FOREMAN

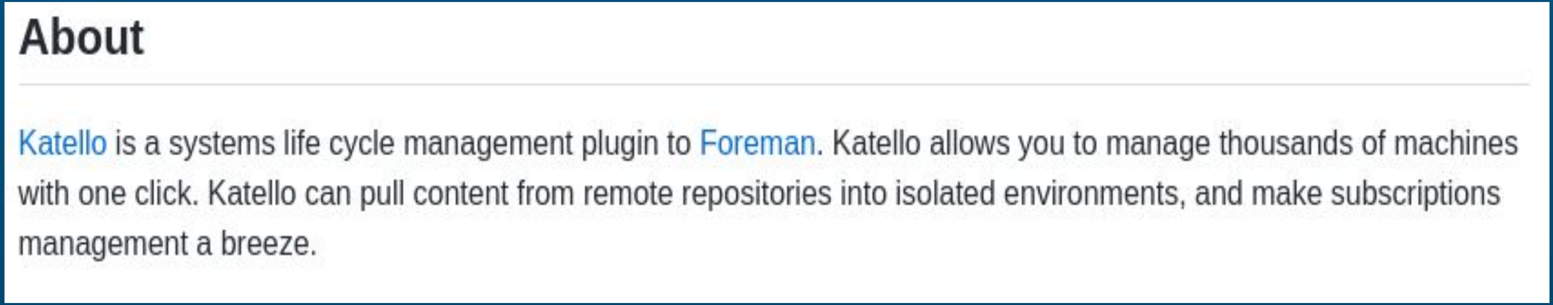
Katello

Pulp integration

By - Katello team
Find us on
#theforeman-dev

What is katello?

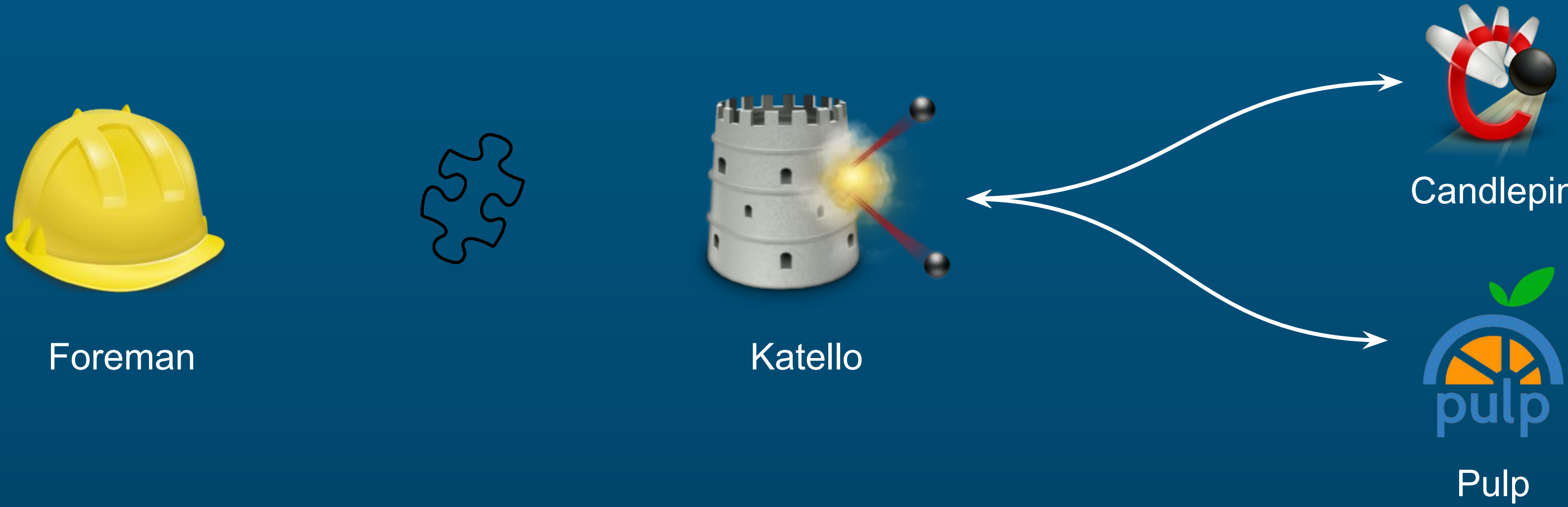
Source of truth: https://github.com/Katello/katello/blob/master/README.md


About

Katello is a systems life cycle management plugin to Foreman. Katello allows you to manage thousands of machines with one click. Katello can pull content from remote repositories into isolated environments, and make subscriptions management a breeze.

Documentation : https://docs.theforeman.org/nightly/Quickstart/index-katello.html

The Stack



Overview:

FOREMAN

Subscription Management

1. The right to receive the associated content from Katello.
2. Subscriptions are most easily managed with activation keys.
3. Access to content can be isolated using a mix of organizations, lifecycle environments and content views.
4. Entitlements are generated for all content hosts registered to Katello.

Content Management

1. Ability to create Repositories, download content and curate the content into products and content views.
2. Snapshot a set of repositories using content views.
3. Provide remote execution of content actions like installing/removing packages/errata on host(s).
4. Deploy content mirrors via “smart proxies” anywhere

Content management demo

1. Content workflow:
   a) Repositories - Creation and sync
   b) Content views , versions, filters
   c) Look at publications, distributions
2. Register a host using an activation key
3. Consume content on the host
4. Perform host actions from the Katello UI
5. Smart proxy workflow.

2022 in review:

Pulp versions in use:

1. Katello 4.5, 4.6 -> Pulpcore 3.18
2. Katello 4.7 (Not yet released) -> Pulpcore 3.21

Upstream source of truth: https://yum.theforeman.org/katello/
Downstream source of truth: Ohsnap

Current Pulp Packages on Katello

FOREMAN

```
pulpcore-selinux-1.3.2-1.el8.x86_64
python39-pulp-rpm-3.18.5-1.el8.noarch
python39-pulp-ansible-0.15.0-1.el8.noarch
python39-pulp-file-1.11.1-1.el8.noarch
python39-pulp-cli-0.14.0-4.el8.noarch
python39-pulp-certguard-1.5.5-1.el8.noarch
python39-pulp-ostree-2.0.0-0.8.a6.el8.noarch
python39-pulpcore-3.21.0-1.el8.noarch
python39-pulp-container-2.14.0-2.el8.noarch
python39-pulp-python-3.7.2-2.el8.noarch
python39-pulp-deb-2.20.0-1.el8.noarch
```

2022 in review:

Notable new features for content management:

1. Content view comparison
2. Alternate content sources (ACS)
3. Content import/export

Content view comparison

Demo:

1. Create Content View. (Add all content type repos)
2. Publish
3. Remove a couple of content repos and publish again
4. Show comparison

ACS

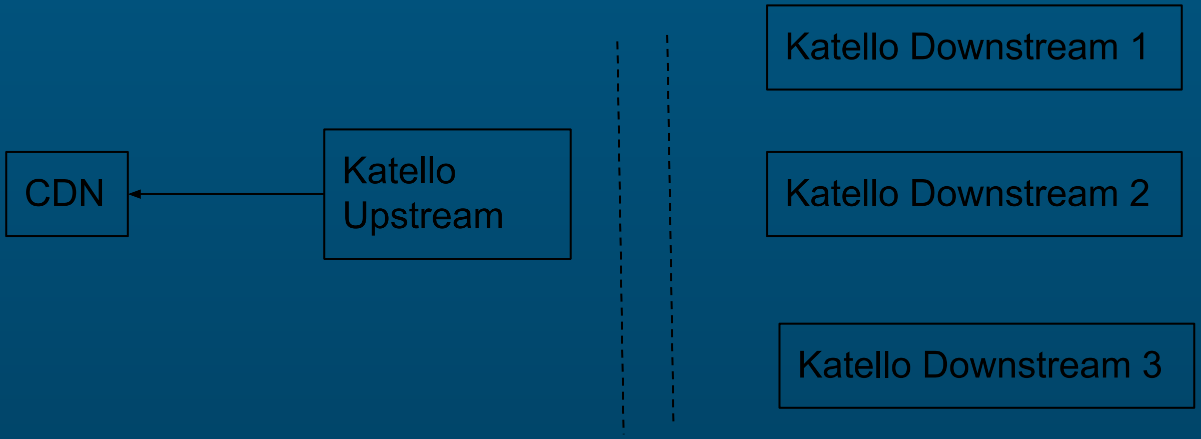
Demo:

1. Create ACS. (Custom/RHUI and Simplified)
2. Show orchestration/actions
3. Show pulp ACS / remote

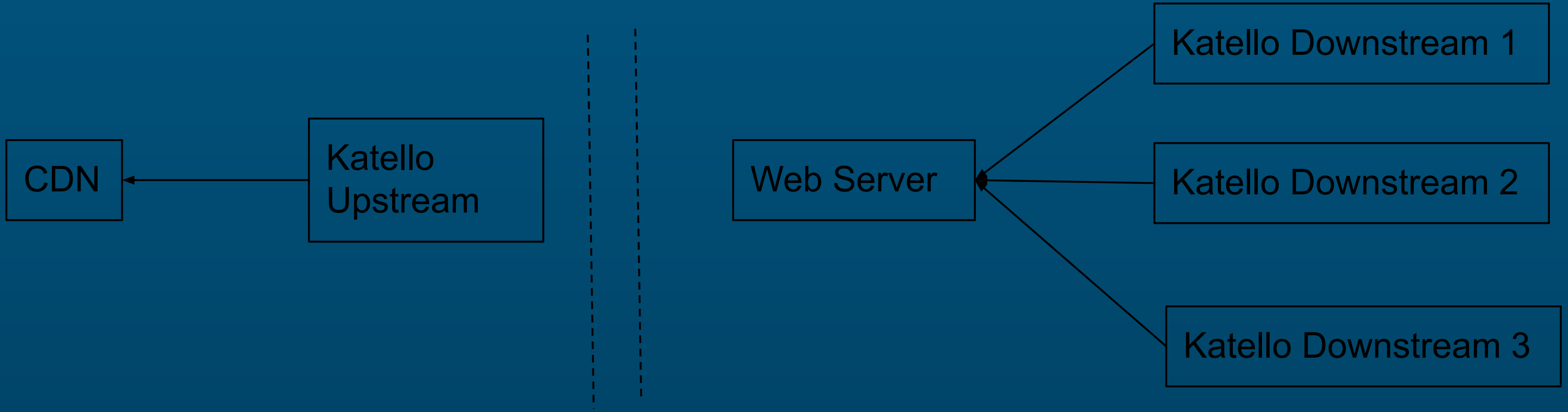
Content Import/Export

Scenarios

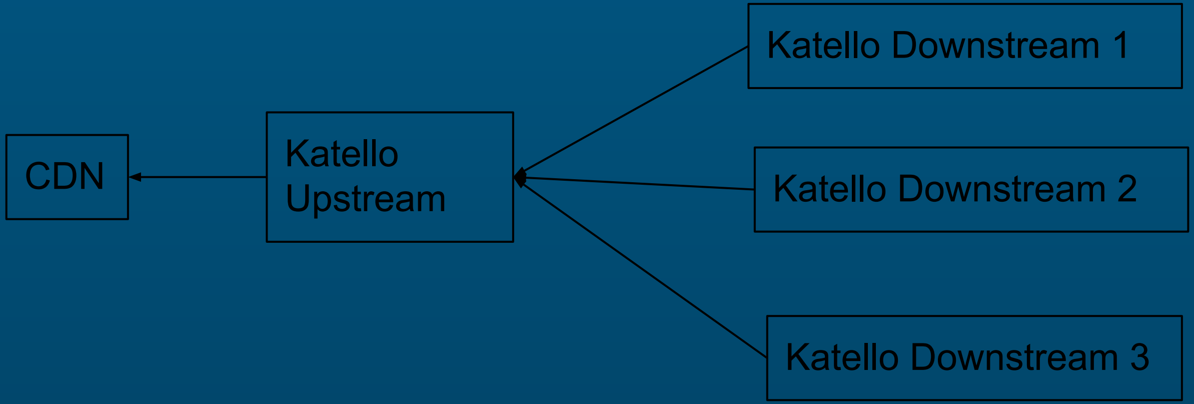
Air Gapped - AKA Export Sync



Air Gapped + Webserver



Connected


CDN
Katello Upstream
Katello Downstream 1
Katello Downstream 2
Katello Downstream 3

Export Types

1) Complete
2) Incremental

Formats

1) Default - aka Importable
2) Syncable

Export in the Default Importable format

1) Uses the Pulp Core Exporter (specifies versions or start versions in the case of incremental)
2) Exports via pulp core export
3) Generates some metadata to be used by Katello on the import side

Import in the Default Importable format

1) Auto creates repositories, content views and other artifacts based on the metadata.
2) Uses the Pulp Core Importer and Import to import versions
3) Copies the contents of CV repositories to the library repos.

Export/Import in the Default format

Pros

1) More natural, pulp is aware of exactly what is being exported and imported.
2) Versions and Changesets are used in incremental exports which offer more reliable way to track import/exports. import side

Cons

1) Needs at least 2X space to export. Which can be expensive. For example: RHEL 7 is 50GB
2) The plugins need to be same X.Y versions.
3) The export format is not similar to a yum repo which affects the Air Gapped with web server use case.

Export in the Syncable format

1) Uses the Pulp Core FS Exporter and Exports (specifies publication to export for metadata.)
2) Uses hard links instead of write.
3) Exports look like proper a yum repository with syncable content
4) Generates `Listing` files and additional metadata to be used by Katello on the import side.

Import in the Syncable format

1) Auto creates repositories, content views and other artifacts based on the metadata.
2) Repository remotes point to a file system/https directory with the exported data.
3) Uses pulp to sync repos and publishes content view etc.

Export/Import in the Syncable format

Pros

1) Hardlinks imply 1X space and time (assuming same inode.)
2) Exported contents look similar to what you'd see in a CDN.
3) More flexibility for Air Gapped with web server use case.

Cons

1) Needs katello to generate listing files (ideally pulp could add that option)
2) No incremental exports
3) Harder to evaluate the contents of the import since it is synced from a remote repository.

Wishes and what could be improved

FOREMAN

1. Better logging and better error messages
   a. Ex: Refresh an ACS with wrong URL
2. Consistent support for N-1 smart proxy syncs
   a. Requires newer API bindings to talk to an older Pulp https://github.com/pulp/pulp_container/issues/1122
3. Help with content copying
   a. Katello currently tells Pulp what to copy during content view publishing
   b. Copy commands can be large since each unit is specified
   c. Pulp has more knowledge about the content
4. Import/Export
   a. Incremental Export for the fs exporter.
   b. FS exporter ideally should also generate cdn style listing files (may be some flag).