



Using Puppet Enterprise

(Generated on September 16, 2011, from git revision
8830f89918de137c687bd1616c0054e57ca66394)

NAVIGATION

- Introduction
- [Overview](#)
- [Installing](#)
- [Upgrading](#)
- [Using Puppet Enterprise](#)
- The Accounts Module
 - [The accounts::user Type](#)
 - [The accounts Class](#)
- Puppet Compliance
 - [Puppet Compliance Basics and UI](#)
 - [Using the Puppet Compliance Workflow](#)□
 - [Compliance Workflow Tutorial](#)□
- [Known Issues](#)
- [Troubleshooting](#)
- [Answer File Reference](#)

Welcome to Puppet Enterprise!

Thank you for choosing Puppet Enterprise 1.2. This distribution is designed to help you get up and running with a fully operational and highly scalable Puppet environment as quickly as possible.

About This Manual

This manual covers Puppet Enterprise version 1.2, and documents its installation and maintenance. In its current form, it is not intended to teach the use of Puppet or MCollective to new users. For an introduction to Puppet, we recommend starting with the [Learning Puppet](#) series; for an introduction to MCollective, we recommend starting with the [MCollective documentation](#).

System Requirements

Puppet Enterprise 1.2 supports the following operating system versions:

Operating system	Version	Arch	Support
Red Hat Enterprise Linux	5 and 6	x86 and x86_64	master/agent
Red Hat Enterprise Linux	4	x86 and x86_64	agent
CentOS	5 and 6	x86 and x86_64	master/agent
CentOS	4	x86 and x86_64	agent
Ubuntu LTS	10.04	32- and 64-bit	master/agent
Debian	Lenny (5) and Squeeze (6)	i386 and amd64	master/agent
Oracle Enterprise Linux	5 and 6	x86 and x86_64	master/agent
Scientific Linux	5 and 6	x86 and x86_64	master/agent
SUSE Linux Enterprise Server	11	x86 and x86_64	master/agent
Solaris	10	SPARC and x86_64	agent

Puppet's hardware requirements will vary widely depending on the number of resources you are managing, but in general, you can expect a 2-4 CPU puppet master with 4 GB of RAM to manage approximately 1,000 agent nodes.

The ActiveMQ server, which runs on the puppet master, will require at least 512 MB of RAM for its Java VM instance. It also requires very accurate timekeeping, which can potentially present problems when running the puppet master role under some virtualization platforms. For this reason, we recommend running the puppet master role on a Xen, KVM, or physical server if you plan to use MCollective.

The puppet agent role has very modest requirements, and should run without problems on nearly any physical or virtualized hardware.

What's New in PE 1.2

Version 1.2 is a major release of Puppet Enterprise, which adds the following new features:

MCollective integration

Puppet Enterprise now bundles version 1.2.1 of Marionette Collective, Puppet Labs' server orchestration framework.

Puppet Compliance

Puppet Compliance is a new auditing workflow that uses Puppet Dashboard to track changes to resources.

Puppet Dashboard 1.2

This release upgrades Puppet Dashboard to version 1.2, which boasts vastly improved performance and can export most node views to CSV.

Puppet 2.6.9

This release updates Puppet to version 2.6.9, the latest release in the stable 2.6 series.

Notable new features and fixes since the last version of Puppet Enterprise (which shipped with Puppet 2.6.4) include:

- A new shell exec provider that passes the command through /bin/sh, mimicking the behavior of Puppet 0.25.
- Dramatically improved puppet master performance under Passenger.
- Puppet master under Passenger no longer requires two runs to detect changes to manifests.
- A stable inventory service API, which is used by the new version of Puppet Dashboard.
- Selectors can now use hashes.
- Nested hashes are now allowed.
- Creation of system users is now allowed.
- External node classifiers can now support parameterized classes.
- New puppet inspect application for the Puppet Compliance workflow.
- \$name can now be used to set default values in defined resource types

For more detailed information about the differences between Puppet 2.6.4 and 2.6.9, please see the [Puppet release notes](#).

Accounts 1.0 Module

PE now includes a ready-to-use Puppet module for managing user accounts. This module provides an account defined type and an optional wrapper class that lets you store all account data in an external YAML file.

Stdlib 2.0 Module

This helper module extends Puppet with:

- A new `facts.d` Facter extension that can read system facts from an arbitrary mix of text, yaml, json, and executable script files stored in `/etc/puppetlabs/facter/facts.d` or `/etc/facter/facts.d`.
- Several data validation functions, including `validate_array`, `validate_bool`, `validate_hash`, `validate_re`, `validate_string`, and `has_key`.
- Two data loading functions: `getvar` and `loadyaml`.
- The `file_line` resource type, which ensures that a given line exists verbatim somewhere in a file.□
- The `anchor` resource type, which is used to enhance readability in certain Puppet Labs modules.
- A set of standardized run stages (available when class `stdlib` is declared).

How to Get Help and Support

If you run into trouble, you can take advantage of Puppet Labs' enterprise technical support.

- To file a support ticket, go to support.puppetlabs.com.
- To reach our support staff via email, contact support@puppetlabs.com.
- To speak directly to our support staff, call 1-877-575-9775.□

Other Documentation

For help with features not specific to Puppet Enterprise, please see the [main Puppet documentation](#) and the [MCollective documentation](#).

NAVIGATION

- [Introduction](#)
- [Overview](#)
- [Installing](#)
- [Upgrading](#)
- [Using Puppet Enterprise](#)
- The Accounts Module
 - [The accounts::user Type](#)
 - [The accounts Class](#)
- Puppet Compliance
 - [Puppet Compliance Basics and UI](#)
 - [Using the Puppet Compliance Workflow](#)□
 - [Compliance Workflow Tutorial](#)□

- [Known Issues](#)
- [Troubleshooting](#)
- [Answer File Reference](#)

Overview

What Is Puppet Enterprise?

Puppet Enterprise is a fully supported software distribution of the Puppet family of systems management tools. It adds several premium enterprise-only features, and deploys out-of-the-box into a highly scalable production-ready configuration.□

- Puppet (version 2.6.9) is an industry-leading configuration management platform that lets you□ describe a desired system state as code and enforce that state on any number of machines.
- MCollective (version 1.2.1) is a message-based server orchestration framework for fast parallel command execution.
- Puppet Dashboard (version 1.2) is a web interface to Puppet. It can view and analyze Puppet's reports, and can simplify the process of assigning your existing Puppet classes to nodes.
- Puppet Compliance is an enterprise-only extension to Puppet Dashboard that enables a new workflow for auditing changes to resources.□
- Facter (version 1.6.0) is a system data discovery utility used by both Puppet and MCollective.
- Accounts (version 1.0.0) is a ready-to-use Puppet module for managing user accounts.

To get maximum compatibility and performance on your existing systems, PE bundles and maintains its own versions of Ruby, Apache, and all required libraries, and installs all included software in the `/opt/puppet` directory. This lets us enable advanced features while still shielding users from dependency conflicts, and it offers a relatively seamless transition path for users□ migrating from a manually maintained Puppet installation.

Roles

Puppet Enterprise's features are divided into three main roles, any or all of which can be installed on a single computer:

- The puppet agent role should be installed on every node Puppet will be managing; it installs Puppet, and enables the puppet agent service (`pe-puppet`) that checks in with the puppet master every half-hour and applies the node's catalog. This role also installs (but doesn't automatically enable) the MCollective server, which listens and responds to messages on the ActiveMQ Stomp bus; to enable MCollective on a node, use Puppet to assign the `mcollectivepe` class to it.
- The puppet master role should be installed on exactly one server at your site; it installs Puppet, Apache, the ActiveMQ server, and the MCollective control client. Servers with this role will respond to catalog requests from puppet agent nodes (using instances of puppet master managed by the `pe-httpd` Apache service), and will act as the hub for all MCollective traffic at the site. Puppet master can be configured during installation to submit reports to and request node□ classifications from Puppet Dashboard.□
- The Puppet Dashboard role should be installed on exactly one server at your site; it installs Puppet Dashboard (with the Puppet Compliance extension), Puppet, and Apache, and configures□

them to respond to requests from the puppet master, serve a web interface to your site's administrators, and act as a machine inventory browser and file content viewer. Unlike the other two roles, Puppet Dashboard is an optional component of your site. The Dashboard role should usually be installed on the same machine as the puppet master role; splitting the two roles between different machines can increase performance in some situations, but will require some additional configuration.

Licensing

Puppet Enterprise can be evaluated with a complementary ten-node license; beyond that, a commercial per-node license is required for use. A license key file will have been emailed to you after your purchase, and the puppet master will look for this key at `/etc/puppetlabs/license.key`. Puppet will log warnings if the license is expired or exceeded, and you can view the status of your license by running `puppet license` at the command line on the puppet master.

To purchase a license, please see the [Puppet Enterprise pricing page](#), or contact Puppet Labs at sales@puppetlabs.com or (877) 575-9775. For more information on licensing terms, please see [the licensing FAQ](#). If you have misplaced or never received your license key, please contact sales@puppetlabs.com.

NAVIGATION

- [Introduction](#)
- [Overview](#)
- Installing
- [Upgrading](#)
- [Using Puppet Enterprise](#)
- The Accounts Module
 - [The accounts::user Type](#)
 - [The accounts Class](#)
- Puppet Compliance
 - [Puppet Compliance Basics and UI](#)
 - [Using the Puppet Compliance Workflow](#)
 - [Compliance Workflow Tutorial](#)
- [Known Issues](#)
- [Troubleshooting](#)
- [Answer File Reference](#)

Installing Puppet Enterprise

Puppet Enterprise is the easiest way to install Puppet in a production-ready state — after some quick pre-install sanity checks, simply run the interactive installer and get started immediately.

Preparing to Install

Before installing Puppet Enterprise at your site, you should:

- Decide in advance which server will fill the role of puppet master, and decide whether it will also run Puppet Dashboard. If you are splitting these two roles, you must perform some [extra configuration](#) after installation.
- Ensure that the puppet master server can be reached via domain name lookup by all of the future puppet agent nodes at the site.
- Optionally, add a CNAME record to your site's DNS configuration (or an alias in the relevant `/etc/hosts` files) to ensure that your puppet master can also be reached at the hostname `puppet` — since this is the default puppet master hostname, using it can simplify installation on your agent nodes.
- Configure your firewalls to allow network traffic on ports 8140 (for Puppet), 61613 (for MCollective's Stomp messages), and 3000 (for Puppet Dashboard).

You should plan to install PE on the puppet master (and the Dashboard server, if you have chosen to separate the two) before installing on any agent nodes.

Choosing Your Installer Tarball

Puppet Enterprise can be downloaded in tarballs specific to your OS and version, or as a universal tarball. Although the universal archive can be more convenient, it is roughly ten times the size of the version-specific archives.

Filename ends with...	Will install...
-all.tar	anywhere
-debian-<version and arch>.tar	on Debian
-el-<version and arch>.tar	on RHEL, CentOS, Scientific Linux, or Oracle Enterprise Linux
-sles-<version and arch>.tar	on SUSE Linux Enterprise Server
-solaris-<version and arch>.tar	on Solaris
-ubuntu-<version and arch>.tar	on Ubuntu LTS

Installing PE

To install PE, unarchive the tarball, navigate to the resulting directory, and run the `./puppet-enterprise-installer` script with root privileges in your shell; this will start the installer in interactive mode and guide you through the process of customizing your installation. After you've finished, it will save your answers in a file called `answers.lastrun`, install the selected software, and configure and enable all of the necessary services.

The installer can also be run non-interactively; [see below](#) for details.

Customizing Your Installation

The PE installer configures Puppet by asking a series of questions. Most questions have a default answer (displayed in brackets), which you can accept by pressing enter without typing a replacement. For questions with a yes or no answer, the default answer is capitalized (e.g. “[y/N]”).

Roles

First, the installer will ask which of PE's [three roles](#) (puppet master, Puppet Dashboard, and puppet agent) to install. The roles you apply will determine which other questions the installer will ask.

If you choose the puppet master or Puppet Dashboard roles, the puppet agent role will be installed automatically.

Puppet Master Options

CERTNAME

The puppet master's certname — its unique identifier — should always be its fully-qualified domain name. The installer will automatically detect this, but offers the chance to change it if necessary.

CERTIFIED HOSTNAMES

The puppet master's certified hostnames should be a colon-separated list of every hostname (short and fully-qualified) you use to reach it. This list will be included in the master's SSL certificate, and agent nodes will refuse to connect to a puppet master unless the hostname at which they reached it is included in the certificate.

This list should at least include the master's fully-qualified domain name and its short hostname, but may also include aliases like `puppet` and `puppet.<domain>.com`. If you ensure that the hostname `puppet` resolves to your puppet master and is included in the master's certificate, you can use the default puppet master hostname when installing subsequent agent nodes.

INTEGRATION WITH PUPPET DASHBOARD

The installer will ask if you wish to:

- Send reports to Dashboard
- Use Dashboard as a node classifier
- Forward facts to Dashboard's inventory service

(The latter question won't appear unless you chose not to install Dashboard on the same machine as the master; if they are running on the same machine, inventory viewing will be enabled by default.)

For most users, we recommend doing all three. If you selected any of them, the installer will then ask for Dashboard's hostname and port.

Puppet Dashboard Options

Puppet Dashboard is usually run on the same server as the puppet master, but can also be installed on a separate machine; if you choose to do so, you will need to do some [additional configuration](#) after installing.

PORT

By default, Dashboard will run on port 3000. If the server you install it on is not running any other webserver instances, you may wish to use the standard HTTP port of 80 instead; if so, remember to [point puppet master to the correct port during installation](#). If you need to change puppet's settings later, edit the `reporturl` setting in [puppet.conf](#) and the `BASE=` line in `/etc/puppetlabs/puppet-dashboard/external_node`.

INVENTORY CERTNAME/CERTIFIED HOSTNAMES

Dashboard will automatically use the puppet master's inventory service if you install it on the same server, but will have to maintain the inventory service¹ itself if you're installing it on a separate server.

Like puppet master, the inventory service needs a certname and a list of certified hostnames, which include the Dashboard server's short hostname and fully-qualified domain name. Unlike puppet master, its certified hostnames shouldn't include `puppet`.

Note that if you split Dashboard and master and wish to enable Dashboard's inventory and filebucket features, you will need to [exchange certificates between Dashboard and the puppet master after installation](#).

MYSQL SERVER

Dashboard needs a MySQL server, user, and database to operate. If MySQL isn't already installed on Dashboard's server, the installer can automatically configure everything Dashboard needs; just confirm that you want to install a new database server, and answer the questions about:

- MySQL's root user's password
- A name for Dashboard's database
- Dashboard's database user (default: "dashboard")
- A password for Dashboard's user

If you choose not to install a new database server, you'll need to have already created a database and MySQL user² for Dashboard prior to installation, and to have granted that user all privileges on

the database. If the database server is on a remote machine, you'll also need to provide the installer with:

- The database server's hostname
- The database server's port (default: 3306)

Puppet Agent Options

UNIQUE IDENTIFIER ("CERTNAME")

Like the puppet master, each puppet agent node needs a unique identifier for its SSL certificate.□ Unlike the master, this doesn't have to be a valid hostname, and can be any unique string that identifies the node.□

If domain names change frequently at your site or are otherwise unreliable, you may wish to investigate other schemes of node identification, such as UUIDs or hashes of some permanent□ firmware attribute.□

PUPPET MASTER HOSTNAME

By default, puppet agent will attempt to connect to a puppet master server at the hostname puppet. If your puppet master isn't reachable at that hostname, you should override this.

USING PLUGINSYNC

Puppet agent's pluginsync feature will download executable Ruby code (such as custom facts, types, and providers) from the puppet master. This should almost always be enabled, as Puppet Enterprise requires it to activate MCollective on agent nodes.

Development Libraries

As Puppet Enterprise maintains its own copy of Ruby and gem, you will need PE-specific copies of□ the Ruby development libraries if you intend to install any gems with native bindings.

Most users will not need to install the Ruby development libraries on any of their nodes. If you do not install the development libraries when installing Puppet Enterprise and find that you need them□ later, you can install them manually; see "Configuring and Troubleshooting" below.□

Convenience Links

Puppet Enterprise installs the Puppet software and manpages into subdirectories of /opt/puppet, which aren't included in your system's default \$PATH or \$MANPATH. For ease of use, the installer can create symbolic links to the Puppet executables in /usr/local/bin.

Additional Packages

Puppet Enterprise requires certain software from your operating system vendor's package repositories. If any of this software isn't yet installed, the installer will list which packages (if any) are needed and offer to automatically install them. If you decline, the installer will exit so you can□ install the necessary packages manually.

Post-install Configuration

Signing Agent Certificates

When installing the puppet agent role on a new node, the installer will automatically submit a certificate request to the puppet master. Before the agent will be able to receive any configurations, a user will have to sign the certificate from the puppet master. To view the list of pending certificate signing requests, run:

```
puppet cert list
```

To sign one of the pending requests, run:

```
puppet cert sign <name>
```

Configuring a Stand-Alone Dashboard Server

Most users should install Puppet Dashboard on the same server as the puppet master. However, if you need to split the two, you should install Dashboard after the puppet master, and do the following after installing both:

On the Dashboard server:

```
# cd /opt/puppet/share/puppet-dashboard
# export PATH=/opt/puppet/sbin:/opt/puppet/bin:$PATH
# rake RAILS_ENV=production cert:request
```

On the puppet master server:

```
# puppet cert sign dashboard
# puppet cert sign <inventory service's certname>
```

On the Dashboard server (in the same shell as before, with the modified PATH):

```
# rake RAILS_ENV=production cert:retrieve
# receive_signed_cert.rb <inventory service's certname> <puppet master's
hostname>
# service pe-httpd start
```

Advanced Installation

Installer Options

The Puppet Enterprise installer will accept the following command-line flags:

- -a ANSWER_FILE – Read answers from file and quit with error if an answer is missing.□
- -A ANSWER_FILE – Read answers from file and prompt for input if an answer is missing.□
- -D – Display debugging information.
- -h – Display a brief help message.
- -l LOG_FILE – Log commands and results to file.□
- -n – Run in ‘noop’ mode; show commands that would have been run during installation without running them.
- -s ANSWER_FILE – Save answers to file and quit without installing.□

Non-interactive Installation

To streamline your deployment, the Puppet Enterprise installer can run non-actively. When run with the -a or -A flags, the installer will load an answer file specified by the user.□

Answer files are simply shell scripts; they set a number of predefined variables and are sourced by□ the installer at runtime. Although static answer files are easily generated by running the installer□ with the -s ANSWER_FILE option (or by retrieving the answers.lastrun file from a previous□ installation), they can also be constructed by hand or with other tools, and can contain arbitrary shell code. This feature of the installer can be used to, for example, automate the generation of puppet agent certnames for sites that do not identify nodes by FQDN.

Further details and a full description of the available variables can be found in the [answer file□ reference](#).

1. Under the hood, this is a private instance of puppet master that doesn’t serve catalogs to agent nodes. ↪
2. The SQL commands to do so will likely resemble the following:

```
CREATE DATABASE dashboard CHARACTER SET utf8;
CREATE USER 'dashboard'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON dashboard.* TO 'dashboard'@'localhost';↪
```

NAVIGATION

- [Introduction](#)
- [Overview](#)
- [Installing](#)
- Upgrading
- [Using Puppet Enterprise](#)
- The Accounts Module
 - [The accounts::user Type](#)
 - [The accounts Class](#)
- Puppet Compliance

- [Puppet Compliance Basics and UI](#)
- [Using the Puppet Compliance Workflow](#)□
- [Compliance Workflow Tutorial](#)□
- [Known Issues](#)
- [Troubleshooting](#)
- [Answer File Reference](#)

Upgrading From Puppet Enterprise 1.x

Puppet Enterprise ships with an upgrade script that will do a large part of the work of upgrading your installation. However, you will have to finish the configuration of PE 1.2 manually.□

To upgrade to PE 1.2, you must:

- Download and unarchive the PE tarball.
- Run the `puppet-enterprise-upgrader` script.
- Create a new database for the inventory service and grant all permissions on it to the dashboard MySQL user.
- Manually edit the `puppet.conf`, `auth.conf`, `site.pp`, and `settings.yml` files on your puppet□ master.
- Generate and sign certificates for Puppet Dashboard to enable inventory and filebucket viewing.□
- Restart `pe-httpd`.

Choosing Your Installer Tarball

Puppet Enterprise can be downloaded in tarballs specific to your OS and version, or as a universal□ tarball. Although the universal archive can be more convenient, it is roughly ten times the size of the version-specific archives.□

Filename ends with...	Will install...
-all.tar	anywhere
-debian-<version and arch>.tar	on Debian
-el-<version and arch>.tar	on RHEL, CentOS, Scientific Linux, or Oracle Enterprise Linux□
-sles-<version and arch>.tar	on SUSE Linux Enterprise Server
-solaris-<version and arch>.tar	on Solaris
-ubuntu-<version and arch>.tar	on Ubuntu LTS

Running the Upgrader

Once you've retrieved a PE tarball, you should unarchive it, navigate to the resulting directory, and run `./puppet-enterprise-upgrader`. This script will examine your system to determine which Puppet Enterprise roles are currently installed, then list the packages these roles will require and ask if you want to continue with the upgrade. Note that the list of packages shown is not complete: any new dependencies from your operating system's repositories will be installed without confirmation, and the upgrade may fail if you are not connected to the source of these packages.□ Note also that upgrades to PE 1.x systems with the Puppet Dashboard role and no puppet agent

role may not upgrade cleanly, as this configuration is no longer supported under PE 1.2. We recommend that you run the `puppet-enterprise-installer` script in this situation, although this upgrade path has not been thoroughly tested.

After receiving confirmation, the upgrader will update existing packages, install new packages added in this version of PE, and run additional scripts or puppet manifests to make the system similar (though not necessarily identical) to a new installation of PE 1.2.

Upgrading Puppet Agent

The puppet agent role requires no additional upgrade steps. For machines running only the agent role, you don't need to do anything beyond run the upgrader script.

As with a new installation, you need to [assign the `mcollectivepe` class](#) to an agent node in order to enable MCollective.

Upgrading Puppet Master and Puppet Dashboard

If you are running the puppet master and Puppet Dashboard roles on the same machine, you will need to perform some additional steps to complete the upgrade. Puppet and Puppet Dashboard will continue to perform properly even if you skip these steps, but they are necessary to enable new features added in version 1.2.

Create a New Inventory Database

To support the inventory service, you must manually create a new database for puppet master to store node facts in. To do this, use the `mysql` client on the server providing the database. The server providing the database will almost always be the server running puppet master and Dashboard.

```
# mysql -uroot -p
Enter password:
mysql> CREATE DATABASE dashboard_inventory_service;
mysql> GRANT ALL PRIVILEGES ON dashboard_inventory_service.* TO
'dashboard'@'localhost';
```

If you chose a different MySQL user name for Puppet Dashboard when you originally installed PE, use that user name instead of “dashboard”. If the database is served by a remote machine, use the hostname of the master/Dashboard server instead of “localhost”.

Edit `/etc/puppetlabs/puppet/puppet.conf`

- To use the new accounts, `mcollectivepe`, `stdlib`, and `baselines` modules, you must add the `/opt/puppet/share/puppet/modules` directory to Puppet's `modulepath`:

```
[main]
modulepath =
```

```
/etc/puppetlabs/puppet/modules:/opt/puppet/share/puppet/modules
```

Note that if you were previously using an older version of the `stdlib` module, or any modules with the same name as the `accounts`, `mcollectivepe`, or `baselines` modules, you will have to delete them in order to use the modules included with PE 1.2.

- To support the inventory service, you must configure Puppet to save facts to a MySQL database.□

```
[master]
  facts_terminus = inventory_active_record
  dbadapter = mysql
  dbname = dashboard_inventory_service
  dbuser = dashboard
  dbpassword = <MySQL password for dashboard user>
  dbserver = localhost
```

If you chose a different MySQL user name for Puppet Dashboard when you originally installed PE, use that user name as the `dbuser` instead of “dashboard”. If the database is served by a remote machine, use that server’s hostname instead of “localhost”.

- To support filebucket viewing when using the Puppet Compliance workflow, you must set `archive_files` to `true` for puppet inspect:□

```
[main]
  archive_files = true
```

Edit `/etc/puppetlabs/puppet/auth.conf`

To support the inventory service, you must add the following stanzas to your [auth.conf](#) file:□

```
# Allow Dashboard to retrieve inventory facts:

path /facts
auth yes
method find, search
allow dashboard

# Allow puppet master to save facts to the inventory:

path /facts
auth yes
method save
allow <puppet master's certname>
```

These stanzas must be inserted before the final stanza:□

```
...method save
```

```
allow <puppet master's certname>

# final auth.conf stanza:

path /
auth any
```

If you paste these stanzas at the end of `auth.conf`, they will have no effect.□

Edit `/etc/puppetlabs/puppet/manifests/site.pp`

Even if you don't use `site.pp` to classify nodes, you must add the following resource and resource default in order to support Puppet Dashboard's filebucket viewing capabilities:□

```
# specify remote filebucket
filebucket { 'main':
  server => '<puppet master's hostname>',
  path => false,
}

File { backup => 'main' }
```

This will cause all agent nodes to back up their file contents to the puppet master, which will then□ serve the files to Dashboard on demand.□

Edit `/etc/puppetlabs/puppet-dashboard/settings.yml`

To turn on inventory and filebucket viewing, you must ensure that the following two options in□ `settings.yml` are set to `true`:

```
enable_inventory_service: true
use_file_bucket_diffs: true
```

You'll also need to ensure that the following three settings point to one of the puppet master's certified hostnames:□

```
ca_server: '<puppet master's hostname>'
inventory_server: '<puppet master's hostname>'
file_bucket_server: '<puppet master's hostname>'
```

Generate and Sign Certificates for Puppet Dashboard□

To support Dashboard's inventory and filebucket viewing capabilities, you must generate and sign□ certificates to allow it to request data from the puppet master.□

First, navigate to Dashboard's installation directory:

```
$ cd /opt/puppet/share/puppet-dashboard
```

Next, create a keypair and request a certificate:□

```
$ sudo /opt/puppet/bin/rake cert:create_key_pair  
$ sudo /opt/puppet/bin/rake cert:request
```

Next, sign the certificate request:□

```
$ sudo /opt/puppet/bin/puppet cert sign dashboard
```

Next, retrieve the signed certificate:□

```
$ sudo /opt/puppet/bin/rake cert:retrieve
```

And finally, make puppet-dashboard the owner of the certificates directory:□

```
$ sudo chown -R puppet-dashboard:puppet-dashboard certs
```

Restart pe-httpd

To reload all of the relevant puppet master and Dashboard config files, restart Apache:□

```
$ sudo /etc/init.d/pe-httpd restart
```

And you're done!

Upgrading Puppet Master Without Dashboard

If you are not using Puppet Dashboard at your site, upgrading puppet master is significantly□ simpler.

Edit /etc/puppetlabs/puppet/puppet.conf

- To use the new accounts, mcollectivepe, stdlib, and baselines modules, you must add the the /opt/puppet/share/puppet/modules directory to Puppet's modulepath:

```
[main]  
  modulepath =  
  /etc/puppetlabs/puppet/modules:/opt/puppet/share/puppet/modules
```

Upgrading Separated Puppet Master and Puppet Dashboard Servers

Coming Soon

Upgrading to PE 1.2 in cases where puppet master and Puppet Dashboard are being run on different servers is significantly more complicated. This upgrade procedure is not yet documented, and will be added to the manual at a later date.

NAVIGATION

- [Introduction](#)
- [Overview](#)
- [Installing](#)
- [Upgrading](#)
- Using Puppet Enterprise
- The Accounts Module
 - [The accounts::user Type](#)
 - [The accounts Class](#)
- Puppet Compliance
 - [Puppet Compliance Basics and UI](#)
 - [Using the Puppet Compliance Workflow](#)
 - [Compliance Workflow Tutorial](#)
- [Known Issues](#)
- [Troubleshooting](#)
- [Answer File Reference](#)

Using Puppet Enterprise

Puppet Enterprise is Puppet

If you're an experienced Puppet or MCollective user, Puppet Enterprise will work almost exactly as you expect.

Users

Puppet Enterprise creates and uses the following users for services and interaction:

- `mco` — The Mcollective client user, which exists on the puppet master server. To issue MCollective commands, you must be acting as the `mco` user using `sudo -H -u mco <command>`, `sudo -H -s -u mco`, or `su mco` — root cannot send MCollective messages.
- `pe-puppet` — The Puppet user, which runs the puppet agent service (`pe-puppet`).
- `pe-apache` — The PE web server user, which runs the copy of Apache (`pe-httpd`) that manages puppet master and Puppet Dashboard.
- `pe-activemq` — The ActiveMQ user, which exists on the puppet master server and runs the message bus used by MCollective.

Directories and Locations

All of Puppet Enterprise's components are installed in the `/opt/puppet` directory.

- The Puppet and MCollective library files are installed in `/opt/puppet/lib/ruby/site_ruby/1.8`.
- Executables are installed in `/opt/puppet/bin` and `/opt/puppet/sbin`.
- Puppet Dashboard is installed in `/opt/puppet/share/puppet-dashboard`.
- The bundled Puppet modules are installed in `/opt/puppet/share/puppet/modules`, which is included in Puppet's `modulepath`.

Similarly, PE's configuration files are kept in the `/etc/puppetlabs` directory.

- Puppet's `confdir` is `/etc/puppetlabs/puppet`, and its user modules directory is `/etc/puppetlabs/puppet/modules`.
- PE's license key should be placed at `/etc/puppetlabs/license.key`.
- MCollective's config files are in `/etc/puppetlabs/mcollective`.
- Puppet Dashboard's `settings.yml` and `database.yml` files are in `/etc/puppetlabs/puppet-dashboard`.

Services

Puppet Enterprise uses the following services, which can be managed with your OS's system tools:

- `pe-puppet` — The puppet agent daemon. Runs on every agent node.
- `pe-mcollective` — The MCollective server. Runs on agent nodes where [MCollective has been enabled](#).
- `pe-httpd` — Apache with Passenger, which manages instances of puppet master and Puppet Dashboard and responds to HTTP requests. Runs on servers with the puppet master and/or Puppet Dashboard roles.
- `pe-puppet-dashboard-workers` — Puppet Dashboard's background worker manager. Runs on servers with the Puppet Dashboard role.
- `pe-activemq` — The ActiveMQ message server, which the MCollective servers on agent nodes communicate with. Runs on servers with the puppet master role.

Reading Man Pages

For this version of PE, you should access the Puppet man pages using the `pe-man` tool, which accepts the complete name of a puppet command:

```
# pe-man puppet agent
```

Using Puppet and MCollective

Teaching the use of Puppet and MCollective is currently outside the scope of this document. We recommend the following resources:

- For new users learning Puppet from scratch, we recommend the [Learning Puppet](#) series, which explains the theory of Puppet and guides you through its basic use with practical exercises.
- For advanced users looking to jump in feet-first, we recommend reading the [language guide](#), the [guide to module layout](#), and the [core types cheat sheet](#) (PDF link).
- To learn how to use MCollective, we recommend starting with the [MCollective documentation](#).
- To learn how to use Puppet Dashboard, please see the [Puppet Dashboard 1.2 Manual](#).

Enabling MCollective

Puppet Enterprise uses a Puppet class called `mcollectivepe` to manage MCollective. To enable MCollective on an agent node, you must apply this class to the node.

If you are managing node classes in Puppet Dashboard, you should first add the `mcollectivepe` class to its list of known Puppet classes...

Class
Add class

Group
Add group

...then add that class to each node or group where you want MCollective enabled:

Edit node

Hostname
barn1.magpie.lan

Description
Enter a description for this host here...

Parameters
key value
Add parameter

Classes
Type in a search term
base x

Save changes or Cancel

Classes
mcol|
mcollectivepe

Classes
mcollectivepe x |

Maintaining and Tweaking Puppet Enterprise

Manually Installing the Ruby Development Libraries

If you find that you need the Ruby development libraries but skipped them during installation, you can install the packages manually rather than re-running the installer. The method for this will depend on your operating system's package management tools, but in each case, you must first navigate in your shell to the directory containing the packages for your operating system and

architecture, which will be inside the packages subdirectory of the Puppet Enterprise distribution tarball. (If you deleted the installation files, you will need to re-download them.)□

For systems using apt and dpkg (Ubuntu and Debian), run the following commands:

```
dpkg -i *ruby*dev*  
apt-get install --fix-broken
```

For systems using rpm and yum (Red Hat Enterprise Linux, CentOS, and Oracle Enterprise Linux), run the following commands:

```
yum localinstall --nogpgcheck *ruby*dev*
```

Enabling Reporting on Non-PE Agent Nodes

If you are integrating puppet agent nodes running a non-supported operating system into your Puppet Enterprise site, you'll need to enable reporting on these nodes in order for them to appear in Puppet Dashboard. In each such agent node's puppet.conf file, ensure that the [agent] or [puppetd] section contains report = true.

Maintaining Puppet Dashboard in Puppet Enterprise

Puppet Dashboard can require periodic maintenance; most notably, old reports should be cleaned occasionally. All of these tasks are done via rake tasks, and in Puppet Enterprise, they must be done with PE's copy of rake.

Before running any of the rake tasks described in the [maintenance chapter of the Dashboard manual](#), you should navigate to /opt/puppet/share/puppet-dashboard in your shell and alter your \$PATH to ensure that the proper copy of rake will be called:

```
# cd /opt/puppet/share/puppet-dashboard  
# export $PATH=/opt/puppet/bin:/opt/puppet/sbin:$PATH
```

NAVIGATION

- [Introduction](#)
- [Overview](#)
- [Installing](#)
- [Upgrading](#)
- [Using Puppet Enterprise](#)
- The Accounts Module
 - The accounts::user Type
 - [The accounts Class](#)

- Puppet Compliance
 - [Puppet Compliance Basics and UI](#)
 - [Using the Puppet Compliance Workflow](#)□
 - [Compliance Workflow Tutorial](#)□
- [Known Issues](#)
- [Troubleshooting](#)
- [Answer File Reference](#)

The `accounts::user` Type

This defined type manages a single user account. Many of its parameters echo those of the standard [user type](#). Unlike the user type, it will also create and manage the user's home directory, create and manage a primary group with the same name as the user, manage a set of SSH public keys for the user, and optionally lock the user's account.

The `accounts::user` type can be used on all of the platforms supported by Puppet Enterprise.

Usage Example

```
# /etc/puppetlabs/puppet/modules/site/manifests/users.pp
class site::users {
  # Declaring a dependency: we require several shared groups from the
  site::groups class (see below).
  Class[site::groups] -> Class[site::users]

  # Setting resource defaults for user accounts:
  Accounts::User {
    shell => '/bin/zsh',
  }

  # Declaring our accounts::user resources:
  accounts::user {'puppet':
    locked  => true,
    comment => 'Puppet Service Account',
    home    => '/var/lib/puppet',
    uid     => '52',
    gid     => '52',
  }
  accounts::user {'sysop':
    locked  => false,
    comment => 'System Operator',
    uid     => '700',
    gid     => '700',
    groups  => ['admin', 'sudonopw'],
    sshkeys => ['ssh-rsa
AAAAB3NzaC1yc2EAAAABIWAAAQEAWLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiINSCL8j+5zE/V
sysop+moduledevkey@puppetlabs.com'],
  }
  accounts::user {'villain':
    locked  => true,
    comment => 'Test Locked Account',
    uid     => '701',
    gid     => '701',
    groups  => ['admin', 'sudonopw'],
    sshkeys => ['ssh-rsa
AAAAB3NzaC1yc2EAAAABIWAAAQEAWLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiINSCL8j+5zE/V
villain+moduledevkey@puppetlabs.com'],
  }
  accounts::user {'jeff':
```

```

    comment => 'Jeff McCune',
    groups => ['admin', 'sudonopw'],
    uid => '1112',
    gid => '1112',
    sshkeys => [
      'ssh-rsa
AAAAB3NzaC1yc2EAAAABIWAAAQEAWLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiINSCL8j+5zE/V
jeff+moduledevkey@puppetlabs.com',
      'ssh-rsa
AAAAB3NzaC1yc2EAAAABIWAAAQEAWLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiINSCL8j+5zE/V
jeff+moduledevkey2@puppetlabs.com',
      'ssh-rsa
AAAAB3NzaC1yc2EAAAABIWAAAQEAWLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiINSCL8j+5zE/V
jeff+moduledevkey3@puppetlabs.com',
      'ssh-rsa
AAAAB3NzaC1yc2EAAAABIWAAAQEAWLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiINSCL8j+5zE/V
jeff+moduledevkey4@puppetlabs.com'
    ],
  }
  accounts::user {'dan':
    comment => 'Dan Bode',
    uid => '1109',
    gid => '1109',
    sshkeys => ['ssh-rsa
AAAAB3NzaC1yc2EAAAABIWAAAQEAWLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiINSCL8j+5zE/V
dan+moduledevkey@puppetlabs.com'],
  }
  accounts::user {'nigel':
    comment => 'Nigel Kersten',
    uid => '2001',
    gid => '2001',
    sshkeys => ['ssh-rsa
AAAAB3NzaC1yc2EAAAABIWAAAQEAWLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiINSCL8j+5zE/V
nigel+moduledevkey@puppetlabs.com'],
  }
}

# /etc/puppetlabs/puppet/modules/site/manifests/groups.pp
class site::groups {
  Group { ensure => present, }
  group {'developer':
    gid => '3003',
  }
  group {'sudonopw':
    gid => '3002',
  }
  group {'sudo':
    gid => '3001',
  }
  group {'admin':
    gid => '3000',
  }
}

```

Parameters

name

The user's name. While limitations differ by operating system, it is generally a good idea to restrict user names to 8 characters, beginning with a letter. Defaults to the resource's title.

ensure

Whether the user and its primary group should exist. Valid values are present and absent. Defaults to present.

shell

The user's login shell. The shell must exist and be executable. Defaults to `/bin/bash`.

comment

A description of the user. Generally a user's full name. Defaults to the user's name.

home

The home directory of the user. Defaults to `/home/<user's name>`

uid

The user's uid number. Must be specified numerically; defaults to being automatically determined (undef).

gid

The gid of the primary group with the same name as the user; the `accounts::user` type will create and manage this group. Must be specified numerically; defaults to being automatically determined (undef).

groups

An array of groups the user belongs to. The primary group should not be listed. Defaults to an empty array.

membership

Whether specified groups should be considered the complete list (inclusive) or the minimum list (minimum) of groups to which the user belongs. Valid values are `inclusive` and `minimum`; defaults to `minimum`.

password

The user's password, in whatever encrypted format the local machine requires. Be sure to enclose

any value that includes a dollar sign (\$) in single quotes ('). Defaults to '!!!', which prevents the user from logging in with a password.

locked

Whether the user should be prevented from logging in; set this to true for system users and users whose login privileges have been revoked. Valid values are true and false; defaults to false.

sshkeys

An array of SSH public keys associated with the user. Unlike with the [ssh_authorized_key](#) type, these should be complete public key strings that include the type and name of the key, exactly as the key would appear in its `id_rsa.pub` or `id_dsa.pub` file. Defaults to an empty array.□

NAVIGATION

- [Introduction](#)
- [Overview](#)
- [Installing](#)
- [Upgrading](#)
- [Using Puppet Enterprise](#)
- The Accounts Module
 - [The accounts::user Type](#)
 - The accounts Class
- Puppet Compliance
 - [Puppet Compliance Basics and UI](#)
 - [Using the Puppet Compliance Workflow](#)□
 - [Compliance Workflow Tutorial](#)□
- [Known Issues](#)
- [Troubleshooting](#)
- [Answer File Reference](#)

The accounts Class

The accounts class can do any or all of the following:

- Create and manage a set of `accounts::user` resources
- Create and manage a set of shared group resources
- Maintain a pair of rules in the `sudoers` file granting privileges to the `sudo` and `sudoNOPW` groups

This class is designed for cases where your account data is maintained separately from your Puppet manifests. This usually means it is being read on demand from a non-Puppet directory service or CMDB, managed manually by a user who does not write Puppet code, or generated by an out-of-band process. If your site's account data will be maintained manually by a sysadmin able to write Puppet code, it will make more sense to maintain it as a normal set of `accounts::user` and group resources, although you may still wish to use the accounts class to maintain `sudoers` rules.

To manage users and groups with the accounts class, you must prepare a data store and configure the class for the data store when you declare it.

Usage Example

To use YAML files as a data store:

```
class {'accounts':  
  data_store => yaml,  
}
```

To use a Puppet class as a data store and manage `sudoers` rules:

```
class {'accounts':  
  data_store      => namespace,  
  data_namespace => 'site::accounts::data',  
  manage_sudoers => true,  
}
```

To manage `sudoers` rules without managing any users or groups:

```
class {'accounts':  
  manage_users   => false,  
  manage_groups  => false,  
  manage_sudoers => true,  
}
```

Data Stores

Account data can come from one of two sources: a Puppet class that declares three variables, or a set of three [YAML](#) files stored in `/etc/puppetlabs/puppet/data`.

Using a Puppet Class as a Data Store

This option is most useful if you are able to generate or import your user data with a [custom function](#), which may be querying from an LDAP directory or some other arbitrary data source.

The Puppet class containing the data must have a name ending in `::data`. (We recommend `site::accounts::data`.) This class must declare the following variables:

- `$users_hash` should be a hash in which each key is the title of an `accounts::user` resource and each value is a hash containing that resource's attributes and values.
- `$groups_hash` should be a hash in which each key is the title of a group and each value is a hash containing that resource's attributes and values.
- `$users_hash_default` is an inert variable resulting from a [known issue](#), and should be an empty hash (`{}`). Note that part of this variable's name is the reverse of the corresponding file name.□

[See below](#) for examples of the data formats used in these variables.

When declaring the `accounts` class to use data in a Puppet class, use the following attributes:

```
data_store      => namespace,  
data_namespace => {name of class},
```

Using YAML Files as a Data Store

This option is most useful if your user data is being generated by an out-of-band process or is being maintained by a user who does not write Puppet manifests.

When storing data in YAML, the following valid [YAML](#) files must exist in `/etc/puppetlabs/puppet/data`:

- `accounts_users_hash.yaml`, which should contain an anonymous hash in which each key is the title of an `accounts::user` resource and each value is a hash containing that resource's attributes and values.
- `accounts_groups_hash.yaml`, which should contain an anonymous hash in which each key is the title of a group and each value is a hash containing that resource's attributes and values.
- `accounts_users_default_hash.yaml`, which is an inert file resulting from a [known issue](#) and should contain an anonymous hash with at least one dummy key/value pair. Note that part of this file's name is the reverse of the corresponding Puppet variable name.□

[See below](#) for examples of the data formats used in these variables.

When declaring the accounts class to use data in YAML files, use the following attribute:□

```
data_store => yaml,
```

Data Formats

This class uses three hashes of data to construct the `accounts::user` and `group` resources it manages.

THE USERS HASH

The users hash represents a set of `accounts::user` resources. Each key should be the title of an `accounts::user` resource, and each value should be another hash containing that resource's attributes and values.

PUPPET EXAMPLE

```
$users_hash = {
  sysop => {
    locked => false,
    comment => 'System Operator',
    uid => '700',
    gid => '700',
    groups => ['admin', 'sudonopw'],
    sshkeys => ['ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAwLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiINSCmL8j+5zE/V
sysop+moduledevkey@puppetlabs.com'],
  },
  villain => {
    locked => true,
    comment => 'Test Locked Account',
    uid => '701',
    gid => '701',
    groups => ['admin', 'sudonopw'],
    sshkeys => ['ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAwLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiINSCmL8j+5zE/V
villain+moduledevkey@puppetlabs.com'],
  },
}
```

YAML EXAMPLE

```
---
sysop:
  locked: false
  comment: System Operator
  uid: '700'
  gid: '700'
  groups:
  - admin
  - sudonopw
```

```

sshkeys:
- ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAWLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiiNSCmL8j+5zE/V
sysop+moduleddevkey@puppetlabs.com
villain:
  locked: true
  comment: Test Locked Account
  uid: '701'
  gid: '701'
  groups:
  - admin
  - sudonopw
  sshkeys:
  - ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAWLBhQefRiXHSbVNZYKu2o8VWJjZJ/B4LqICXuxhiiNSCmL8j+5zE/V
villain+moduleddevkey@puppetlabs.com

```

THE GROUPS HASH

The groups hash represents a set of shared group resources. Each key should be the title of a group resource, and each value should be another hash containing that resource's attributes and values.

PUPPET EXAMPLE

```

$groups_hash = {
  developer => {
    gid      => 3003,
    ensure   => present,
  },
  sudonopw  => {
    gid      => 3002,
    ensure   => present,
  },
  sudo      => {
    gid      => 3001,
    ensure   => present,
  },
  admin     => {
    gid      => 3000,
    ensure   => present,
  },
}

```

YAML EXAMPLE

```

---
developer:
  gid: "3003"
  ensure: "present"
sudonopw:
  gid: "3002"
  ensure: "present"
sudo:
  gid: "3001"
  ensure: "present"

```

```
admin:
  gid: "3000"
  ensure: "present"
```

THE USER DEFAULTS HASH

The user defaults hash has no effect on the behavior of the `accounts` module, but is required due to a [known issue](#). It should be either an empty hash or a hash with one or more dummy key/value pairs. (The function that loads YAML from disk is unable to recognize an empty hash, as an empty YAML hash is indistinguishable from an empty YAML string.)

PUPPET EXAMPLE

```
$users_hash_default = {}
```

YAML EXAMPLE

```
---
key: "value"
```

Parameters

manage_groups

Whether to manage a set of shared groups, which can be used by all `accounts::user` resources. If true, your data store must define these groups in the `$groups_hash` variable or the `accounts_groups_hash.yaml` file. Allowed values are `true` and `false`; defaults to `true`.

manage_users

Whether to manage a set of `accounts::user` resources. If true, your data store must define these users in the `$users_hash` variable or the `accounts_users_hash.yaml` file. Default attributes that apply to all users should be defined in the `$users_hash_default` variable or the `accounts_users_default_hash.yaml` file. Allowed values are `true` and `false`; defaults to `true`.

manage_sudoers

Whether to add sudo rules to the node's sudoers file. If true, the class will add `%sudo` and `%sudonopw` groups to the sudoers file and give them full sudo and passwordless sudo privileges respectively. You will need to make sure that the `sudo` and `sudonopw` groups exist in the groups hash, and that your chosen users have those groups in their groups arrays. Managing sudoers is not supported on Solaris.

Allowed values are `true` and `false`; defaults to `false`.

data_store

Which data store to use for accounts and groups.

When set to `namespace`, data will be read from the puppet class specified in the `data_namespace` parameter. When set to `yaml`, data will be read from specially-named YAML files in the `/etc/puppetlabs/puppet/data` directory. (If you have changed your `$confdir`, it will look in `$confdir/data`.) Example YAML files are provided in the `ext/data/` directory of this module.

Allowed values are `yaml` and `namespace`; defaults to `namespace`.

`data_namespace`

The Puppet namespace from which to read data. This must be the name of a Puppet class, and must end with `::data` (we recommend using `site::accounts::data`), which will automatically be declared by the `accounts` class. The class must be a non-parameterized class that declares variables named:

- `$users_hash`
- `$groups_hash`
- `$users_hash_default`

See the `accounts::data` class included in this module (in `manifests/data.pp`) for an example; see below for information on each hash's data structure.

Defaults to `accounts::data`.

`sudoers_path`

The path to the `sudoers` file on this system. Defaults to `/etc/sudoers`.

NAVIGATION

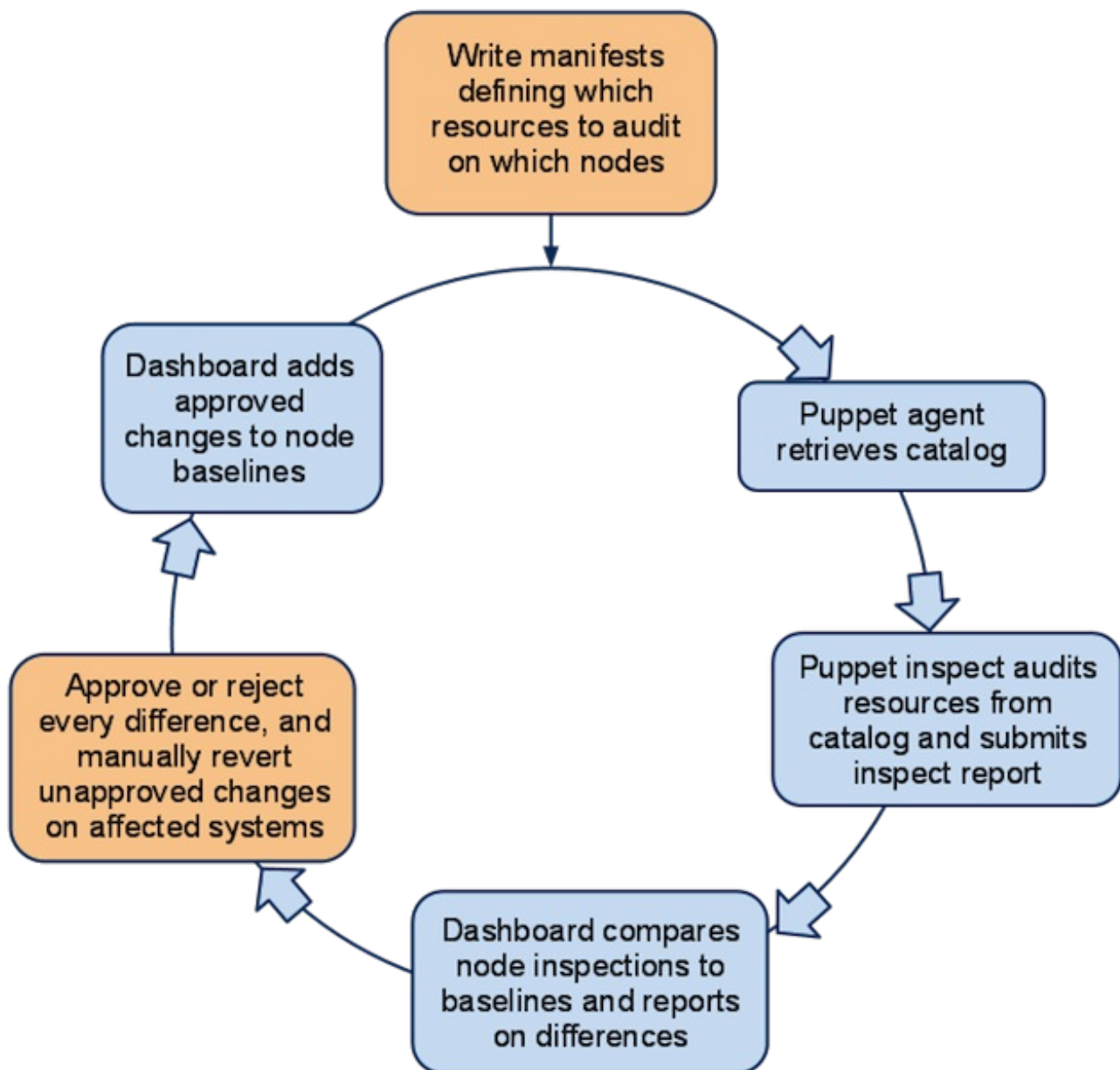
- [Introduction](#)
- [Overview](#)
- [Installing](#)
- [Upgrading](#)
- [Using Puppet Enterprise](#)
- The Accounts Module
 - [The accounts::user Type](#)
 - [The accounts Class](#)
- Puppet Compliance
 - Puppet Compliance Basics and UI
 - [Using the Puppet Compliance Workflow](#)
 - [Compliance Workflow Tutorial](#)

- [Known Issues](#)
- [Troubleshooting](#)
- [Answer File Reference](#)

Puppet Compliance Basics and UI

The Compliance Workflow Cycle

The puppet compliance workflow is designed to audit changes to systems managed by ad-hoc manual administration. It can also be used to audit changes to resources already managed by Puppet.



- A sysadmin writes manifests defining which resources to audit on which nodes.
- Puppet agent retrieves and caches a catalog compiled from those manifests.
- Puppet inspect reads that catalog to discover which resources to audit, then submits an inspect report, which the puppet master forwards to Puppet Dashboard.
- Dashboard then calculates a daily report of differences between the inspected system state and the approved baseline state, creating a baseline if one didn't already exist.

- A sysadmin uses the Dashboard interface to approve or reject every difference, then manually reverts any unapproved changes as necessary.
- Dashboard then modifies the baseline to include any approved changes, and awaits the next day's inspect reports.

Concepts

Auditing

When using this workflow, Puppet audits the state of resources, rather than enforcing a desired state; it does not make changes to any audited resources. Instead, changes are to be made manually (or by some out-of-band process) and reviewed for approval after the fact.

After changes have been reviewed in Puppet Dashboard, any approved changes will be considered the baseline state in future reports. Rejected changes will continue to be reported as non-baseline states until they are reverted manually on the affected machines.

Resources and Attributes

Any native Puppet resource type can be used in the baseline compliance workflow. As with similar compliance products, you can audit the content and metadata of files, but you can also audit user accounts, services, cron jobs, and anything for which a custom native type can be written.

Resources are audited by attribute — you can choose one or more attributes you wish to audit, or audit all attributes of that resource.

Compliance Manifests

The set of resources to audit is declared in standard Puppet manifests on the master and retrieved as a catalog by the agent. Instead of (or in addition to) declaring the desired state of a resource, these manifests should declare the `audit` metaparameter.

Inspect Reports

Each node being audited for compliance will routinely report the states of its audited resources. The documents it sends are called inspect reports, and differ from standard Puppet reports.

To enable puppet inspect, which sends these reports, you will need to apply the `baselines::agent` class to each node you are auditing.

Baselines

Conceptually, a baseline is a blessed inspect report for a single node: it lists the approved states for every audited resource on that node. Each node is associated with one and only one baseline, and nodes cannot share baselines. However, nodes with similar baselines can be grouped for convenience.

Baselines are maintained by Puppet Dashboard. They change over time as administrators approve

changes to audited resources. Nodes reporting for the first time are assumed to be in a compliant state, and their first inspect report will become the baseline against which future changes are compared.

Groups

Although nodes cannot share baselines, nodes in a Dashboard group can have similar changes approved or rejected en masse.

The Compliance Dashboard UI

Puppet Compliance's UI lives in Puppet Dashboard, and consists of:

- A new summary and custom report control on each group's page
- A set of dedicated compliance pages, accessible from the "Compliance" link in Dashboard's header menu

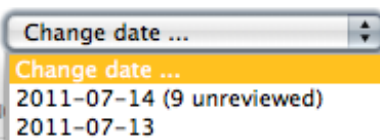


Each of these pages shows compliance information for a single day, and contains a date changer drop-down for changing the view. The most recently selected date will persist while you navigate these pages.

Friday, July 15, 2011

9 unreviewed changes on other days

 **Group**



New Controls on Group Pages

Group: hawks
Edit
Delete

Parameters
— No parameters —

Groups
— No groups —

Classes
— No classes —

Derived groups
— No child groups —

Daily run status
Number and status of runs during the last 30 days:

Nodes for this group
Friday, July 15, 2011
Unreviewed: 0 nodes, 0 differences
Accepted: 0
Rejected: 0
Change date ...
hawk1.magpie.lan
Generate Custom Report

Export nodes as CSV
Resources

Hostname	Source	Latest report	Total	Failed	Pending	Changed	Unchanged
Total			106	0	0	2	104
✓ hawk3.magpie.lan	hawk3.magpie.lan	2011-07-13 17:53 UTC	20	0	0	0	20
✓ hawk2.magpie.lan	hawk2.magpie.lan	2011-07-13 17:53 UTC	20	0	0	0	20
✓ hawk1.magpie.lan	hawk1.magpie.lan	2011-07-13 17:53 UTC	20	0	0	0	20

Per page: 20 100 all

Each group page will now have:

- A compliance summary in its node information section
- A control for generating [custom reports](#)

hawk2.magpie.lan
Generate Custom Report

Compliance Overview

puppet dashboard v1.1.9 • Home • Nodes • Groups • Classes • Reports • File Search • Inventory Search • Compliance

Background Tasks

✓ All systems go

Nodes

5 Unresponsive

0 Failed

0 Pending

0 Changed

0 Unchanged

0 Unreported

5 All

Add node

Group

hawks 3

Add group

Class

Add class

Compliance

Thursday, July 14, 2011

Change date ...

Group	NODES	UNREVIEWED	ACCEPTED	REJECTED
hawks	3	6	—	—
Total	3	6	—	—

Node (not in a group)	UNREVIEWED	ACCEPTED	REJECTED
eagle.magpie.lan	1	—	—
osprey.magpie.lan	2	—	—
Total	3	—	—

© Copyright 2011 Puppet Labs

The main compliance overview page shows a single day's comparison results, with aggregate summaries for grouped nodes and individual summaries for groupless nodes.

Compliance Node Page

puppet dashboard v1.1.9 • Home • Nodes • Groups • Classes • Reports • File Search • Inventory Search • Compliance

« Compliance • hawk1.maggie.lan node

Thursday, July 14, 2011
 Unreviewed: 2
 Accepted: 0
 Rejected: 0

Change date ...

Accept or reject all differences for this node
 Accept all differences for this node
 Reject all differences for this node

File[/etc/profile]

PROPERTY	BASLINE	INSPECTED	Differences
content	See file contents • Search for file	See file contents • Search for file	
ctime	2010-04-09 13:52 UTC	2011-07-14 19:36 UTC	
mtime	2009-09-21 23:27 UTC	2011-07-14 19:36 UTC	

User[nick]

PROPERTY	BASLINE	INSPECTED	Differences
comment	—	—	
ensure	present	absent	
expiry	absent	—	
gid	506	—	
groups	—	—	
home	/home/nick	—	
password	ll	—	
password_max_age	99999	—	
password_min_age	0	—	
shell	/bin/bash	—	
uid	506	—	

Per page: 25 100

© Copyright 2011 Puppet Labs

Individual node pages show one node’s off-baseline inspection results for a single day. You can accept or reject changes from this page.

Links to the node pages of groupless nodes are displayed on the main compliance overview page. To see the details of a node which is in at least one group, navigate to its group page and use the “Individual Differences” tab.

Compliance Group Page

puppet dashboard v1.1.9 • Home • Nodes • Groups • Classes • Reports • File Search • Inventory Search • Compliance

« Compliance • hawks group

Thursday, July 14, 2011
 Unreviewed: 3 nodes, 6 differences
 Accepted: 0
 Rejected: 0

Change date ...

Common Differences Individual Differences

This is a list of all resources with the same inspected state on more than one node.

File[/etc/profile]

⚠ There are 2 conflicting inspected states for this resource. You must accept 1, or reject them all.

Nodes	Property	Baseline	Inspected
1 nodes	content	See file contents • Search for file	See file contents • Search for file
	ctime	2010-04-09 13:52 UTC	2011-07-14 19:36 UTC
	mtime	2009-09-21 23:27 UTC	2011-07-14 19:36 UTC
2 nodes	ctime	2010-04-09 13:52 UTC	2011-07-13 19:31 UTC
	mtime	2009-09-21 23:27 UTC	2011-07-13 19:31 UTC

PROPERTY BASELINE INSPECTED

User[nick]

Nodes	Property	Baseline	Inspected
3 nodes	comment	—	—
	ensure	present	absent
	expiry	absent	—
	gid	506	—
	groups	—	—
	home	/home/nick	—
	password	!!	—
	password_max_age	99999	—
	password_min_age	0	—
	shell	/bin/bash	—
	uid	506	—

PROPERTY BASELINE INSPECTED

puppet dashboard v1.1.9 • Home • Nodes • Groups • Classes • Reports • File Search • Inventory Search • Compliance

« Compliance • hawks group

Thursday, July 14, 2011
 Unreviewed: 3 nodes, 6 differences
 Accepted: 0
 Rejected: 0

Change date ...

Common Differences Individual Differences

Node	Unreviewed	Accepted	Rejected
hawk3.magpie.lan	2	0	0
hawk2.magpie.lan	2	0	0
hawk1.magpie.lan	2	0	0
Total	6	0	0

Compliance group pages show the collected differences for a group of nodes. Two tabs are available:

- Use the “Common Differences” tab to approve and reject aggregate changes en masse
- Use the “Individual Differences” tab to access node pages and act individually

Groups are one of Dashboard’s core constructs, and nodes can be added to or removed from groups in Dashboard’s main node pages.

Although Dashboard allows groups to contain other groups, a compliance group page will only list nodes that are direct children of that group.

NAVIGATION

- [Introduction](#)
- [Overview](#)
- [Installing](#)
- [Upgrading](#)
- [Using Puppet Enterprise](#)
- The Accounts Module
 - [The accounts::user Type](#)
 - [The accounts Class](#)
- Puppet Compliance
 - [Puppet Compliance Basics and UI](#)
 - Using the Puppet Compliance Workflow□
 - [Compliance Workflow Tutorial](#)□
- [Known Issues](#)
- [Troubleshooting](#)
- [Answer File Reference](#)

Using the Puppet Compliance Workflow□

The Puppet Compliance workflow consists of the following routine tasks:□

- Preparing new agent nodes for compliance reporting
- Writing compliance manifests
- Reviewing changes
- Comparing whole groups against a single baseline

Preparing Agent Nodes For Compliance Reporting

Since agent nodes under compliance auditing need to submit a slightly different type of report, you will need to apply the `baselines::agent` Puppet class to any node whose resources you want to audit. This class installs a cron job that runs `puppet inspect` every day at 8pm.

You can apply this class in Puppet Dashboard. Refer to the [instructions for enabling MCollective](#) for instructions on how to apply a Puppet class with Dashboard.

Writing Compliance Manifests

Once compliance reporting has been enabled on your agent nodes, a sysadmin familiar with the Puppet language should write a collection of manifests defining the resources to be audited on the site's various computers.

In the simplest case, where you want to audit an identical set of resources on every computer, this can be done directly in the site manifest (`/etc/puppetlabs/puppet/manifests/site.pp`). More likely, you'll need to create a set of classes and modules that can be composed to describe the different kinds of computers at your site, then assign those classes using Puppet Dashboard or `site.pp`.

To mark a resource for auditing, declare its `audit` metaparameter. The value of `audit` can be one attribute, an array of attributes, or `all`. You can also have Puppet manage some attributes of an audited resource.

```
file {'hosts':  
  path => '/etc/hosts',  
  audit => 'content',  
}  
file {'/etc/sudoers':  
  audit => [ensure, content, owner, group, mode, type],  
}  
user {'httpd':  
  audit => 'all',  
}  
  
# Allow a user to change their password, but notify Puppet Compliance when
```

they do:

```
user {'admin':  
  ensure => present,  
  gid    => 'wheel',  
  groups => ['admin'],  
  shell  => '/bin/zsh',  
  audit  => 'password',  
}
```

Reviewing Changes

To audit changes, first scan the day's node and group summaries to see which nodes have ☐ unreviewed changes.

Compliance				
Thursday, July 14, 2011				
Change date ...				
Group	NODES	UNREVIEWED	ACCEPTED	REJECTED
hawks	3	6	--	--
Total	3	6	--	--
Node (not in a group)				
		UNREVIEWED	ACCEPTED	REJECTED
eagle.maggie.lan		1	--	--
osprey.maggie.lan		2	--	--
Total		3	--	--

Once you've seen the overview, you can navigate to any pages with unreviewed changes. If there are any unreviewed changes on previous days, there will be a warning next to the date changer drop-down, and the drop-down will note which days aren't fully reviewed.

Friday, July 15, 2011

9 unreviewed changes on other days

Change date ...

Change date ...

2011-07-14 (9 unreviewed)

2011-07-13

Changes can be accepted or rejected. Accepted changes will become part of the baseline for the next day's comparisons. Rejecting a change does nothing, and if an admin doesn't manually revert the change, it will appear as a difference again on the next day's comparisons.

When accepting multiple days' changes to the same resource, the most recently accepted change will win.

Reviewing Individual Nodes

Changes to individual nodes are straightforward to review: navigate to the node's page, view each change, and use the green plus and red minus buttons when you've decided whether the change was legitimate.

File[/etc/syslog.conf]

	content	See file contents • Search for file		See file contents • Search for file	Differences
	ctime	2011-07-07 22:58 UTC		2011-07-13 19:33 UTC	
	mtime	2011-07-07 22:58 UTC		2011-07-13 19:33 UTC	
PROPERTY		BASELINE		INSPECTED	

Service[crond]

	ensure	running	stopped	
	PROPERTY		BASELINE	
		INSPECTED		

Each change is displayed in a table of attributes, with the previously approved (“baseline”) state on the left and the inspected state on the right. You will also see links for viewing the original and modified contents of any changed files, as well as a “differences” link for showing the exact changes.

You can also accept or reject all of the day’s changes to this node at once with the controls below the node’s summary. Note that rejecting all changes is “safe,” since it makes no edits to the node’s baseline; if you reject all changes without manually reverting them, you’re effectively deferring a decision on them to the next day.

Reviewing Groups

If you’ve collected similar nodes into Dashboard groups, you can greatly speed up the review of similar changes with the “Common Differences” tab. You can also use the “Individual Differences” tab to navigate to the individual nodes.

SAME CHANGE ON MULTIPLE NODES

User[nick]				
	comment	—	—	
	ensure	present	absent	
	expiry	absent	—	
	gid	506	—	
	groups	—	—	
	home	/home/nick	—	
	password	!!	—	
	password_max_age	99999	—	
	password_min_age	0	—	
	shell	/bin/bash	—	
	uid	506	—	
PROPERTY		BASELINE		INSPECTED

If the same change was made on several nodes in a group, you can:

- Accept or reject the change for all affected nodes
- View the individual node pages to approve or reject the change selectively

DIFFERENT CHANGES TO THE SAME RESOURCE




File[/etc/profile]				
There are 2 conflicting inspected states for this resource. You must accept 1, or reject them all.				
	content	See file contents • Search for file		See file contents • Search for file
	ctime	2010-04-09 13:52 UTC	2011-07-14 19:36 UTC	
	mtime	2009-09-21 23:27 UTC	2011-07-14 19:36 UTC	
	ctime	2010-04-09 13:52 UTC	2011-07-13 19:31 UTC	
	mtime	2009-09-21 23:27 UTC	2011-07-13 19:31 UTC	
PROPERTY		BASELINE		INSPECTED

If different changes were made to the same resource on several nodes, the changes will be grouped for easy comparison. You can:

- Accept or reject each cluster of changes
- View the individual node pages to approve or reject the changes selectively

CONVERGENCE OF DIFFERING BASELINES

File[/etc/profile]

  3 nodes	content	 Multiple states	See file contents • Search for file
	ctime		2011-07-14 19:36 UTC
	mtime		2011-07-14 19:36 UTC
	PROPERTY	BASELINE	INSPECTED

If several nodes in a group had a different baseline value for one resource but were recently brought into an identical state, you can click through to examine the previous baselines, and can:

- Approve or reject the single new state for all affected nodes
- View the individual node pages to approve or reject the changes selectively

Comparing Groups Against a Single Baseline

Puppet Compliance can also generate custom reports which compare an entire group to a single member node's baseline. While the day-to-day compliance views are meant for tracking changes to a node or group of nodes over time, custom reports are meant for tracking how far a group of nodes have drifted away from each other.

- Custom reports only examine the most recent inspection report for each group member
- Custom reports do not allow you to approve or reject changes

Dashboard will maintain one cached custom report for each group; generating a new report for that group will erase the old one.



Custom reports are generated from Dashboard's core group pages — not from the compliance group pages. To generate a report, choose which baseline to compare against and press the generate button; the report will be queued and a progress indicator will display. (The indicator is static markup that does not automatically poll for updates; you will need to reload the page periodically for updated status on the report.)

Once generated, custom reports can be viewed from Dashboard's core group pages, the main compliance overview page, and the compliance group pages.

Nodes for this group

Saturday, July 23, 2011

Unreviewed: 0 nodes, 0 differences

Accepted: 0

Rejected: 2

Change date ...

Current custom report: [Comparison against hawk2.magpie.lan on 2011-07-23](#)

hawk1.magpie.lan

Generate Custom Report

« Compliance • hawks group

Saturday, July 23, 2011

Unreviewed: 0 nodes, 0 differences

Accepted: 0

Rejected: 2

Change date ...

Current custom report: [Comparison against hawk2.magpie.lan on 2011-07-23](#)

Compliance

Compliance differences for Saturday, July 23, 2011

Change date ...

 Group	NODES	UNREVIEWED	ACCEPTED	REJECTED
hawks	3	–	–	2
Total	3	–	–	2

On-demand Reports

hawks	AGAINST BASELINE hawk2.magpie.lan on 2011-07-23	CREATED AT Saturday, July 23, 2011
-------	--	---------------------------------------

A custom report is split into a “Common Differences” tab and an “Individual Differences” tab. This is very similar to the layout of the group compliance review pages, and should be read in the same fashion; the only difference is that all comparisons are to a single baseline instead of per-node baselines.

hawk1.magpie.lan

Common Differences

Individual Differences

This is a list of all resources with the same inspected state on more than one node.

Resource Type Title

Service[**crond**]

2 nodes	enable	false	true
	PROPERTY	BASELINE	INSPECTED

User[**nick**]

2 nodes	ensure	absent	present
	PROPERTY	BASELINE	INSPECTED

NAVIGATION

- [Introduction](#)
- [Overview](#)
- [Installing](#)
- [Upgrading](#)
- [Using Puppet Enterprise](#)
- The Accounts Module
 - [The accounts::user Type](#)
 - [The accounts Class](#)
- Puppet Compliance
 - [Puppet Compliance Basics and UI](#)
 - [Using the Puppet Compliance Workflow](#)□
 - Compliance Workflow Tutorial□
- [Known Issues](#)
- [Troubleshooting](#)
- [Answer File Reference](#)



Compliance Workflow Tutorial□

This brief walkthrough shows a compliance workflow auditing the state of the following resources:□

- File['/etc/profile']

- File['/etc/syslog.conf']
- Service['crond']
- Service['sshd']
- User['nick']

Morning, July 14, 2011

Compliance				
Thursday, July 14, 2011				
Change date ... ▾				
 Group	NODES	UNREVIEWED	ACCEPTED	REJECTED
hawks	3	6	—	—
Total	3	6	—	—
 Node (not in a group)		UNREVIEWED	ACCEPTED	REJECTED
eagle.magpie.lan		1	—	—
osprey.magpie.lan		2	—	—
Total		3	—	—

On Thursday morning, the admin notices unreviewed changes in a group of three nodes and a pair of ungrouped nodes. She checks the group first.□

Common Differences

Individual Differences

This is a list of all resources with the same inspected state on more than one node.

File[/etc/profile]

There are 2 conflicting inspected states for this resource. You must accept 1, or reject them all.

1 nodes	content	See file contents • Search for file	See file contents • Search for file
	ctime	2010-04-09 13:52 UTC	2011-07-14 19:36 UTC
	mtime	2009-09-21 23:27 UTC	2011-07-14 19:36 UTC
2 nodes	ctime	2010-04-09 13:52 UTC	2011-07-13 19:31 UTC
	mtime	2009-09-21 23:27 UTC	2011-07-13 19:31 UTC
PROPERTY	BASELINE		INSPECTED

User[nick]

3 nodes	comment	—	—
	ensure	present	absent
	expiry	absent	—
	gid	506	—
	groups		—
	home	/home/nick	—
	password	!!	—
	password_max_age	99999	—
	password_min_age	0	—
	shell	/bin/bash	—
	uid	506	—
	PROPERTY	BASELINE	INSPECTED



There, she notices that a user was completely deleted from all three nodes, and something odd happened with a file. She immediately rejects the deletion of the user...□

User[nick]



3 nodes	comment	—	—
	ensure	present	absent
	expiry	absent	—
	gid	506	—
	groups		—
	home	/home/nick	—
	password	!!	—
	password_max_age	99999	—
	password_min_age	0	—
	shell	/bin/bash	—
	uid	506	—
	PROPERTY	BASELINE	INSPECTED

...and manually SSHes to the affected nodes to re-instate the account.□

User[nick]



  3 nodes	comment
	ensure
	expiry
	gid
	groups
	home

hawk1.magpie.lan node

 	comment
	ensure
	expiry
	gid
	groups
	home
	password
	password_max_age
	password_min_age
	shell
	uid



PROPERTY

hawk2.magpie.lan node

 	comment
	ensure
	expiry
	gid
	groups
	home
	password
	password_max_age
	password_min_age
	shell
	uid

PROPERTY

hawk3.magpie.lan node

 	comment
	...

```
[root@hawk1.magpie.lan ~]# puppet resource group nick ensure=present gid=506
[root@hawk1.magpie.lan ~]# puppet resource user nick ensure=present uid=506
gid=506
...
```

Then she takes a look at the file. It looks like two nodes had the ctime and mtime of the `/etc/profile` file changed, but no edits were made. This was probably nothing, but it piques her interest and she'll ask around about it later; in the meantime, she approves the change, since there's no functional difference. The other node, however, had its content changed. She drills down into the node view and checks the contents before and after:

```
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"

fi

HOSTNAME=`/bin/hostname`
HISTSIZE=1000

if [ -z "$STDINTRO" ] && [ -f "$HOME/.intro"
```

```
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"

fi

HOSTNAME='hawk1.magpie.lan'
HISTSIZE=1000

if [ -z "$STDINTRO" ] && [ -f "$HOME/.intro"
```

That's not OK. It looks like someone was trying to resolve a DNS problem or something, but that's definitely not how she wants this machine configured. She rejects and manually reverts, and makes a note to find out what the problem they were trying to fix was.

Next, the admin moves on to the individual nodes.

« Compliance • osprey.magpie.lan node

Thursday, July 14, 2011
 Unreviewed: 0
 Accepted: 0
 Rejected: 2

Change date ...

Accept or reject all differences for this node

Accept all differences for this node
 Reject all differences for this node

File[/etc/syslog.conf]

PROPERTY	BASELINE	INSPECTED	Differences
content	See file contents • Search for file	See file contents • Search for file	
ctime	2011-07-07 22:58 UTC	2011-07-13 19:33 UTC	
mtime	2011-07-07 22:58 UTC	2011-07-13 19:33 UTC	

Service[crond]

PROPERTY	BASELINE	INSPECTED	Differences
ensure	running	stopped	

On the osprey server, something has stopped crond, which is definitely not good, and someone has made an edit to /etc/syslog.conf. She rejects the cron stoppage and restarts it, then checks the diff on the syslog config:

```
7c7
< *.info;mail.none;authpriv.none;cron.none    /var/log/messages
---
> *.info;mail.none;authpriv.none;cron.none    /etc/apache2/httpd.conf
```

That looks actively malicious. She rejects and reverts, then moves on to the eagle server to check on a hunch.

« Compliance • eagle.magpie.lan node

Thursday, July 14, 2011
 Unreviewed: 1
 Accepted: 0
 Rejected: 0

Change date ...

Accept or reject all differences for this node

Accept all differences for this node
 Reject all differences for this node

File[/etc/syslog.conf]

PROPERTY	BASELINE	INSPECTED	Differences
content	See file contents • Search for file	See file contents • Search for file	
ctime	2011-07-07 22:58 UTC	2011-07-13 19:33 UTC	
mtime	2011-07-07 22:58 UTC	2011-07-13 19:33 UTC	

Per page: 25 100


```
7c7
< *.info;mail.none;authpriv.none;cron.none          /var/log/messages
---
> *.info;mail.none;authpriv.none;cron.none          /etc/apache2/httpd.conf
```

Yup: same dangerous change. She rejects and reverts, then spends the rest of her day figuring out how the servers got vandalized.

Morning, July 15, 2011

The next day, the admin's previous fixes to the syslog.conf and profile files on the various servers have caused changes to the ctime and mtime of those files. She approves her own changes, then decides that she should probably edit her manifests so that all but a select handful of file resources use audit => [ensure, content, owner, group, mode, type] instead of audit => all, in order to suppress otherwise meaningless audit events.

It's an otherwise uneventful day.

NAVIGATION

- [Introduction](#)
- [Overview](#)
- [Installing](#)
- [Upgrading](#)
- [Using Puppet Enterprise](#)
- The Accounts Module
 - [The accounts::user Type](#)
 - [The accounts Class](#)
- Puppet Compliance
 - [Puppet Compliance Basics and UI](#)
 - [Using the Puppet Compliance Workflow](#)
 - [Compliance Workflow Tutorial](#)
- Known Issues
- [Troubleshooting](#)
- [Answer File Reference](#)

Known Issues in Puppet Enterprise 1.2

In PE ≤ 1.2.1

Upgrading From PE 1.1 Breaks Node Classification on Debian and Ubuntu Systems

([Issue #9444](#))

When upgrading a puppet master with Dashboard from PE 1.1 to 1.2 on Debian and Ubuntu systems, a permissions change to `/etc/puppetlabs/puppet-dashboard` and `/etc/puppetlabs/puppet-dashboard/external_node` breaks Dashboard's node classification abilities. The symptom of this issue is that classes that applied properly to agents on PE 1.1 will no longer be applied.

The workaround for this issue is to ensure that the permissions of `/etc/puppetlabs/puppet-dashboard` are 755 and that the permissions of `/etc/puppetlabs/puppet-dashboard/external_node` are 755. Running the following command should be sufficient:

```
# sudo chmod 755 /etc/puppetlabs/puppet-dashboard /etc/puppetlabs/puppet-dashboard/external_node
```

This issue only affects Debian/Ubuntu systems being upgraded from PE 1.1 to 1.2.x with a combined master/Dashboard installation.

Accounts Class Requires an Inert Variable/File

The accounts class — a data-separation wrapper that uses external data to declare a set of `accounts::user` resources — will not function unless a `$users_hash_default` variable (if using the namespace data store) or `accounts_users_default_hash.yaml` file (if using the `yaml` data store) is present, even though this variable/file is never used when creating resources.

The workaround is to ensure that this variable/file is present in the namespace or data directory.

In PE 1.2.0

Puppet Inspect Does Not Archive Files

In `puppet.conf`, the `archive_files = true` setting was incorrectly placed in an inert `[inspect]` block. This caused puppet inspect to not upload files when submitting compliance reports.

To repair this issue, edit your `puppet.conf` file to include `archive_files = true` under the `[main]` block. This will not happen automatically when upgrading from PE 1.2.0 to 1.2.1, but is fixed in new installations of PE 1.2.1.

NAVIGATION

- [Introduction](#)
- [Overview](#)
- [Installing](#)
- [Upgrading](#)
- [Using Puppet Enterprise](#)
- The Accounts Module
 - [The accounts::user Type](#)
 - [The accounts Class](#)
- Puppet Compliance
 - [Puppet Compliance Basics and UI](#)
 - [Using the Puppet Compliance Workflow](#)□
 - [Compliance Workflow Tutorial](#)□
- [Known Issues](#)
- Troubleshooting
- [Answer File Reference](#)

Troubleshooting

This document explains several common problems that can prevent a Puppet Enterprise site from working as expected. Additional troubleshooting information can be found at the [main Puppet documentation site](#), on the [Puppet Users mailing list](#), and in #puppet on freenode.net.

Furthermore, please feel free to contact Puppet Labs' enterprise technical support:

- To file a support ticket, go to support.puppetlabs.com.
- To reach our support staff via email, contact support@puppetlabs.com.
- To speak directly to our support staff, call 1-877-575-9775.□

When reporting issues with the installer itself, please run the installer using the `-D` debugging flag□ and provide a transcript of the installer's output along with your description of the problem.

Firewall Issues

If not configured properly, your system's firewall can interfere with access to Puppet Dashboard and with communications between the puppet master server and puppet agent nodes. In particular, CentOS and Red Hat Enterprise Linux ship with extremely restrictive iptables rules, which may need to be modified.□

When attempting to debug connectivity issues (especially if puppet agent nodes are generating log messages like `err: Could not request certificate: No route to host - connect(2)`), first□ examine your firewall rules and ensure that your master server is allowing tcp connections on port□ 8140 and your Dashboard server is allowing tcp connections on port 3000.

If you use the REST API and have configured puppet agent to listen for incoming connections, your□ agent nodes will need to allow tcp connections on port 8139.

Certificate Issues□

The learning curve of SSL certificate management can lead to a variety of problems. Watch out for□ the following common scenarios:

Agent Node Was Installed Before the Puppet Master

If you install the puppet agent role on a node before getting the puppet master up and running, the installer won't be able to request a certificate during installation, and you won't immediately see a□ pending certificate request when you run `puppet cert list`. To request a certificate manually, you□ can log into the agent node and run:

```
# sudo puppet agent --test
```

...after which you should be able to sign the certificate on the puppet master. Alternately, you can simply wait 30 minutes, as the puppet agent service will request a certificate on its next run.

Agent Nodes Refuse to Trust the Master's Certificate

Symptom: Puppet agent is unable to retrieve a configuration, and logs a series of errors ending in "hostname was not match with the server certificate."

Agent nodes determine the validity of the master's certificate based on hostname; if they're contacting it using a hostname that wasn't included when the certificate was signed, they'll reject the certificate.

Solution: Either:

Modify your agent nodes' settings to point to one of the master's certified hostnames. (This may also require adjusting your site's DNS.) To see the puppet master's certified hostnames, run:

```
# sudo puppet master --configprint certname,certdnsnames
```

...on the puppet master server.

Or:

Re-generate the puppet master's certificate:

- Stop the pe-httpd service:

```
# sudo /etc/init.d/pe-httpd stop
```

- Delete the puppet master's certificate, private key, and public key:

```
# sudo find /etc/puppetlabs/puppet/ssl -name $(puppet master --configprint certname) -delete
```

- Edit the certname and certdnsnames settings in the puppet master's /etc/puppetlabs/puppet/puppet.conf file to match the puppet master's actual hostnames.
- Start a non-daemonized WEBrick puppet master instance, and wait for it to generate and sign a new certificate:

```
# sudo puppet master --no-daemonize --verbose
```

You should stop the temporary puppet master with ctrl-C after you see the "notice: Starting Puppet master version 2.6.9" message.

- Start the pe-httpd service:

```
# sudo /etc/init.d/pe-httpd start
```

Puppet master is rejecting an agent node with a valid certificate

Symptom: Puppet agent logs a series of errors that end with “SSL_connect returned=1 errno=0 state=SSLv3 read server certificate B: certificate verify failed” or “SSL_connect returned=1 errno=0 state=SSLv3 read finished A: sslv3 alert bad certificate.”

The most common cause of this error is time drift in a virtual machine — if either the puppet master’s or the agent’s system time have been reset to before the master or agent certificates were signed, or if the certificates were accidentally signed while the puppet master’s clock was set to the future, the certificates will be rejected as invalid.

Solution:

To avoid this error, maintain accurate timekeeping on the agent nodes and puppet masters; keep in mind that NTP can behave unreliably in virtual machines.

To repair this error, you will need to re-generate the affected certificates. Use `puppet cert print <certname>` on the puppet master to examine the valid dates of a certificate, and when you have confirmed that an agent certificate needs to be regenerated, run:

- `puppet cert clean <certname>` on the puppet master
- `rm -rf $(puppet agent --configprint ssldir); puppet agent --test` on the affected agent node.
- `puppet cert sign <certname>` on the puppet master

If you need to re-generate the puppet master’s certificate, see [the instructions in the previous question](#)

An agent node’s OS has been re-installed, and the master will no longer communicate with it

The puppet master stores signed agent certificates based on nodes’ certnames (unique IDs), which are usually fully-qualified domain names. If a node loses its certificates but retains its unique ID (certname) — as would happen if the OS and Puppet were re-installed from scratch — it will re-generate its certificate and send a signing request to the master. However, since the master already has a signed certificate cached for that node, it will ignore the signing request and expect the node to connect using the old (and now lost) certificate.

A similar situation can arise if an agent node is rebuilt while a previous signing request is still pending.

Solution: On the master server, run `puppet cert --clean {agent certname}`. The master should

now accept the node's signing request and wait for permission to sign it.

An agent's hostname has changed, and it can no longer contact its master

Normally, a hostname change will not affect a node's certname. However, if the certname is changed in `puppet.conf` or at the command line, the node's previously signed certificate won't be used when contacting the puppet master.

Solution: The agent will have submitted a new certificate signing request during the first attempted agent run with the new certname. Simply run `sudo puppet cert sign <new certname>` on the puppet master to allow the agent to connect under its new name. You can also run `puppet cert clean` on the previous certificate if you expect to re-use the old certname for a new agent node.

Dashboard Was Installed Before the Puppet Master

If you're splitting the Dashboard and puppet master servers and you installed Dashboard first, you'll need to modify the post-installation certificate exchange. On the Dashboard server, run the following commands:

```
# puppet agent --test
# cd /opt/puppet/share/puppet-dashboard
# export PATH=/opt/puppet/sbin:/opt/puppet/bin:$PATH
# rake RAILS_ENV=production cert:generate
```

After that, you should be able to follow [the normal post-install instructions for a stand-alone Dashboard server](#).

Puppet Master Server Was Replaced

All nodes in a given domain must be using certificates signed by the same CA certificate. Since puppet master generates a new CA cert during installation, a new master node will, in its default state, be rejected by any agent nodes which were aware of the previous master.

You have two main options: either delete the `/etc/puppetlabs/puppet/ssl` directory from each agent node (thereby forcing each node to re-generate certificates and issue new signing requests), or recover the CA certificate and private key from the previous master server and re-generate and re-sign the new master's certificate. The short version of how to do the latter is as follows:

- Stop puppet master using your system's service utilities (e.g., on CentOS: `service pe-httpd stop`)
- Run `puppet cert --clean {master's certname}`
- Replace the `/etc/puppetlabs/puppet/ssl/ca` directory with the same directory from the old master
- Run `puppet master --no-daemonize`, wait approximately 30 seconds, then end the process with `ctrl-C`

- Start puppet master using your system's service utilities

(As you can see, replacing a master server requires a certain amount of planning, and if you expect it to be a semi-regular occurrence, you may wish to investigate certificate chaining and the use of external certificate authorities.)

Miscellaneous Issues and Additional Tasks

Dashboard Has a Very Large Number of Pending Background Tasks

Background tasks are processed by worker processes, which can sometimes die and leave invalid PID files. To restart the worker tasks, run:

```
# sudo /etc/init.d/pe-puppet-dashboard-workers restart
```

This should immediately start reducing the number of pending tasks.

Change The Port Used By Puppet Dashboard

If you chose to use port 3000 when you installed Puppet Dashboard but later decide you want to use port 80 (or any other port), you will need to stop the pe-httpd service and make the following changes:

- In `/etc/puppetlabs/httpd/conf.d/puppetdashboard.conf`, change the `Listen 3000` and `<VirtualHost *:3000>` directives to use port 80.
- In `/etc/puppetlabs/puppet/puppet.conf`, change the `reporturl` setting to use your preferred port instead of port 3000.
- In `/etc/puppetlabs/puppet-dashboard/external_node`, change the `BASE="http://localhost:3000"` line to use your preferred port instead of port 3000.
- Allow access to the new port in your system's firewall rules.

After making these changes, start the pe-httpd service again.

Import Existing Reports Into Puppet Dashboard

If you are upgrading to PE from a self-maintained Puppet environment, you can add value to your older reports by importing the reports into Puppet Dashboard. To run the report importer, first locate the reports directory on your previous puppet master (`puppet master --configprint reportdir`) and copy it to the server running Puppet Dashboard. Then, on the Dashboard server, run the following commands with root privileges:

included stored reports on the puppet master (that is, if agent nodes were configured with `report = true` and the master was configured with `reports = store`), can add value to this existing knowledge.


```
cd /opt/puppet/share/puppet-dashboard
PATH=/opt/puppet/bin:$PATH REPORT_DIR={old reports directory} rake
reports:import
```

If you have a significant number of existing reports, this task can take some time, so plan accordingly.

NAVIGATION

- [Introduction](#)
- [Overview](#)
- [Installing](#)
- [Upgrading](#)
- [Using Puppet Enterprise](#)
- The Accounts Module
 - [The accounts::user Type](#)
 - [The accounts Class](#)
- Puppet Compliance
 - [Puppet Compliance Basics and UI](#)
 - [Using the Puppet Compliance Workflow](#)
 - [Compliance Workflow Tutorial](#)
- [Known Issues](#)
- [Troubleshooting](#)
- Answer File Reference

Answer File Reference

When run with the `-a` or `-A` flag and a filename argument, the Puppet Enterprise installer will read its installation instructions from an answer file. `-a` will use the provided answer file to perform a non-interactive installation (which will fail if any required variables are not set), and `-A` will perform a partially-interactive installation, prompting the user for any missing answers.

Answer files consist of normal shell script variable assignments (e.g. `q_puppetdashboard_database_port=3306`), and can include arbitrary control logic or assign variables based on the output of backtick-wrapped shell commands. One of the most common uses of answer files is to automatically set a puppet agent node's unique identifier (`certname`) to something other than its fully-qualified domain name; this is most efficiently done by running the installer with `-s ANSWER_FILE`, then editing the resultant file to include, for example:

```
q_puppetagent_certname=`uuidgen`
```

The answer file could then be used for any number of puppet agent installations without further modification or interaction.

Boolean answers should use `Y` or `N` (case-insensitive) rather than `true`, `false`, `1`, or `0`.

A variable can be omitted if another answer ensures that it won't be used (i.e. `q_puppetmaster_certname` can be left blank if `q_puppetmaster_install = n`).

Allowed Variables In Answer Files

puppet master Answers

- `q_puppetmaster_install` – `Y` or `N` (Default: `Y`) – Sets whether the puppet master service will be installed on this computer.
- `q_puppetmaster_certname` – String (Default: {computer's fully-qualified domain name}) – Sets the puppet master's SSL certificate common name (CN). This should usually be set to either "puppet" or the server's fully-qualified domain name.
- `q_puppetmaster_certdnsnames` – String (Default: `puppet:puppet.{computer's domain}:{computer's hostname}:{computer's fully-qualified domain name}`) – Sets the puppet master's certified hostnames. Must be a colon-separated list of hostnames and fully-qualified domain names.
- `q_puppetmaster_use_dashboard_reports` – `Y` or `N` (Default: `Y`) – Sets whether the puppet master will send reports received from puppet agent nodes to Puppet Dashboard.
- `q_puppetmaster_use_dashboard_classifier` – `Y` or `N` (Default: `Y`) – Sets whether the puppet master will request node definitions from Puppet Dashboard.
 - `q_puppetmaster_dashboard_hostname` – String (Default: `localhost`) – Sets the hostname where

Puppet Dashboard is located. Used when `q_puppetmaster_use_dashboard_reports = y` or `q_puppetmaster_use_dashboard_classifier = y`.

- `q_puppetmaster_dashboard_port` – String (Default: 3000) – Sets the port to use when connecting to Puppet Dashboard. Used when `q_puppetmaster_use_dashboard_reports = y` or `q_puppetmaster_use_dashboard_classifier = y`.

Puppet Dashboard Answers

- `q_puppetdashboard_install` – Y or N (Default: Y) – Sets whether Puppet Dashboard will be installed on this computer.
- `q_puppetdashboard_httpd_port` – String (Default: 3000) – Sets which port Puppet Dashboard will use.
- `q_puppetdashboard_database_install` – Y or N (Default: Y) – Sets whether to install and configure a new MySQL database from the OS's repositories.
 - `q_puppetdashboard_database_root_password` – Y or N (Default: Y, unless MySQL is already installed on this system) – Sets the password that will be assigned to MySQL's root user. Used when `q_puppetdashboard_database_install = y`.
 - `q_puppetdashboard_database_remote` – Y or N (Default: Y) – Sets whether the MySQL database server is running on a remote host; if set to N, Dashboard will be configured to connect to a database on localhost. Used when `q_puppetdashboard_database_install = n`.
 - `q_puppetdashboard_database_host` – String (Default: localhost) – Sets the hostname of the remote MySQL database server. Used when `q_puppetdashboard_database_remote = y`.
 - `q_puppetdashboard_database_port` – String (Default: 3306) – Sets the port used by the remote MySQL database server. Used when `q_puppetdashboard_database_remote = y`.
- `q_puppetdashboard_database_name` – String (Default: dashboard) – Sets the name of the database Dashboard will use.
- `q_puppetdashboard_database_user` – String (Default: dashboard) – Sets the MySQL user Dashboard will connect to the database server as. This user must already exist on the MySQL server, and must have the necessary privileges (see "Puppet Dashboard" above) for the database Puppet Dashboard will be using.
- `q_puppetdashboard_database_password` – String or blank (Default: {blank password}) – Sets the password with which Puppet Dashboard's MySQL user will connect to the database.

puppet agent Answers

- `q_puppetagent_install` – Y or N (Default: Y) – Sets whether the puppet agent service will be installed on this computer.
- `q_puppetagent_certname` – String (Default: {computer's fully-qualified domain name}) – Sets the unique identifier ("certname") for this puppet agent node's SSL certificate.
- `q_puppetagent_server` – String (Default: puppet; however, if `q_puppetmaster_install = y`, this will default to the value of `q_puppetmaster_certname`.) – Sets the hostname of the puppet

master.

- `q_puppetagent_pluginsync` – Y or N (Default: Y) – Sets whether to sync executable Ruby code (e.g. custom facts, types, and providers) from the puppet master.

Other Answers

- `q_rubydevelopment_install` – Y or N (Default: N) – Sets whether to install the Ruby development libraries for Puppet Enterprise’s copy of Ruby.
- `q_vendor_packages_install` – Y or N (Default: Y) – Sets whether the installer has permission to install additional packages from the operating system’s repositories. If this is set to “n,” the installation will only go forward if the installer detects no missing dependencies.
- `q_puppet_symlinks_install` – Y or N (Default: Y) – Sets whether `/usr/local/bin` should contain symbolic links to the Puppet executables (located in `/opt/puppet/bin`).
- `q_install` – Y or N (Default: Y) – Sets whether the installation should occur. Setting this to “n” is not considered useful for non-interactive installation.

© 2010 [Puppet Labs](http://puppetlabs.com) info@puppetlabs.com 411 NW Park Street / Portland, OR 97209 1-877-575-9775