KATELLO AND ANSIBLE FOR AUTOMATED TESTING AND RELEASING OF PACKAGES

$ WHOAMI

Evgeni Golov

Software Engineer at Red Hat

ex-Consultant at Red Hat

Debian and Grml Developer

♥ FOSS ♥

♥ automation ♥

MOTIVATION

- you build a software product
- you ship the product as distribution packages to your customers
- the product has dependencies outside a base OS (Ruby? node.js? Django?)
- unit tests are great, but you also need to test the shipped bits

ANSIBLE

$ WHATIS ANSIBLE

- radically simple IT automation engine
- contains a big number of modules to execute actions and ensure state on target hosts
- easily extended by self-written modules
- integrates well with REST APIs

ANSIBLE TERMINOLOGY

- **Module** - discrete units of code that can be used from the command line or in a playbook task to execute an action or ensure a state
- **Task** - *Module* invocation with a set of parameters
- **Play** - list of *Task*s to be executed against a set of hosts
- **Playbook** - file containing one or more *Play*s

KATELLO

$ WHATIS KATELLO

- plug-in to Foreman
- adds content management functionality (RPM, DEB, Puppet, Containers, Files)
- allows to group content for tailored presentation to consumers
- allows snapshots of content for versioning

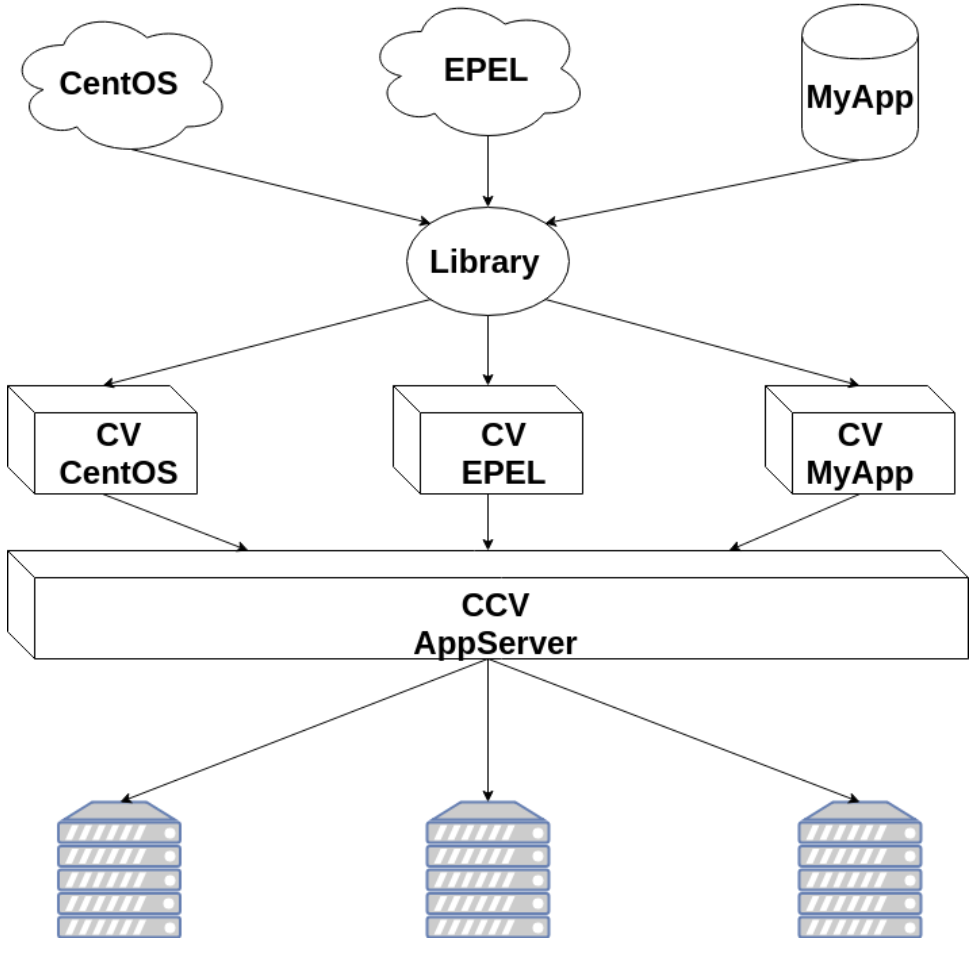
KATELLO TERMINOLOGY

- **Repository** - Collection of content
- **Product** - Collection of related repositories (CentOS 7 distribution with repositories for `i686` and `x86_64`)
- **Lifecycle Environment** - Environment/stage in your deployment cycle (Test, QA, Production)
  - **Library** - special LE that receives the content first

KATELLO TERMINOLOGY

- **Content View** - Selection of repositories (CentOS 7 + EPEL 7)
  - **Publish** creates a snapshot (*Version*) of the selected repositories available to *Library*
  - **Promote** copies a published *Content View Version* to another LE
- **Composite Content View** - Selection of Content Views (base OS + Application)
  - can be *published* and *promoted* like a CV

KATELLO EXAMPLE



STAGING CHANGES WITH KATELLO

- every time a (Composite) Content View is published, a new *Version* is created
- this version can be made available to clients by promoting it to a certain Lifecycle Environment
- you can revert to older versions, if problems are found after a promotion

STAGING CHANGES WITH KATELLO (EXAMPLE)

- **DEV** moving fast, getting changes on every commit
- **TEST** getting changes daily, after a minimal gating happened
- **QA** getting changes weekly, after a basic set of tests passed
- **PROD** getting changes whenever **QA** is happy

TESTING WITH KATELLO AND ANSIBLE

ARCHITECTURE OVERVIEW

- Source in Git (GitLab)
- Jenkins is the main executor, triggered by GitLab
- Katello is the package store
- Ansible is used by Jenkins to interact with the Katello API

TEST WORKFLOW

- Jenkins builds packages on every change (using Koji)
- Packages are synced to Katello
- Katello also syncs external packages (RHEL, RHSCL)
- Jenkins creates/updates ContentView (RHEL, RHSCL, Packages from Koji)
- Jenkins tests the content in *Library* by installing the software and running end-to-end tests
- Jenkins promotes ContentView to *Test* and *QA*

PACKAGE BUILDING

On every change to the source, the following steps are executed:

- a new source tarball is generated
- the RPM `.spec` is updated
- the RPM is built using Koji

PACKAGE TESTING

Jenkins runs a daily pipeline which:

- Synchronizes the packages from Koji into Katello (*Library*)
- Executes a test Ansible playbook in a Vagrant VM
- When the playbooks finishes successfully, the Content is promoted to *Test*

PACKAGE TESTING

- We use forklift for testing
- Set of Ansible playbooks and Vagrant files
  - Create Vagrant VMs
  - Configure package sources
  - Install Katello and a Content Proxy
  - Execute `bats` tests that verify the functionality of the setup
- Same setup can be used on your laptop (if it has enough RAM)

PACKAGE TESTING

- Synchronization is executed via `katello_sync` from foreman-ansible-modules
- Content View is published via `katello_content_view_publish`
- Promotion happens via `katello_content_view_version_promote`

FURTHER TESTING AND RELEASING

- Daily tests are limited and take "only" ~1h
- Once a week content from *Test* is promoted to *QA*
- This triggers a large test-suite (>24h!)
- Plus manual verification of features and fixed bugs that have no automated tests
- After successful verification, the software is released

ARCHIVING RELEASES

- Each weekly snapshot is archived
  - to an own Lifecycle Environment (created with `katello_lifecycle_environment`)
  - referenced by an own Activation Key (created with `katello_activation_key`)
- this allows to reproduce older environments and re-test bugs

REFERENCES

- our Jenkins jobs
- our Ansible playbooks

THANKS!

evgeni@golov.de

die-welt.net

@zhenech

@zhenech@chaos.social

@evgeni

zhenech