Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Katello 3.7 Installation
These instructions are for installing Katello 3.7, but the latest stable is 3.18.

Note: After installation of Katello, be sure to trust Katello's CA certificate on your system. This is required for the encrypted NoVNC connections. You will find `katello-server-ca.crt` in the `/pub` directory of your Katello server (e.g. http://katello.example.com/pub/katello-server-ca.crt).

**Important Note for Existing Installations**

Katello does not currently support installation on existing Foreman deployments. **DO NOT attempt to install Katello on an existing Foreman deployment**, unless you are a Foreman developer and willing to debug the broken configuration that will result from attempting an install on existing system.

**Hardware Requirements**

Katello may be installed onto a baremetal host or on a virtual guest. The minimum requirements are:

- Two Logical CPUs
- 8 GB of memory (12 GB highly recommended)
- The filesystem holding `/var/lib/pulp` needs to be large, but may vary depending on how many different Operating Systems you wish to synchronize:
  - Allocate 30 GB of space for each operating system. Even though an operating system may not take up this much space now, this allows space for future updates that will be synchronized later.
- The path `/var/spool/squid` is used as a temporary location for some types of repository syncs and may grow to consume 10s of GB of space before the files are migrated to `/var/lib/pulp`. You may wish to put this on the same partition as `/var/lib/pulp`.
- The filesystem holding `/var/lib/mongodb` needs at least 4 GB to install, but will vary depending on how many different Operating Systems you wish to synchronize:
  - Allocate around 40% of the capacity that has been given to the `/var/lib/pulp` filesystem
- The root filesystem needs at least 20 GB of Disk Space

**Required Ports**

The following ports need to be open to external connections:

- 80 TCP - HTTP, used for provisioning purposes
- 443 TCP - HTTPS, used for web access and api communication
- 5647 TCP - qdrouterd - used for client and Smart Proxy actions
- 9090 TCP - HTTPS - used for communication with the Smart Proxy

**Production**

Katello provides a puppet based installer for deploying production installations. Production installations are supported on the following OSes:

<table>
<thead>
<tr>
<th>OS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS 7</td>
<td>X</td>
</tr>
<tr>
<td>RHEL 7</td>
<td>X</td>
</tr>
</tbody>
</table>

Katello can only run on an x86_64 operating systems.

Installation may be done manually or via our recommended approach of using forklift.

**Required Repositories**

Select your Operating System: [Red Hat Enterprise Linux 7]
**Installation**

After setting up the appropriate repositories, update your system:

```
yum -y update
```

Then install Katello:

```
yum -y install katello
```

At this point the `foreman-installer` should be available to setup the server. The installation may be customized, to see a list of options:

```
foreman-installer --scenario katello --help
```

**Note**

Prior to running the installer, the machine should be set up with a time service such as ntpd or chrony, since several Katello features will not function well if there is minor clock skew.

These may be set as command line options or in the answer file (/etc/foreman-installer/scenarios.d/katello-answers.yaml). Now run the options:

```
foreman-installer --scenario katello <options>
```

**Forklift**

Foreman provides a git repository designed to streamline setup by setting up all the proper repositories. Forklift provides the ability to deploy a virtual machine instance via Vagrant or direct deployment on an already provisioned machine. For details on how to install using forklift, please see the README.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Smart Proxy Installation
Hardware Requirements
The Smart Proxy server is only supported on x86_64 Operating Systems

- 2 Two Logical CPUs
- 8 GB of memory
- Disk space usage is similar to that of the main Katello server

Required Ports
At a minimum, the following ports need to be open to external connections for installation:

- 80 TCP - HTTP, used for provisioning purposes
- 443 TCP - HTTPS, used for web access and api communication
- 9090 TCP - HTTPS - used for communication with the Smart Proxy

See the User Guide for additional information about Smart Proxy services and required ports.

Installation
Install needed packages:
The same yum repositories need to be configured on the Smart Proxy server as the main Katello server. See the installation guide for the list of required repositories.

Once you get the repositories configured, install the formean-proxy-content package on the Smart Proxy

```
yum install -y foreman-proxy-content python-django
```

Generate Certificates for the Smart Proxies
Prior to installing the Smart Proxy, we need to generate certificates on the main Katello server:

```
foreman-proxy-certs-generate --foreman-proxy-fqdn "myproxy.example.com"
--certs-tar "~/myproxy.example.com-certs.tar"
```

In the above example, replace ‘myproxy.example.com’ with your Smart Proxy's fully qualified domain name. This will generate a tar file containing all the needed certificates. You will need to transfer those certificates to the server that you will install your Smart Proxy on using whatever method you prefer (e.g. SCP).

The foreman-proxy-certs-generate command will output an example installation command. For example:

```
Installing Success! [100%] [........................]

To finish the installation, follow these steps:

1. Ensure that the foreman-installer-katello package is installed on the system.
2. Copy ~/myproxy.example.com-certs.tar to the system myproxy.example.com
3. Run the following commands on the Smart Proxy (possibly with the customized parameters, see foreman-installer --scenario formean-proxy-content --help and documentation for more info on setting up additional services):

```
yum -y localinstall http://katello.example.com/pub/katello-ca-consumer-latest.noarch.rpm
subscription-manager register --org "Default_Organization"
foreman-installer --scenario formean-proxy-content
  --foreman-proxy-content-parent-fqdn "katello.example.com"
  --foreman-proxy-register-in-foreman "true"
  --foreman-proxy-base-url "https://katello.example.com"
  --foreman-proxy-trusted-hosts "katello.example.com"
  --foreman-proxy-trusted-hosts "myproxy.example.com"
  --foreman-proxy-consumer-key "UVrAZfMaCfBiiWejoUVLYCZHT2xhuFV"
  --foreman-proxy-consumer-secret "ZH8p7M57tNU3WmUGWASag3JeXKgUX"
  --foreman-proxy-oauth-consumer-key "UVrAZfMaCfBiiWejoUVLYCZHT2xhuFV"
  --foreman-proxy-oauth-consumer-secret "ZH8p7M57tNU3WmUGWASag3JeXKgUX"
  --foreman-proxy-content-certs-tar "~/myproxy.example.com-certs.tar"
```

The full log is at /var/log/foreman-proxy-certs-generate.log

Install Smart Proxy
Use the provide installation command from formean-proxy-certs-generate, and tailor for your own purposes as needed. The defaults will give you a Smart Proxy ready for Content-related services.

See the User Guide to learn about setting up provisioning related services, as well as the Foreman manual
Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Client Installation
Client machines can be added in one of two ways: manually or via a provisioned host.

**Manual**

Install the appropriate Katello client release packages. For CentOS 6, you will also need to enable the COPR repository for subscription-manager.

Select your Operating System: **Enterprise Linux 5 (CentOS, etc.)**

```bash
yum install -y http://fedorapeople.org/groups/katello/releases/yum/3.7/client/el5/x86_64/katello-client-repos-latest.rpm
```

Now you are ready to install the client package:

The **katello-host-tools** package reports errata & package profile information, but does not allow you to run remote actions on the clients.

```bash
yum install katello-host-tools
```

We generally recommend using Foreman Remote Execution or Ansible for remote actions, but we also offer a messaging bus based client that does have some limitations when used with a large number of clients.

```bash
yum install katello-agent
```

Optionally you can also install **katello-host-tools-tracer** and the client will report processes that need restarting after an update back to the Katello server.

```bash
yum install katello-host-tools-tracer
```

For Suse Clients, only katello-host-tools is supported:

```bash
zypper install katello-host-tools
```

**Provisioned**

In order to install the katello-agent package on a host you are provisioning, you will need to make the appropriate client repository available within your Katello. The first step is to either create a new product or add to an existing product, the appropriate client repository from the dropdown in the **manual** section above. After you create the new repositories, they will need to be synced locally. Next, you will then need to add them to the relevant content view(s) for the hosts you are wanting to provision. At this point, a new version of the content view can be published and promoted to the appropriate environments that you are wanting to provision a host into. At this point, you can go provision a host and the host will install the katello-agent package during setup.

When provisioning new clients that should use Puppet 4, set a parameter called ‘enable-puppet4’ to ‘true’, so the templates know which package to install and where to place the configuration. This parameter can be placed at the host, host group, or another appropriate level of the hierarchy.
Katello 3.7 supports upgrades from version 3.6 and Puppet version 4. For users transitioning from versions prior to 3.6, first upgrade to Katello 3.6, upgrade Puppet to version 4, and then proceed to upgrade to Katello 3.7 following the instructions below.

**Pre-upgrade considerations**

Before you upgrade, you need to run the upgrade check script that will check for any active tasks, your version of Katello, and if there are any content hosts that will be deleted (see below).

To run the script:

```
foreman-rake katello:upgrade_check
```

**Step 1 - Backup**

If Katello is running on a Virtual Machine, we recommend to take a snapshot prior to upgrading. Otherwise, take a backup of the relevant databases by following the instructions here.

**Step 2 - Operating System**

Ensure your operating system is fully up-to-date:

```
yum -y update
```

**NOTE:** If kernel packages are updated here (e.g. upgrading el 6.6 to 6.7), you must reboot and ensure the new kernel and SELinux policy is loaded before upgrading Katello.

**Step 3 - Repositories**

Update the Foreman and Katello release packages:

- RHEL7 / CentOS 7:

  ```
yum update -y http://fedorapeople.org/groups/katello/releases/yum/3.7/katello/el7/x86_64/katello-repos-latest.rpm
  yum update -y http://yum.theforeman.org/releases/1.18/el7/x86_64/foreman-release.rpm
  yum update -y foreman-release-scl
  ```

**Step 4 - Update Packages**

Stop the Katello services

```
katello-service stop
```

Clean the yum cache

```
yum clean all
```

Update the required packages:

```
yum -y update
```

**Step 5 - Run Installer**

The installer with the --upgrade flag will run the right database migrations for all component services, as well as adjusting the configuration to reflect what's new in Katello 3.7

```
foreman-installer --scenario katello --upgrade
```

Congratulations! You have now successfully upgraded your Katello to 3.7 For a rundown of what was added, please see the release notes.

If for any reason, the above steps failed, please review /var/log/foreman-installer/katello.log – if any of the “Upgrade step” tasks failed, you may try to run them manually below to aid in troubleshooting.
Foreman 2.3.3 has been released! Follow the quick start to install it.

Foreman 2.2.2 has been released! Follow the quick start to install it.
Smart Proxy Upgrade
Step 1 - Operating System

Ensure your operating system is fully up-to-date:

```
yum -y update
```

**NOTE:** If kernel packages are updated here (e.g. upgrading el 6.6 to 6.7), you must reboot and ensure the new kernel and SELinux policy is loaded before upgrading Katello.

Step 2 - Repositories

Update the Foreman and Katello release packages:

- RHEL7 / CentOS 7:

```
yum update -y http://fedorapeople.org/groups/katello/releases/yum/3.7/katello/el7/x86_64/katello-repos-latest.rpm
yum update -y http://yum.theforeman.org/releases/1.18/el7/x86_64/foreman-release.rpm
```

Step 3 - Update Packages

Clean the yum cache

```
yum clean all
```

Update packages:

```
yum update -y
```

```
yum install foreman-proxy-content
```

Step 4 - Regenerate Certificates

On the Katello server, regenerate the certificates tarball for your Smart Proxy:

```
foreman-proxy-certs-generate --foreman-proxy-fqdn "myproxy.example.com" --certs-update-all --certs-tar "~/myproxy.example.com-certs.tar"
```

And copy them to your Smart Proxy:

```
scp ~/myproxy.example.com-certs.tar myproxy.example.com:
```

Step 5 - Run Installer

The installer with the --upgrade flag will run the right database migrations for all component services, as well as adjusting the configuration to reflect what’s new in Katello 3.7

```
foreman-installer --scenario foreman-proxy-content --upgrade --foreman-proxy-content-certs-tar ~/myproxy.example.com-certs.tar --certs-update-all --certs-regenerate true --certs-deploy true
```

Congratulations! You have now successfully upgraded your Smart Proxy to 3.7. For a rundown of what was added, please see release notes.

If for any reason, the above steps failed, please review /var/log/foreman-installer/foreman-proxy.log – if any of the “Upgrade step” tasks failed, you may try to run them manually below to aid in troubleshooting.
Foreman 2.3.3 has been released! Follow the quick start to install it.

Foreman 2.2.2 has been released! Follow the quick start to install it.

View our Privacy Policy

This web site is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. Source available: github/theforeman/theforeman.org

Thanks to our sponsors for hosting this website and our project services.
Client Upgrade
When upgrading clients there are 2 scenarios: manually added clients and provisioned clients.

**Step 1 - Update Repositories**

**Manually Added Clients**

Update the Katello client release packages:

Select your Operating System: [Enterprise Linux 5 (RHEL, CentOS, etc.)]

```
yum update -y http://fedorapeople.org/groups/katello/releases/yum/3.7/client/el5/x86_64/katello-client-repos-latest.rpm
```

**Provisioned Clients**

If the katello-agent was setup during provisioning from a locally synced repository then you will need to go through some **initial setup** to add the 3.7 client repositories to your Katello for each version needed. After you create the new repositories, they will then need to be added to the relevant content view(s) and the older versions removed. At this point, a new version of the content view can be published and promoted to the appropriate environments. Once the new package is available the clients can be updated following the next steps.

**Step 2: Update Packages**

Clean the yum cache

```
yum clean all
```

Update packages:

```
yum update katello-agent
```

---

**Foreman 2.3.3 has been released!** Follow the quick start to install it.

**Foreman 2.2.2 has been released!** Follow the quick start to install it.

---

View our Privacy Policy

This web site is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. Source available: [github/theforeman/theforeman.org](https://github/theforeman/theforeman.org)

Thanks to our sponsors for hosting this website and our project services.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Katello 3.7 (Blonde Ale) Release Notes
For the full release notes, see the Changelog.

Features

New Red Hat Repositories & Subscriptions Pages
Completely redesigned with an eye toward usability and functionality.

New Upstream Subscriptions API
For users with Red Hat content we now provide an API for managing subscriptions against existing manifests. Check the API documentation for details.

Content Views
Composite content views can be configured to auto-publish when one of its components is changed.

Client Tooling
Zypper and DNF plugins have been added to support errata applicability similar to Yum clients

Backup and Restore Utilities
We are beginning to adopt Foreman Maintain for Katello. Our first endeavor has been to utilize it for backup and restore capabilities.

The primary goals of this change are to:

- Centralize system operations in a single tool
- Support remote databases
- Support new Mongo versions (3.4)

*katello-backup* and *katello-restore* are completely removed so please be sure to update any cron jobs or scripts using them.

Contributors

- agrawal
- Adam Ruzicka
- Andrew Kofink
- Bernhard Suttner
- Bryan Kearney
- Chris Duryee
- Chris Hale
- Christine Fouant
- Daniel Lobato Garcia
- Dan Seethaler
- Dmitri Dolguikh
- Eric D. Helms
- Evgeni Golov
- Ido Kanner
- Ivan Nečas
- John Mitsch
- Jonathon Turel
- Justin Sherrill
- kgaikwad
- Idjebran
- Lukas Zapletal
- Marek Hulan
- Masatake YAMATO
- Matthias Dellweg
- Michael Johnson
- Michael Moll
- Ondrej Prazak
- Partha Aji
- Paul Donohue
- Quirin Pamp
- Ranjan Kumar
Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests
What is the CLI?

The Hammer CLI provides users with a command-line interface for interacting with Katello. It's our goal to make all functionality that's accessible through Katello's Web UI also available through Hammer so that users may use Hammer for their entire Katello workflow.

Installation

The first step to install the CLI is to setup the appropriate repositories: foreman, katello and epel.

Select your Operating System: Red Hat Enterprise Linux 6

```
yum -y --disablerepo=* --enablerepo=rhel-6-server-rpms install yum-utils wget
yum-config-manager --disable ***
yum-config-manager --enable rhel-6-server-rpms epel
yum-config-manager --enable rhel-6-server-optional-rpms
```

After setting up the appropriate repositories, install Katello:

```
yum -y install tfm-rubygem-hammer_cli_katello
```

How do I use Hammer?

To get started with hammer, view the help:

```
hammer -u <user> -p <password> --help
```

How do I contribute to Hammer?

See the Katello Hammer CLI project if you want to get setup for contributing to the hammer code.

---

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.

---

View our Privacy Policy

This web site is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. Source available: github/theforeman/theforeman.org

Thanks to our sponsors for hosting this website and our project services.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. API Documentation
8. Annotated Backend Requests

Activation Keys
Activation Keys provide a mechanism to define properties that may be applied to Content Hosts during registration. This includes properties such as:

- Lifecycle Environment
- Content View
- Subscriptions
- Repository Enablement
- Host Collections

Definitions

- **Content Host**
- Host Collection - A statically defined group of Content Hosts.
- **Subscription** - The right to receive the associated content from Katello.

General Features

The following is a high-level summary of the Activation Key features:

- Create an Activation Key
- Add subscriptions to an Activation Key
- Change repository enablement for an Activation Key
- Add Host Collections to an Activation Key
- Register a Content Host using an Activation Key

Create an Activation Key

To create a new key,

- navigate to: Content > Activation Keys
- click **New Activation Key**

- **Name**: This required option is used to identify the activation key to command line tools, such as `subscription-manager`.

- **Content Host Limit**: This option will control how many Content Hosts may be registered using the key. An “unlimited” value will not place any limits on usage. Specifying a quantity will limit the number of registered content hosts.
Registering with an activation key consumes one of the available limit quantity, while unregistering makes it available again. (i.e. this quantity is not a usage counter but a limit of actively registered content hosts)

- **Description:** A free form text field that can be used to store a description of the key for later reference or for pseudo-tagging that can be used to search.
- **Environment** and **Content View:** Although optional, at least one activation key used during registration must specify a content view. Activation keys are used in the order specified to `subscription-manager` meaning the last activation key with a content view takes precedence.

The following example would use CV_B's content view:

```
subscription-manager register --org Default_Organization --activationkey NO_CV --activationkey CV_A --activationkey CV_B
```

Or equivalently:

```
subscription-manager register --org Default_Organization --activationkey NO_CV,CV_A,CV_B
```

For registration to succeed, at least one activation key must be successfully applied. For an activation key to succeed, at least one of the listed subscriptions must be successfully attached to the registering content host.

### Add Subscriptions to an Activation Key

To add subscriptions to a key:

- navigate to: Content > Activation Keys
- select the desired key from the list
- click **Subscriptions**
- click **Add**
- select the Subscriptions you would like to add
- click **Add Selected**

The **Auto-Attach** setting controls how the group of subscriptions are processed during registration.

When 'Auto-Attach' is enabled but no subscriptions are added to the activation key, subscriptions will be automatically added to cover the installed products. This is equivalent to passing the `--auto-attach` flag to the `subscription-manager` command:

```
subscription-manager register --org=Default_Organization --auto-attach
```

When 'Auto-Attach' is enabled and subscriptions are listed for the activation key, two things will happen. First all subscriptions for custom products will be attached to the registering content host. Second, the group of Red Hat subscriptions will be attached as needed to cover the content host's installed Red Hat products. This is most commonly used when there is a group of similar subscriptions (e.g. several Red Hat Enterprise Linux from different contracts, or guest subscriptions from different hypervisors) and which one used is not important. Katello's subscription tooling, **Candlepin**, will automatically choose the minimal proper subscriptions from the group.

Finally, when 'Auto-Attach' is disabled, all subscriptions on the activation key will attached to the registering content host, regardless of whether needed to cover an installed product or not. For example, adding an OpenStack Platform subscription would then allow that product to be installed after registration.
To change repository enablement settings using a key:

- **navigate to**: Content > Activation Keys
- **select** the desired key from the list
- **click** Product Content
- **click** the edit icon for the repository content set that you would like to modify
- **select** the desired value (e.g. 'Override to Yes', 'Override to No', 'Defaults to Yes', 'Defaults to No')
- **click** Save

**View current settings:**
Change current settings:

Add Host Collections to an Activation Key
To add Host Collections to a key:

- navigate to: Content > Activation Keys
- select the desired key from the list
- click Host Collections
- click Add
- select the Host Collections you would like to add
- click Add Selected

Register a Content Host using an Activation Key

The simplest form of registering a content host with an activation key is this:

```
subscription-manager register --org=Default_Organization --activationkey=$KEY_NAME
```

Click here for more information

Note that modifying an activation key does not change anything on content hosts previously registered with the key.

Foreman 2.3.3 has been released! Follow the quick start to install it.

Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Backup
Please use Foreman Maintain for backup and restore functionality. Foreman-maintain is a dependency of Katello starting in 3.7.

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. API Documentation
8. Annotated Backend Requests

Smart Proxies
What are Smart Proxies?

The Smart Proxy server is a Katello component that provides federated services to discover, provision, control, and configure hosts. Each Katello server includes a Default Smart Proxy, and you may deploy additional Smart Proxies to remote data centers. A Smart Proxy server provides the following features:

- Content features, including:
  - Repository synchronization
  - Content delivery
  - Host action delivery (package installation updates, etc)
  - Subscription management proxy (RHSM)
- Foreman Smart Proxy features, including:
  - DHCP, including ISC DHCP servers
  - DNS, including Bind and MS DNS servers
  - Realm, including FreeIPA
  - Any UNIX-based TFTP server
  - Puppet Master servers
  - Puppet CA to manage certificate signing and cleaning
  - Baseboard Management Controller (BMC) for power management
  - Provisioning template proxy

The Katello Smart Proxy server is a means to scale out the Katello installation. Organizations can create various Smart Proxies in different geographical locations. These are centrally managed through the Katello server. When a Katello user promotes content to a particular environment, the Katello server will push the content to each of the Smart Proxy servers subscribed to that environment. Hosts pull content and configuration from the Katello Smart Proxy servers in their location and not from the central server.

In a fully configured Smart Proxy, communication is completely isolated between hosts and the Katello server.

What is a Foreman Proxy with Content?

A Katello Smart Proxy is a Foreman Smart Proxy with the addition of content-related services.

Deployment

In the simplest use case, a user may only want to use the Default Smart Proxy. Larger deployments would have a single Katello server with multiple Smart Proxies attached, with these remote Smart Proxies deployed to various datacenters. Smart Proxies can also be used to scale the number of hosts attached to a single Katello server.

Installation

See Smart Proxy Installation

Removal

To stop all services and remove all Katello and Foreman related packages, run the following command as root on the Smart Proxy:

```
katello-remove
```

Smart Proxy Isolation

The goal of Smart Proxy Isolation is to provide a single endpoint for all of a client’s communication, so that in remote network segments, you need only open Firewall ports to the Smart Proxy itself. The following section details the communication clients need to have with a Smart Proxy. The installation options mentioned are the default starting with Katello 2.2.
Content and Configuration Services

There are five primary areas that require client communication:

1 - Content Delivery

That is, yum. Katello Smart Proxies by default have the Pulp feature, which mirrors content for the selected Lifecycle Environments.

Install Option:

- `--pulp=true`

Required Connectivity:

- Clients need to be able to communicate with the Smart Proxy on port 443/tcp.
2 - Katello Agent

The Katello agent is a goferd plugin which allows you to schedule remote actions on hosts such as package installation, updates, etc. A Smart Proxy must be running the Qpid Dispatch Router service for this feature to work.

Install Option:
- `--qpid-router=true`

Required Connectivity:
- Clients need to be able to communicate with the Smart Proxy on port 5647/tcp

3 - Puppet & Puppet CA

By default, the Puppet CA feature on the Smart Proxy is an independent CA which will manage the certificates for all the clients registered against the Smart Proxy. Simply select the Puppetmaster and Puppet CA to be the Smart Proxy when creating a host.

Install Option:
- `--puppet=true` `--puppetca=true`

Required Connectivity:
- Clients need to communicate with the Smart Proxy on port 8140/tcp

4 - Subscription Management

Content Hosts utilize Subscription Manager for registration to Katello and enabling/disabling specific repositories.

Install Option:
- `--reverse-proxy=true`

Required Connectivity:
- Clients need to talk to the Smart Proxy on port 8443/tcp

5 - Provisioning Services

When provisioning a host using DHCP/PXE, you will need, at a minimum, the TFTP feature enabled on the Smart Proxy, and a DHCP server available. While not required, the Smart Proxy can provide the DHCP service. In order for the installer to obtain its kickstart template from the Smart Proxy, you should enable the templates feature.

If a TFTP proxy has the Templates feature as well, Foreman will automatically make the communication isolated. Your clients need to talk to the Smart Proxy on port 67/udp and 68/udp for DHCP, 69/udp for TFTP, and 8000/tcp for Templates.

Consult the installer’s `--help` for the full range of provisioning options.

Foreman 2.3.3 has been released! Follow the quick start to install it.

Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Content
Katello can currently host two different types of content, RPMs and puppet modules. RPMs and Puppet Modules can be synced from an external resource or can be uploaded directly.

The advantages to using Katello to mirror your local content are:

- Reduce bandwidth usage and increase download speed by having client machines pull updates from Katello
- Provision hosts using local Repositories covered HERE TODO
- Customize content locally, covered HERE TODO

Definitions

- Repository - Collection of content (either RPM or puppet)
- Product - Collection of Repositories, Content Hosts subscribe to a product
- Library - A special pre-created Lifecycle Environment where Repositories are created and content is synced or uploaded to. A Content Host can subscribe to library and receive content as soon as the content is synced or uploaded.

Creating a Product

From the web UI, navigate to:

Content > Products > New Product (top right)

Creating a Repository

From the web UI, navigate to:

Content > Products > Select desired product > Create Repository (right hand side)

Note the following options:

- Publish via HTTP: allows access to the Repository without any restriction. Unless you desire to restrict access to your content in this Repository, we recommended to leave this checked.
- URL: If you are syncing from an external Repository (yum or puppet), this would be filled in. This can be changed, added, or removed later. For example if you are wanting to create a mirror of EPEL, you would set this to 'http://dl.fedoraproject.org/pub/epel/6/x86_64/.'
Syncing a Repository

From the web UI, navigate to:

Content > Products > Select desired product > Select the Repository > Sync Now
The progress will be displayed:

Syncing multiple repositories
To easily sync multiple repositories at once and track their progress, navigate to:

Content > Sync Status

From here you can expand the desired products, and select multiple repositories to sync.

Uploading RPM Content

Uploading RPM content directly is not currently supported. You will need to build a custom yum Repository. TODO Provide instructions on creating a custom yum repo

Uploading Puppet Content

To upload puppet modules, first create a Repository with type puppet (similarly to creating a yum Repository above):
When creating this Repository the URL field can be left blank.

Puppet modules can be uploaded via the Web UI, navigate to:

Content > Repositories > Products > Select desired Product > Select desired Puppet Repository > Select file on the right
Subscribing a System to a Product for yum content
To read about registering systems and subscribing them to the Product click TODO.

Scheduling Repository Synchronization
Sync plans give you the ability to schedule Repository synchronization on a hourly, daily or weekly basis. Sync Plans are applied to Products and thus all Repositories within a Product will be synchronized according to the products plan.

Creating a Sync Plan
If you would like to schedule certain repositories to sync on a hourly, daily or weekly basis, Sync Plans give you this capability.
To create a Sync Plan, navigate to:
Content > Sync Plans > click “New Sync Plan” on the upper right
The Start Date and Start Time fields are used as the day of the week/month and time of the day to run the re-occuring syncs.
For example a sync plan that starts on Sunday 2014-04-06 at 2:30 will occur every Sunday at 2:30 every week if it has a weekly interval. If on a daily interval it would sync every day at 2:30.

Assigning a Sync Plan to a Product
Navigate to:
Content > Sync Plans > Select your Sync Plan > Products > Add
Then select the Products you want to add and click “Add Selected” in the upper right.
GPG Keys

To learn about securing your packages and Repositories with GPG Keys click here. TODO

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.

View our Privacy Policy

This web site is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. Source available: github/theforeman/theforeman.org

Thanks to our sponsors for hosting this website and our project services.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Content Hosts
What is a Host?
A Host is a Foreman concept that represents a server/host/system/computer (whatever you want to call it). In addition to holding facts about the system, it:

- Stores which operating system the system should be running
- Stores which puppet classes should be assigned
- Stores which parameters apply to which puppet classes
- Allows you to re-provision the machine

What are Content Hosts?
Content Hosts are the part of a host that manages Content and Subscription related tasks. As time goes on more and more of this functionality will be moved to the Host object. A Host's Content Host:

- Stores which Products are assigned (i.e. which Repositories will the system pull content from)
- Initiates package install/upgrade/removal
- Determines which errata apply to a system
- Initiates errata installation

How is a Content Host registered?
Subscription Manager is the client for Katello that handles registration.

Installing Subscription Manager
Depending on your Operating System, for:

- RHEL, subscription-manager is installed by default
- Fedora, subscription-manager is available from the Everything repo for its release: `yum install subscription-manager`
- CentOS 7, subscription-manager is available in the ‘os’ repo for its release: `yum install subscription-manager`
- CentOS 5/6, enable the upstream subscription-manager repo and then install subscription-manager (be sure to change ‘6’ to ‘5’ if you're on ELS, as the version from 6 will not work):

```
yum install subscription-manager -y
```

Registering with Subscription Manager
First install the bootstrap rpm from your Katello server:

```
rpm -Uvh http://$KATELLO_HOSTNAME/pub/katello-ca-consumer-latest.noarch.rpm
```

Then register:

```
subscription-manager register --org=Default_Organization --environment=Library
```

Subscription manager will prompt for your username and password. You can also specify `--username $USER --password $PASS` on the command line.

Registering to a Content View
To register to Content View “MyView” in a “Devel” Lifecycle Environment:

```
subscription-manager register --org=Default_Organization --environment=Devel/MyView
```

Registering without using a username and password
Activation Keys allow you to register and consume content without using a username and password. To create an Activation Key see the Activation Key Guide

Once you have created an activation key, register with:

```
subscription-manager register --org=Default_Organization --activationkey=$KEY_NAME
```
Actions with registered Content Hosts
To see the list of your Content Hosts, navigate to Hosts > Content Hosts

Changing the Lifecycle Environment and Content View of a Content Host:
Navigate to the Content Host Details page, Host > Content Hosts > Click the name of the desired Content Host
Look in the upper right corner for the “Content Host Content”:
Then select the new Lifecycle Environment you desire, select the new Content View you desire, and click save.

Assigning a Content Host to a Product
In order for a Content Host to receive package updates and access Repositories hosted on Katello, it needs to be subscribed to a product.
Navigate to Hosts > Content Hosts > Select Content Host > Subscriptions > Click the “Add” tab
Check the checkbox under the Products you want to add and select “Add Selected” in the upper right.
To see existing attached Products, click the “List/Remove” tab. To remove a Product, select the checkbox under the desired Product in this list and click “Remove Selected”.

Package Management
To perform package actions on a single Content Host, navigate to: Hosts > Content Hosts > Select Content Host > Packages
From here you can:
- See a list of installed packages
- Perform a yum install/update/remove of a Package or Package Group
- Update all packages (equivalent of running ‘yum update’)

View and Install Applicable Errata
If your synced Repositories contain Errata, you can use Katello’s Errata management to track and install Errata.
Navigate to: Hosts > Content Hosts > Select Content Host > Errata
To apply errata, search for the errata you want and select the checkbox beside each errata. Then click “Apply Selected” at the top right.
The “Show From” filters what applicable errata to show:
- Current Environment - Shows only Applicable Errata available in the Host’s Content View & Lifecycle Environment.
- Previous Environment - Shows Applicable Errata that are available from the Host’s Content View but in the previous Lifecycle Environment. Promoting the Content View Version from that previous Lifecycle Environment to the current Lifecycle Environment for this Host would cause all Applicable Errata shown to then be available.
- Library Synced Content - Shows Applicable Errata which have been synced to the Library. This shows you what is applicable even when the Errata have not been published into a Content View. All applicable Errata are shown regardless of availability to the Content Host.
Change Host Collection Assignments

To change Host Collection assignments for a Content Host, navigate to: Hosts > Content Hosts > Select Content Host > Host Collections

Bulk Actions

Katello provides the ability to perform actions on many Content Hosts at once such as:

- Package installation/upgrade/removal
- Listing and applying applicable errata
- Assigning Host Collections
- Changing Lifecycle Environment and Content View assignments

In order to use the bulk actions, perform whatever search you desire and select which Content Hosts you want to modify. If you want to select all Content Hosts from a search result, click the ‘checkbox’ above the table:

This will select all Content Hosts on that page (only the ones that are visible). To select all that correspond to that search query, notice a bar has now appeared:

Next select the ‘Bulk Actions’ button in the top right.

From here you can select the tab corresponding to any action you wish to perform.

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests
What can a Content View be used for?

- To stage content through environments (Dev, Test, Production).
- To filter the contents of a repository (include a package or exclude certain errata, for example).
- To have multiple snapshots of the same repository and/or puppet modules.

Definitions

- Content View - snapshot of one or more repositories and/or puppet modules.
- Composite Content View - a Content View that contains a collection of other Content Views.
- Filter - provides finer grained control over content in a Content View. Can be used to include or exclude specific packages, package groups, or errata.
- Publishing - Content Views are 'published' in order to lock their contents in place. The content of the Content View is cloned and all filters applied. Publishing creates a new version of the Content View.
- Promoting - Content Views can be cloned to different Lifecycle Environments (Dev, Test, Production).

General Workflow

First create a product and repository in the library environment and populate the repository with content (by syncing it or uploading content). A Content Host can now register directly to library and be attached to the content therein. Updates will be available as soon as new content is synced or uploaded.

To utilize Content Views for filtering and snapshotting:

1. Create a Content View
2. Add the desired repository and/or puppet modules to the Content View
3. Optionally create one or more Filters to fine tune the content of the Content View.
4. Publish the Content View
5. Attach the Content Host to the Content View
6. Optionally promote the Content View to another environment

At this point the Content Host will no longer be getting content directly from Library, but from the Content View. Updates to library will not affect this Content Host.

Note that all of the actions below can also done with hammer, the CLI tool, and examples are given at the end of each section.

Creating a Content View

To create a Content View using the web UI, navigate to:

Content > Content Views

Click the Create New View button on the top right of the screen.
Creating a Composite Content View

To create a Composite Content View using the web UI follow the above steps for Creating a Content View but check the “Composite View?” checkbox.

From the CLI:

```
hammer content-view create
  --organization="Default Organization"
  --name="New Composite Content View"
  --description="This is my new composite content view."
```

Adding Repositories

Adding a repository to a Content View means whenever a Content View is published, all of the content contained within the repository at that time is included in the Content View. If the repository is synced after publishing the Content View, the Content View will contain the state of the repository prior to syncing. A new version of the Content View must be published in order for the new version to get the contents of the newly synced repository.

To add a repository using the web UI, navigate to:

Content > Content Views > Select the desired Content View > Content (within sub navigation) > Repositories

From the CLI, adding a repository:

```
hammer content-view add-repository
  --organization="Default Organization"
  --name="New Content View"
  --repository="CentOS 6.5"
```

Adding a Puppet Module

Adding a puppet module to a Content View means that whenever the Content View is published the puppet module is locked to the version selected. If the “Use Latest” version is selected then the puppet module will be “frozen” at the latest version available when the Content View is published. A new version of the Content View must be published in order for the new
version to get any updated puppet module.

To add a puppet module using the web UI, navigate to:

Content > Content Views > Select the desired Content View > Puppet Modules (within sub navigation)
From the CLI, first find the UUID of your puppet module from the list:

```bash
hammer puppet-module list --organization="Default Organization" --repository "Puppet Modules"
```

Then add the puppet module:

```bash
hammer content-view puppet-module add --organization="Default Organization" --uuid=91cc9bb7-dbb3-4798-b50a-45173b763cbb
```

**Adding Content Views to a Composite Content View**

Adding a version of a Content View to a Composite Content View means whenever the Composite Content View is published, all of the content contained within the specific version of that Content View is contained in the Composite Content View. If the Content Views contained within the Composite Content View are updated (i.e. a new version is published) or if their content is
updated after publishing the Composite Content View, the Composite Content View will only contain the versions of the Content View(s) prior to syncing. A new version of the Composite Content View must be published in order for it to get the updated Content Views.

To add a Content View to a Composite Content View using the web UI, navigate to:

Content > Content Views > Select the desired Content View > Content (within sub navigation) > Repositories

Find the Content View ID of the specific version of the Content View to add:

```
hammer content-view version list \
  --organization="Default Organization" \
  --content-view="New Content View"
```

From the CLI, add a Content View to a composite Content View:

```
hammer content-view update \
  --organization="Default Organization" \
  --content-view="New Composite Content View" \
  --component-ids=2
```

Creating a filter

If only using Content Views as snapshots, Filters are unnecessary. If the desire is to filter what content make it into the view, such as blacklisting a package by name or version, or blacklisting errata by date or type, Filters can help accomplish these tasks.

To create a new Content View Filter using the web UI, navigate to:

Content > Content Views > Select the desired Content View > Content (within sub navigation) > Filters > New Filter
From the CLI, adding a Content View Filter:

```
hammer content-view filter create \\
  --organization="Default Organization" \\
  --content-view="New Content View" \\
  --name="New Filter" \\
  --inclusion=false \\
  --type=rpm
```

From the CLI, adding a Content View Filter rule:

```
hammer content-view filter rule create \\
  --organization="Default Organization" \\
  --content-view="New Content View" \\
  --content-view-filter="New Filter" \\
  --name="something-else" \\
  --max-version="10.0.0" \\
  --min-version="10.0.0"
```

Selecting whichRepositories to Filter
By default a Filter applies to all repositories (present and future) in the Content View. It's possible to select which repositories within the Content View apply to the filter. This is useful, for example, if the desire is to exclude errata from only certain repositories in a view.

To select which repositories to Filter in the web UI, navigate to:

Content > Content Views > Select the desired Content View > Content (within sub navigation) > Filters > Select the desired Filter > Affected repositories (within sub navigation)

From the CLI, adding a Content View Filter:

```
hammer content-view filter update \
    --organization="Default Organization" \
    --name="New Filter" \
    --repository-ids=2,3,7
```

Publishing a Content View

Publishing a Content View produces a new version of the content view that is subsequently promoted to the Library lifecycle environment. This newly published version of the content view is now available to any content host registered to Library.

To publish a Content View, in the web UI, navigate to:

Content > Content Views > Select the desired Content View > Publish New Version
From the CLI:

```
hammer content-view publish
   --organization="Default Organization" \
   --name="New Content View"
```

Registering a Content Host

To register a Content Host that is not currently registered to the Content View, simply use subscription manager on the client Content Host and run:

```
subscription-manager register --org=ACME_Corporation --environment=Library/my_rhel_view
```

This would register the Content Host to the Library environment and the my_rhel_view Content View.

If the Content Host is already registered, from the UI:

Hosts > Content Hosts > Select the desired Content Host

From the CLI:

```
hammer content-host update \
   --organization="Default Organization" \
   --name="dhcp129-211.rdu.redhat.com" \
   --content-view="New Content View" \
   --lifecycle-environment="Library"
```

Promoting a Content View

Initially a Content View is published to Library as version 1. If there are Content Hosts in other environments that would like to consume this Content View, a version of the content view will need to be promoted to those environments. For example, given the Content View “New Content View”, version 1 of which has been promoted to the Dev environment. Any Content Hosts in Dev attached to the Content View would remain at version 1 until a version 2 is both published and promoted to the Dev environment.

To promote a Content View in the Web UI, navigate to:

Content > Content Views > Select the desired Content View > Versions (within sub navigation) > Click promote for desired version
New Content View

Version | Status | Environments | Content | Author
---|---|---|---|---
Version 1 | Published | Library | 487 Packages | 112 Errors | 44 | 305

Promote Version 1

Choose one or more lifecycle environments from the existing promotion paths available in "Library".
To promote a Content View in the CLI:

```
hammer content-view version promote
   --organization="Default Organization" \
   --content-view="New Content View" \
   --to-lifecycle-environment="Test" \
   --version 1
```

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Katello Inter-Server Sync
NOTE: This feature is intended to replace the 'katello-disconnected' script.

Intro

If you are working in an air-gapped network environment where some of your Katello servers do not have Internet connectivity, you may be interested in using the Katello Inter-Server Sync (ISS) feature. This allows you to export repos, including repos in content views, on your “upstream” Katello server, and then import said repos into your “downstream” server that does not have connectivity. Individual repos can be exported, or all of the repos in a content view.

List of currently supported repo content types:

- yum

Future releases will enable support for additional content types.

The diagram above shows an example scenario where a user wants to export all Yum content in a content view and then import to another Katello server. The ISO file is burned to media and then walked across the air-gap in the network.

Detailed Operation

Exporting

The ISS feature allows users to move Yum content from one Katello server to another, in a way that is compatible with air-
gapped networks. Typically users will set up an upstream server that is connected to the Internet, and then create a content view that contains Yum content that they would like to present to the downstream server (step 1).

Content is exported via either hammer repository export or hammer content-view version export (step 2). It is exported to the location set in “pulp_export_destination” in the Settings page, under the Katello tab. This defaults to /var/lib/pulp/katello_export. Please be aware that the location needs to be readable and writable by the foreman user. SELinux permissions also need to be set on the export location with the type httpd_sys_rw_content_t as well as foreman user and group ownership.

You can select to either export as a plain set of directories, or as a set of ISO files. The “iso_size_mb” parameter sets how large you would like each ISO file to be. It defaults to 4380 MB, which is the size of a single-side, single-layer DVD.

Importing

Importing (step 3) can be done in one of two ways. The first way is to make the export available via HTTP to the importing Katello instance. Simply put the export in /var/www/html/pub/export, either via copy or symlink. After that, edit your CDN location from the manifest import page to point to “http://export/path/to/export” and the Red Hat Repos page will then work as expected, using your exported data. Please be sure to use 'http' and not 'https' when altering the CDN url. Katello by default only supports the CA certificate for `cdn.redhat.com`. This is a [known limitation](http://projects.theforeman.org/issues/16392) that will be addressed in a future version.

The second way is to perform a repository sync via hammer, specifying the source location. Please see the hammer repository sync command for more information. This method is the only way to import custom content, and is the only way to import incremental content.

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Docker Management
Katello can be used to manage and deploy Docker content. Katello can retrieve Docker content from a variety of sources such as Docker hub, private Docker registries, the Red Hat CDN, and so forth. Docker content can then be published and promoted via Content Views and then pulled or provisioned to a server running Docker.

What is Docker?
Docker is a tool used to manage Linux containers. To read more about Docker, check out the official Docker site. Docker repositories, which contain images and tags, can be retrieved, stored, managed, and deployed from Katello.

How to sync a Docker repository
The easiest way to get Docker content into Katello is to sync it in. You can either sync Docker content from the Red Hat CDN (if you have subscriptions for the content) or from a registry such as Docker Hub.

Red Hat Docker Images
Content can be synced into Katello using a Red Hat manifest in much the same way as yum content. See our guide on how to manage Red Hat content for more information.

Docker Hub/Docker Registry
To sync content from a Docker registry such as Docker Hub (which is the official Docker-run registry), simply start by creating a new Repository.

On the new Repository screen, select “Docker” as the content type. Once you do that, you'll be given two options: upstream name and URL. The URL will be the registry URL; for Docker Hub, this would be https://registry.hub.docker.com.

For the upstream name, you want to use the fully qualified upstream name which also includes any namespace such as the username. This can be just “busybox” if the Repository is an official Docker Hub Repository or it can be something like “fedora/ssh” where “fedora” is the username/namespace.

Then click save and then sync the Repository as you normally would. Katello will fetch all the images and tags contained within that Repository.

How to Upload Docker Images
In versions of Katello prior to 3.0, Docker images could be uploaded directly via either the UI or CLI. However, Katello 3.0 only supports the Docker Registry v2 format, which is significantly different than the Docker Registry v1 format. The docker save command outputs a Docker image in v1 format, which cannot be uploaded directly to a v2 repository.

As a workaround, you can create a local Docker registry like so:

docker run -p 5030:5000 --name registry registry:2

Note the :2 above, which specifies a v2 registry. Push your changes to your newly created local registry then follow the instructions in the section above to sync this registry to Katello. This will ensure that your Docker content stays in Docker's v2 registry format.

How to Publish and Promote Docker Content
Docker content can be published and promoted via Content Views much like yum or puppet content.

After creating a Content View, visit the Docker Content tab. Here you can select any Docker repositories you want to add to your Content View. After you've added Docker Repositories to your view, you may proceed as normal. Visit the Content View user guide for more information.

How to View and Pull Docker Content
To view Docker content contained with Katello, visit the Docker Tags page. This can be accessed under the Content menu at the top of any page.
On the Docker Tags page, you can see a list of Docker Tags grouped by Repository in Katello. This shows you Tags grouped across Content Views and Lifecycle Environments. Suppose I wanted to pull the latest Tag from my redis repository, I would click the latest row for my redis repository.

I can see here that my redis Repository has been added to a published Content View called redisv. If I want to use the tag from that Content View, I would just copy the Published At URL and then on my docker server I would run:

```bash
$ docker pull localhost:5000/default_organization-library-redisv-Tester-redis:latest
Pulling repository localhost:5000/default_organization-library-redisv-Tester-redis...
```

### How to Provision Docker Content

See how to provision content in the documentation in the [foreman-docker documentation](https://theforeman.org/docs/). Provisioning content from Katello works in much the same way.

First, proceed to the new Container page by accessing it from the Containers menu at the top. Then, select the Local Content tab on the second step. This will allow you to select a Docker image from a published Katello repository which is in an environment/content view/Smart Proxy. Then just proceed in the wizard as per the Foreman Docker instructions. When you are finished, you should have a new container running from an image in Katello.
Email Notifications
Types of Email Notifications

In addition to the Email Notifications that Foreman provides:

- Puppet run summary (Daily/Weekly/Monthly)
- Puppet errors

Katello provides a few addition reports:

- Katello Host Advisory (Daily/Weekly/Monthly) - A report of all of the Errata applicable to all readable Content Hosts
- Katello Promote Errata - A report generated at Content View promotion time showing what Errata applicable to the Content Hosts within that Content View.
- Katello Sync Errata - A report generated after each Repository sync listing new Errata synced and how many Content Hosts are applicable.

Configuring the Foreman/Katello to send emails:

The configuration of how the Foreman/Katello service sends email is located in Administer > Settings > Email.

For more information see: Email Configuration

Opting in to the emails

By default a user will receive no email notifications. Each notification must be opted into.

To opt in for your own user, at the very top right of the web interface, hover over your Username, click “My Account” and then click the “Mail Preferences” tab.

To opt in for other users, navigate to “Administer” > “Users” > Click the desired User > click the “Mail Preferences” tab.

Select which emails and frequency you would like the user to have and click “Submit”.

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Errata
Errata are updates between major releases. An Erratum is metadata about a group of packages that explains the importance of the package updates. Errata may be released individually on an as-needed basis or aggregated as a minor release. There are three main types of errata:

- **Enhancement**: the new packages contain one or more added features
- **Bugfix**: the new packages contain one or more bug fixes
- **Security**: the new packages fix one or more security vulnerabilities

With regard to Content Hosts, Errata is divided into two distinct classifications depending on whether or not the Erratum is present in the Content Host's Lifecycle Environment and Content View:

- **Applicable**: the errata applies to one or more Content Hosts
- **Installable**: the errata applies to one or more Content Hosts and is present in the Content Host's Lifecycle Environment and Content View

**Definitions**

- **Content Host**
- **Content View**
- **Lifecycle Environment**

**General Features**

The following is a high-level summary of the Errata features:

- **View List of Errata**
- **View Errata Details**
- **View Affected Content Hosts**
- **View Repositories Containing Errata**
- **Applying Errata**

**View List of Errata**

To view the list of Errata in the Organization:

- navigate to: Content > Errata

**View Errata Details**

To view the details of an Errata:
View Affected Content Hosts

To view the Affected Content Hosts of an Errata:

- navigate to: Content > Errata
- Click on an Errata ID
- Click on the Content Hosts Tab

Note the following option:

- Checking the box limits the display of Content Hosts to those which already have the Errata available in their Lifecycle Environment and Content View.
View Repositories Containing Errata

To view the Repositories Containing an Errata:

- navigate to: Content > Errata
- Click on an Errata ID
- Click on the Repositories Tab

Note that you can filter by Lifecycle Environment and Content View.
Applying Errata

How Errata is applied to a Content Host(s) depends on whether the Errata is installable.

- If the Errata is already installable then the Errata is applied to the Content Host(s).
- If the Errata is not installable then an Incremental Update is generated. An Incremental Update creates a point release of the Content View with the Errata included. The Errata can also be applied to the Content Host(s) as part of this process.

There are two ways to apply Errata:

- A single Errata can be applied to one or more Content Hosts
- Several Errata can be applied to one or more Content Hosts via a bulk operation

Applying a Single Errata

To apply a single Errata:

- Navigate to: Content > Errata
- Click on an Errata ID
- Click on the Content Host tab
- Select the desired Content Hosts
- Click “Apply to Hosts”
- Confirm the action
Applying Several Errata

To apply several Errata:

- Navigate to: Content > Errata
- Select the desired Errata
- Click “Apply Errata”
- Select the intended Content Hosts
- Click “Next”
- Confirm the action
Foreman 2.3.3 has been released! Follow the quick start to install it.

Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Content Credentials
Content Credentials can be associated with Katello Products and Repositories. Two types of Content Credentials are supported, each having a distinct purpose:

GPG Keys
GPG Keys are used by Content Hosts in order to verify the signature of packages that have been retrieved from a Repository and ensure they haven't been corrupted. For more information on GPG Keys see The GNU Privacy Guard.

SSL Certificates
SSL Certificates are used by the Katello server to sync content from upstream repositories that require client SSL authentication.

Using Content Credentials
- Create a Content Credential
- Associate Content Credential with a Product
- View Associated Products
- View Associated Repositories

Create a Content Credential
To create a new Content Credential:
- navigate to Content > Content Credentials
- click Create Content Credential
- Enter a Name and select a Type
- You may either upload your Credential or paste its content into the text area.

When creating SSL Certificates for Products or Repositories that require them, a separate credential must be created for the CA, cert, and key respectively.

Associate Content Credential with a Product
To add a Content Credential to a Product:
Note that adding a Credential to a Product adds it to all current and future repositories unless a repository already has a Credential assigned. The Credential currently assigned to a Repository can be modified or overridden from the detail view of the repository.
- navigate to Content > Products
- select the desired Product from the list
- click **Details**
- click the edit icon on the GPG Key, SSL CA Cert, SSL Client Cert, or SSL Client Key field
- select the desired Content Credential

The steps for adding a Credential to a Repository are the same but performed from the repository’s detail view.

---

**View Associated Products**

To view all Products that have been assigned a Content Credential:

- navigate to Content > Content Credentials
- select the desired Credential from the list
- click **Products**

---

**View Associated Repositories**

To view all Repositories that have been assigned a Content Credential:

- navigate to Content > Content Credentials
• select the desired Credential from the list
• click Repositories

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Managing Content Hosts using Host Collections
Host Collections provide a mechanism to statically group multiple Content Hosts. This enables administrators to group Content Hosts based on the needs of their organization. For example, Content Hosts could be grouped by function, department or business unit.

Once a Host Collection is created, it can be used to perform various actions on the Content Hosts contained within it. This includes actions such as the following:

- Package installation, removal and update
- Errata installation
- Changing of assigned Lifecycle Environment or Content View

Definitions

- Content Host

General Features

The following is a high-level summary of the Host Collection features:

- Create a Host Collection
- Add Content Hosts to a Host Collection
- Copy a Host Collection
- Perform actions on a Host Collection

Create a Host Collection

To create a new collection,

- navigate to: Hosts > Host Collections
- click New Host Collection

Note the following option:

- Content Host Limit: This option will control how many Content Hosts are allowed to be added to the collection.

Add Content Hosts to a Host Collection
To add Content Hosts to a collection:

- navigate to: Hosts > Host Collections
- select the desired collection from the list
- click **Content Hosts**
- click **Add**
- select the Content Hosts you would like to add
- click **Add Selected**

Copy a Host Collection

Copying a Host Collection allows a user to quickly create a new collection that is a copy of an existing one.

To copy a Host Collection:

- navigate to: Hosts > Host Collections
- select the desired collection from the list
- click **Copy Collection**
- enter a name for the new collection
- click **Create**
Perform Actions on a Host Collection

To perform an action on Content Hosts within a collection:

- navigate to: Hosts > Host Collections
- select the desired collection from the list
- click **Collection Actions**
- click on the action that you would like to perform
### Hosts

<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
<th>Content Hosts</th>
<th>Collection Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>development</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>research</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following actions can be performed on content hosts in this host collection:

- Package Installation, Removal, and Update
- Errata Installation
- Host Collection Membership
- Change assigned Environment or Content View

Note: clicking on an action will take the user to the appropriate Content Hosts Bulk Actions page, where all Content Hosts associated with the collection have been selected. Click here, for more information on performing Content Host Bulk Actions.

---

**Foreman 2.3.3 has been released!** Follow the quick start to install it.

**Foreman 2.2.2 has been released!** Follow the quick start to install it.

---

View our [Privacy Policy](#).

This web site is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-sa/3.0/). Source available: [github/theforeman/theforeman.org](https://github/theforeman/theforeman.org).

Thanks to our sponsors for hosting this website and our project services.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Lifecycle Environments
What can a Lifecycle Environments be used for?

- Hold content view versions.
- To manage the lifecycle of Content Hosts.
- Establish workflow containers and promote content views.

Definitions

- Lifecycle Environment - containers for content view versions which are consumed by content hosts.
- Library - a special kind of Lifecycle Environment that does not have a parent. The library serves as the main container for synced content such as products, puppet modules, and published content views. Every organization has a library. Subsequent environments are derived from the library. The first node of an environment is the Library, all future environments are derived from the library and follow the library in promotion order.
- Lifecycle Environment Path - Sequence of lifecycle environments that form the content promotion order.

General Workflow

First create a lifecycle environment connected to the library life cycle environment and promote content views to the new lifecycle environment. A Content Host can now register directly to the promoted content view in the promoted environment or library therein. Updates will be available as soon as new content is synced and promoted.

Viewing the list of lifecycle environments

From the web UI, navigate to:

Content -> Lifecycle Environments

Creating a lifecycle environment

Click on the + next to the Library or the prior environment to add a new path
Creating a lifecycle environment path

Click on the New Environment Path
View/Updating environment name
Click on the name of the environment.

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Provisioning
See the Foreman manual for general information on configuring provisioning.

Templates
Katello ships a number of templates in addition to the standard Foreman ones. When using these templates, if a host has a Host group with an Activation Key, it will register as a Content Host automatically.

- **Katello Kickstart Default** - Kickstart template for Fedora, CentOS, RHEL, and other Red Hat-compatible operating systems.
- **Katello Kickstart Default Finish** - image-based provisioning
- **Katello Kickstart Default User Data** - cloud-init template for EC2 and OpenStack
- **subscription_manager_registration** - Snippet for registering a host for content

To customize any of the above templates, simply clone them and add your changes.

When you synchronize a repository with a distribution such as Fedora or CentOS, Katello will automatically create the operating system and assign these default templates. You may change the defaults by going to Administer > Settings, and selecting the Katello tab.
Managing Puppet Content
Importing the Puppet Forge

The Puppet Forge is a collection of puppet modules written by the community which can be used to manage hosts in Katello. These modules can be used in content views as described in the content views guide in order to configure the running hosts.

To import the puppet forge navigate to

Content > Products

Click on the +New Product button.

Once the product is created, select the product and click the Create Repository button. Fill out the repository as shown:

This can be done via the CLI:

```
hammer product create
--organization "Default Organization"
--name Puppet

hammer repository create
--organization "Default Organization"
--product Puppet
--name forge
--content-type puppet
--url "https://forge.puppetlabs.com/

```

The repository can now be synced.

Importing Puppet Modules from Git

In order to allow users to import puppet modules from Git repositories, Katello comes with a tool called 'pulp-puppet-module-builder' from the pulp-puppet-tools RPM. This utility will be available on the Katello server but it can also be installed on another machine if desired. By running the 'pulp-puppet-module-builder' against a Git repository, it will checkout the repository, build all of the modules, and publish them in a structure Katello can synchronize.

The most common method is to run the utility on the Katello server itself and publish to a local file system directory and sync against that directory.

```
mkdir /modules
chmod 755 /modules

pulp-puppet-module-builder --output-dir=/modules --url="git@mygitserver.com:mymodules.git" --branch=develop
```

This will checkout the 'develop' branch of the Git repository located at 'git@mygitserver.com:mymodules.git' and publish them
to the `/modules` directory. If you have SELinux enabled, in order to sync from the file system, you'll need to apply a label to the files in order for the system to access them. Two options are `httpd_sys_r_content_t` or `pulp_tmp_t`. Note: if you choose `httpd_sys_r_content_t` then the webserver can also read the files so that may or may not be good. One way to apply these labels would be to use the `chcon` command.

Next, from within Katello, simply set the url on your Puppet Repository to `file://modules`. You can now sync the Repository just like any other Repository.

If you are running this on a remote machine, you will need to publish the containing to folder to a location accessible by HTTP or HTTPS.

```
mkdir /var/www/html/modules/
chmod 755 /var/www/html/modules/
pulp-puppet-module-builder --output-dir=/var/www/html/modules --url=git@mygitserver.com:mymodules.git --branch=develop
```

Then in Katello, simply enter `http://HOSTNAME/modules/` for the Repository url and sync it like you normally would.

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Red Hat Content
Katello can be used to manage content associated with Red Hat products based upon available subscriptions. This includes content such as RPMs, package groups, errata and distributions.

Definitions

- Subscription Manifest - An archive file containing certificates and data that represent the subscriptions that are available. A subscription manifest is created and downloaded from the Red Hat Customer Portal.
- Repository - Collection of content (either rpm or puppet).
- Product - Collection of repositories (content hosts attach to a product).
- Library - The initial lifecycle environment where repositories are created. Content that is synced or uploaded lands in the library.

General Workflow

The following is a high-level summary of the workflow:

- Create a subscription manifest using the Red Hat Customer Portal
- Import the subscription manifest
- Enable Red Hat repositories
- Synchronize repositories
- Schedule repository synchronization
- Attach a content host to a product for Red Hat content

Create a Subscription Manifest Using the Red Hat Customer Portal

If you are a Red Hat customer, you should have access to the Red Hat Customer Portal to create and download a subscription manifest. Once created, the manifest can be imported into a Katello Organization.

To access the Red Hat Customer Portal, click here

For details on how to create a subscription manifest, click here

Import the Subscription Manifest

Importing a subscription manifest will allow for Red Hat content associated with purchased subscriptions to be enabled and synchronized to Katello.

To import a manifest,

- navigate to: Content > Red Hat Subscriptions
- click Choose File
- navigate to the file containing the manifest (e.g. manifest.zip)
- click Open
- click Upload
Enable Red Hat Repositories

Once a subscription manifest is imported, access is available to potentially hundreds of Red Hat Repositories (e.g. Red Hat Enterprise Linux Server, Red Hat Enterprise Virtualization...etc). This process allows you to select only those that you are interested in for your enterprise.

To enable Red Hat repositories,

- navigate to: Content > Red Hat Repositories
- select the content type: RPMs, Source RPMs, Debug RPMs, Beta, ISOs or Other
- select one or more Red Hat products (e.g. Red Hat Enterprise Linux Server)
- select one or more Repository Sets (e.g. Red Hat Enterprise Linux 6 Server (RPMs))
- select one or more Repositories (e.g. Red Hat Enterprise Linux 6 Server RPMs x86_64 6Server)

Note:

- When enabling a RHEL repository, Red Hat recommends selecting the Server repo (e.g. 6Server, 5Server) versus a specific release (e.g. 6.2). When a specific release is necessary, the preferred way is to create a Content View with filters that narrow the content to the desired version (e.g. 6.2)
- If you plan to provision content hosts, be sure to enable both the RPM and Kickstart repositories.
Expand each Red Hat Product below to examine the different repository sets available. When enabling a repository set, the different repositories within are discovered and may be enabled individually.

### Enable Red Hat Repositories

<table>
<thead>
<tr>
<th>PRODUCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat Enterprise Linux High Availability for RHEL Server</td>
</tr>
<tr>
<td>Red Hat Enterprise Linux Load Balancer for RHEL Server</td>
</tr>
<tr>
<td>Red Hat Enterprise Linux Resilient Storage for RHEL Server</td>
</tr>
<tr>
<td>Red Hat Enterprise Linux Server</td>
</tr>
<tr>
<td>Red Hat Enterprise Linux Workstation</td>
</tr>
</tbody>
</table>
Synchronize Repositories

Synchronizing a repository will retrieve all associated content and mirror the content in the Katello Library lifecycle environment.

To sync multiple repositories as well as track their progress,

- navigate to: Content > Sync Status
- expand the desired products
- select the repositories to sync
- click Synchronize Now
Creating a Sync Plan

Sync plans provide the ability to schedule repository synchronization on a daily, weekly or a monthly basis. Sync plans can be applied individually or to a set of repositories.

To create a Sync Plan:

- navigate to: Content > Sync Plans
- click New Sync Plan on the upper right

Note the following options:

- **Start Date** and **Start Time**: specify the day of the week/month and time of the day to run the recurring syncs. For example, a sync plan that starts on Sunday 2014-04-06 at 2:30 will occur every Sunday at 2:30 every week if it has a weekly interval. If on a monthly interval it would sync every month on the 6th day at 2:30.
Assigning a Sync Plan to a Red Hat Product

To assign a sync plan to a product,

- navigate to Content > Sync Plans
- select your Sync Plan
- click Products
- click Add
- select the products that you want to add
- click Add Selected on the upper right
Attach a Content Host to a Product for Red Hat Content

To read about registering a content host and subscribing it to a product, click TODO.

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello can be used to manage content associated with SUSE products based upon contents mirrored via the smt tool. This includes content such as RPMs, errata.

Definitions
- Repository - Collection of content (either rpm).
- Product - Collection of repositories (content hosts attach to a product).
- Library - The initial lifecycle environment where repositories are created. Content that is synced or uploaded lands in the library.

General Workflow
The following is a high-level summary of the workflow:
- Set up SUSE account and smt server
- Set up the smt mirroring and Content
- Create Products and Repositories

Set up SUSE account and SMT server
Assuming you have an SCC account, setup the SMT server, [click here](#) for more documentation.

- Install smt

```
# zypper install smt
# yast smt-server
```

- A TUI will pop up and you are going to have to [add credentials from your account](#), found in scc.suse.com, as shown in screenshot below

![SUSE, Customer Center](image)

- Click ok and proceed with the steps provided, it will finish up and exit

Set up the SMT Content and mirroring
On a SLES 12 Box

- Sync repo data

```
# smt-sync
```

- List Repositories

```
# smt-repos
```

- Install the repositories you need to mirror via the smt-repos -e flag. For example to mirror SLES12-SP3-Pool

```
# smt-repos -e SLES12-SP3-Pool
```

- Check the enabled repos

```
# smt-repos -o
```

- Mirror the repositories

```
# smt-mirror
```

- This should install an apache2 on that host
You should be able to navigate to the mirror by browsing to http://<fqdn>/repo/

Create Products and Repositories

- Follow the steps listed here to create Products and Repositories
- Follow the steps listed on the same link as above to create a yum repository with the repository url pointing to the appropriate location. For example http://<fqdn>/repo/SUSE/Products/SLE-SERVER/12-SP3/x86_64/product/

Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Setup Remote Databases in Foreman with
Katello Plugin Installed

Foreman with Katello plugin can be installed with remote databases for both postgresql and mongo. These instructions are for a Foreman with Katello plugin server, where remote databases are currently supported. This guide will refer to the server as “Foreman”, with the assumption that the Katello plugin is installed.

High level

There are two ways to deploy Foreman with remote databases:

1. **Fresh install**
   - prepare Postgres server with databases for Foreman and Candlepin and dedicated users owning them
   - prepare Mongo DB with user owning the pulp_database
   - prepare box where the Foreman will be installed and make sure the databases are accessible from the box
   - run foreman-installer with right parameters pointing to the databases

2. **Migration of existing Foreman installation**
   - prepare Postgres server with databases for Foreman and Candlepin and dedicated users owning them
   - prepare Mongo DB with user owning the pulp_database
   - make sure the databases are accessible from the box where Foreman is installed
   - shut down the services except the dbs you want to move (mongod, postgresql)
   - dump the DBs
   - restore the DBs on remote servers
   - run foreman-installer with right parameters pointing to the databases. It re-configures the databases and start all the services with new DB locations

In either scenario, both of the databases don’t have to be remote. You can opt to use only a remote mongo database or only a remote postgresql database. Both postgresql and mongo databases can be on the same host, but this isn’t recommended due to the amount of resources mongo can use.

Prepare remote Postgres

**GOAL:** To use remote Postgres database with Foreman we have to:

- be able to access the databases from foreman box
- the database user we use to connect to the database needs to own the database, i.e. it can create, alter and delete the tables, indexes and constraints. Note it is not required to be able to create the database itself.

Install Postgres

**Warning:** This is just minimal testing setup which is not suitable for production, please adjust the settings to your environment as needed.

Assume our postgres server has hostname `postgres.example.com`.

First, we install postgresql.

```
yum install -y postgresql-server postgresql-contrib
postgresql-setup initdb
systemctl enable --now postgresql
```

Now we need to make Postgres listen to inbound connections, please adjust these parameters to your own networking and security requirements.

Edit `/var/lib/pgsql/data/postgresql.conf`. Uncomment `listen_address` and modify its value to look like:

```
listen_address = "***"
```

The next step we need to take is to add a proper client authentication for remote client to our postgres server. To achieve the same, edit `/var/lib/pgsql/data/pg_hba.conf`. Append the following line at the end of the file

```
host all all <katello.example.com ip>/24 md5
```

Now restart the postgres service for changes to take effect

```
systemctl restart postgresql
```

Create the databases
Switch the user role to postgres and start postgres client

```
su - postgres -c psql
```

Once inside the client, we need to create two databases and dedicated roles, one for foreman and one for candlepin.

```
CREATE USER "foreman" WITH PASSWORD '<FOREMAN_PASSWORD>';
CREATE USER "candlepin" WITH PASSWORD '<CANDLEPIN_PASSWORD>';
CREATE DATABASE foreman OWNER foreman;
CREATE DATABASE candlepin OWNER candlepin;
```

Test it works

From katello.example.com test the DB is accessible:

```
PGPASSWORD='<FOREMAN_PASSWORD>' psql -h postgres.example.com  -p 5432 -U foreman -d foreman -c "SELECT 1 as ping"
PGPASSWORD='<CANDLEPIN_PASSWORD>' psql -h postgres.example.com  -p 5432 -U candlepin -d candlepin -c "SELECT 1 as ping"
```

If there are no errors we are done with database preparation.

Prepare remote Mongo

GOAL: To use remote Mongo database with Foreman we have to:

- be able to access the databases from foreman box
- the database user we use to connect to the database needs to own the database

Install Mongo DB

Warning: This is just minimal testing setup which is not suitable for production.

Assume our Mongo server has hostname mongo.example.com. Install and enable Mongo server

```
yum install -y centos-release-scl
yum install -y rh-mongodb34-syspaths
```

Enable authentication in `/etc/mongod.conf`

```
auth=true
```

Enable and start the service

```
systemctl enable --now mongod
```

Create Pulp user and database

```
mongo admin -u admin -p admin --eval "db.createUser({user:'pulp',pwd:'<PULP_PASSWORD>',roles:[{role:'dbOwner', db:'pulp_database'}, { role: 'readWrite', db: 'pulp_database'}]})"
mongo --host mongo.example.com -u pulp -p <PULP_PASSWORD> --port 27017 --eval 'ping:1' pulp_database
```

Test it works

From katello.example.com test the mongo DB is accessible:

```
mongo --host mongo.example.com -u pulp -p <PULP_PASSWORD> --port 27017 --eval 'ping:1' pulp_database
```

If there are no errors we are done with database preparation.

Fresh install

Install katello package

We assume the box where the Foreman server will be installed has hostname katello.example.com.
Follow the documentation to install the katello package and do not run foreman-installer. We need to use the remote database flags with the installer. Use the following steps once the katello rpm is installed.

**Prepare remote databases**

Follow the instructions to prepare remote mongo and prepare remote postgres to make the remote database servers ready for installation.

**Run the installer**

To install and configure Foreman we just need to run

```
foreman-installer --scenario katello \  
--foreman-db-host postgres.example.com \  
--foreman-db-password <FOREMAN_PASSWORD> \  
--foreman-db-database foreman \  
--foreman-db-manage false \  
--katello-candlepin-db-host postgres.example.com \  
--katello-candlepin-db-name candlepin \  
--katello-candlepin-db-password <CANDLEPIN_PASSWORD> \  
--katello-candlepin-manage-db false \  
--katello-pulp-db-username pulp \  
--katello-pulp-db-password <PULP_PASSWORD> \  
--katello-pulp-db-seeds "mongo.example.com:27017" \  
--katello-pulp-db-name pulp_database \  
--katello-pulp-manage-db false
```

Note: for more related options and tips on SSL configuration see Full list of options

**Migration of existing Foreman**

Migrating an existing installation to remote databases can take time, so plan for some outage time (length depending on database size) while a backup is taken and the databases are migrated.

In this example, we assume that Foreman was installed and is running on katello.example.com.

**Prepare remote databases**

Follow the instructions to prepare remote mongo and prepare remote postgres to make the remote database servers ready for migration.

**Stop the Foreman server**

Stop the Foreman related services to minimize risk of the data changes during the migration

```
katello-service stop
systemctl start postgresql mongod
```

**Dump databases**

Dump the local databases

```
foreman-maintain backup online --skip-pulp-content --preserve-directory -y /tmp/migration_backup
```

**Restore data in remote databases**

You can restore the SQL dumps to the remote databases from the foreman system.

```
PGPASSWORD='<FOREMAN_PASSWORD>' pg_restore -h postgres.example.com -U foreman -d foreman < /tmp/migration_backup/foreman.dump
PGPASSWORD='<CANDLEPIN_PASSWORD>' pg_restore -h postgres.example.com -U candlepin -d candlepin < /tmp/migration_backup/candlepin.dump
mongorestore --host mongo.example.com --db pulp_database --username pulp --password <PULP_PASSWORD> /tmp/migration_backup/mongo_dump
```

Now the copy of the local database is also at the remote locations.

**Update the configuration**

To update existing configuration of Foreman we just need to run
The installer starts services aside from the database related services. Everything should be up and ready at this point, and you can clean up the local databases if you would like.

Full list of remote database related options in the installer

Use `foreman-installer --full-help` for all up-to-date installer options

Foreman database related:

- `-foreman-db-manage`\ If enabled, will install and configure the database server on this host
- `-foreman-db-host`\ Database 'production' host
- `-foreman-db-password`\ Database 'production' password, default is randomly generated
- `-foreman-db-database`\ Database 'production' size of connection pool (current: 5)
- `-foreman-db-port`\ Database 'production' port
- `-foreman-db-root-cert`\ Root cert used to verify SSL connection to postgres
- `-foreman-db-salmode`\ Database 'production' ssl mode (disable|allow|prefer|required|verify-full)
- `-foreman-db-username`\ Database 'production' user (e.g. foreman)

Candlepin database related:

- `-katello-candlepin-db-host`\ Host with Candlepin DB
- `-katello-candlepin-db-name`\ Name of the Candlepin DB
- `-katello-candlepin-db-password`\ Candlepin DB password
- `-katello-candlepin-db-port`\ Port accepting connections to Candlepin DB
- `-katello-candlepin-db-ssl`\ Boolean indicating if the connection to the database should be over SSL
- `-katello-candlepin-db-ssl-verify`\ Boolean indicating if the SSL connection to the database should be verified
- `-katello-candlepin-db-user`\ Candlepin DB user
- `-katello-candlepin-manage-db`\ Boolean indicating whether a database should be installed, this includes db creation and user

Mongo database related:

- `-katello-pulp-db-ca-path`\ The ca_certs file contains a set of concatenated "certification authority" certificates,
- `-katello-pulp-db-name`\ The name of the database to use
- `-katello-pulp-db-password`\ The password to use for authenticating to the MongoDB server
- `-katello-pulp-db-replica-set`\ The name of replica set configured in MongoDB, if one is in use
- `-katello-pulp-db-seeds`\ Comma-separated list of hostname:port of database replica seed hosts
- `-katello-pulp-db-ssl-certfile`\ The certificate file used to identify the local connection against mongod.
- `-katello-pulp-db-ssl-keyfile`\ A path to the private keyfile used to identify the local connection against mongod. If not specified and `--katello-pulp-db-ssl` is true, mongod will be configured to allow insecure connection.
- `-katello-pulp-db-unsafe-autoretry`\ If true, retry commands to the database if there is a connection error.
- `-katello-pulp-db-verify-ssl`\ Specifies whether a certificate is required from the other side of the connection, and allows the database to connect to an untrusted server
- `-katello-pulp-db-write-concern`\ Write concern of 'majority' or 'all'. When 'all' is specified, 'w' is set to number of replicas.

SSL configuration

Here is sample installer command that sets up Postgres databases with SSL verification. The Postgres server has its own CA. The CA cert used by Candlepin needs to be stored in system trust (`/etc/pki/java/cacerts`) as there is no other way to pass it to Candlepin.

```bash
foreman-installer --scenario katello
--foreman-admin-password changeme
--foreman-db-host postgres.example.com
--foreman-db-password foreman
--foreman-db-database foreman
--foreman-db-manage false
--katello-candlepin-db-host postgres.example.com
--katello-candlepin-db-name candlepin
--katello-candlepin-db-password <CANDLEPIN_PASSWORD
--katello-candlepin-db-manage-db false
--katello-pulp-db-username pulp
--katello-pulp-db-password <PULP_PASSWORD
--katello-pulp-db-seeds "mongo.example.com:27017"
--katello-pulp-db-name pulp_database
--katello-pulp-manage-db false
```

The actual option names may vary between versions. Check the actual naming with `foreman-installer --full-help`.
Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.

View our Privacy Policy
This web site is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. Source available: github/theforeman/theforeman.org
Thanks to our sponsors for hosting this website and our project services.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. API Documentation
8. Annotated Backend Requests

Katello Troubleshooting
For general support information, see here.
Sub-services status

Katello uses a set of back-end services to perform the actual job. The status of these services can negatively influence the whole system and it's one of the first things to check when some errors occur.

The status of back-end services can be found either from the Web UI on the `/about` page:

![System Information](image)

Alternatively, the `hammer ping` command can be used to get this information.

`katello-service` tool can be used to restart Katello related services. See `man katello-service` for more details.

Tasks

Katello uses Foreman Tasks for orchestration between the underlying services (local database, Pulp, Candlepin...). The tasks are modeled as Dynflow processes. When something goes wrong (and there might be many reasons for this happening), Dynflow gives us the tools to recover from these errors to get to the consistent state.

Health checking

There are two properties used for identifying issues with a task:

- **state** - what phase of execution is the task in, possible values are:
  - **planning** - the planning phase of the task is performed: the operations performed in this phase shouldn't modify anything outside Katello's database. The execution of this phase happens in the web-process thread and usually should not take more than a few seconds
  - **planned** - the planning phase finished and the task is waiting for the executor process (foreman-tasks service) to pick it up
  - **running** - the executor is performing the orchestration action, modifying the state of external services to converge to the final state
  - **paused** - something went wrong during running the task and it's waiting for the resolution (further details below)
  - **stopped** - the execution of the task finished (the success is determined by the `result` value)
  - **result** - how the task ended up (or is going to end up if we already know it)
  - **pending** - task is in the process of executing
  - **success** - no errors occurred during the execution
  - **error** - unresolved errors occurred during the execution
  - **warning** - there were errors during the execution, but they did not prevent the task from finishing or were skipped
To see all the tasks in the system, one can go to `/foreman_tasks/tasks` page. To see all the tasks that failed, one can search on `result = error`:

Failed tasks include those in the 'stopped' or 'paused' state. The stopped tasks are already considered as resolved, there is no risk of inconsistency. The tasks in the 'stopped' state and the 'error' result are usually those failed during the planning phase (usually locking error or bad input data).

To see all the tasks requiring further assistance, filter on `state = paused`:

Dealing with paused task

Once the paused task is identified, one can investigate the problem causing the errors:
The resolution of the problem is dependent on the error details. The task may be resolvable by resuming the task: make sure the sub-services are running (see Sub-services status for more details) and then click ‘Resume’ within the web interface.

If this still doesn’t help, one possible step is going to a Dynflow console (the button from task details takes you there):

Caution: Dynflow console is considered a low-level tool and should be used very carefully, ideally discussing other options before using its features.

If the failed task was taken care of by other means (performing the failed steps manually) or it was identified as not critical to the whole task, one can skip the failed step and resume the task to continue. These tasks end up with warning result at the end, to indicate there was some difficulty during the run.

Dealing with Long Running Tasks

In some cases, there might be an issue with sub-services that make it appear as if the task is running for too long without any obvious evidence that something is occurring within the task.

The first place to look in this case is filtering the tasks on state = running and looking at Running Steps in the task details:
In this case, the "start_time" => nil indicates that the task was not picked up by Pulp, which usually means some issues with running the Pulp workers. See (see Sub-services status for more details).

One can also go to the Dynflow console for even more details: the suspended state means that the step is waiting for the external task to finish - the suspended state itself doesn't have to indicate any error:

If you're sure the underlying services are running fine, depending on the type of task, there might be a possibility to cancel the running step and possibly following dealing with paused tasks instead.
Locking

Foreman tasks provides a locking mechanism which is important to prevent the possibility of operations colliding that are being performed concurrently on the same resource (such as synchronizing and deleting a repository at the same time).

When trying to run an operation on a resource that another task is already running, one can get required lock is already taken by other running tasks.

A locked resource is one where another task that is related to the same resource is already running. Thus, the task being attempted will result in that task being tried in running or paused state. This means that the error is triggered also in cases, where there is a task with unresolved failure (see dealing with paused tasks for more details).

In rare cases, it might be hard to get into the stopped state. There is a possibility to unlock the resource in the running/paused task. This will switch the task into stopped state, freeing the resources for other tasks. Caution: unlocking allows running other tasks to run on potentially inconsistent data, which might lead into further errors. It's still possible to go to the Dynflow console and resume the tasks, even after using the unlock feature. There are two unlock-related buttons: Unlock and Force Unlock. The only difference between these two is the second one is allowed even when the task is in running state, and therefore is potentially even more dangerous than the Unlock button. See dealing with tasks running too long before attempting to use the Force Unlock option.

Debug Certificate

Debug certificates (also called Ueber Certificates) can be used to unlock all the content for a given Organization. These are meant to be used by sysadmins who are debugging issues with the Katello install.

Generating a Debug Certificate

To generate a debug certificate for a given Organization from the UI, navigate to the organizations page and click on the organization for which you want a debug certificate. Click on the button to generate and download the certificate as highlighted below:
To generate a debug certificate using the API see the API docs located on your server running at `/apidoc`.

In either case, you will get the Private Key and Certificate returned to you in a format such as:

```
Key: -----BEGIN RSA PRIVATE KEY-----
<<<<DER ENCODED TEXT>>>>
-----END RSA PRIVATE KEY-----
```

```
Cert: -----BEGIN CERTIFICATE-----
<<<<DER ENCODED TEXT>>>>
-----END CERTIFICATE-----
```

Using Firefox to browse content

If you wish to use the certificate to browse content via Firefox, do the following:

1. Copy the contents of the above file from `-----BEGIN RSA PRIVATE KEY-----` to `-----END RSA PRIVATE KEY-----` inclusive to a file called `key.pem`
2. Copy the contents of the above file from `-----BEGIN CERTIFICATE-----` to `-----END CERTIFICATE-----` inclusive to a file called `cert.pem`
3. Run the following command to create a pkcs12 file:

```
openssl pkcs12 -keypbe PBE-SHA1-3DES -certpbe PBE-SHA1-3DES -export -in cert.pem -inkey key.pem -out [NAME].pfx -name [NAME]
```

4. Provide a password when prompted.
5. Using the preferences tab, import the resulting pfx file into your browser (Edit->Preferences->Advanced Tab -> View Certificates -> Import)
6. Point your browser at `http://[FQDN]/pulp/repos/[ORG_NAME]`

To use curl to access the repository, you can provide `-cert` and `-key` options. Provided the cert is in `~/cert.pem` and key in `~/key.cert`, the following command will let you access any repository data in the organization. To check the access to a repository, checking the availability of `repodata/repomd.xml` is usually a good idea (make sure `key.pem` and `cert.pem` are "absolute paths" otherwise it silently fails):

```
```

Frequently Asked Questions

Can I use pulp-admin with Katello?
We do not encourage the use of pulp-admin because it has the potential to get data out of sync. However, pulp-admin can be useful when troubleshooting Katello.

1. Install needed packages
   
   ```sh
   yum install -y pulp-admin-client pulp-rpm-admin-extensions
   ```

2. Edit `/etc/pulp/admin/admin.conf`
3. Uncomment the `host:` line and add your server's hostname:
   
   ```plaintext
   host: katello-hostname.example.com
   ```

4. Run `grep default_password /etc/pulp/server.conf` to lookup the admin password

   ```sh
   sudo grep default_password /etc/pulp/server.conf
   ```

5. Use pulp-admin by specifying the admin username and password:

   ```sh
   pulp-admin -u admin -p default_password repo list
   ```

Using pulp-admin without password

Using the `pulp-admin login` command does not function and is not supported with Katello in an attempt to limit access to the certificate authority generated at installation time.

Katello 3.0 generates a client cert at installation time which allows usage of pulp-admin without specifying the username and password. To use this:

1. `mkdir ~/.pulp`
2. Copy the public client cert and private key to a file together:

   ```sh
   # sudo cat /etc/pki/katello/certs/pulp-client.crt /etc/pki/katello/private/pulp-client.key > ~/.pulp/user-cert.pem
   ```
3. Run pulp-admin without username and password:

   ```sh
   pulp-admin repo list
   ```

How can I sync a repository like Katello does directly from the console?

Sometimes you want to debug why a synchronization of a repository from Katello is failing and rather than dig through log files and error messages it can often be easier to try to sync the repo with the "grinder" tool which is what Katello uses to download repositories. The tool can be ran from a terminal on your Katello server:

```sh
$ grinder yum --label=sync-test --url=http://fedorapeople.org/groups/katello/releases/yum/1.0/RHEL/6Server/x86_64/
grinder.RepoFetch: INFO    fetchYumRepo() repo_label = sync-test, repo_url = http://fedorapeople.org/groups/katello/releases/yum/1.0/RHEL/6Server/x86_64/, basepath = ./, verify_options = {}
grinder.RepoFetch: INFO    sync-test, http://fedorapeople.org/groups/katello/releases/yum/1.0/RHEL/6Server/x86_64/, Calling RepoFetch with: cacert=<None>, clicert=<None>, clikey=<None>, proxy_url=<None>, proxy_port=<3128>, proxy_user=<None>, proxy_pass=<NOT_LOGGED>, ssverify=<1>, max_speed=<None>, verify_options=<{}>, filter=<None>
   grinder.ParallelFetch: INFO 5 threads are active. 8 items left to be fetched
   grinder.ParallelFetch: INFO 4 threads are active. 4 items left to be fetched
   grinder.ParallelFetch: INFO WorkerThread deleting ActiveObject
   grinder.ParallelFetch: INFO 3 threads are active. 3 items left to be fetched
   grinder.ParallelFetch: INFO WorkerThread deleting ActiveObject
   grinder.ParallelFetch: INFO Thread ending
   grinder.ParallelFetch: INFO Thread ending
   grinder.ParallelFetch: INFO Thread ending
   $ ls sync-test/
   converge-ui-devel-0.8.3-1.el6.noarch.rpm
   elasticsearch-0.18.4-13.el6.noarch.rpm
   katello-1.0.6-1.el6.noarch.rpm
   katello-agent-1.0.6-1.el6.noarch.rpm
   katello-all-1.0.6-1.el6.noarch.rpm
   katello-certs-tools-1.1.7-1.el6.noarch.rpm
   lucene3-contrib-3.4.0-2.el6.noarch.rpm
   repodata
   rubygem-actionmailer-3.0.10-3.el6.noarch.rpm
   ...
```
Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Foreman API v2 is currently the default API version.

## Resources

### Activation keys

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /katello/api/activation_keys</td>
<td>List activation keys</td>
</tr>
<tr>
<td>GET /katello/api/organizations/organization_id/activation_keys</td>
<td></td>
</tr>
<tr>
<td>GET /katello/api/environments/environment_id/activation_keys</td>
<td></td>
</tr>
<tr>
<td>POST /katello/api/activation_keys</td>
<td>Create an activation key</td>
</tr>
<tr>
<td>PUT /katello/api/activation_keys/id</td>
<td>Update an activation key</td>
</tr>
<tr>
<td>DELETE /katello/api/activation_keys/id</td>
<td>Destroy an activation key</td>
</tr>
<tr>
<td>GET /katello/api/activation_keys/id</td>
<td>Show an activation key</td>
</tr>
<tr>
<td>POST /katello/api/activation_keys/id/copy</td>
<td>Copy an activation key</td>
</tr>
<tr>
<td>GET /katello/api/activation_keys/id/host_collections/available</td>
<td>List host collections the activation key does not belong to</td>
</tr>
<tr>
<td>GET /katello/api/activation_keys/id/releases</td>
<td>Show release versions available for an activation key</td>
</tr>
<tr>
<td>GET /katello/api/activation_keys/id/product_content</td>
<td>Show content available for an activation key</td>
</tr>
<tr>
<td>POST /katello/api/activation_keys/id/host_collections</td>
<td></td>
</tr>
<tr>
<td>PUT /katello/api/activation_keys/id/add_subscriptions</td>
<td>Attach a subscription</td>
</tr>
<tr>
<td>PUT /katello/api/activation_keys/id/remove_subscriptions</td>
<td>Unattach a subscription</td>
</tr>
<tr>
<td>PUT /katello/api/activation_keys/id/content_override</td>
<td>Override content for activation_key</td>
</tr>
</tbody>
</table>

### Architectures

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /api/architectures</td>
<td>List all architectures</td>
</tr>
</tbody>
</table>
Foreman 2.3.3 has been released! Follow the quick start to install it.
Foreman 2.2.2 has been released! Follow the quick start to install it.
Katello 3.7 Documentation

1. Installation
   1.1 Katello
   1.2 Smart proxy
   1.3 Clients
2. Upgrade
   2.1 Katello
   2.2 Smart proxy
   2.3 Clients
3. Release Notes
4. CLI
5. User Guide
   5.1 Activation Keys
   5.2 Backup
   5.3 Foreman proxy
   5.4 Content
   5.5 Content Hosts
   5.6 Content Views
   5.7 Disconnected
   5.8 Docker Management
   5.9 Email Notifications
   5.10 Errata
   5.11 Glossary
   5.12 Content Credentials
   5.13 Host Collections
   5.14 Lifecycle Environments
   5.15 Provisioning
   5.16 Puppet Integration
   5.17 Red Hat Content
   5.18 SUSE Content
   5.19 Remote Databases
6. Troubleshooting
7. Api Documentation
8. Annotated Backend Requests

Repository Sync (default settings)
Request # 1:

**POST /pulp/api/v2/repositories/scenario_test/actions/sync/**

Backend Service: pulp

Description:

Request body

```json
{
  "override_config": {
    "num_threads": 4,
    "validate": true
  }
}
```

Response body

```json
{
  "spawned_tasks": [
    {
      "_href": "/pulp/api/v2/tasks/8b9c7591-4d1b-48b7-b37b-bce4fde9c2fd/",
      "task_id": "8b9c7591-4d1b-48b7-b37b-bce4fde9c2fd",
      "result": null,
      "error": null
    }
  ],
  "result": null,
  "error": null
}
```

Request # 2: Poll Sync task

**GET /pulp/api/v2/tasks/8b9c7591-4d1b-48b7-b37b-bce4fde9c2fd/**

Backend Service: pulp

Description:

Total Requests for this URL: 32

Request body

None

Response body

```json
{
  "exception": null,
  "task_type": "pulp.server.managers.repo.sync.sync",
  "href": "/pulp/api/v2/tasks/8b9c7591-4d1b-48b7-b37b-bce4fde9c2fd/",
  "task_id": "8b9c7591-4d1b-48b7-b37b-bce4fde9c2fd",
  "tags": [
    "pulp:repository:scenario_test",
    "pulp:action:sync"
  ],

  "finish_time": "2017-03-30T21:16:05Z",
  "ns": "task_status",
  "start_time": "2017-03-30T21:16:05Z",
  "traceback": null,
  "spawned_tasks": [
    {
      "_href": "/pulp/api/v2/tasks/0837f608-4696-449a-811f-70ddabe59025/",
      "task_id": "0837f608-4696-449a-811f-70ddabe59025"
    }
  ],
  "progress_report": {
    "yum_importer": {
      "content": {
        "items_total": 0,
        "state": "FINISHED",
        "error_details": [
          {
            "rpm_total": 0,
            "rpm_done": 0,
            "drpm_total": 0,
            "drpm_done": 0,
            "size_total": 0,
            "size_left": 0,
            "items_left": 0
          }
        ]
      },
      "comps":
    }
  }
}
```
Request # 3: Poll Publish task

GET /pulp/api/v2/tasks/0837f608-4696-449a-811f-70ddabe59025/

Backend Service: pulp

Description:
Total Requests for this URL: 12

Request body
None

Response body

```
{
  "exception": null,
  "task_type": "pulp.server.managers.repo.publish.publish",
  "_href": "/pulp/api/v2/tasks/0837f608-4696-449a-811f-70ddabe59025/",
  "task_id": "0837f608-4696-449a-811f-70ddabe59025",
  "tags": [],
  "_ns": "task_status",
  "finish_time": "2017-03-30T21:16:05Z",
  "start_time": "2017-03-30T21:16:05Z",
  "traceback": null,
  "spawned_tasks": [],
  "progress_report": {
    "scenario_test": [
      {
        "num_success": 1,
        "description": "Copying files",
        "step_type": "save_tar",
        "items_total": 1,
        "state": "FINISHED",
        "error_details": []
      },
      {
        "num_success": 1,
        "description": "Initializing repo metadata",
        "step_type": "initialize_repo_metadata",
        "items_total": 1,
        "state": "FINISHED",
        "error_details": []
      },
      {
        "num_success": 1,
        "description": "Publishing Distribution files",
        "step_type": "distribution",
        "items_total": 1,
        "state": "FINISHED",
        "error_details": []
      },
      {
        "num_success": 8,
        "description": "Copying files",
        "step_type": "save_tar",
        "items_total": 8,
        "state": "FINISHED",
        "error_details": []
      }
    ],
    "repository_test": [
      {
        "num_success": 1,
        "description": "Copying files",
        "step_type": "save_tar",
        "items_total": 1,
        "state": "FINISHED",
        "error_details": []
      }
    ]
  },
  "metadata": {
    "state": "FINISHED"
  }
}
```
<table>
<thead>
<tr>
<th>Step ID</th>
<th>Description</th>
<th>Step Type</th>
<th>Items Total</th>
<th>State</th>
<th>Error Details</th>
<th>Details</th>
<th>Num Failures</th>
<th>Num Processed</th>
</tr>
</thead>
<tbody>
<tr>
<td>197b0894-07fd-470b-8bb4-5a55b9713d18</td>
<td>Publishing RPMs</td>
<td>rpms</td>
<td>8</td>
<td>FINISHED</td>
<td></td>
<td></td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>54795382-de70-4de3-a8ce-5f726b8f9cb2</td>
<td>Publishing Delta RPMs</td>
<td>drpms</td>
<td>1</td>
<td>SKIPPED</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1996449c-9471-4988-b3ee-b7ceae0c6b41</td>
<td>Publishing Errata</td>
<td>errata</td>
<td>3</td>
<td>FINISHED</td>
<td></td>
<td></td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>f12472f5-4012-4112-afe0-49f57348e569</td>
<td>Publishing Comps file</td>
<td>comps</td>
<td>3</td>
<td>FINISHED</td>
<td></td>
<td></td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>31efb1ef-a413-48be-99f2-7f4788a1cebd</td>
<td>Publishing Metadata</td>
<td>metadata</td>
<td>0</td>
<td>FINISHED</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22b9313d-ffae-49e3-8027-c86ef68f34de</td>
<td>Closing repo metadata</td>
<td>close_repo_metadata</td>
<td>1</td>
<td>FINISHED</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>fbc75485-e400-4cfa-bfa7-693edd8a832c</td>
<td>Generating sqlite files</td>
<td>generate sqlite</td>
<td>1</td>
<td>SKIPPED</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```json
{
  "step_id": "fbc75d85-e400-4cla-bfa7-693edd8a832c",
  "num_processed": 0,
  "num_success": 0,
  "description": "Generating HTML files",
  "step_type": "repoview",
  "items_total": 1,
  "state": "SKIPPED",
  "error_details": [
    
  ],
  "details": "",
  "num_failures": 0,
  "step_id": "bf1215fe-c617-4481-894a-1ff1c7c7043b",
  "num_processed": 0,
  "num_success": 1,
  "description": "Publishing files to web",
  "step_type": "publish_directory",
  "items_total": 1,
  "state": "FINISHED",
  "error_details": [
    
  ],
  "details": "",
  "num_failures": 0,
  "step_id": "605ae2e4-c443-4a5b-ba9f-f313d23c4822",
  "num_processed": 1,
  "num_success": 1,
  "description": "Writing Listings File",
  "step_type": "initialize_repo_metadata",
  "items_total": 1,
  "state": "FINISHED",
  "error_details": [
    
  ],
  "details": "",
  "num_failures": 0,
  "step_id": "9fa215e9-b465-4f33-9ddd-e05c5a9e314d",
  "num_processed": 1,
  "num_success": 1,
  "description": "Copying files",
  "step_type": "save_tar",
  "items_total": 1,
  "state": "FINISHED",
  "error_details": [
    
  ],
  "details": "",
  "num_failures": 0,
  "step_id": "8fa5d08e-3473-4127-80f6-223bee0a0409",
  "num_processed": 1,
  "queue": "reserved_resource_worker-1@dev.example.com.dq",
  "state": "finished",
  "worker_name": "reserved_resource_worker-1@dev.example.com",
  "result": {
    "result": "success",
    "exception": null,
    "repo_id": "scenario_test",
    "started": "2017-03-30T21:16:05Z",
    "ns": "repo_publish_results",
    "completed": "2017-03-30T21:16:05Z",
    "traceback": null,
    "distributor_type_id": "yum_distributor",
    "summary": {
      "generate sqlite": "SKIPPED",
      "initialize_repo_metadata": "FINISHED",
      "rpm": "FINISHED",
      "repoview": "SKIPPED",
      "close_repo_metadata": "FINISHED",
      "drpms": "SKIPPED",
      "comps": "FINISHED",
      "distribution": "FINISHED",
      "save_tar": "FINISHED",
      "publish_directory": "FINISHED",
      "errata": "FINISHED",
      "metadata": "FINISHED"
    },
    "error_message": null,
    "distributor_id": "scenario_test",
    "id": "58dd7595418a8a0648b9bc53",
    "details": {
      "num_success": 1,
      "description": "Copying files",
      "step_type": "save_tar",
      "items_total": 1,
      "state": "FINISHED",
      "error_details": [
        
      ],
      "details": "",
      "num_failures": 0,
      "step_id": "8fa5d08e-3473-4127-80f6-223bee0a0409",
      "num_processed": 1
    }
  }
}```
{ "description": "Initializing repo metadata", "step_type": "initialize_repo_metadata", "items_total": 1, "state": "FINISHED", "error_details": [ ] }, 
{ "details": "", "num_failures": 0, "step_id": "32ce4d1-6e51-419a-9833-5f2c88a3ef09", "num_processed": 1 }, 
{ "num_success": 1, "description": "Publishing Distribution files", "step_type": "distribution", "items_total": 1, "state": "FINISHED", "error_details": [ ] }, 
{ "details": "", "num_failures": 0, "step_id": "d562a1815-acd5-439c-b511-dcbd69edade4", "num_processed": 1 }, 
{ "num_success": 1, "description": "Publishing RPMs", "step_type": "rpms", "items_total": 1, "state": "FINISHED", "error_details": [ ] }, 
{ "details": "", "num_failures": 0, "step_id": "197b0894-07fd-470b-8bb4-5a55b9713d18", "num_processed": 1 }, 
{ "num_success": 8, "description": "Publishing Delta RPMs", "step_type": "drpms", "items_total": 1, "state": "SKIPPED", "error_details": [ ] }, 
{ "details": "", "num_failures": 0, "step_id": "54779385-de70-4de3-8f9c-b8f97f26b89", "num_processed": 1 }, 
{ "num_success": 3, "description": "Publishing Errata", "step_type": "errata", "items_total": 3, "state": "FINISHED", "error_details": [ ] }, 
{ "details": "", "num_failures": 0, "step_id": "1996449c-9471-4988-b3ee-b7ce4e0c6b41", "num_processed": 3 }, 
{ "num_success": 0, "description": "Publishing Metadata.", "step_type": "metadata", "items_total": 0, "state": "FINISHED", "error_details": [ ] }, 
{ "details": "", "num_failures": 0, "step_id": "1947215-4012-41e2-a0e0-49f57348e569", "num_processed": 3 }, 
{ "num_success": 0, "description": "Publishing Metadata.", "step_type": "metadata", "items_total": 0, "state": "FINISHED", "error_details": [ ] }, 
{ "details": "", "num_failures": 0, "step_id": "31eb1ef-a413-48be-9ff2-7f4788a1ced", "num_processed": 3 }
Request # 4: Distribution Search

POST /pulp/api/v2/repositories/scenario_test/search/units/

Backend Service: pulp

Description: Search for distribution information with all fields

Request body

```json
{
  "error": null,
  "id": {
    "$oid": "58dd7595e6919db96421ea23"
  },
  "id": "58dd7595e6919db96421ea23"
}
```
Request # 5: Fetch rpm unit ids for this repo

POST /pulp/api/v2/repositories/scenario_test/search/units/

Backend Service: pulp

Description:

Request body

```json
{
    "criteria": {
        "type_ids": [
            "rpm"
        ]
    }
}
```
Response body
Request # 6: Fetch rpm units for this repository

POST /pulp/api/v2/content/units/rpm/search/

Backend Service: pulp

Description: Search for all rpms with the previously fetched ids. We do this because historically we’ve wanted the list of all repo ids for each unit, which is only available via this endpoint.

Request body

```json
{
    "criteria": {
        "limit": 8,
        "skip": 0,
        "fields": [
            "name",
            "version",
            "release",
            "arch",
            "epoch",
            "summary",
            "sourcerpm",
            "checksum",
            "filename",
            "_id"
        ],
        "filters": {
            "_id": {
                "$in": [
                    "085babbb-3a43-4b9c-bdfb-915fd78c7bec",
                    "2b00d383-f2c7-462c-bfe3-25a008caaa0fe",
                    "5ee1ec3a-7f81-47e6-bbba-11b443251f29",
                    "6b8e61f7-c155-4cb9-a291-a93d05c8eac3",
                    "7f14f62-7a47-4b20-072d-0d1549c527",
                    "96b939b6-8411-093d0b1d1ec3",
                    "d35640a0-8613-44bd-9e95-14122834087",
                    "f521967-a646-436e-83-1ac9e5l87cd8"
                ]
            }
        }
    },
    "include_repos": true
}
```

Response body

```json
{
    "repository_memberships": [
        {
            "sourcerpm": "walrus-0.3-0.8.src.rpm",
            "name": "walrus",
            "checksum": "6b8e61f7-c155-4cb9-a291-a93d05c8eac3",
            "summary": "A dummy package of walrus",
            "filename": "walrus-0.3-0.8.noarch.rpm",
            "epoch": 0,
            "version": "0.3",
            "release": "0.8",
            "_id": "085babbb-3a43-4b9c-bdfb-915fd78c7bec",
            "arch": "noarch",
            "children": [
                "href": "pulp/api/v2/content/units/rpm/085babbb-3a43-4b9c-bdfb-915fd78c7bec"
            ]
        },
        {
            "sourcerpm": "penguin-0.3-0.8.src.rpm",
            "name": "penguin",
            "checksum": "7f14f62-7a47-4b20-072d-0d1549c527",
            "summary": "A dummy package of penguin",
            "filename": "penguin-0.3-0.8.noarch.rpm",
            "epoch": 0,
            "version": "0.3",
            "release": "0.8",
            "_id": "2b00d383-f2c7-462c-bfe3-25a008caaa0fe",
            "arch": "noarch",
            "children": [
                "href": "pulp/api/v2/content/units/rpm/2b00d383-f2c7-462c-bfe3-25a008caaa0fe"
            ]
        },
        {
            "sourcerpm": "elephant-0.3-0.8.src.rpm",
            "name": "elephant",
            "checksum": "d35640a0-8613-44bd-9e95-14122834087",
            "summary": "A dummy package of elephant",
            "filename": "elephant-0.3-0.8.noarch.rpm",
            "epoch": 0,
            "version": "0.3",
            "release": "0.8",
            "_id": "d35640a0-8613-44bd-9e95-14122834087",
            "arch": "noarch",
            "children": [
                "href": "pulp/api/v2/content/units/rpm/d35640a0-8613-44bd-9e95-14122834087"
            ]
        }
    ]
}
```
"sourcerpm": "elephant-0.3-0.8.src.rpm",
"name": "elephant",
"checksum": "3e1c70cd1b421328aca6397c5d316145306bb95f65db0955c31372a0a70f3",
"summary": "A dummy package of elephant",
"filename": "elephant-0.3-0.8.noarch.rpm",
"epoch": "0",
"version": "0.3",
"release": "0.8",
"_id": "See1ec3a-7f81-47e6-bbba-11b443251f29",
"arch": "noarch",
"children": []
}
}
]
]
repository_memberships": [
 "scenario_test"
]
]
]
 sourcerpm": "monkey-0.3-0.8.src.rpm",
"name": "monkey",
"checksum": "0e8fa50d0128fbabc7ccc5632e3fa25d39b0280169f6166cb8e2c8d8e8501db1",
"summary": "A dummy package of monkey",
"filename": "monkey-0.3-0.8.noarch.rpm",
"epoch": "0",
"version": "0.3",
"release": "0.8",
"_id": "Bea869197-c155-4c8c-a931-aa5b85bd218c",
"arch": "noarch",
"children": []
}
}
]
]
repository_memberships": [
 "scenario_test"
]
]
]
 sourcerpm": "lion-0.3-0.8.src.rpm",
"name": "lion",
"checksum": "12400dc95c23a4c160725a908716cd3f891585437ab64cd62ea3e8ae4",
"summary": "A dummy package of lion",
"filename": "lion-0.3-0.8.noarch.rpm",
"epoch": "0",
"version": "0.3",
"release": "0.8",
"_id": "7f4f862-7ae7-4b20-b072-0d154a9cc527",
"arch": "noarch",
"children": []
}
}
]
]
repository_memberships": [
 "scenario_test"
]
]
]
 sourcerpm": "cheetah-0.3-0.8.src.rpm",
"name": "cheetah",
"checksum": "422d0baa0cd9d7713ae796e886a23e171578f924f74880debdb7d65b368dse",
"summary": "A dummy package of cheetah",
"filename": "cheetah-0.3-0.8.noarch.rpm",
"epoch": "0",
"version": "0.3",
"release": "0.8",
"_id": "9f6b8fda-dd97-453e-8841-093305c8eac3",
"arch": "noarch",
"children": []
}
}
]
]
repository_memberships": [
 "scenario_test"
]
]
]
 sourcerpm": "giraffe-0.3-0.8.src.rpm",
"name": "giraffe",
"checksum": "f25d67d1d9d9a04f12e657ca323247b43891ac46533e355b82de6d1922009f14",
"summary": "A dummy package of giraffe",
"filename": "giraffe-0.3-0.8.noarch.rpm",
"epoch": "0",
"version": "0.3",
"release": "0.8",
"_id": "9f6b8fda-dd97-453e-8841-093305c8eac3",
"arch": "noarch",
"children": []
}
]
]
repository_memberships": [
 "scenario_test"
]
]
]
 sourcerpm": "squirrel-0.3-0.8.src.rpm",
"name": "squirrel",
"checksum": "25176b9d15f378487c27638aa8aecd015511e253756093cde1c0a878a17d2",
"summary": "A dummy package of squirrel",
"filename": "squirrel-0.3-0.8.noarch.rpm",
"epoch": "0",
"version": "0.3",
"release": "0.8",
"_id": "d3964a9-8613-44bd-9e99-141af2843087",
"arch": "noarch",
"children": []
}
]
]
repository_memberships": [
 "scenario_test"
]
]
]
 sourcerpm": "squirrel-0.3-0.8.src.rpm",
"name": "squirrel",
"checksum": "25176b9d15f378487c27638aa8aecd015511e253756093cde1c0a878a17d2",
"summary": "A dummy package of squirrel",
"filename": "squirrel-0.3-0.8.noarch.rpm",
"epoch": "0",
"version": "0.3",
"release": "0.8",
"_id": "d3964a9-8613-44bd-9e99-141af2843087",
"arch": "noarch",
"children": []
}
]
]
repository_memberships": [
 "scenario_test"
]
]
]
Request # 7: Fetch Errata ids for repository

POST /pulp/api/v2/repositories/scenario_test/search/units/

Backend Service: pulp

Description:

Request body

```
{
  "criteria": {
    "type_ids": ["erratum"],
    "fields": {
      "unit": [ ]
    },
    "association": {
      "unit_id": [ ]
    }
  }
}
```

Response body

```
[
  {
    "metadata": {
      "_id": "4b12197f-28b0-4d5c-bf8d-057dc0b378f1",
      "_content_type_id": "erratum"
    },
    "id": {"$oid": "58dd7595e6919db96421ea1e"},
    "unit_id": "4b12197f-28b0-4d5c-bf8d-057dc0b378f1",
    "unit_type_id": "erratum"
  },
  {
    "metadata": {
      "_id": "5bc4a860-2872-461a-8061-b30626274f61",
      "_content_type_id": "erratum"
    },
    "id": {"$oid": "58dd7595e6919db96421ea1f"},
    "unit_id": "5bc4a860-2872-461a-8061-b30626274f61",
    "unit_type_id": "erratum"
  },
  {
    "metadata": {
      "_id": "c1227ae-b619-40cc-afbc-75c92e78ca13",
      "_content_type_id": "erratum"
    },
    "id": {"$oid": "58dd7595e6919db96421ea1d"},
    "unit_id": "c1227ae-b619-40cc-afbc-75c92e78ca13",
    "unit_type_id": "erratum"
  }
]
```

Request # 8: Fetch errata units.

POST /pulp/api/v2/content/units/erratum/search/

Backend Service: pulp

Description: Using the previously fetched ids, we fetch all errata in the repo.
Request body

```json
{
  "criteria": {
    "limit": 3,
    "skip": 0,
    "filters": {
      "$in": [
        "4b12197f-28b0-4d5c-bf8d-057dc0b378f1",
        "5bc4a860-2872-461a-8061-b30626274615",
        "c12277ae-b619-40cc-afbc-75c92a78ca13"
      ]
    }
  },
  "include_repos": true
}
```

Response body

```json
[
  {
    "repository_memberships": [
      "scenario_test"
    ],
    "_href": "https://rhn.redhat.com/errata/RHSA-2010-0858.html",
    "issued": "2010-11-10 00:00:00",
    "references": [
      {
        "href": "https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=627882",
        "type": "bugzilla",
        "id": "627882",
        "title": "CVE-2010-0405 bzip2: integer overflow flaw in BZ2_decompress"
      },
      {
        "href": "https://www.redhat.com/security/data/cve/CVE-2010-0405.html",
        "type": "cve",
        "id": "CVE-2010-0405",
        "title": "CVE-2010-0405"
      },
      {
        "href": "http://www.redhat.com/security/updates/classification/#important",
        "type": "other",
        "id": null,
        "title": null
      }
    ],
    "pulp_user_metadata": {
      "_content_type_id": "erratum",
      "id": "RHSA-2010:0858",
      "from": "security@redhat.com",
      "severity": "Important",
      "title": "Important: bzip2 security update",
      "version": "3",
      "reboot_suggested": false,
      "type": "security",
      "pkglist": [
        {
          "_pulp_repo_id": "scenario_test",
          "packages": [
            {
              "src": "bzip2-1.0.5-7.el6_0.src.rpm",
              "name": "bzip2-devel",
              "sum": {
                "sha256": "ea67c664da1ff96a6dc94d33009b73d8fab31b59824183fb45e9ba2ebf82d583"
              },
              "filename": "bzip2-devel-1.0.5-7.el6_0.i686.rpm",
              "epoch": 0,
              "version": "1.0.5",
              "release": "7.el6_0",
              "arch": "i686"
            },
            {
              "src": "bzip2-1.0.5-7.el6_0.src.rpm",
              "name": "bzip2-libs",
              "sum": {
                "sha256": "c9f064a8862573fb9f2a6af7c3621f1940b492df2edfc2ebbdcc0b8305ff51147"
              },
              "filename": "bzip2-libs-1.0.5-7.el6_0.i686.rpm",
              "epoch": 0,
              "version": "1.0.5",
              "release": "7.el6_0",
              "arch": "i686"
            }
          ]
        }
      ]
    }
  }
]```
```
"filename": "bzip2-1.0.5-7.el6_0.i686.rpm",
"epoch": "0",
"version": "1.0.5",
"release": "7.el6_0",
"arch": "i686"
],
"src": "bzip2-1.0.5-7.el6_0.src.rpm",
"name": "bzip2",
"sum": {
"sha256": "b8a3f72cb2b0d89ba737099a98bfb82a4f68e02d31b884c02db97f766c3d5c2"
},
"filename": "bzip2-1.0.5-7.el6_0.x86_64.rpm",
"epoch": "0",
"version": "1.0.5",
"release": "7.el6_0",
"arch": "x86_64"
],
"src": "bzip2-1.0.5-7.el6_0.src.rpm",
"name": "bzip2-devel",
"sum": {
"sha256": "7f63124e4655b7c92d23ec4c382265f5d3746568853df7501c85e058e74b5cf6"
},
"filename": "bzip2-devel-1.0.5-7.el6_0.x86_64.rpm",
"epoch": "0",
"version": "1.0.5",
"release": "7.el6_0",
"arch": "x86_64"
],
"src": "bzip2-1.0.5-7.el6_0.src.rpm",
"name": "bzip2-libs",
"sum": {
"sha256": "802f4399dbdd01476e254c3b32c40a0ff59c5f23e45fa488c6917ce8904d6b4d"
},
"filename": "bzip2-libs-1.0.5-7.el6_0.x86_64.rpm",
"epoch": "0",
"version": "1.0.5",
"release": "7.el6_0",
"arch": "x86_64"
],
"name": "Red Hat Enterprise Linux Server (v. 6 for 64-bit x86_64)",
"short": "rhel-x86_64-server-6"
],
"status": "final",
"updated": "2010-11-10 00:00:00",
"description": "bzip2 is a freely available, high-quality data compressor. It provides both libbz2 library must be restarted for the update to take effect.",
"last_updated": "2017-03-30T21:16:05Z",
"pushcount": "",
"rights": "Copyright 2010 Red Hat Inc",
"solution": "Before applying this update, make sure all previously-released errata relevant to your system have been applied. This update is available via the Red Hat Network. Details on how to use the Red Hat Network to apply this update are available at http://kbase.redhat.com/faq/docs/DOC-11259",
"summary": "Updated bzip2 packages that fix one security issue",
"release": "",
"id": "RHEA-2010:0002",
"from": "lzap+pub@redhat.com",
"severity": "",
"title": "One package errata",
"children": []
}]```
Request # 9: Fetch package group units for repository

POST /pulp/api/v2/repositories/scenario_test/search/units/

Backend Service: pulp

Description:

Request body

```json
{
  "criteria": {
    "type_ids": [
      "package_group"
    ],
    "fields": {
      "unit": [ ]
    }
  }
}
```

Response body

```json
"filename": "elephant-0.3-0.8.noarch.rpm",
"epoch": null,
"version": "0.3",
"release": "0.8",
"arch": "noarch"
```
Request # 10: Fetch package group ids for repository

POST /pulp/api/v2/content/units/package_group/search/

Backend Service: pulp

Description: Using the previously fetched ids, we fetch all package groups in the repo.

Request body

```json
{
  "criteria": {
    "limit": 2,
    "skip": 0,
    "filters": {
      "_id": {
        "$in": [
          "919baa7e-e944-4602-b3ed-3aef2ae5b509",
          "c55b5b16-7501-4863-8e3f-7f520bc795fd"
        ]
      }
    }
  },
  "include_repos": true
}
```

Response body

```json
[
  {
    "metadata": {
      "_id": "919baa7e-e944-4602-b3ed-3aef2ae5b509",
      "_content_type_id": "package_group"
    },
    "_id": {
      "$oid": "58dd7595e6919db96421ea21"
    },
    "unit_id": "919baa7e-e944-4602-b3ed-3aef2ae5b509",
    "unit_type_id": "package_group"
  },
  {
    "metadata": {
      "_id": "c55b5b16-7501-4863-8e3f-7f520bc795fd",
      "_content_type_id": "package_group"
    },
    "_id": {
      "$oid": "58dd7595e6919db96421ea20"
    },
    "unit_id": "c55b5b16-7501-4863-8e3f-7f520bc795fd",
    "unit_type_id": "package_group"
  }
]
```
Request # 11: Request applicability generation for consumers bound to the repository

POST /pulp/api/v2/repositories/actions/content/regenerate_applicability/

Backend Service: pulp

Description:

Request body
Response body

```json
{
    "parallel": true,
    "repo_criteria": {
        "filters": [
            {
                "id": {
                    "$in": ["scenario_test"]
                }
            }
        ]
    }
}
```

Request # 12: Monitor task group status
GET /pulp/api/v2/task_groups/b0e268a7-f4bf-4598-90a3-5fc3b562cc95/state_summary/
Backend Service: pulp
Description: Monitor status of the applicability generation
Request body
None
Response body

```json
{
    "accepted": 0,
    "finished": 0,
    "running": 0,
    "canceled": 0,
    "waiting": 0,
    "skipped": 0,
    "suspended": 0,
    "error": 0,
    "total": 0
}
```

Response body

```json
{
    "group_id": "b0e268a7-f4bf-4598-90a3-5fc3b562cc95",
    "_href": "/pulp/api/v2/task_groups/b0e268a7-f4bf-4598-90a3-5fc3b562cc95/"
}
```

Request # 13: Fetch repository details
GET /pulp/api/v2/repositories/scenario_test/
Backend Service: pulp
Description: Unclear why
Request body
None
Response body

```json
{
    "scratchpad": {
        "checksum_type": "sha256"
    },
    "display_name": "Scenario yum product",
    "description": null,
    "distributors": [
        {
            "repo_id": "scenario_test",
            "last_updated": "2017-03-30T21:15:43Z",
            "_href": "/pulp/api/v2/repositories/scenario_test/distributors/scenario_test_clone/"
        },
        {
            "last_override_config": {},
            "last_publish": null,
            "distributor_type_id": "yum_clone_distributor",
            "auto_publish": false,
            "scratchpad": {},
            "_ns": "repo_distributors",
            "_id": {
                "$oid": "58dd757f418a8a8a04f88de7da"
            }
        }
    ]
}
```
"config": {  
  "destination_distributor_id": "scenario_test_clone"  
},  
"repo_id": "scenario_test",  
"last_updated": "2017-03-30T21:16:05Z",  
"_href": "/pulp/api/v2/repositories/scenario_test/distributors/scenario_test/",  
"last_overwrite_config": {  
},  
"last_publish": "2017-03-30T21:16:05Z",  
"distributor_type_id": "yum_distributor",  
"auto_publish": true,  
"scratchpad": {  
},  
"_ns": "repo_distributors",  
"_id": {  
  "$oid": "58dd757f418a8a04f88de7d8"  
},  
"config": {  
  "checksum_type": "sha256",  
  "protected": true,  
  "http": false,  
  "https": true,  
  "relative_url": "scenario_test"  
},  
"id": "scenario_test"  
},  
"repo_id": "scenario_test",  
"last_updated": "2017-03-30T21:15:43Z",  
"_href": "/pulp/api/v2/repositories/scenario_test/distributors/export_distributor/",  
"last_overwrite_config": {  
},  
"last_publish": null,  
"distributor_type_id": "export_distributor",  
"auto_publish": false,  
"scratchpad": {  
},  
"_ns": "repo_distributors",  
"_id": {  
  "$oid": "58dd757f418a8a04f88de7d9"  
},  
"config": {  
  "http": false,  
  "https": false,  
  "relative_url": "scenario_test"  
},  
"id": "export_distributor"  
],  
"last_unit_added": "2017-03-30T21:16:05Z",  
"notes": {  
  "repo-type": "rpm-repo"  
},  
"last_unit_removed": null,  
"content_unit_counts": {  
  "package_group": 2,  
  "distribution": 1,  
  "package_category": 1,  
  "rpm": 8,  
  "erratum": 3  
},  
"_ns": "repos",  
"importers": [  
{  
  "repo_id": "scenario_test",  
  "last_updated": "2017-03-30T21:15:43Z",  
  "_href": "/pulp/api/v2/repositories/scenario_test/importers/yum_importer/",  
  "_ns": "repo_importers",  
  "importer_type_id": "yum_importer"  
},  
"_id": {  
  "$oid": "58dd757f418a8a04f88de7d7"  
},  
"config": {  
  "feed": "file:///var/www/test_repos/zoo",  
  "ssl_validation": true,  
  "remove_missing": true,  
  "download_policy": "immediate"  
},  
"id": "yum_importer"  
},  
"locally_stored_units": 15,
Repository Create

Request # 1: Create Content

POST /candlepin/owners/scenario_test/content/

Backend Service: candlepin

Description: Create Content object for repository, for subscription-manager content access

Request body

```json
{
    "name": "Scenario yum product",
    "contentUrl": "/custom/Scenario_Product/Scenario_yum_product",
    "type": "yum",
    "label": "scenario_test_Scenario_Product_Scenario_yum_product",
    "metadataExpire": 1,
    "vendor": "Custom"
}
```

Response body

```json
{
    "created": "2017-03-30T21:15:43+0000",
    "updated": "2017-03-30T21:15:43+0000",
    "uuid": "4028f9515b20f31e015b2112fba4000d",
    "id": "1490908543901",
    "type": "yum",
    "label": "scenario_test_Scenario_Product_Scenario_yum_product",
    "name": "Scenario yum product",
    "vendor": "Custom",
    "contentUrl": "/custom/Scenario_Product/Scenario_yum_product",
    "requiredTags": null,
    "gpgUrl": null,
    "metadataExpire": 1,
    "modifiedProductIds": [],
    "arches": null,
    "releaseVer": null
}
```

Request # 2: Associate content object

POST /candlepin/owners/scenario_test/products/272869743822/content/1490908543901

Backend Service: candlepin

Description: Add the Content object to the product

Request body

None

Response body
Request # 3: Retrieve candlepin environment

GET /candlepin/environments/119c4753ff6d3b7bd0b76de6d5a5f94a

Backend Service: candlepin

Description: Retrieve the environment object (TODO WHY?)

Request body

None

Response body

```json
{
    "owner": {
        "id": "4028f9515b2031e015b2112fc03000e",
        "key": "scenario_test",
        "displayName": "scenario_test",
        "href": "/owners/scenario_test"
    },
    "name": "Library",
    "description": null,
    "id": "119c4753ff6d3b7bd0b76de6d5a5f94a",
    "environmentContent": {
        "created": "2017-03-30T21:15:41+0000",
        "updated": "2017-03-30T21:15:41+0000"
    }
}
```

Request # 4: Create Pulp Repository

POST /pulp/api/v2/repositories/

Backend Service: pulp

Description:

```json
{
    "created": "2017-03-30T21:15:42+0000",
    "updated": "2017-03-30T21:15:44+0000",
    "uuid": "4028f9515b2031e015b2112fc03000e",
    "id": "27269743822",
    "name": "Scenario Product",
    "multiplier": 1,
    "attributes": {
        "created": "2017-03-30T21:15:42+0000",
        "updated": "2017-03-30T21:15:44+0000",
        "name": "arch",
        "value": "ALL"
    },
    "dependentProductIds": [],
    "href": "/products/4028f9515b2031e015b2112fc03000e",
    "productContent": [
        {
            "created": "2017-03-30T21:15:43+0000",
            "updated": "2017-03-30T21:15:43+0000",
            "uuid": "4028f9515b2031e015b2112b40000d",
            "id": "149000545891",
            "type": "yum",
            "name": "Scenario yum product",
            "vendor": "Custom",
            "contentUrl": "/custom/Scenario_Product/Scenario_yum_product",
            "requiredTags": null,
            "gpgUrl": null,
            "metadataExpire": 1,
            "modifiedProductIds": [],
            "arches": null,
            "releaseVer": null,
            "enabled": true
        }
    ],
    "owner": {
        "id": "4028f9515b2031e015b21122550004",
        "key": "scenario_test",
        "displayName": "scenario_test",
        "href": "/owners/scenario_test"
    },
    "name": "Library",
    "description": null,
    "id": "119c4753ff6d3b7bd0b76de6d5a5f94a",
    "environmentContent": {
        "created": "2017-03-30T21:15:41+0000",
        "updated": "2017-03-30T21:15:41+0000"
    }
}
```
Request body

```
{
    "id": "scenario_test",
    "display_name": "Scenario yum product",
    "importer_type_id": "yum_importer",
    "importer_config": {
        "feed": "file:///var/www/test_repos/zoo",
        "ssl_ca_cert": null,
        "ssl_client_cert": null,
        "ssl_client_key": null,
        "ssl_validation": true,
        "download_policy": "immediate",
        "remove_missing": true
    },
    "notes": {
        "_repo-type": "rpm-repo"
    },
    "distributors": [
        {
            "distributor_type_id": "yum_distributor",
            "distributor_config": {
                "relative_url": "scenario_test",
                "http": false,
                "https": true,
                "protected": true
            },
            "auto_publish": true,
            "distributor_id": "scenario_test"
        },
        {
            "distributor_type_id": "export_distributor",
            "distributor_config": {
                "http": false,
                "https": false
            },
            "auto_publish": false,
            "distributor_id": "export_distributor"
        },
        {
            "distributor_type_id": "yum_clone_distributor",
            "distributor_config": {
                "destination_distributor_id": "scenario_test"
            },
            "auto_publish": false,
            "distributor_id": "scenario_test_clone"
        }
    ]
}
```

Response body

```
{
    "scratchpad": {
        "display_name": "Scenario yum product",
        "description": null,
        "last_unit_added": null,
        "notes": {
            "_repo-type": "rpm-repo"
        },
        "last_unit_removed": null,
        "content_unit_counts": {
            "_ns": "repos",
            "_id": {
                "$oid": "58dd757f418a8a04f88de7d6"
            },
            "id": "scenario_test",
            "_href": "/pulp/api/v2/repositories/scenario_test/"
        }
    }
}
```

Request # 5: Add Content To environment

POST /candlepin/environments/119c4753ff6d3b7bd0b76de6d5a5f94a/content

Backend Service: candlepin

Description: Associates content object to “Library” environment, so it is accessible by clients registered to Library

Request body
Request # 6: Fetch repository information

GET /pulp/api/v2/repositories/scenario_test/

Backend Service: pulp

Description:

Request body

None

Response body

```json
{
    "contentId": "1490908543901"
}
```
Request # 7: Publish repository metadata

POST /pulp/api/v2/repositories/scenario_test/actions/publish/

Backend Service: pulp

Description: Publish the repository, generating empty metadata so that existing clients of that product do not get a 404.

Request body

```
{
  "id": "scenario_test",
  "override_config": {
    "force_full": false
  }
}
```

Response body
Request # 8: Poll Task

GET /pulp/api/v2/tasks/dfdb7cae-e465-4885-b3a4-7bb29343b69e/

Backend Service: pulp

Description: Monitor the task status of the publish

Total Requests for this URL: 6

Request body
None

Response body

```json
{
  "exception": null,
  "task_type": "pulp.server.managers.repo.publish.publish",
  "_href": "/pulp/api/v2/tasks/dfdb7cae-e465-4885-b3a4-7bb29343b69e/",
  "task_id": "dfdb7cae-e465-4885-b3a4-7bb29343b69e",
  "tags": [
    "pulp:repository:scenario_test",
    "pulp:action:publish"
  ],
  "finish_time": "2017-03-30T21:15:44Z",
  "_ns": "task_status",
  "start_time": "2017-03-30T21:15:44Z",
  "traceback": null,
  "spawned_tasks": []
}
```

```json
{ "progress_report": { "scenario_test": [ { "num_success": 1, "description": "initializing repo metadata", "step_type": "initialize_repo_metadata", "items_total": 1, "state": "FINISHED", "error_details": [ ] }, { "details": "", "num_failures": 0, "step_id": "e3a30d24-1111-4080-92f8-0a3dc979ae9c", "num_processed": 1 } ] }, { "num_success": 0, "description": "Publishing Distribution files", "step_type": "distribution", "items_total": 0, "state": "FINISHED", "error_details": [ ] }, { "details": "", "num_failures": 0, "step_id": "c2edbc90-716d-4570-a442-bd0c22748188", "num_processed": 0 } ] }, { "num_success": 0, "description": "Publishing RPMs", "step_type": "rpms", "items_total": 0, "state": "FINISHED", "error_details": [ ] }, { "details": "", "num_failures": 0, "step_id": "bfeaa715-21d9-4cf0-af2a-5c5e0c1c172b", "num_processed": 0 } ] }
```
"num_success": 0,
"description": "Publishing Delta RPMs",
"step_type": "drpms",
"items_total": 1,
"state": "SKIPPED",
"error_details": [

],
"details": "",
"num_failures": 0,
"step_id": "47621356-70b3-4e77-a8f2-ee1ec0c5c2ee",
"num_processed": 0
],

"num_success": 0,
"description": "Publishing Errata",
"step_type": "errata",
"items_total": 0,
"state": "FINISHED",
"error_details": [

],
"details": "",
"num_failures": 0,
"step_id": "8e46b5e2-89f5-4ef5-80bd-0d2aab9dd2a0",
"num_processed": 0
],

"num_success": 0,
"description": "Publishing Comps file",
"step_type": "comps",
"items_total": 0,
"state": "FINISHED",
"error_details": [

],
"details": "",
"num_failures": 0,
"step_id": "050aba89-abec-4949-b3cf-28409d9519e0",
"num_processed": 0
],

"num_success": 0,
"description": "Publishing Metadata.",
"step_type": "metadata",
"items_total": 0,
"state": "FINISHED",
"error_details": [

],
"details": "",
"num_failures": 0,
"step_id": "70efd383-905d-4fe2-8c9b-4e18311e76d5",
"num_processed": 0
],

"num_success": 1,
"description": "Closing repo metadata",
"step_type": "close_repo_data",
"items_total": 1,
"state": "FINISHED",
"error_details": [

],
"details": "",
"num_failures": 0,
"step_id": "e3ae5910-8f8d-4e53-abd6-007a9495e0b4",
"num_processed": 1
],

"num_success": 0,
"description": "Generating sqlite files",
"step_type": "generate_sqlite",
"items_total": 1,
"state": "SKIPPED",
"error_details": [

],
"details": "",
"num_failures": 0,
"step_id": "ae023865-a176-4434-b8ab-77a080306602",
"num_processed": 0
],

"num_success": 0,
"description": "Generating HTML files",
"step_type": "repoview",
"items_total": 1,
"state": "SKIPPED",
"error_details": [

],
"details": "",
"num_failures": 0,
"step_id": "9e023865-a176-4434-b8ab-77a080306602",
"num_processed": 0
]
null
"num_failures": 0,
"step_id": "4d7ca329-7b33-478e-a0a5-777771d8c7c6",
"num_processed": 0
],

"num_success": 1,
"description": "Publishing files to web",
"step_type": "publish_directory",
"items_total": 1,
"state": "FINISHED",
"error_details": []
],

details": 
"num_failures": 0,
"step_id": "0986744f-85dc-43ff-9e0f-10c814a8d5e8",
"num_processed": 1
],

"num_success": 1,
"description": "Writing Listings File",
"step_type": "initialize_repo_metadata",
"items_total": 1,
"state": "FINISHED",
"error_details": []
],

details": 
"num_failures": 0,
"step_id": "04b588ee-11fe-4b09-8a42-2ea465247c0b",
"num_processed": 1
],

null
"result": 
"result": "success",
"exception": null,
"repo_id": "scenario_test",
"started": "2017-03-30T21:15:44Z",
"ns": "repo_publish_results",
"updated": "2017-03-30T21:15:44Z",
"traceback": null,
"distributor_type_id": "yum_distributor",
"summary": 
"generate sqlite": "SKIPPED",
"initialize_repo_metadata": "FINISHED",
"rpms": "FINISHED",
"close_repo_metadata": "FINISHED",
"drpms": "SKIPPED",
"comps": "FINISHED",
"distribution": "FINISHED",
"repopview": "SKIPPED",
"publish_directory": "FINISHED",
"errata": "FINISHED",
"metadata": "FINISHED"
],
"error_message": null,
"distributor_id": "scenario_test",
"id": "58dd7580418a8a0648b9bc40",
"details": []
]
<table>
<thead>
<tr>
<th>ID</th>
<th>Step Type</th>
<th>Step Details</th>
<th>Items Total</th>
<th>State</th>
<th>Error Details</th>
<th>Num Failures</th>
<th>Num Success</th>
<th>Details</th>
<th>Num Processed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>rpms</td>
<td>Publishing RPMs</td>
<td>0</td>
<td>FINISHED</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>drpms</td>
<td>Publishing Delta RPMs</td>
<td>1</td>
<td>SKIPPED</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>errata</td>
<td>Publishing Errata</td>
<td>0</td>
<td>FINISHED</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>comps</td>
<td>Publishing Comps file</td>
<td>0</td>
<td>FINISHED</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>metadata</td>
<td>Publishing Metadata</td>
<td>0</td>
<td>FINISHED</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>close_repo_metadata</td>
<td>Closing repo metadata</td>
<td>1</td>
<td>FINISHED</td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>generate_sqlite</td>
<td>Generating sqlite files</td>
<td>1</td>
<td>SKIPPED</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>