**Upgrade to 2017.3.0***Prepared for the Home Depot*

This document is intended to provide a high-level overview of notable changes to be aware of when upgrading from 2015.3.3. For the most up-to-date details please leverage the [2017.3.x docs](https://puppet.com/docs/pe/2017.3/overview/pe_user_guide.html).

The [2015.3 Docs](https://docs.puppet.com/pe/2015.3/overview_about_pe.html) and [2017.3 Docs](https://puppet.com/docs/pe/2017.3/overview/overview.html) were used to create this document.

# High-Level Overview

Below are some notable configuration considerations for you to be aware of prior to your upgrade to 2017.3.x.

## Ports

Orchestrator - Two new ports must be opened in your firewall relating to the Orchestrator - 8142, 8143.

Code Manager - For Code Manager to work properly port 8170 must be open.

## Agent Upgrades

Recommended Agent version is 5.3.2. For this upgrade it’s not recommended to have an Agent below version 1.3.2 to ensure that Orchestrator jobs can run.

* Orchestrator Jobs - Requires Agent version 1.3.2+, which shipped with 2015.3.0.
* Ad-Hoc Tasks - Requires Agent version 5.3.2+ which shipped with 2017.3.0.
* Current Infrastructure State - Requires Agent version 1.71+ which shipped with 2016.4.0.
* No-Op Reporting - Requires Agent version 1.71+ which shipped with 2016.4.0.
* Package Inspector - Requires Agent version 1.6.0+ which shipped with 2017.2.0.

## Hiera 5

Was introduced in 2017.1.1, requires Puppet 4.9.0 and a minimum Agent version of 1.9.3+. Hiera 5 is backwards compatible.

## Hiera 5 Overrides in the Console

This feature is new to 2017.3 and will be enabled by default in **new installs**. When upgrading you will **need to manually enable and configure**. Additional information on how to do this [can be found in the docs](https://puppet.com/docs/pe/2017.3/managing_nodes/grouping_and_classifying_nodes.html#enable-data-editing-in-the-console).

## PE Client Tools

New PE Client Tools have been introduced since 2015.3.3. These are detailed in the PE Client Tools section below and include information on the following new client tools:

* Orchestrator
* Puppet Access
* Code Manager CLI
* PuppetDB CLI

## Orchestrator

Four major areas to note when configuring the Orchestrator are below.

Item 1 - Orchestrator communication relies on ports 8142 and 8143 being open.   
  
Item 2 - Configuring the Orchestrator has a prerequisite of having PE Client Tools installed.

Item 3 - After PE Client Tools are installed the following configurations must be completed:

* Setting PE RBAC permissions and token authentication for Puppet orchestrator
* Enable cached catalogs for use with the orchestrator (optional)
* Review the orchestrator configuration files and adjust them as needed

Item 4 - Compile Master configuration. If you want to use tasks, you must either set master\_uris or you server\_list on agents to point to to your compile masters.

## Package Inspector

Package Inspector was introduced in 2017.2.0 and must be enabled. Upon upgrading [please refer to the docs](https://puppet.com/docs/pe/2017.3/inspecting_infrastructure/viewing_packages_in_use.html#enable-package-data-collection) for more information on how to enable the Package inspector.

## Default global hiera.yaml no longer includes common.yaml or nodes/clientcert

In order to keep environment-specific data within the environment folder, the default global hiera.yaml now supports console data only and doesn't include the previous YAML file hierarchy. If you rely on the default global YAML file hierarchy of nodes/%{clientcert}.yaml and common.yaml, create a Hiera 5 compatible hiera.yaml file in your control-repo environment folder instead. For example: <https://github.com/puppetlabs/control-repo/blob/production/hiera.yaml>

## PostgreSQL 9.6 upgrade

This version of PE upgrades PostgreSQL to version 9.6. If you use an external PostgreSQL instance, you must upgrade it before you upgrade PE.

Note: If you're upgrading a split installation of a PE version earlier than 2016.4.3 with an external PostgreSQL instance, you must upgrade with the --force flag, for example:

/opt/puppetlabs/puppet/bin/puppet infrastructure configure --detailed-exitcodes --modulepath=/opt/puppetlabs/server/data/enterprise/modules --no-noop --upgrade-from=<PREVIOUS PE VERSION> --force

## PuppetDB "time-to-live" default

PuppetDB now uses 14 days for a default “time-to-live” value (node-purge-ttl) before it deletes nodes that have been deactivated or expired. This default can be changed as needed, as documented in the Configuring PE section.

## New location for MCollective logs

On non- Windows systems, MCollective server logs now appear in /var/log/puppetlabs/mcollective, consistent with other log files.

## Puppet Server enhancements

* The Puppet Servermax\_requests\_per\_instance setting, which controls the maximum number of requests per instance of a JRuby interpreter, increased from 10,000 to 100,000 with this release of PE. This change provides a performance boost while still clearing short-lived environments.
* Puppet Server now uses Jetty 9.4 instead of Jetty 9.2.
* The sample Grafana dashboard for Puppet Server metrics visualizes new metrics added in this release.

## Security enhancements

* Connections to PE databases can now be made only with certificates. Usernames and passwords are no longer allowed by default.
* Unlabeled RBAC tokens stored in the database are now hashed. If you label tokens, they are stored unencrypted.

# Configurations

## Ports

Below are the ports that you’ll want to ensure are open. I’ve listed both new and existing ports for completeness.

Ports that will be new are:

* 8142 on the master to allow for incoming traffic/responses from agents
* 8143 on the agents
* 8170 for Code Manager status checks

System configuration information [can be found in the docs here](https://puppet.com/docs/pe/2017.3/installing/system_configuration.html#system_configuration).

| **Port** | **Use** |
| --- | --- |
| 8140 | The master uses this port to accept inbound traffic/requests from agents.  The console sends requests to the master on this port.  Certificate requests are passed over this port unless ca\_port is set differently.  Puppet Server status checks are sent over this port.  In a large environment installation, the master uses this port to send status checks to compile masters. (Not required to run PE.)  Classifier group: **PE Master** |
| 443 | This port provides host access to the console  The console accepts HTTPS traffic from end users on this port.  Classifier group: **PE Console** |
| 61613 | MCollective uses this port to accept inbound traffic/requests from agents.  Any host used to invoke commands must be able to reach MCollective on this port.  Classifier group: **PE ActiveMQ Broker** |
| 4433 | This port is used as a classifier / console services API endpoint.  The master communicates with the console over this port.  Classifier group: **PE Console** |
| 8081 | PuppetDB accepts traffic/requests on this port.  The master and console send traffic to PuppetDB on this port.  PuppetDB status checks are sent over this port.  Classifier group: **PE PuppetDB** |
| 8142 | Orchestrator and the **Run** button for Puppet and task runs use this port on the master of masters to accept inbound traffic/responses from agents via the Puppet Execution Protocol agent.  Classifier group: **PE Orchestrator** |
| 8143 | Orchestrator uses this port to accept connections from Puppet Communications Protocol brokers to relay communications. The orchestrator client also uses this port to communicate with the orchestration services running on the master of masters. If you install the client on a workstation, this port must be available on the workstation.  Classifier group: **PE Orchestrator** |
| 5432 | PostgreSQL runs on this port.  The console node needs to connect to the PuppetDB node hosting the PostgreSQL database on this port.  Classifier group: **PE PuppetDB** |
| 61616 | This port is used for ActiveMQ hub and spoke communication.  Classifier group: **PE ActiveMQ Broker** |
| 8170 | Code Manager status checks are sent over this port. |

# Agent Upgrades

A complete list of Server Components installed with 2017.3 [can be found here](https://puppet.com/docs/pe/2017.3/installing/what_gets_installed_and_where.html#software-components-installed).

Agent Version: 5.3.2 for 2017.3.0 and 2017.3.1. Version 5.3.3 for 2017.3.2.

Consider using the [Puppet Agent module](https://forge.puppet.com/puppetlabs/puppet_agent) for upgrading agents.

If you decline to upgrade your agents the impact will be that you will not be able to leverage certain PE 2017.3 capabilities.

# Hiera 5

Hiera 5 was introduced in 2017.1 which leverages Puppet 4.9.0 and Agent 1.9.3+.

Hiera 5 is a backwards-compatible evolution of Hiera, which is built into Puppet 4.9.0 and higher. To provide some backwards-compatible features, it uses the classic Hiera 3.x.x codebase version listed in this table.

For more information on migrating to Hiera 5 [please read this docs page](https://puppet.com/docs/puppet/5.3/hiera_migrate.html).

# User Groups and Accounts

You’ll notice new Orchestration services related groups and users now created.

Most notably the pe-orchestration-services user group and account for running orchestration services.

For a full list of user groups and accounts please reference this [page on the docs site](https://puppet.com/docs/pe/2017.3/installing/what_gets_installed_and_where.html#user-and-group-accounts-installed).

# Logs

New logs have been added since 2015.3.3. Below is a complete list of present logs with new additions highlighted.

## Master logs

The master has these logs.

* /var/log/puppetlabs/puppetserver/puppetserver.log — The master application logs its activity, including compilation errors and deprecation warnings, here.
* /var/log/puppetlabs/puppetserver/puppetserver-daemon.log — This is where fatal errors or crash reports can be found.
* /var/log/puppetlabs/puppetserver/pcp-broker.log — The log file for Puppet Communications Protocol brokers on compile masters.
* /var/log/puppetlabs/puppetserver/code-manager-access.log
* /var/log/puppetlabs/puppetserver/file-sync-access.log
* /var/log/puppetlabs/puppetserver/masterhttp.log
* /var/log/puppetlabs/puppetserver/puppetserver-access.log
* /var/log/puppetlabs/puppetserver/puppetserver.log
* /var/log/puppetlabs/puppetserver/puppetserver-status.log

## Agent logs

The locations of agent logs depend on the agent operating system.

On \*nix nodes, the agent service logs its activity to the syslog service. Your syslog configuration dictates where these messages are saved, but the default location is /var/log/messages on Linux, /var/log/system.log on Mac OS X, and /var/adm/messages on Solaris.

On Windows nodes, the agent service logs its activity to the event log. You can view its logs by browsing the event viewer.

## ActiveMQ logs

ActiveMQ has these logs.

* /var/log/puppetlabs/activemq/wrapper.log
* /var/log/puppetlabs/activemq/activemq.log
* /var/log/puppetlabs/activemq/data/audit.log

## MCollective logs

MCollective has these logs.

* /var/log/puppetlabs/mcollective.log — Maintained by the MCollective service, which is installed on all nodes.
* /var/log/puppetlabs/mcollective-audit.log — Exists on all nodes that have MCollective installed. Logs any MCollectiveactions run on the node, including information about the client that called the node

## Console and console services logs

The console and pe-console-services has these logs.

* /var/log/puppetlabs/nginx/error.log — Contains errors related to nginx. Console errors that aren't logged elsewhere can be found in this log.
* /var/log/puppetlabs/nginx/access.log
* /var/log/puppetlabs/console-services.log
* /var/log/puppetlabs/console-services-access.log
* /var/log/puppetlabs/console-services/console-services-api-access.log
* /var/log/puppetlabs/console-services-daemon.log — This is where fatal errors or crash reports can be found.

## Installer logs

The installer has these logs.

* /var/log/puppetlabs/installer/http.log — Contains web requests sent to the installer. This log is present only on the machine from which a web-based installation was performed.
* /var/log/puppetlabs/installer/installer-<timestamp>.log — Contains the operations performed and any errors that occurred during installation.
* /var/log/puppetlabs/installer/install\_log.lastrun.<hostname>.log — Contains the contents of the last installer run.

## Database logs

The database has these logs.

* /var/log/puppetlabs/puppetdb/puppetdb-access.log
* /var/log/puppetlabs/puppetdb/puppetdb-status.log
* /var/log/puppetlabs/puppetdb/puppetdb.log
* /var/log/puppetlabs/postgresql/pgstartup.log
* /var/log/puppetlabs/postgresql/postgresql-Fri.log
* /var/log/puppetlabs/postgresql/postgresql-Mon.log
* /var/log/puppetlabs/postgresql/postgresql-Sat.log
* /var/log/puppetlabs/postgresql/postgresql-Sun.log
* /var/log/puppetlabs/postgresql/postgresql-Thu.log
* /var/log/puppetlabs/postgresql/postgresql-Tue.log
* /var/log/puppetlabs/postgresql/postgresql-Wed.log

## Orchestration logs

The orchestration service and related components have these logs.

* /var/log/puppetlabs/orchestration-services.log
* /var/log/puppetlabs/orchestration-services-access.log
* /var/log/puppetlabs/orchestration-services-status.log
* /var/log/puppetlabs/orchestration-services-daemon.log — This is where fatal errors or crash reports can be found.
* /var/log/puppetlabs/orchestration-services/pcp-broker.log — The log file for Puppet Communications Protocolbrokers on the master of masters.
* /var/log/puppetlabs/orchestration-services/pcp-broker-access.log
* /var/log/puppetlabs/pxp-agent/pxp-agent.log (on \*nix) or C:/ProgramData/PuppetLabs/pxp-agent/var/log/pxp-agent.log (on Windows) — Contains the Puppet Execution Protocol agent log file.

# Compile Masters

While Compile Masters haven’t changed that much in theory they are now used to support other areas of PE such as Orchestration services. Whereas in 2015.3.3 your Compile Masters only had pe\_admin and pe\_repo, there are two new additions highlighted below:

* The MCollective client, peadmin
* The repository for agent installation, pe\_repo
* The controller profile used with PE client tools
* Puppet Communications Protocol (PCP) brokers to enable orchestrator scale

Logs for your Compile Masters are still found at: /var/log/puppetlabs/puppetserver/.

New with your upgrade you’ll find that logs for PCP brokers on compile masters are located at /var/log/puppetlabs/puppetserver/pcp-broker.log

## Compile Master Configuration for Orchestration at Scale

As you start to leverage Orchestration services you will want to think about scaling.

The orchestrator communicates with PCP brokers on compile masters over port 8143, and sends job-related messages to the brokers, which are then relayed by the brokers to Puppet Execution Protocol (PXP) agents. As you add compile masters, you also scale the number of PCP brokers that can send orchestration messages to agents.

When adding compile masters, you should load balance PXP agent connections to the PCP brokers running on them. PXP agents connect to PCP brokers running on compile masters over port 8142. Status checks on compile masters must be sent to port 8140.

For a more detailed explanation please [reference the docs here](https://puppet.com/docs/pe/2017.3/installing/installing_compile_masters.html#configure-compile-masters-for-orchestrator-scale).

# Client Tools

The pe-client-tools package collects a set of CLI tools that extend the ability for you to access services from the master or a workstation. While pe-client-tools are included in the PE install, if you’d like to leverage this on your individual nodes please [refer to this section of the docs](https://puppet.com/docs/pe/2017.3/installing/installing_pe_client_tools.html#install-pe-client-tools-on-a-managed-workstation).

Worth mentioning, pe-client-tools can be installed on both managed and unmanaged nodes.

The pe-client-tools package includes:

## Orchestrator

The orchestrator is a set of interactive command line tools that provide the interface to the orchestration service. Orchestrator also enables you to enforce change on the environment level. Tools include puppet job and puppet app.

## Puppet Access

Users can generate tokens to authenticate their access to certain command line tools and API endpoints.

## Code Manager CLI

The puppet-code command allows you to trigger Code Manager from the command line to deploy your environments.

## PuppetDB CLI

This a tool for working with PuppetDB, including building queries and handling exports.

# Current Infrastructure State

While not new to 2017.3, and present in 2016.4.x, there are a few new node statuses that may be new as you upgrade from 2015.3.3. Below is a high-level overview of these statuses.

### Nodes run in enforcement mode

**With failures** - This node’s last Puppet run failed, or Puppet encountered an error that prevented it from making changes. The error is usually tied to a particular resource (such as a file) managed by Puppet on the node. The node as a whole might still be functioning normally. Alternatively, the problem might be caused by a situation on the Puppet master, preventing the node's agent from verifying whether the node is compliant.

**With corrective changes** - During the last Puppet run, Puppet found inconsistencies between the last applied catalog and this node’s configuration, and corrected those inconsistencies to match the catalog.

*Note: Corrective change reporting is only available on agent nodes running PE 2016.4 and later. Agents running earlier versions will report all change events as "with intentional changes."*

**With intentional changes** - During the last Puppet run, changes to the catalog were successfully applied to the node.

**Unchanged** - This node's last Puppet run was successful, and it was fully compliant. No changes were necessary.

### Nodes run in no-op mode

Note: No-op mode reporting is only available on agent nodes running PE 2016.4 and later. Agents running earlier versions will report all no-op mode runs as "would be unchanged."

**With failures** - This node’s last Puppet run in no-op mode failed, or Puppet encountered an error that prevented it from simulating changes.

**Would have corrective changes** - During the last Puppet run, Puppet found inconsistencies between the last applied catalog and this node’s configuration, and would have corrected those inconsistencies to match the catalog.

**Would have intentional changes** - During the last Puppet run, catalog changes would have been applied to the node.

**Would be unchanged** - This node’s last Puppet run was successful, and the node was fully compliant. No changes would have been necessary.

### Nodes not reporting

**Unresponsive** - The node hasn't reported to the Puppet master recently. Something might be wrong. The cutoff for considering a node unresponsive defaults to one hour, and can be configured.

Check the run status table to see the timestamp for the last known Puppet run for the node and an indication of whether its last known run was in no-op mode. Correct the problem to resume Puppet runs on the node.

**Have no reports** - Although Puppet Server is aware of this node's existence, the node has never submitted a Puppet report for one or more of the following reasons: it's a newly commissioned node; it has never come online; or its copy of Puppet is not configured correctly.

Note: Expired or deactivated nodes are displayed on the Overview page for seven days. To extend the amount of time that you can view or search for these nodes, change the node-ttl setting in PuppetDB. Changing this setting affects resources and exported resources.

# Orchestrator

## Configuring and Setting up the Orchestrator

Once you've installed PE or the client tools package, there are a few tasks you need to do to prepare your PE infrastructure for orchestration services.

* Setting PE RBAC permissions and token authentication for Puppet orchestrator
* Enable cached catalogs for use with the orchestrator (optional)
* Review the orchestrator configuration files and adjust them as needed

All of these instructions assume that PE client tools are installed.

The details for each of these steps [can be found in the docs here](https://puppet.com/docs/pe/2017.3/orchestrator/configuring_puppet_orchestrator.html).

## Puppet orchestrator architecture

The functionality of the orchestrator is derived from the Puppet Execution Protocol (PXP) and the Puppet Communications Protocol (PCP).

* PXP: A message format used to request that a task be executed on a remote host and receive responses on the status of that task. This is used by the pe-orchestration services to run Puppet on agents.
* PXP agent: A system service in the agent package that runs PXP.
* PCP: The underlying communication protocol that describes how PXP messages get routed to an agent and back to the orchestrator.
* PCP broker: A JVM-based service that runs in pe-orchestration-services on the master of masters (MoM) and in the pe-puppetserver service on compile masters. PCP brokers route PCP messages, which declare the content of the message via message type, and identify the sender and intended recipient. PCP brokers on compile masters connect to the orchestrator, and the orchestrator uses the brokers to direct messages to PXP agents connected to the compile masters. When using compile masters, PXP agents running on PE components (the Puppet master, PuppetDB, and the PE console) connect directly to the orchestrator, but all other PXP agents connect to compile masters via load balancers.

## Configuration notes for the orchestrator and related components

Configuration and tuning for the components in the orchestrator happen in various files.

* pe-orchestration-services: The underlying service for the orchestrator. The main configuration file is /etc/puppetlabs/orchestration-services/conf.d.
* Additional configuration for large infrastructures may include tuning the pe-orchestration-services JVM heap size, increasing the limit on open file descriptors for pe-orchestration-services, and tuning ARP tables.
* PCP broker: Part of the pe-puppetserver service. The main configuration file is /etc/puppetlabs/puppetserver/conf.d.
* The PCP broker requires JVM memory and file descriptors, and these resources scale linearly with the number of active connections. Specifically, the PCP broker requires:
  + Approximately 40 KB of memory (when restricted with the `-Xmx` JVM option)
  + One file descriptor per connection
  + An approximate baseline of 60 MB of memory and 200 file descriptors
* For a deployment of 100 agents, expect to configure the JVM with at least -Xmx64m and 300 file descriptors. Message handling requires minimal additional memory.
* PXP agent: Configuration is managed by the agent profile (puppet\_enterprise::profile::agent).
* The PXP agent is configured to use Puppet’s SSL certificates and point to one PCP broker endpoint. If high availability (HA) is configured, the agent will point to additional PCP broker endpoints in the case of failover.

## Debugging the orchestrator and related components

If you need to debug the orchestrator or any of its related components, the following log locations may be helpful.

* pe-orchestration-services: The main log file is /var/log/puppetlabs/orchestration-services/orchestration-services.log.
* PCP: The main log file for PCP brokers on compile masters is /var/log/puppetlabs/puppetserver/pcp-broker.log. You can configure logback through the Puppet server configuration.
* The main log file for PCP brokers on the MoM is /var/log/puppetlabs/orchestration-services/pcp-broker.log.
* You can also enable an access log for messages.
* PXP agent: The main log file is /var/log/puppetlabs/pxp-agent/pxp-agent.log (on \*nix) or C:/ProgramData/PuppetLabs/pxp-agent/var/log/pxp-agent.log (on Windows). You can configure this location as necessary.
* Additionally, metadata about Puppet runs triggered via the PXP agent are kept in the spool-dir, which defaults to /opt/puppetlabs/pxp-agent/spool (on \*nix) and or C:/ProgramData/PuppetLabs/pxp-agent/var/spool (on Windows). Results are kept for 14 days.

## Jobs - Running Puppet on Nodes

The Puppet orchestrator is a set of interactive tools used to deploy configuration changes when and how you want them. You can use the orchestrator to run Puppet from the console, command line, or API.

You can use the orchestrator to enforce change based on a:

* selection of nodes – from the console or the command line:  
  puppet job run --nodes <COMMA-SEPARATED LIST OF NODE NAMES>
* PQL nodes query – from the console or the command line, for example:  
  puppet job run --query 'nodes[certname] { facts {name = "operatingsystem" and value = "Debian" }}'
* application or application instance - from the command line.

If you're putting together your own tools for running Puppet or want to enable CI workflows across your infrastructure, use the orchestrator API. For more information reference the [docs for running puppet on nodes.](https://puppet.com/docs/pe/2017.3/managing_nodes/run_puppet_on_nodes.html)

## Tasks

PE 2017.3 adds tools for running ad hoc tasks on nodes in your infrastructure.

You can run tasks from the console, on the command line, or by the orchestrator API /command/task endpoint. You can target an individual node, a list of nodes, or a set of nodes derived from a PQL query. Watch task execution in real time from the console or command line, and take advantage of PE's built-in job reporting and activity service that shows a complete history of task jobs. Role-based access control allows you to define who can run which tasks.

Tasks are packaged in modules, and this release includes some default tasks—facter\_task, package, service, and puppet\_conf—to get you started. You can install many more from the Forge, or write your own.

### Configuration

The orchestrator gives you the ability to set up jobs in the console or on the command line to run ad hoc or arbitrary tasks across systems in your infrastructure.

You can install pre-existing tasks, and run tasks from the console and from the command line.

Note: If you have set up compile masters and you want to use tasks, you must either set master\_uris or you server\_list on agents to point to to your compile masters. This setting is described in the section on configuring compile masters for orchestrator scale.

### Pre-Installed Tasks

Puppet Enterprise comes with some pre-installed tasks, but you must install any other tasks you want to use.

PE comes with facter\_task, package, service, and puppet\_conf tasks. Use the package task to inspect, install, upgrade, and manage packages. Use the service task to start, stop, restart, and check the status of services running on your systems.

Tasks are packaged in Puppet modules, so you can install them directly from the Forge or install and manage them with a Puppetfile and Code Manager.

If you plan to run tasks from the console, those tasks must be installed into the production environment.

# Hiera overrides in the console

## Summary

PE 2017.3 adds configuration data in the console, where you can now set parameters on node groups without declaring the class. Data that is set in the console is used for automatic parameter lookup, which promotes code regularity and predictability.

Console configuration data overrides default or missing parameter values on classes that are already declared. This capability resolves the class conflicts that previously prevented using both the node classifier and the roles and profiles method.

You can set configuration data on the **Configuration** tab (formerly the **Classes** tab). A new node group permission, **Edit configuration data**, controls the ability to create and edit configuration data. Data editing is enabled by default on new installations, but you must enable it on upgrades from previous versions.

## Configuration

To take advantage of Hiera overrides in the console some configuration is required.

This feature **is not enabled for upgrades by default**.

Please reference [the following instructions](https://puppet.com/docs/pe/2017.3/managing_nodes/grouping_and_classifying_nodes.html#enable-data-editing-in-the-console) on how to enable and configure Hiera overrides in the console.

# Package inventory and management

PE 2017.3 includes improvements to the **Packages** page, where you can now view and manage your complete package inventory.

The new packages details page allows you to:

* Separate package instances that are managed with Puppet from those that are not managed with Puppet.
* Filter packages by version, and generate lists of affected nodes.
* See where packages managed with Puppet are declared in your Puppet code, and quickly navigate to that location.
* Run tasks to update packages that are not managed with Puppet.

## Configuring Package Inspector

Package data collection is disabled by default, so the Packages page in the console will initially appear blank. In order to view a node's current package inventory, enable package data collection.

For information on how to enable Package Inspector [refer to the docs](https://puppet.com/docs/pe/2017.3/inspecting_infrastructure/viewing_packages_in_use.html#enable-package-data-collection).

# Code Manager / FileSync / r10k

If you are already using r10k to manage your Puppet code, we suggest that you upgrade to Code Manager. Code Manager works in concert with r10k, so when you switch to Code Manager, you no longer interact directly with r10k.

If you're using r10k and aren't ready to switch to Code Manager yet, you can continue using r10k alone. You push your code changes to your version control repo, deploy environments from the command line, and r10k creates environments and installs the modules for each one.

Both Code Manager and the r10k code management tool are built into PE, so you don't have to install anything.

# 2017.3 and Beyond

The following section can be used for reference when deciding which version fo 2017.3.z to upgrade to next.

## 2017.3.0

[Default global hiera.yaml no longer includes common.yaml or nodes/clientcert (2017.3.0)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#default-global-hiera-yaml-no-longer-includes-common-yaml-or-nodes-clientcert-2017-3-0)

In order to keep environment-specific data within the environment folder, the default global hiera.yaml now supports console data only and doesn't include the previous YAML file hierarchy. If you rely on the default global YAML file hierarchy of nodes/%{clientcert}.yaml and common.yaml, create a Hiera 5 compatible hiera.yaml file in your control-repo environment folder instead. For example: <https://github.com/puppetlabs/control-repo/blob/production/hiera.yaml>

[PostgreSQL 9.6 upgrade (2017.3.0)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#postgresql-9-6-upgrade-2017-3-0)

This version of PE upgrades PostgreSQL to version 9.6. If you use an external PostgreSQL instance, you must upgrade it before you upgrade PE.

Note: If you're upgrading a split installation of a PE version earlier than 2016.4.3 with an external PostgreSQL instance, you must upgrade with the --force flag, for example:

/opt/puppetlabs/puppet/bin/puppet infrastructure configure --detailed-exitcodes --modulepath=/opt/puppetlabs/server/data/enterprise/modules --no-noop --upgrade-from=<PREVIOUS PE VERSION> --force

[Purge nodes without restart (2017.3.0)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#purge-nodes-without-restart-2017-3-0)

[PuppetDB "time-to-live" default (2017.3.0)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#puppetdb-time-to-live-default-2017-3-0)

PuppetDB now uses 14 days for a default “time-to-live” value (node-purge-ttl) before it deletes nodes that have been deactivated or expired. This default can be changed as needed, as documented in the Configuring PE section.

[Simplified file sync reset (2017.3.0)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#simplified-file-sync-reset-2017-3-0)

[New location for MCollective logs (2017.3.0)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#new-location-for-mcollective-logs-2017-3-0)

On non- Windows systems, MCollective server logs now appear in /var/log/puppetlabs/mcollective, consistent with other log files.

[Puppet Server enhancements (2017.3.0)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#puppet-server-enhancements-2017-3-0)

* The Puppet Servermax\_requests\_per\_instance setting, which controls the maximum number of requests per instance of a JRuby interpreter, increased from 10,000 to 100,000 with this release of PE. This change provides a performance boost while still clearing short-lived environments.
* Puppet Server now uses Jetty 9.4 instead of Jetty 9.2.
* The sample Grafana dashboard for Puppet Server metrics visualizes new metrics added in this release.

[Orchestrator enhancements (2017.3.0)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#orchestrator-enhancements-2017-3-0)

[Console enhancements (2017.3.0)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#console-enhancements-2017-3-0)

[Security enhancements (2017.3.0)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#security-enhancements-2017-3-0)

* Connections to PE databases can now be made only with certificates. Usernames and passwords are no longer allowed by default.
* Unlabeled RBAC tokens stored in the database are now hashed. If you label tokens, they are stored unencrypted.

## 2017.3.1

[MSI properties in Windows simplified agent install script (2017.3.1)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#msi-properties-in-windows-simplified-agent-install-script-2017-3-1)

[Package task fields prepopulated](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#package-task-fields-prepopulated-2017-3-1)

## 2017.3.2

[Control the state of the Puppet service during agent install (2017.3.2)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#control-the-state-of-the-puppet-service-during-agent-install-2017-3-2)

[Reinitialize an high availability replica (2017.3.2)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#easier-entry-and-editing-of-console-node-data-2017-3-2)

[Structured fact keys in the console are sorted alphabetically (2017.3.2)](https://puppet.com/docs/pe/2017.3/release_notes/enhancements.html#structured-fact-keys-in-the-console-are-sorted-alphabetically-2017-3-2)