to make sure your cluster meets minimum system requirements. When the preflight check is complete:

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

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

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

6. Once the version is ready to deploy, click **Deploy**. On the **Application** tab, monitor the application for readiness.

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

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

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

Update a license for online installations

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

To update the license for an online application:

1. Log in to Puppet Application Manager, click the **License** tab, and then **Sync License**.
2. On the **Version history** tab, click **Deploy**.

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

Update a license for offline installations

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

To update the license for an offline application:

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

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

Upgrade an automated online application installation

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

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

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

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

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

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

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

   Replace `<NAMESPACE>` with the name of the namespace in which you installed PAM (usually `default`).
3. Wait five minutes to allow the software time to process the change.
4. Navigate to `http://<NODE IP ADDRESS>:8800` and log in with the Puppet Application Manager password.

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

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

Related information

Automate PAM and Puppet application offline installations on page 45
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 45
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. Reboot the node.

Load balancer health checks

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

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

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`
- **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.

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

   For standalone deployments, use:

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

   For HA deployments, use:

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

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

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

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).

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. Run the installation command:

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

   > **Note:** This script issues a prompt to run the `task.sh` and `upgrade.sh` scripts on your secondary nodes. Use the versions of these scripts from the downloaded bundle in step 2.

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.

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.

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

1. On your node (or control plane node if you have a HA deployment), rerun the installation script, passing in any arguments you included when installing for the first time:
   - For standalone installs:

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

   - For HA installs:

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

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.

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.

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

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

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

2. Copy the installation bundle to your primary and secondary Puppet Application Manager nodes and unpack it:

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

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

   For standalone installs use:

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

   For HA installs use:

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

> **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:

```
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`

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.

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.

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

1. Upgrade kubectl KOTS:

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

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

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

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

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

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

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

Backing up PAM using snapshots

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

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 yourPAM 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 61.

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:

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

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

Field	Description
Note: Only connections via service principals are currently supported.	
Bucket	The name of the Azure Blob Storage container where snapshots are stored.
Path	**Optional.** The path within the container where all snapshots are stored.
Subscription ID	**Required only for access via service principal or AAD Pod Identity.** The subscription ID associated with the target container.
Tenant ID	**Required only for access via service principal .** The tenant ID associated with the Azure account of the target container.
Client ID	**Required only for access via service principal .** The client ID of a Service Principle with access to the target container.
Client secret	**Required only for access via service principal .** The Client Secret of a Service Principle with access to the target container.
Cloud name	The Azure cloud for the target storage (options: AzurePublicCloud, AzureUSGovernmentCloud, AzureChinaCloud, AzureGermanCloud)
Resource group	The resource group name of the target container.
Storage account	The storage account name of the target container

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

Field	Description
Bucket	The name of the GCS bucket where snapshots are stored.
Path	**Optional.** The path within the bucket where all snapshots are stored.
Service account	The GCP IAM Service Account JSON file that has permissions to read from and write to the storage location.

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 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 Puppet Application Manager.
- For offline installs, both the old and new clusters must use the same version of Puppet Application Manager.

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.
   - 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 Puppet Application Manager 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=5m
> ```
>
> This command requires the jq CLI tool to be installed. It is available in most Linux OS repositories.

7. If the new cluster's hostname is different from the old one, in the Puppet Application Manager UI, update the **Hostname** on the **Config** tab, click **Save Config**, and then redeploy the application.

> **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):

   ```
   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):

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

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

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

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

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

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:

```
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:

```
/usr/local/bin/kubectl-kots reset-password default
```

Update the PAM TLS certificate

A self-signed TLS certificate secures the connection between your browser and Puppet Application Manager (PAM). Once the initial Puppet Application Manager setup process is complete, you can upload new certificates by enabling changes to the installation's Kubernetes secrets.

Use this process if you chose not to add a TLS certificate when installing Puppet Application Manager, or if you need to update your existing TLS certificate.

1. Enable changes to your installation's `kotsadm-tls` Kubernetes secret by running:

```
kubectl -n default annotate secret kotsadm-tls acceptAnonymousUploads=1
```

2. Restart the `kurl-proxy` pod to deploy the change by running:

```
kubectl delete pods $(kubectl get pods -A | grep kurl-proxy | awk '{print
 $2}')
```

3. Once the kurl-rpoxy pod restarts and is back up and running, navigate to `https://<HOSTNAME>:8800/tls` and upload your new TLS certificate.

Reduce recovery time when a node fails

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

How much downtime depends on the following factors:

- Timeout for communication between Kubernetes services (at least one minute to mark the node as unreachable).
- Timeout for the ekco service to determine that pods need to be rescheduled. The default is five minutes after node is marked unreachable.
- Time to restart services (at least two minutes, possibly up to five minutes, if there are complex dependencies).

The ekco service can be configured to reschedule pods more quickly by configuring the installation with a `patch.yaml`similar to the following:

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

```
metadata:
  name: patch
spec:
  ekco:
    nodeUnreachableToleration: 1m
```

Apply the patch during an install or upgrade by including `installer-spec-file=patch.yaml` as an install option.

Important: This patch needs to be included during all future upgrades to avoid resetting the option.

PAM components

Puppet Application Manager (PAM) uses a range of mandatory and optional components.

Support services

- Database: PostgreSQL (single instance) - https://www.postgresql.org/
- Object storage: previously MinIO - https://min.io, now Ceph - https://ceph.io
- tlser for basic TLS cert management - https://github.com/puppetlabs/tlser
- kurl_proxy for HTTPS proxying outside the Ingress (ports besides 80/443): https://github.com/replicatedhq/kots/tree/v1.36.1/kurl_proxy

Kubernetes components

- Networking (CNI): Weave - https://github.com/weaveworks/weave
- Storage (CSI): Rook - https://rook.io, Ceph - https://ceph.io
- Ingress: Project Contour - https://projectcontour.io
- Kubernetes Cluster: kURL - https://kurl.sh
- Embedded kURL Cluster Operator: ekco - https://github.com/replicatedhq/ekco
- Admin Console: KOTS - https://kots.io
- Snapshots: Velero - https://velero.io, Restic - https://restic.net
- Monitoring: Prometheus - https://prometheus.io
- Registry: Docker Registry - https://docs.docker.com/registry/

Optional components

Prometheus (+Grafana) and Velero (+Restic) are optional components:

- Prometheus+Grafana uses 112m/node + 600m CPU, 200MiB/node + 1750MiB RAM
- Velero+Restic uses 500m/node + 500m CPU, 512MiB/node + 128MiB RAM

If you do not need these optional components, they can be omitted from the initial install and further upgrades with a patch similar to the following:

```
apiVersion: cluster.kurl.sh/v1beta1
kind: Installer
metadata:
  name: patch
spec:
  prometheus:
    version: ''
  velero:
    version: ''
```

Important: This patch needs to be included during upgrades to avoid adding the components later.

If you want to remove optional components that are already installed, use the following command:

```
kubectl delete ns/monitoring ns/velero
```

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.
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
```

Installing

To begin using Puppet Comply, you must first complete the initial setup process.

- System requirements on page 67

Refer to these system requirements to allow your Puppet Comply application to connect to Puppet Enterprise (PE).

- Set up Comply on page 68

To start using Puppet Comply, you must complete the setup process, using both Puppet Application Manager (PAM) and Puppet Enterprise (PE).

- Uninstall Comply on page 73

Uninstall Comply by deleting the Comply application and purging the Kubernetes cluster.

System requirements

Refer to these system requirements to allow your Puppet Comply application to connect to Puppet Enterprise (PE).

Open port requirements

Comply is deployed on a Kubernetes cluster, which requires the following ports:

Port	Protocol	Purpose	Source	Destination
PE ports				
8143	TCP	PE integration	Comply	PE Orchestrator
8081	TCP	PE integration	Comply	PuppetDB
4433	TCP	PE integration	Comply	PE RBAC
Comply ports				
443	TCP	Comply access	User browser	Comply UI
443	TCP	Sending reports	Scan target node	Comply server
30303	TCP	Assessor downloads and sending reports	Scan target node	Comply

Note: Port 30303 is not needed if you bring your own ingress when configuring automatic assessor upgrades.

Supported Puppet Enterprise versions

The following versions of Puppet Enterprise (PE) are supported for use with Comply:

PE version
2019.8.4 and later

For more on PE versions, see Puppet Enterprise support lifecycle.

Set up Comply

To start using Puppet Comply, you must complete the setup process, using both Puppet Application Manager (PAM) and Puppet Enterprise (PE).

Important: Before you set up Comply, make sure you have installed Puppet Application Manager (PAM), Puppet Enterprise (PE) and reviewed the system requirements.

Setting up Comply involves the following steps:

- Configure Comply in Puppet Application Manager (PAM) — in an online or offline environment
- Configure Comply certificates in Puppet Enterprise (PE).
- Install the `comply` module.
- Classify the nodes you want to scan in PE.
- Deploy Comply.
- Add your PE credentials to Comply.

- Configure Comply in Puppet Application Manager on page 68

The Comply configuration process creates a Kubernetes cluster and installs the application on that cluster.

- Generate Comply certificates in PE on page 70

You need to generate certificates for Comply in Puppet Enterprise (PE) to enable automatic upgrades of the CIS-CAT assessor and for tasks to upload reports.

- Install the Comply module on page 70

Install the Comply module from Puppet Forge.

- Classify the nodes you want to scan on page 71

In Puppet Enterprise (PE), classify the nodes you want to scan.

- Deploy Comply on page 72

Now that you have completed the setup process, you can deploy Comply.

- Add your PE credentials to Comply on page 72

To allow Comply to communicate with PE, you must add your PE credentials to Comply.

Configure Comply in Puppet Application Manager

The Comply configuration process creates a Kubernetes cluster and installs the application on that cluster.

You can configure Comply in an online or offline environment.

Configure Comply in an online environment

The Comply configuration process creates a Kubernetes cluster and installs the application on that cluster.

Before you begin

Follow the instructions to install Puppet Application Manager.

1. In Puppet Application Manager, upload your Comply license and follow the prompts.

   You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets Comply system requirements.

   > **Note:** The license file is issued by Puppet. If you do not have a license file, contact your Puppet representative. You must also agree to our license agreement. If your licence terms update, for example the expiry date or number of licensed nodes, upload your updated license file to Puppet Application Manager.

2. To configure your installation, click **Config**.
   a) In the **Hostname** field, enter the fully qualified domain name (FQDN) that you want to use to access Comply.

      For example, this could be the name of the node you have installed Comply on. If you choose to use an FQDN that is different from the name of this node, you must configure your domain name system (DNS) to resolve the FQDN to the IP address of the Comply node.
   b) Configure any other settings on the page relevant to your installation.
   c) When you have finished making any necessary changes to the configuration, click **Continue**.
3. 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.

Generate Comply certificates in PE.

Configure Comply in an offline environment

Configure Puppet Comply in an air-gapped or offline environment where the Comply host server does not have direct access to the internet.

Before you begin

Follow the instructions to install Puppet Application Manager.

1. In Puppet Application Manager, upload your Comply licence and follow the prompts.

   You'll be guided through the process of setting up SSL certificates, uploading a license, and checking to make sure your infrastructure meets Comply system requirements.

   > **Note:** The license file is issued by Puppet. If you do not have a license file, contact your Puppet representative. You must also agree to our license agreement. If your licence terms update, for example the expiry date or number of licensed nodes, upload your updated license file to Puppet Application Manager.

2. When prompted, upload the `.airgap` bundle for the most recent version of Comply.

   > **Note:** Puppet provides you with a custom URL from which to download the Comply `.airgap` file. Please open a ticket with Puppet Support to receive your custom download URL.

3. To configure your installation, click **Config**.
   a) In the **Hostname** field, enter the fully qualified domain name (FQDN) that you want to use to access Comply.

      For example, this could be the name of the node you have installed Comply on. If you choose to use an FQDN that is different from the name of this node, you must configure your domain name system (DNS) to resolve the FQDN to the IP address of the Comply node.
   b) Configure any other settings on the page relevant to your installation.
   c) When you have finished making any necessary changes to the configuration, click **Continue**.

4. Monitor the new version's preflight checks. The **Running Checks** indicator is shown on the screen while Comply checks your system 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.

Generate Comply certificates in PE.

Generate Comply certificates in PE

You need to generate certificates for Comply in Puppet Enterprise (PE) to enable automatic upgrades of the CIS-CAT assessor and for tasks to upload reports.

> **Before you begin**
> Make sure you have configured Comply in Puppet Application Manager (PAM).

This process involves generating certificates in Puppet Enterprise (PE) and setting up Mutual Transport Layer Security (MTLS) in Puppet Application Manager (PAM). MTLS enables a secure authenticated connection between your nodes and Comply.

1. SSH into your PE primary server and generate the certificates:

   ```
   puppetserver ca generate --certname <COMPLY-HOSTNAME>
   ```

   This command does the following:
   - Saves the private key to `/etc/puppetlabs/puppet/ssl/private_keys/<COMPLY-HOSTNAME>.pem`
   - Saves the certificate to `/etc/puppetlabs/puppet/ssl/certs/<COMPLY-HOSTNAME>.pem`
2. Log in to Puppet Application Manager, click on the **Version history** tab, and click **Check for update**.
3. Click on the **Config** tab, and scroll down to **Transport layer security (TLS) certificates to interact with PE**.
4. Upload the signed certificate public key, the private key files, and the CA certificate, with the following locations:
   - Paste the contents of `/etc/puppetlabs/puppet/ssl/certs/<COMPLY-HOSTNAME>.pem` to the **TLS certificate** field.
   - Paste the contents of `/etc/puppetlabs/puppet/ssl/private_keys/<COMPLY-HOSTNAME>.pem` to the **TLS private key** field.
   - Paste the contents of `/etc/puppetlabs/puppet/ssl/ca/ca_crt.pem` to the **CA certificate** field.

     > **Note:** To host the assessor on your own supported cluster via NGINX ingress, click the **Bring your own NGINX ingress** check box and enter the FQDN in the **PE TLS FQDN field** — using the same FQDN that you used to generate the TLS certificates.

5. Click **Save Config**.

Install the `comply` module.

Install the Comply module

Install the Comply module from Puppet Forge.

> **Before you begin**
> Make sure you have generated the Comply certificates in PE.

Modules are self-contained, shareable bundles of code and data. The Comply module contains a Bolt task — the tool that runs the CIS assessor on your nodes.

The Comply module lives on Puppet Forge, a repository of thousands of modules. If you're new to PE and Comply, see Managing environment content with a Puppetfile for more information on the Puppetfile and installing modules.

1. Go to the comply module on the Forge.

   Follow the instructions in the **r10k or Code Manager** drop down to add the module declaration to your Puppetfile. You also need to add its dependencies. For example:

   ```
   # Puppet comply module
   mod 'puppetlabs-comply',           '2.1.0'

   # dependencies for comply
   mod 'puppet/archive',              '5.0.0'
   mod 'puppetlabs/chocolatey',       '6.0.1'
   mod 'puppetlabs/inifile',          '5.1.0'
   mod 'puppetlabs/java',             '7.1.0'
   mod 'puppetlabs/ruby_task_helper', '0.6.0'
   mod 'puppetlabs/stdlib',           '7.1.0'
   ```

   If you don’t specify options, Code Manager installs the latest version and does not update it automatically. To always have the latest version installed, specify `:latest` and it updates automatically when a new version is released. Make sure you are always running the latest version of Comply if you intend to use the `:latest` keyword to update the Comply module. To install a specific version of the module that does not update automatically, specify the version number as a string.

   > **Important:** If you choose a specific version of the module, it MUST be the same as the Comply version. For example, version 2.1.0 of the module must be installed for Comply 2.1.0.

2. SSH into your PE primary server and deploy the code:

   ```
   puppet-code deploy --all
   ```

Classify the nodes you want to scan in Puppet Enterprise (PE).

Classify the nodes you want to scan

In Puppet Enterprise (PE), classify the nodes you want to scan.

> **Before you begin**
> Make sure you have installed the `comply` module.

Classification is when you create a node group, add nodes to the group, and assign *classes* to the group — in this case, the `comply` class. Classes are the blocks of Puppet code used to configure nodes and assign resources to them. If you are new to Puppet, see Grouping and classifying nodes for more information.

1. In the PE console, click **Node groups**.
2. Create a new node group or select an existing node group that you want to scan.
3. On the **Classes** tab — in the **Add new class** field — select the `comply` class.
4. Click **Add class**.
5. In your new `comply` class, select the `scanner_source` **Parameter**.

   > **Note:** *Parameters* allow a class to request external data.

6. Change the default parameter **Value** to one of the following assessor distribution files:
   - If using the Puppet supported cluster: `https://<COMPLY-HOSTNAME>:30303/assessor`
   - If using NGINX Ingress: `https://<PE-TLS-FQDN>/assessor`
7. Click **Add to node group**, and then commit the changes.
8. Run Puppet **twice**.

Deploy Comply.

Deploy Comply

Now that you have completed the setup process, you can deploy Comply.

Before you begin
Make sure you have classified the nodes you want to scan in Puppet Enterprise (PE).

1. Navigate to Puppet Application Manager. 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.

   > **Tip:** You can monitor the deployment's progress by running `kubectl get pods --watch`.

   When the deployment is complete, the application status changes to **Ready**. Comply is now deployed!
2. Navigate to `https://<COMPLY-HOSTNAME>` using the name of the FQDN you created in Configure Comply and sign into Comply.

Add your PE credentials to Comply.

Add your PE credentials to Comply

To allow Comply to communicate with PE, you must add your PE credentials to Comply.

Before you begin
Make sure you have deployed Comply.

Adding your PE credentials authenticates Comply with Role Based Access Control (RBAC). Your PE account requires the following permissions:

Type	Action	Instance
Console	View	-
Job Orchestrator	Start, stop and view jobs	-
Node Groups	View	All
Nodes	View node data from PuppetDB	-
Tasks	Run Tasks	Task: `comply::backup_assessor` Permitted on : All nodes
Tasks	Run Tasks	Task: `comply::ciscat_scan` Permitted on: All nodes

For more information on permissions, see RBAC permissions.

1. In Comply — located at `https://<COMPLY-HOSTNAME>/` — click **Settings**.
2. Enter your PE **hostname**, **username**, and **password**.
3. Click **Submit**.

   > **Tip:** You can refresh the PE node and fact information by clicking **Refresh data**.

You'll now see a list of your classified nodes on the **Nodes** page.

You have completed the Comply setup process! You can now start running CIS scans on your nodes. If you're new to Comply, try out the getting started guide.

Uninstall Comply

Uninstall Comply by deleting the Comply application and purging the Kubernetes cluster.

1. From the command line of the node where you have Comply installed, run the following command to delete the Comply application:

```
# kubectl delete $(kubectl api-resources --verbs=delete -o name | paste -
sd "," -) -A -l app.kubernetes.io/part-of=comply
```

2. On the same node, run the following command to uninstall the embedded Kubernetes cluster:

```
curl https://k8s.kurl.sh/comply/tasks.sh | sudo bash -s reset
```

> **Note:** This command resets the Replicated installation with a purge.

3. Reboot your node to clear the `kube-ipvs0` device.

Upgrading

New versions of Puppet Comply are released regularly. Upgrading to the current version ensures you are always taking advantage of the latest features, fixes, and improvements.

> **Important:** The CIS-CAT assessor setup process is embedded in the `comply` module. To ensure you always have the latest version, upgrade the `comply` module *before* you upgrade the Comply application. Note that you cannot run scans until you complete *both* of these upgrades.

Upgrade from Comply 2.0.0 to 2.1.0

Comply 2.1.0 automatically upgrades the CIS-CAT assessor to the latest version every time you upgrade Comply.

The upgrade process involves generating certificates in Puppet Enterprise (PE) and setting up Mutual Transport Layer Security (MTLS) in Puppet Application Manager (PAM). MTLS enables a secure authenticated connection between your nodes and Comply.

1. SSH into your PE primary server and generate the certificates:

```
puppetserver ca generate --certname <COMPLY-HOSTNAME>
```

   This command does the following:

   - Saves the private key to `/etc/puppetlabs/puppet/ssl/private_keys/<COMPLY-HOSTNAME>.pem`
   - Saves the certificate to `/etc/puppetlabs/puppet/ssl/certs/<COMPLY-HOSTNAME>.pem`

2. Log in to Puppet Application Manager, click on the **Version history** tab, and click **Check for update**.
3. Click on the **Config** tab, and scroll down to **Transport layer security (TLS) certificates to interact with PE**.

4. Upload the signed certificate public key, the private key files, and the CA certificate, with the following locations:
   - Paste the contents of `/etc/puppetlabs/puppet/ssl/certs/<COMPLY-HOSTNAME>.pem` to the **TLS certificate** field.
   - Paste the contents of `/etc/puppetlabs/puppet/ssl/private_keys/<COMPLY-HOSTNAME>.pem` to the **TLS private key** field.
   - Paste the contents of `/etc/puppetlabs/puppet/ssl/ca/ca_crt.pem` to the **CA certificate** field.

   > **Note:** To host the assessor on your own supported cluster via NGINX ingress, click the **Bring your own NGINX ingress** check box and enter the FQDN in the **PE TLS FQDN field** — using the same FQDN that you used to generate the TLS certificates.

5. If you want to have Comply update the CIS-CAT Assessor automatically, select **Automatically kick off PE jobs on assessor upgrade** on the **Config** page.

   If you select this option, on upgrade Comply kicks off 2 PE agent runs: the first to download the new assessor, and the second update the facts in PE.

   > **Tip:** Because this option starts PE jobs automatically on upgrading Comply, systems administrators, especially of larger implementation, may wish to consider leaving this option unchecked. Assessor upgrade then takes place automatically when the next two PE jobs are run.

   Comply requires the latest version of the assessor on the node in order to perform runs. A background task runs to check if nodes have been upgraded every 15 minutes if this option is selected and every hour if it is not selected. If a node does not upgrade and remains red on the **Inventory** page, run the Puppet agent. If the upgrade continues to fail, see the Puppet agent logs for more information.
6. Click **Save Config**.
7. Navigate to Puppet Enterprise (PE), and update the default **value** of the `comply` class *scanner_source* parameter to one of the following assessor distribution files:
   - If using the Puppet supported cluster: `https://<COMPLY-HOSTNAME>:30303/assessor`
   - If using NGINX Ingress: `scanner_source` to: `https://<PE-TLS-FQDN>/assessor`

   For more information, see Classify the nodes you want to scan in PE.
8. Click **Add to node group**, and then commit the changes.
9. Upgrade the `comply` module.
   a) Update your Puppetfile with the latest version of the `comply` module and its dependencies.
   b) Deploy code by running the `puppet-code deploy --all` command.

   For more information, see upgrade the `comply` module.
10. Navigate back to Puppet Application Manager. After pre-flight checks have completed successfully, click **Go to updated version**, and then click **Deploy**.

> **Note:** If the upgrade of an assessor on a node fails, the node is marked in red on the **Inventory** page. Failures may be due to network issues. If that is the case, Comply attempts to upgrade the node once connectivity returns. An hourly background task runs to check if nodes have been upgraded or not. If a node does not upgrade and remains red on the **Inventory** page, run the Puppet agent. If the upgrade continues to fail, see the Puppet agent logs for more information.

Upgrade the Comply module

Upgrade to the latest version of the `comply` module in Puppet Enterprise (PE).

> **Note:** Take note of module dependencies when upgrading to a new major version — you need to upgrade these as well.

1. Update your Puppetfile with the latest version of the `comply` module and its dependencies. For example:

```
# Puppet comply module
mod 'puppetlabs-comply',           '2.1.0'

# dependencies for comply
mod 'puppet/archive',              '5.0.0'
mod 'puppetlabs/chocolatey',       '6.0.1'
mod 'puppetlabs/inifile',          '5.1.0'
mod 'puppetlabs/java',             '7.1.0'
mod 'puppetlabs/ruby_task_helper', '0.6.0'
mod 'puppetlabs/stdlib',           '7.1.0'
```

2. SSH into your PE primary server and deploy code by running the `puppet-code deploy --all` command.

Upgrade Comply in an online environment

Check for download and deploy updates from the **Version history** tab in the Puppet Application Manager UI.

> **Before you begin**
> Upgrade the `comply` module.

1. In the Puppet Application Manager UI, click **Version history**.
2. Click **Check for updates**.

   Configure an automatic update check by clicking **Configure automatic updates**. You can check for updates hourly, every four hours, daily, weekly, or at a custom interval.
3. If an update is available, Puppet Application Manager downloads it for you and performs preflight checks on your system to make sure your cluster meets system requirements for the new version. Review the outcome of these checks by clicking **View preflight**.
4. When you're ready to upgrade to the new version of Comply, click **Deploy**.

Upgrade Comply in an offline environment

If your environments do not have direct access to the internet, use the links below to upgrade to the latest version of Comply.

> **Before you begin**
> Upgrade the `comply` module.

1. Navigate to the portal provided to you by Puppet in the licence email, for example, `https://get.replicated.com/airgap/#/kots/comply/`, and login with the password.
2. Select **Embeded cluster** and download the latest Comply release `.airgap` file.
3. Log into Puppet Application Manager — `https://<PLATFORM-ADMIN-CONSOLE-ADDRESS>:8800`.
4. Select **Version history**, and upload the new version of the `.airgap` file that you downloaded in step 2.
5. Click **Deploy**.

Desired compliance

Set your desired compliance. This is the benchmark and profile that you to assign to a particular node. It is what is scanned on that node by default. Most of the time, you only need to set this once for your nodes.

There are two ways to specify desired compliance:

- **Option 1: Allow Comply to automatically set desired compliance.** Based on fact information from PE, Comply can automatically assign an appropriate benchmark for each operating system, along with a Level 1 profile, to nodes that have not been set. This is the quickest way to get up and running with desired compliance.
- **Option 2: Manually set desired compliance.** If you want to choose a specific benchmark and profile for your nodes, or even a custom profile, option two provides this flexibility.

> **Note:** Only one benchmark and profile can be assigned to each node.

Option 1: Allow Comply to automatically set desired compliance

1. In Comply, click **Nodes**.

   Comply lists the nodes that have been classified with the `comply` class. If you do not see any nodes, ensure you have classified your nodes correctly.
2. In the message box that appears in the top right corner, click **Apply suggested profiles**.

   Comply automatically assigns the appropriate benchmark, along with a Level 1 profile, to all the nodes that have not already been set, on your *current* page. To apply the suggested profile to all the nodes in your inventory, you must do this on every page.

   > **Tip:** If you want to customize your scans to fit your organization's internally defined standards, see Creating custom profiles, which shows you how to exclude rules in a profile.

The ## sign in the **Profile assigned** column tells you that the desired compliance is set. You can view the node's information, including its assigned benchmark and profile, by clicking on the node. If you want to change a node's desired compliance, use the drop-down menu and click **Update**.

Now that you have applied desired compliance, you can use this option to run a CIS scans.

Option 2: Manually set desired compliance

1. In Comply, click **Nodes**.

   Comply lists the nodes that have been classified with the `comply` class. If you do not see any nodes, ensure you have classified your nodes correctly.
2. Click on the node for which you want to specify desired compliance.

   In the window that appears on the right, you can see facts about the node and whether desired compliance has been set.
3. Choose the CIS benchmark and profile that you want to assign to the node.

   The benchmark and profile you set here is the desired compliance option for future scans.

   If you have created a custom profile, you can set it as the desired compliance by clicking **Use an associated custom profile?**.
4. Click **Update**.

The ## sign in the **Profile assigned** column tells you that the desired compliance is set. You can view the node's information, including its assigned benchmark and profile, by clicking on the node. If you want to change a node's desired compliance, use the drop-down menu and click **Update**.

Now that you have applied desired compliance, you can use this option to run scans.

Related information

Create a custom profile on page 77
Create a custom profile based on an existing benchmark.

Run a CIS scan on page 77

Run your desired compliance scan or an ad-hoc scan on your nodes.

Custom profiles

A custom profile is a benchmark profile that you customize to fit your organization's internally defined standards.

You can base a custom profile on an existing benchmark and profile combination, and then specify which rules you want visible in scan reports.

Create a custom profile

Create a custom profile based on an existing benchmark.

1. Navigate to **Custom profiles**.
2. Click **Create custom profile**.
3. Select a **Benchmark** and **Profile**.
4. Deselect rules in the profile that you **do not** want to scan, and click **Next**.
5. Enter the name of the profile and, optionally, a description.
6. Click **Save custom profile**.

   Your custom profile appears as an option when you assign the associated benchmark to a node.

Navigate to **Nodes** to set your custom profile as the desired compliance for your nodes, or preform an ad-hoc scan by selecting your custom profile on the **Scan** page.

Related information

Option 2: Manually set desired compliance on page 76

Run a CIS scan on page 77

Run your desired compliance scan or an ad-hoc scan on your nodes.

Run a CIS scan

Run your desired compliance scan or an ad-hoc scan on your nodes.

You can run scans on individual nodes by selecting the **Scan node** drop-down on the node's **Node detail** page, and then selecting **desired compliance** or **custom** options if you have those set up. Then follow the scan wizard as outlined in steps 4-7 on this help page.

You can also all nodes by selecting the **Scan all nodes** drop-down on the **Dashboard** page, and then selecting **desired compliance** or **custom** options if you have those set up. Then follow the scan wizard as outlined in steps 4-7 on this help page.

1. In Comply, click **Scan** in the sidebar menu.

   > **Tip:** You can also run a full scan from the **Scan Reports** page by click **Run a new scan**.

2. In the **Benchmark** drop-down, select **Desired compliance** or a benchmark and profile of your choice.

   If you have not set desired compliance, see Setting desired compliance for instructions.

3. Next, select an option from the **Profile** drop-down. If you want to use a custom profile for this scan, select the **Use an associated custom profile?** option and choose the relevant option from the **Custom profile** drop-down.

   For more information on custom profiles, see Custom profiles on page 77

4. Click **Next** to review the PE credentials and environment you want the scan to run on.

5. Click **Next** to see the nodes selected for scanning.

   To only scan a subset of nodes, deselect any that you do not want to include.

6. Click **Scan**, and then **Scan** again to confirm.

   You'll be taken to the **Activity Feed**, which lists each scan. Scans are run as a task in PE. To see the details of the job, click on the job ID to be taken to PE.

   > **Tip:** You can also run a scan by clicking the **Scan nodes** button at the top right corner on several pages. This option uses the nodes listed on the page you are currently viewing.

7. In Comply, navigate to **Compliance dashboard** to see the results of your scans.

   See Viewing scan results for a description of the scan data.

   To find out how you can enforce and automate CIS benchmarks on your failing nodes, see Enforce CIS benchmarks.

Related information

Scan results on page 79
View the results of your CIS scans and find out whether your nodes are compliant.

Create a custom profile on page 77
Create a custom profile based on an existing benchmark.

Enforce CIS benchmarks on page 80
Puppet Comply provides visibility into your compliance status, but it cannot fix your failing nodes. Instead, you can use Puppet's Compliance Enforcement Modules (CEM).

CIS scan reports

The **Scan Report** page provides info on the latest CIS scan and is where you can run CIS scan reports to receive data about the compliance of your nodes.

The **Scan report metrics bar** at the top of the **Scan Report** page is divided into two sections: **Compliance scan status** and **Puppet Enterprise job status** areas.

The **Compliance scan status** area provides a brief overview of the number of nodes that have passed and failed compliance, as well the error percentage, the rules that couldn't be evaluated across nodes, and the scan initiation date and time.

The **Puppet Enterprise job status** area in the **Scan report metrics bar** shows the number of nodes that ran the CIS scanner job successfully, the number that failed to run the scanner job and the number of nodes that showed an error for the scanner job.

Click **Run a new scan** to kick off a new scan of your network.

More detailed information on the success and failure of rules is given on the **Nodes** tab. The **Rules** tab provides detail on the performance of individual rules in the scan.

Rules tab

The **Rules** table on the **Rules** tab lists all the rules that were assessed as part of the latest scan. The **Rules** table provides information on the rule profile, and the number of nodes on which the rule failed. Click the rule name on any given row to visit the **Rule detail** page for the selected node.

Nodes tab

The **Nodes** table on the **Nodes** tab lists all the nodes that were part of the latest scan. The **Nodes** table provides information on the node profile, and the percentage of rules in compliance on each node. Click the node name in any given row to visit the **Node detail** page for the selected node.

Scan results

View the results of your CIS scans and find out whether your nodes are compliant.

Scan report metrics bar

The **Scan report metrics bar** provides a brief overview of the number of nodes that have passed and failed, as well the error percentage, the rules that couldn't be evaluated across nodes, and the scan initiation date and time.

Compliance dashboard

The **Compliance dashboard** provides a breakdown of your latest CIS scan.

It has three components:

- The **Node results** chart shows the percentage of rules that passed and failed across all of your nodes. These percentages are shown with the statuses of **Pass**, **Fail**, **Error**, and **Unknown**.
- The **Desired compliance** chart shows the number of nodes that have desired compliance set. Desired compliance is the default benchmark and profile that you have assigned to that node.
- The **Node results table** lists information about the latest scan for each node.

Node compliance

From the **Compliance dashboard**, click on a node name to get to the **Node compliance** page, and see the results of the latest scan on that node. The data includes:

- The **Rules checked** chart shows a status breakdown for the latest scan on that node — the rules that passed, failed, had an error, or an unknown status. The percentage displayed in the **Rules checked** pie chart is an aggregate of the these four statuses.

  Other statuses that are not included in scoring are included in the table below:

Value	Included in Scoring?	Description
Pass	Yes	The target system or component state satisfied all the conditions of the check(s)/rule(s) for the recommendation.
Fail	Yes	The target system or component state did not satisfy at least one condition of the check(s)/rule(s) for the recommendation.
Error	Yes	The assessor checking engine encountered a system error and could not complete the test. The status of the target's compliance is not certain.
Unknown	Yes	Assessor was unable to collect, interpret, or evaluate against the check/rule conditions associated with the recommendation.
Manual	No	This recommendation cannot be fully automated and requires manual evaluation. On CIS Benchmarks, a recommendation is deemed important during the consensus process but cannot be fully and reliably verified without organizational manual verification. Corresponds to *xccdf* terminology of "Informational".

Not Applicable	No	The rule(s)/check(s) were not applicable to the target. This typically occurs when the wrong benchmark is selected for the platform i.e.: platform mismatch.
Not Checked	No	The recommendation was not evaluated as there are no rule/check properties.
Not Selected	No	This recommendation was not part of the profile selected for the configuration assessment.
Informational	No	This is the same result that is displayed as Manual on the HTML report. The recommendation cannot be fully automated and requires manual evaluation.

- The **Rule scan results** table lists each rule that was checked and the status of that rule from the latest scan.

Rule results

From **Node compliance**, click on a rule title for **Rule results** page, and see the details of that rule and the status of the nodes it is checked on. The data includes:

- A tabbed section that displays information about each rule:
  - **Description** — information on what is being checked.
  - **Rationale** — the reason why it is important to check that rule.
  - **Fix** — the steps you can take to fix the rule if it is failing on a node.
- The **Node results** donut shows the nodes that this rule is checked on, and which nodes passed or failed the rule.
- The **Node compliance result** table lists each node the rule has been checked against and shows the current status — when it was last checked and when it last passed that rule.

Related information

Comply terminology on page 4
Key terms to be familiar with when using Puppet Comply.

Desired compliance on page 75
Set your desired compliance. This is the benchmark and profile that you to assign to a particular node. It is what is scanned on that node by default. Most of the time, you only need to set this once for your nodes.

Enforce CIS benchmarks

Puppet Comply provides visibility into your compliance status, but it cannot fix your failing nodes. Instead, you can use Puppet's Compliance Enforcement Modules (CEM).

Available to premium content subscribers, CEM consists of two modules — `cem_linux` and `cem_windows`. These are supported Puppet modules developed specifically to bring your Puppet Enterprise (PE) managed nodes under CIS (Center for Internet Security) compliance.

By default, CEM enforces the latest CIS Level 1 benchmarks on your nodes, automating hundreds of operating system settings — the default profile depends on your operating system. You can also customize these configurations to suit your organization's policies.

To get started with CEM, you need to add the Forge premium content API key to your primary Puppet server. For instructions, visit cem_linux or cem_windows on Puppet Forge.

Troubleshooting

Use this section to troubleshoot issues with your Puppet Comply installation.

Reset your Comply password

If you forget your password, you can reset it in the user admin console.

1. SSH into your Comply node and run the following commands to retrieve the admin username and password:

```
kubectl exec $(kubectl get pod -l app.kubernetes.io/name=comply-auth -o
 jsonpath="{.items[0].metadata.name}") -- /bin/bash -c 'cat /etc/keycloak/
admin-user'
```

```
kubectl exec $(kubectl get pod -l app.kubernetes.io/name=comply-auth -o
 jsonpath="{.items[0].metadata.name}") -- /bin/bash -c 'cat /etc/keycloak/
admin-password'
```

2. Navigate to `https://<COMPLY-HOSTNAME>/auth/admin` using the FQDN of your Comply node.
3. Login using the credentials from step 1.
4. Navigate to **Users**.
5. Click **View all users** and select the user account you want to update, and click **Edit**.
6. Select the **Credentials** tab and the reset password.

Access logs

If you run into issues with Puppet Comply, you can download the relevant log files. The Comply logs are stored in Puppet Application Manager.

1. Log into Puppet Application Manager — `https://<PUPPET-APPLICATION-MANAGER-ADDRESS>:8800`.
2. Select the **Troubleshoot** tab, and click **Analyse Comply**.
3. Download the bundle of log files.

Resolve Comply domain

If the Puppet Comply gatekeeper is unable to resolve the Comply domain, try the following troubleshooting steps.

When you assign a hostname to Comply, it needs to be resolved by the pods in your Kubernetes cluster. A preflight check verifies the domain you specified in the configuration is resolvable. You must ensure that the nodes can resolve their own hostnames, through either local host mapping or a reachable DNS server.

1. To verify your whether your hostname is resolvable, run the following commands:

```
kubectl exec $(kubectl get pod -l app=kotsadm -o
 jsonpath="{.items[0].metadata.name}") -- /bin/sh -c 'curl --SI
 <hostname>'
```

   If the hostname was resolved, the command returns an exit code 0 with no output.

   If the hostname cannot be resolved, the command returns an exit code 6. Proceed to step 2 to add DNS entries.

2. To add DNS entries for CoreDNS, run the following command to open the CoreDNS configuration maps:

```
kubectl -n kube-system edit configmaps coredns
```

3. Add a `hosts` entry below `kubernetes`. This is where you configure the DNS entry for Comply. For example:

```
kubernetes cluster.local in-addr.arpa ip6.arpa {
   pods insecure
   fallthrough in-addr.arpa ip6.arpa
   ttl 30
}
hosts {
  10.23.24.25 comply.mycompany.net comply // IP_address canonical_hostname
 [aliases...]
  fallthrough
}
prometheus :9153
```

4. Run the command from step 1 to verify whether the DNS entry was updated:

```
kubectl exec $(kubectl get pod -l app=kotsadm -o
 jsonpath="{.items[0].metadata.name}") -- /bin/sh -c 'curl --SI
 <hostname>'
```

5. Re-run the preflight checks.

Resolve failed assessor upgrade

If an upgrade of the assessor has failed on one of your nodes, try the following troubleshooting step.

If the upgrade of an assessor on a node fails, the node is marked in red on the **Inventory** page. Failures may be due to network issues. If that is the case, Comply attempts to upgrade the node once connectivity returns. An hourly background task runs to check if nodes have been upgraded or not. If a node does not upgrade and remains red on the **Inventory** page, run the Puppet agent. If the upgrade continues to fail, see the Puppet agent logs for more information.