Pulp Puppet Support Documentation

Release 2.7.1

Pulp Team

December 10, 2015

Contents

CHAPTER 1

User Guide

1.1 Introduction

Puppet support for Pulp allows you to create and publish repositories of Puppet modules. One common use case is to mirror Puppet Forge. You can synchronize an existing repository such as all or part of Puppet Forge, upload your own Puppet modules, and publish the result as a repository inside your own network.

Another common use case is to copy synced modules into a custom repository and add additional modules by uploading them directly. This allows you to test new modules or new versions of existing modules and then easily promote them into a production repository.

Puppet modules must adhere to the Puppet 3.6+ metadata guidelines, which require a valid *metadata.json* file to be present in the puppet module. Also, each tag.gz file must contain only one Puppet module inside it. Extra directories or Puppet modules can cause unexpected behavior.

Consumers must have Puppet 2.7.14 to 3.4.3 installed, and we recommend getting the Puppet client packages directly from Puppet Labs.

1.2 Release Notes

Contents:

1.2.1 Pulp Puppet 2.7 Release Notes

Pulp Puppet 2.7.0

New Features

- Support for using the Puppet Forge v3 API for installing modules
- The *Install Distributor* removes the environment directory on repo delete

API Changes

- The *unit_key* parameter is ignored by the API when uploading puppet modules. The unit key is always determined through inspection of metadata.json in the top level directory. The parameter *unit key* is still required by the Pulp platform, but its value is ignored for Puppet content.

Bugs Fixed

You can see the list of bugs fixed.

1.2.2 Pulp 2.6 Release Notes

Pulp 2.6.1

List of bugs fixed in 2.6.1.

Pulp 2.6.0

You can see the bugs fixed in 2.6.0.

1.2.3 Pulp 2.5 Release Notes

Pulp 2.5.0

The 2.5.0 release of `pulp-puppet` is a minor update release with bugfixes and two new features.

- The `--skip-dep` and `--modulepath` flags are now supported during install and update operations.
- A new SELinux boolean named `pulp_manage_puppet` is introduced for the Install Distributor. It works with the new SELinux policy introduced in Pulp Platform 2.5.0. Read more in the Plugin Configuration section of the Pulp Puppet Technical Reference for more details on this new boolean.

Users upgrading from 2.3.x or earlier will need to manually delete the `/var/www/pulp_puppet` directory after upgrading. This is the old location for puppet module publishing in Pulp 2.3 but is no longer used. The directory should be empty after the migration is complete.

You can see the list of bugs fixed.

1.2.4 Pulp 2.4 Release Notes

Pulp 2.4.1

The 2.4.1 release is a minor bugfix release. You can see the list of bugs fixed here.

Pulp 2.4.0

New Features

- Install and update operations can now be performed with puppet version 3.3+ against a pulp repository. See *Installing With Puppet Client 3.3+*.
- Pulp's puppet distributor is now updated to publish to /var/lib/pulp/puppet instead of /var/www/pulp_puppet.

API Changes

- The sync progress report's modules section has changed the way it reports errors with individual modules. The `modules` sub-object has an `individual_errors` attribute that used to index a JSON object, with module names (including version) as keys. Pulp stores this data structure in MongoDB, which led to RHBZ #1072580 because the module versions use periods which are illegal characters to use as document keys. The `individual_errors` attribute now indexes a list of JSON objects, and each JSON object has these keys: `module` (which is formatted as "name-version"), `author`, `exception`, and `traceback`. `individual_errors` used to be null when there weren't any errors, but would be an object if there were errors. Now `individual_errors` will always be an array. It will be empty when there are no errors.

Notable Bugs Fixed

All Bugs

1.2.5 Pulp 2.3 Release Notes

Pulp 2.3.0

New Features

- Syncs now support all of the networking settings that have been available for yum repository syncs. This includes proxy config, bandwidth throttling, etc.
- A new "install distributor" is available through the REST API that will install all modules in a repository to a specified path on the local filesystem. See the developer guide for more details.

Notable Bugs Fixed

- Full semantic versions were not supported by the upload process, despite being supported by Puppet Forge and other puppet tools.

All Bugs

You can see the complete list of over 100 bugs that were fixed in Pulp 2.3.0.

Pulp 2.3.1

Bugs Fixed

- Uploading puppet modules resulted in `pulp-admin` exiting with an unhelpful error message.
- Installing puppet modules on consumers also resulted in `pulp-admin` exiting with an unhelpful error message.

1.2.6 Pulp 2.2 Release Notes

Pulp 2.2.0

New Features

For Puppet functionality, this is mostly a bug-fix release.

Many of the commands in the command line interface have improved. In particular, many commands that used to display a task ID will now automatically poll the server and display progress until it is complete.

A technical reference has been added for Puppet support, which is available at http://www.pulpproject.org/docs/

Upgrade Instructions

Please see the Pulp Platform upgrade instructions to upgrade the Pulp and Puppet RPMs and database.

Pulp 2.2.1

No changes directly impact Pulp's Puppet features, but several bug fixes impact the Pulp Platform. See the full list of bug fixes below.

All Bug Fixes

1.2.7 Pulp 2.1 Release Notes

Pulp 2.1.0

New Features

1. Pulp 2.1 now supports Fedora 18 and Apache 2.4.
2. We now support the use of the `puppet module` tool (provided by Puppet) against a Pulp server.
3. Pulp can now manage installation, upgrade, and removal of puppet modules on Pulp consumers."

Upgrade Instructions

Upgrade the Platform and Pulp Puppet Software Please see the Pulp Platform upgrade instructions to upgrade the Pulp and Puppet RPMs and database.

Republish Puppet Repositories The Puppet Forge API will not work for previously published repositories. Those repositories must be republished. You can find out the repository IDs on your system with this command:

```
$ sudo pulp-admin puppet repo list --fields repo_id
```

For each ID in that list, you can republish it with this command, substituting <repo_id> with the ID of the repository you wish to republish:

```
$ sudo pulp-admin puppet repo publish run --repo-id=<repo_id>
```

1.3 Installation

Note: If you followed the Pulp installation instructions you already have Puppet features installed. If not, this document will walk you through the installation.

1.3.1 Prerequisites

Puppet support requires the namespace `/api/v1/` at the root of your web server in order to implement an API compatible with Puppet Forge. We don't like taking that namespace, but it was the only way to support the use of Puppet Labs' command line tool against a Pulp server.

Consumers must have Puppet 2.7.14 to 3.4.3 installed, and we recommend getting packages directly from Puppet Labs.

Please see the Pulp User Guide for other prerequisites including repository setup.

Note: Consumer install and update operations against a repository published over HTTPS will do SSL certificate verification. Thus, you must ensure that the `puppet module` tool is able to verify the server's certificate against a trusted CA in order to publish puppet repositories over HTTPS.

1.3.2 Server

If you followed the Pulp User Guide install instructions, you already have Puppet support installed. If not, just install the following package.

```
$ sudo yum install pulp-puppet-plugins
```

Then run `pulp-manage-db` to initialize the new types in Pulp's database.

```
$ sudo -u apache pulp-manage-db
```

Then restart each pulp component, as documented in the Pulp User Guide.

1.3.3 Admin Client

If you followed the Pulp User Guide install instructions, you already have Puppet support installed. If not, just install the following package.

```
$ sudo yum install pulp-puppet-admin-extensions
```

1.4 Configuration

1.4.1 Importer Configuration

The Puppet importer is configured by editing `/etc/pulp/server/plugins.conf.d/puppet_importer.json`. This file must be valid JSON.

The importer supports the settings documented in Pulp's importer config docs

1.5 Quick Start

1.5.1 Login

The default admin password is “admin”.

```
$ pulp-admin login -u admin
Enter password:
Successfully logged in. Session certificate will expire at Dec 14 18:50:41 2012
GMT.
```

1.5.2 Create a Repository

This creates a basic repository that will fetch modules from Puppet Forge.

```
$ pulp-admin puppet repo create --repo-id=repo1 --description="Mirror of Puppet Forge" --display-name
Successfully created repository [repo1]
```

By default, Pulp will serve this repository over HTTP without SSL. Adding `--serve-https=true` would cause it to also be served over HTTPS. Non-SSL HTTP can similarly be disabled.

1.5.3 Update a Repository

Let’s add a query to limit the scope of how many modules get synced.

Note: The `--query` option was deprecated in version 2.1

```
$ pulp-admin puppet repo update --repo-id=repo1 --queries=libvirt
Repository [repo1] successfully updated
```

1.5.4 List Repositories

```
$ pulp-admin puppet repo list
+----------------------------------------------------------------------+
                          Puppet Repositories
+----------------------------------------------------------------------+

Id:                 repo1
Display Name:       Repo 1
Description:        Mirror of Puppet Forge
Content Unit Count: 0
```

To include more details, use the `--details` flag.

```
$ pulp-admin puppet repo list --details
+----------------------------------------------------------------------+
                          Puppet Repositories
+----------------------------------------------------------------------+

Id:                 repo1
Display Name:       Repo 1
Description:        Mirror of Puppet Forge
Content Unit Count: 0
```

```
Notes:
Importers:
  Config:
    Feed:    http://forge.puppetlabs.com
    Queries: libvirt
  Id:                 puppet_importer
  Importer Type Id: puppet_importer
  Last Sync:          None
  Repo Id:            repo1
  Scheduled Syncs:
  Scratchpad:         None
Distributors:
  Auto Publish:         True
  Config:
  Distributor Type Id: puppet_distributor
  Id:                   puppet_distributor
  Last Publish:         None
  Repo Id:              repo1
  Scheduled Publishes:
  Scratchpad:           None
```

1.5.5 Search Repositories

This is a search for all repositories with more than 0 modules.

```
$ pulp-admin puppet repo search --gt='content_unit_count=0'
+----------------------------------------------------------------------+
                              Repositories
+----------------------------------------------------------------------+

Id:                 forge
Display Name:       forge
Description:        None
Content Unit Count: 669
Notes:
Scratchpad:

Id:                 repo1
Display Name:       Repo 1
Description:        Mirror of Puppet Forge
Content Unit Count: 2
Notes:
Scratchpad:
```

1.5.6 Sync a Repository

This process downloads content from an existing repository and places it into a repository hosted by Pulp. This allows you to make a local copy of all or part of a remote repository.

```
$ pulp-admin puppet repo sync run --repo-id=repo1
+----------------------------------------------------------------------+
                    Synchronizing Repository [repo1]
+----------------------------------------------------------------------+

This command may be exited by pressing ctrl+c without affecting the actual
```

```
operation on the server.

Downloading metadata...
[==================================================] 100%
Metadata Query: 1/1 items
... completed

Downloading new modules...
[==================================================] 100%
Module: 2/2 items
... completed

Publishing modules...
[==================================================] 100%
Module: 2/2 items
... completed

Generating repository metadata...
[-]
... completed

Publishing repository over HTTP...
... completed

Publishing repository over HTTPS...
... skipped
```

At this point, the repository has been published and is available via HTTP. You can see it at http://localhost/pulp/puppet/repo1/ (adjust the hostname as necessary).

1.5.7 List Modules in a Repository

```
$ pulp-admin puppet repo modules --repo-id=repo1
Name:          libvirt
Version:       0.0.1
Author:        thias
Dependencies:
Description:   Install, configure and enable libvirt.
License:       Apache 2.0
Project Page:  http://glee.thias.es/puppet
Source:        git://github.com/thias/puppet-modules/modules/libvirt
Summary:       Libvirt virtualization API and capabilities
Tag List:      rhel, libvirt, kvm, CentOS
Types:

Name:          virt
Version:       1.0.0
Author:        carlasouza
Dependencies:
Description:   None
License:       GPLv3
Project Page:  None
Source:
Summary:       None
Tag List:      virtualization, kvm, xen, openvz, libvirt
Types:
```

To be more specific, we can search by name.

```
$ pulp-admin puppet repo modules --repo-id=repo1 --str-eq='name=libvirt'
Name:          libvirt
Version:       0.0.1
Author:        thias
Dependencies:
Description:   Install, configure and enable libvirt.
License:       Apache 2.0
Project Page: http://glee.thias.es/puppet
Source:        git://github.com/thias/puppet-modules/modules/libvirt
Summary:       Libvirt virtualization API and capabilities
Tag List:      rhel, libvirt, kvm, CentOS
Types:
```

Or by license, and for fun let's use a regex.

```
$ pulp-admin puppet repo modules --repo-id=repo1 --match='license=^GPL.*'
Name:          virt
Version:       1.0.0
Author:        carlasouza
Dependencies:
Description:   None
License:       GPLv3
Project Page: None
Source:
Summary:       None
Tag List:      virtualization, kvm, xen, openvz, libvirt
Types:
```

1.5.8 Copy Modules Between Repositories

Assuming we have repositories "repo1" and "repo2", and "repo1" has two units as a result of the above sync.

```
$ pulp-admin puppet repo create --repo-id=repo2
Successfully created repository [repo2]

$ pulp-admin puppet repo copy --from-repo-id=repo1 --to-repo-id=repo2 --str-eq='name=libvirt'
Progress on this task can be viewed using the commands under "repo tasks".

$ pulp-admin repo tasks list --repo-id=repo1
+----------------------------------------------------------------------+
                                 Tasks
+----------------------------------------------------------------------+

Operations:  associate
Resources:   repo2 (repository), repo1 (repository)
State:       Successful
Start Time:  Unstarted
Finish Time: 2012-12-07T19:04:54Z
Result:      Incomplete
Task Id:     54459b2f-6ed9-4918-94c9-63e2b3370554
```

1.5.9 Upload a module

Assuming we have a repository with repo-id *repo1* we can upload an archive containing a Puppet module. This operation does not auto publish the repository.

```
    $ pulp-admin puppet repo uploads upload --file puppetlabs-apache-1.4.0.tar.gz --repo-id repo1
      +----------------------------------------------------------------------+
                                    Unit Upload
      +----------------------------------------------------------------------+

      Extracting necessary metadata for each request...
      [==================================================] 100%
      Analyzing: puppetlabs-apache-1.4.0.tar.gz
      ... completed

      Creating upload requests on the server...
      [==================================================] 100%
      Initializing: puppetlabs-apache-1.4.0.tar.gz
      ... completed

      Starting upload of selected units. If this process is stopped through ctrl+c,
      the uploads will be paused and may be resumed later using the resume command or
      cancelled entirely using the cancel command.

      Uploading: puppetlabs-apache-1.4.0.tar.gz
      [==================================================] 100%
      147426/147426 bytes
      ... completed

      Importing into the repository...
      This command may be exited via ctrl+c without affecting the request.


      [\]
      Running...

      Task Succeeded


      Deleting the upload request...
      ... completed
```

1.5.10 Publish a Repository

By default, repositories are auto-published following a sync. However, if you create an new repository and populate it with content by copying and/or uploading modules, you will need to publish manually. Since that is the case for “repo2” into which we just copied a module, let’s publish that repo.

```
$ pulp-admin puppet repo publish run --repo-id=repo2
+----------------------------------------------------------------------+
                     Publishing Repository [repo2]
+----------------------------------------------------------------------+

This command may be exited by pressing ctrl+c without affecting the actual
operation on the server.

Publishing modules...
[==================================================] 100%
Module: 1/1 items
... completed

Generating repository metadata...
```

```
[-]
... completed

Publishing repository over HTTP...
... completed

Publishing repository over HTTPS...
... skipped
```

1.5.11 Delete a Repository

```
$ pulp-admin puppet repo delete --repo-id=repo1
Repository [repo1] successfully deleted
```

1.6 Recipes

1.6.1 Mirror Puppet Forge

Start by creating a new repository that includes the URL for Puppet Forge. Use any repo-id you like, as long as it is unique within Pulp.

```
$ pulp-admin puppet repo create --repo-id=forge --feed=http://forge.puppetlabs.com
Successfully created repository [forge]
```

Next synchronize the repository, which downloads all of the modules into the local repository.

```
$ pulp-admin puppet repo sync run --repo-id=forge
+----------------------------------------------------------------------+
                    Synchronizing Repository [forge]
+----------------------------------------------------------------------+

This command may be exited by pressing ctrl+c without affecting the actual
operation on the server.

Downloading metadata...
[==================================================] 100%
Metadata Query: 1/1 items
... completed

Downloading new modules...
[==================================================] 100%
Module: 669/669 items
... completed

Publishing modules...
[==================================================] 100%
Module: 669/669 items
... completed

Generating repository metadata...
[\]
... completed

Publishing repository over HTTP...
```

```
... completed

Publishing repository over HTTPS...
... skipped
```

Let's take a moment to display the repository and admire your work!

```
$ pulp-admin puppet repo list
+----------------------------------------------------------------------+
                          Puppet Repositories
+----------------------------------------------------------------------+

Id:                 forge
Display Name:       forge
Description:        None
Content Unit Count: 669
```

Also point a browser to http://localhost/pulp/puppet/forge/ (adjust the host name as needed) to view the published repository.

1.6.2 Install Modules

Installing With Puppet Client 3.3+

To install from a specific pulp repository, the forge URL is formed as `http://<hostname>/pulp_puppet/forge/repository/<repository_id>`.
To install as a consumer from any bound repository, the URL is formed as `http://<hostname>/pulp_puppet/forge/consumer/<consumer_id>`.

For example, to install module puppetlabs-stdlib from the repository "demo", run the following command.

```
$ puppet module install --module_repository=http://localhost/pulp_puppet/forge/repository/demo puppet
```

Or to install module puppetlabs-stdlib as the consumer "con1" from any repository to which that consumer is bound, run the following command.

```
$ puppet module install --module_repository=http://localhost/pulp_puppet/forge/consumer/con1 puppetla
```

Installing With Puppet Client < 3.3

You might notice that this command does not work:

```
$ puppet module install --module_repository http://localhost/pulp/puppet/forge author/name
```

For technical reasons described in the note below, the `puppet module install` tool in versions prior to 3.3 ignores the part of the URL after the host name, which means we cannot put the repository ID in the URL. We have a work-around that will still allow you to use the `puppet module install` command with Pulp, and it involves the use of basic auth credentials as part of the URL.

Note: Puppet Forge implements a web API that their client uses to obtain dependency data when installing a module. Unfortunately, their command line tool has hard-coded absolute paths instead of relative, which means the API must live at the root of a web server. As a result, we cannot put the repository ID in the path as you would expect with the above example.

- **Consumer ID** For a consumer registered with Pulp, just specify its consumer ID as the username in the URL, and a "." for the password. A consumer's ID is a unique identifier just like a username, so this isn't actually a bad use of that field. When a consumer ID is provided, Pulp searches all of that consumer's bound repositories for either the newest version of the requested module, or if a version is specified, searches for the exact version requested. Once a suitable module has been located in a bound repository, all dependency data returned is scoped to that same repository.

```
$ puppet module install --module_repository http://consumer1:.@localhost
```

- **Repository ID** For machines that are not bound to a repository, or for a bound machine where you want to specify a repository, do so in the password field. If a repository ID is specified, any value in the username field is ignored. To keep the convention, use a single "." as a null value.

```
$ puppet module install --module_repository http://.:forge@localhost
```

The repository URL can be set in `/etc/puppet/puppet.conf` so that it does not need to be provided on the command line every time. See Puppet's own documentation for details.

Note: The dependency API from Puppet Forge has been re-implemented by Pulp and can be accessed at /api/v1/releases.json. Puppet Forge also implements a search API that Pulp has not re-implemented due to even more restrictive use of absolute URLs in the puppet tool.

At this time, Puppet Labs is working on a new version of their API that will include public documentation, and we believe that new API will be much easier to integrate with.

1.6.3 Puppet Consumers

Puppet modules installed on puppet masters can be managed with Pulp's consumer features. Start by registering the system as a consumer. This process only needs to happen once, after which the consumer can bind to repositories of any content type (puppet modules, RPMs, or any other content supported by Pulp). Note that the following command requires root privileges.

```
$ sudo pulp-consumer register --consumer-id=fred
Enter password:
Consumer [fred] successfully registered
```

Next the consumer should be bound to a repository. This can be done with the `pulp-consumer` command from a shell on the consumer machine.

```
$ pulp-consumer puppet bind --repo-id=forge
Bind tasks successfully created:

Task Id: 9531a15f-d19d-4c77-9a61-ac67e1223c93

Task Id: 9f06e091-e54c-47d4-8b17-cebfc4451215
```

The same could be accomplished using the pulp-admin command, which interacts with the Pulp server. The server then notifies the consumer of the binding.

```
$ pulp-admin puppet consumer bind --repo-id=forge --consumer-id=fred
Bind tasks successfully created:

Task Id: 88a49289-2dc8-49f3-9050-92bcd8ddc8de

Task Id: 8e8f3cd7-420e-447c-8feb-8cf5703a2324
```

Either way, we can now see from pulp-admin that the consumer is bound to the repository with ID "forge".

```
$ pulp-admin consumer list
+----------------------------------------------------------------------+
                              Consumers
+----------------------------------------------------------------------+

Id:            fred
Display Name:  fred
Description:   None
Bindings:
  Confirmed:   forge
  Unconfirmed:
Notes:
```

Install

For install requests, Pulp will search all repositories to which the consumer is bound to find the requested module. If no version is specified, it will find the newest version available. Once the module has been found in a repository, dependency resolution will occur only within that repository. The install command will automatically install any dependencies.

This example installs a specific version of the `puppetlabs/stdlib` module.

```
$ pulp-admin puppet consumer install run --consumer-id=fred -u puppetlabs/stdlib/3.1.1
This command may be exited via ctrl+c without affecting the request.

[\]
1 change was made

Install Succeeded
```

Update

Updates follow the same repository matching process as installs. This example updates the `puppetlabs/stdlib` module. Since a version is not specified, the newest available version will be installed.

```
$ pulp-admin puppet consumer update run --consumer-id=fred -u puppetlabs/stdlib
Update task created with id [ 672d34e9-e0c3-40ea-942f-76da2d7dbad1 ]

This command may be exited via ctrl+c without affecting the request.

[|]
1 change was made

Update Succeeded
```

Uninstall

Uninstall requests merely uninstall the specified module.

```
$ pulp-admin puppet consumer uninstall run --consumer-id=fred -u puppetlabs/stdlib
Uninstall task created with id [ 0f040d05-d37d-4a4d-a1aa-1c882aeea771 ]

This command may be exited via ctrl+c without affecting the request.

[-]
```

```
Waiting to begin
1 change was made

Uninstall Succeeded
```

1.6.4 Building and Importing Modules

Start by creating a working directory. The directory will be used for git cloning and for building puppet modules. This directory will be the *feed* for our Pulp repository. Use any directory you like so long as you have *write* and *execute* permissions.

```
$ sudo mkdir -p /opt/puppet/modules
$ sudo chmod -R 777 /opt/puppet
```

Next, create a new repository that specifies a feed URL for the directory that will be created in a subsequent step. Use any repo-id you like, as long as it is unique within Pulp.

```
$ pulp-admin puppet repo create --repo-id=puppet-builds --feed=file:///opt/puppet/modules/
Successfully created repository [puppet-builds]
```

Next, build the puppet modules from source. The `pulp-puppet-module-builder` tool is provided with Pulp puppet support to make this step easier. The tool uses the puppet module tool to build modules. It also supports basic Git repository operations such a cloning and the checkout of branches and tags to simplify the building and importing of pupppet modules from git repositories.

In this example, we will build the `puppetlabs-xinitd` module provided by the Puppet Labs git repository using `pulp-puppet-module-builder`.

```
$ cd /opt/puppet
$ pulp-puppet-module-builder --url=https://github.com/puppetlabs/puppetlabs-xinetd -o ../modules
cd /opt/puppet
git clone --recursive https://github.com/puppetlabs/puppetlabs-xinetd
cd puppetlabs-xinetd
git status
git remote show -n origin
git fetch
git fetch --tags
git pull
find . -name init.pp
puppet module build .
mkdir -p ../modules
cp ./pkg/puppetlabs-xinetd-1.2.0.tar.gz ../modules
cd ../modules
cd /opt/puppet/puppetlabs-xinetd
cd /opt/puppet
```

Listing of `/opt/puppet/modules`:

```
-rw-rw-r-- 1 demo demo  101 Jan 29 09:46 PULP_MANIFEST
-rw-rw-r-- 1 demo demo 6127 Jan 29 09:46 puppetlabs-xinetd-1.2.0.tar.gz
```

The content of PULP_MANIFEST:

```
puppetlabs-xinetd-1.2.0.tar.gz,344bfa47dc88b17d91a8b4a32ab6b8cbc12346a59e9898fce29c235eab672958,6127
```

Next synchronize the repository, which imports all of the modules into the local Pulp repository. When the directory containing the built modules is located on another host and served by http, the feed URL for the manifest may be `http://` instead of *file://*‘ in which case, the manifest and modules are downloaded into a temporary location.

```
$ pulp-admin puppet repo sync run --repo-id=puppet-builds
+----------------------------------------------------------------------+
               Synchronizing Repository [puppet-builds]
+----------------------------------------------------------------------+

This command may be exited by pressing ctrl+c without affecting the actual
operation on the server.

Downloading metadata...
[==================================================] 100%
Metadata Query: 1/1 items
... completed

Downloading new modules...
[==================================================] 100%
Module: 1/1 items
... completed

Publishing modules...
[==================================================] 100%
Module: 1/1 items
... completed

Generating repository metadata...
[\]
... completed

Publishing repository over HTTP...
... completed

Publishing repository over HTTPS...
... skipped
```

Note: The `pulp-puppet-module-builder` requires that module source layout conform to Puppet Labs standard module layout

1.7 Troubleshooting

Note: "unknown error" during a consumer install operation can be caused by several underlying problems. The unknown aspect is unfortunately the result of a limitation of Puppet's own tool. It will occasionally produce output that is not in the expected JSON format, particularly when reporting errors, and then Pulp cannot parse the output. This is a known bug in Puppet that is being worked on.

1.7.1 SSL Certificate Verification Fails for Consumer Install

Symptom

Installing a module on a consumer results in an "unknown error".

```
$ pulp-admin puppet consumer install run --consumer-id client2 -u puppetlabs/stdlib
This command may be exited via ctrl+c without affecting the request.
```

```
[||]
unknown error with module puppetlabs/stdlib

Operation executed, but no changes were made.
```

Problem

This can be caused by an SSL verification error on the client. If the repository is published over HTTPS and the `puppet module install` tool is not able to verify the server's SSL certificate against a trusted CA, the `puppet module install` tool will return an error. Unfortunately, this is one of the cases where that tool offers to return JSON output but then fails to do so, and thus Pulp is not able to parse the error message. As soon as that behavior is fixed upstream, Pulp will pass the error message through instead of reporting "unknown error".

Verification

You can verify that this is the source of the problem by running the following command on the consumer machine and looking for a similar error message about SSL. Adjust the "consumer_id" and "hostname" as appropriate.

```
$ sudo puppet module install --module_repository=http://consumer_id:.@hostname puppetlabs/stdlib
Preparing to install into /etc/puppet/modules ...
Downloading from http://consumer_id:.@hostname ...
Error: SSL_connect returned=1 errno=0 state=SSLv3 read server certificate B: certificate verify faile
Error: Try 'puppet help module install' for usage
```

Solution

Either don't publish repositories over HTTPS, or make sure the `puppet module install` tool is able to verify the server's SSL certificate with a trusted CA. Details on how to install a new trusted CA are outside the scope of this document.

1.7.2 Missing metadata.json file

If uploading a puppet module results in *MissingModuleFile* error, one possible problem is that the tar.gz file being uploaded does not contain *metadata.json* file. Another possible problem is presence of more than one directory (Puppet module) inside the archive.

Solution

Modules must adhere to the 3.6+ metadata guidlines. Also ensure that an uploaded archive contains only one Puppet module.

Incorrect Puppet module metadata

If metadata for a Puppet module in a Pulp repository doesn't match metadata in the *metadata.json* module, the tar.gz archive contains multiple Puppet modules. Ensure that an uploaded tar.gz file contains only one Puppet module.

CHAPTER 2

Technical Reference

2.1 Type

The programmatic identifier for this type is `puppet_module`.

When identifying modules on Puppet Forge or on the command line, the indentifier takes the form `author/name`. For example: `puppetlabs/stdlib`. These "author" and "name" fields are used individually as part of the unit key.

2.1.1 Unit Key

`author` Module's author, in the form of a "username" on Puppet Forge. For example, the contributor "Puppet Labs" has the username "puppetlabs".

`name` Module's name only, not including the author section. For the module identified as "puppetlabs/stdlib", this field would be "stdlib".

`version` Module's version, which according to Puppet Labs' documentation, should follow Semantic Versioning.

2.1.2 Metadata

`dependencies` List of dictionaries describing modules on which this module depends. Each dictionary has a key `name` which includes the full `author/name` notation, and a key `version_requirement` which describes what versions are acceptable to satisfy this dependency. This is an empty list if there are no dependencies. The format for this value is described in detail in Puppet Labs' own documentation.

`description` Longer description of the module.

`license` Name of the license with which the module is distributed.

`project_page` URL to a web site for the module.

`source` URL to the module's source.

`summary` Short description of the module, 1 line only.

`tag_list` List of tags assigned to this module on Puppet Forge. This is an empty list if there are no tags.

2.2 Handler Configuration

These options can be passed to a handler request.

repo_id Unique ID for a repository that should be used to fulfill an install request.

whole_repo Boolean value for an install request indicating if the entire repository should be installed. Defaults to `False`. If `True`, a `repo_id` must also be specified.

2.3 Plugin Configuration

2.3.1 Importer

Type ID: `puppet_importer`

feed URL to an existing repository that should be imported, for example `http://forge.puppetlabs.com`

The repository may be either a Puppet Forge repository or a plain directory containing a pulp manifest and packaged puppet modules. The pulp manifest is a file listing each puppet module contained in the directory. Each module is listed on a separate line which has the following format: <name>,<checksum>,<size>. The *name* is the file name. The *checksum* is SHA-256 digest of the file. The *size* is the size of the file in bytes. The Pulp manifest must be named `PULP_MANIFEST`.

Example:

Directory containing:

- PULP_MANIFEST
- module-a.tar.gz
- module-b.tar.gz
- module-c.tar.gz

The PULP_MANIFEST:

```
module-a.tar.gz,2d711642b726b04401627ca9fbac32f5c8530fb1903cc4db02258717921a4881,1763
module-b.tar.gz,5dde896887f6754c9b15bfe3a441ae4806df2fde94001311e08bf110622e0bbe,1431
module-c.tar.gz,cd2eb0837c9b4c962c22d2ff8b5441b7b45805887f051d39bf133b583baf6860,2213
```

The URL: `file://myhost/modules/PULP_MANIFEST`

queries Comma-separated list of queries that should be run against the upstream repository. Each query is used separately to retrieve a result set, and each resulting module will be imported.

remove_missing Boolean indicating whether or not previously-synced modules should be removed from the local repository if they were removed in the upstream repository. Defaults to `False`.

2.3.2 Distributor

Type ID: `puppet_distributor`

This distributor publishes a forge-like API. The user guide explains in detail how to use the `puppet module` tool to install, update, and remove modules on a puppet installation using a repository hosted by Pulp. This distributor does not support the search functionality that Puppet Forge offers, primarily because that feature is not compatible with the concept of hosting multiple repositories at one FQDN.

absolute_path Base absolute URL path where all Puppet repositories are published. Defaults to `/pulp/puppet`.

http_dir Full path to the directory where HTTP-published repositories should be created. Defaults to `/var/lib/pulp/published/puppet/http/repos`.

`https_dir` Full path to the directory where HTTPS-published repositories should be created. Defaults to `/var/lib/pulp/published/puppet/https/repos`.

`serve_http` Boolean indicating if the repository should be served over HTTP. Defaults to `True`.

`serve_https` Boolean indicating if the repository should be served over HTTPS. Defaults to `False`.

2.3.3 Install Distributor

Type ID: `puppet_install_distributor`

This distributor publishes modules by actually installing them into a given `install_path` on the Pulp server's filesystem. The use case is that you want the contents of a repository to exactly be the collection of modules installed in a puppet environment. This allows you to use Pulp's repository management features to manage which modules are installed in puppet.

This distributor performs these operations in the following order:

1. Creates a temporary directory in the parent directory of `install_path`.
2. Extracts each module in the repository to that temporary directory.
3. Deletes every directory it finds in the `install_path`.
4. Moves the content of temporary directory into the `install_path`.
5. Removes the temporary directory.

When this distributor gets removed from a repository, such as when the repository gets deleted, the `install_path` and everything in it will be deleted.

Warning: This distributor deletes everything in the `install_path`!

`install_path` This is a full path to the directory where modules should be installed. It is the user's responsibility to ensure that Pulp can write to this directory. The web server user (for example, `apache`) must be granted filesystem permissions to write to this path and the parent directory. If the directory does not exist, it will be created. Additionally, the system SELinux policy must permit Pulp to write to this directory. Pulp's SELinux policy includes a `pulp_manage_puppet` boolean that allows Pulp to write to paths that have the `puppet_etc_t` label. You must ensure that the `install_path` and its parent directory have this label applied to it. This boolean is disabled by default for safety. If you wish to enable it, you can do this:

```
$ sudo semanage boolean --modify --on pulp_manage_puppet
```

`/etc/puppet/` has the `puppet_etc_t` label by default, so if you use this or a sub directory of it as your `install_path` and you enable the `pulp_manage_puppet` boolean, SELinux will allow Pulp to write to that path.

2.3.4 File Distributor

Type ID: `puppet_file_distributor`

This distributor publishes modules by making them available in a flattened format in a single directory on the file system and served via HTTPS. The files are published to the `https_files_dir` specified in the plugin configuration. A repository is placed in a subdirectory of the '`https_files_dir` with the same name as the repository id. The base URL path where all Puppet repositories are published is `/pulp/puppet/files`.

`https_files_dir` Full path to the directory where HTTPS published file repositories will be created. Defaults to `/var/lib/pulp/published/puppet/files`.

2.4 Plugin Reports

2.4.1 Install Distributor

The publish report's `summary` value is a string describing the success or failure of the operation.

The publish report's `details` value has the following format:

```
{
  "errors": [
    [
      {
        "version": "3.2.0",
        "name": "stdlib",
        "author": "puppetlabs"
      },
      "failed to download: not found"
    ]
  ],
  "success_unit_keys": [
    {
      "version": "4.1.0",
      "name": "stdlib",
      "author": "puppetlabs"
    }
  ]
}
```

The `details` report object has two keys:

`errors` An array containing error reports. Each error report is a 2-member array: the first position is an object representing a unit key, and the second position is an error message.

`success_unit_keys` An array containing objects representing unit keys of modules that were successfully published.

2.5 Forge API

Puppet Forge implements a basic API that prior to Puppet 3.6 was not documented. After version 3.6 the documentation for the Forge API can be found at https://forgeapi.puppetlabs.com/.

The challenging aspect of re-implementing the API is that prior to puppet version 3.3, the `puppet module` tool used hard-coded absolute paths, so the API must exist at the root of the web server. This also prevents the inclusion of a repository ID in the URL. After puppet version 3.5 the use of hard-coded paths was reintroduced so the methods of specifying the repository or consumer outlined here apply.

2.5.1 Search

Pulp does not implement the search API, so using `puppet module search` against a Pulp repository will not work. This was not implemented because of the URL namespace problem.

2.5.2 Dependency

When the `puppet module` tool needs to know what the dependencies are for a particular module (such as at install time), it queries the dependency API. For a module named `puppetlabs/java`, the following request would be made against the Puppet Forge repository.

```
http://forge.puppetlabs.com/api/v1/releases.json?module=puppetlabs/java
```

Because of the URL namespace limitation described above, Pulp had to take a creative approach to identifing which repositories should be considered when determining the dependencies for a module.

Basic Auth

For puppet versions prior to 3.3, basic authentication credentials included in the URL are used to specify either a repository ID or a consumer ID. When a consumer ID is specified, all repositories to which it is bound are searched for the specified module. If a version was not specified, the repository with the newest version is then queried for dependency information.

This is an example request with a consumer ID:

```
http://consumer1:.@localhost/api/v1/releases.json?module=puppetlabs/java
```

This is an example with a repository ID and a version:

```
http://.:repo1@localhost/api/v1/releases.json?module=puppetlabs/java&version=0.2.0
```

When specifying a repository ID or a consumer ID, use a single "." in place of the other value.

Under the Hood

When a Puppet repository is published by Pulp, a small gdbm database is generated and placed at the root of the repository containing all of the data necessary to respond to dependency queries. This ensures that when dependency data is returned, it corresponds to the state of the repository at the time it was published, and does not reflect any changes made in the database since. The name of this file is `.dependency_db`, and it is not visible when accessing the repository over HTTP because Apache excludes files whose names begin with ".".

CHAPTER 3

Indices and tables

- genindex
- search