Katello and Ansible for automated testing and

$ whoami

Evgeni Golov

Software Engineer at Red Hat

One-time 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

Katello

$ whatis katello

- “Katello brings the full power of content management alongside the provisioning and configuration capabilities of Foreman”
- 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

staging changes with Katello

- Katello presents “frozen” Content Views to the clients
- Clients can belong to different Environments, each with a different frozen view
- Synchronization and publication of content can be automated
- This will be our main driver

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

- Each change executes a unit test suite before it’s merged
- Packages are built and synchronized after a merge
- Empty VM is presented with the newly built packages plus RHEL and RHSCL
- Software is installed and end-to-end tests run
- Succesful tests promote the content to the *nightly* environment

source testing

Every merge-request executes the tests of the sub-project:

- Ruby test suite
- UI/Javascript test suite
- Whatever else that sub-project offers (webpack, Puppet, …)

These tests don’t cover the production deployment as it’s done on the users machine.

Neither do they cover integration with other parts of the product.

package building

After every merge, we build packages:

- New source tarball is generated
- RPM `.spec` is updated
- RPM is built using Koji
- All automated using Ansible, go watch Ewoud's talk about `obal` at 17:30!

package testing

All packages are tested daily together:

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

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 tests that verify the functionality of the setup
- Same setup can be used on your laptop (if it has enough RAM)

package testing - forklift

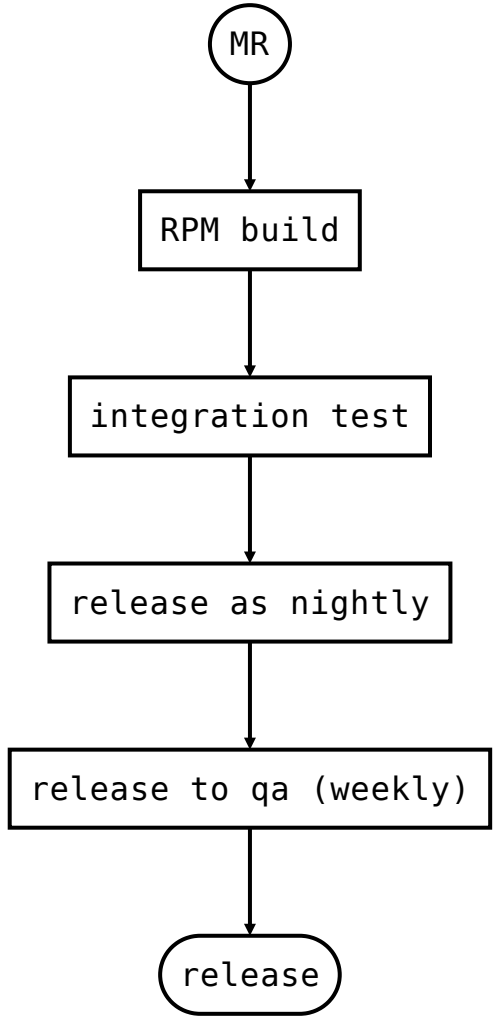
- Ansible playbooks and roles to deploy Foreman/Katello/Proxy
- Can be used for your production setup
- Or together with Vagrant for dev/test/qa tasks
- Repository also includes a set of integration tests (`bats` and `robottelo`)
  - Content related tests for Katello (and Proxy)
  - Plugin specific tests for other Plugins (BIND and PowerDNS)
  - Easily extendable for other plugins

package testing - content management

- Heavy use of foreman-ansible-modules for interacting with the Katello API
- Synchronization: `katello_sync`
- Publishing: `katello_content_view_publish`
- Promotion: `katello_content_view_version_promote`
- No `uri` module! ♥

further testing and releasing

- Daily tests are limited in scope 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


MR
RPM build
integration test
release as nightly
release to qa (weekly)
release

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