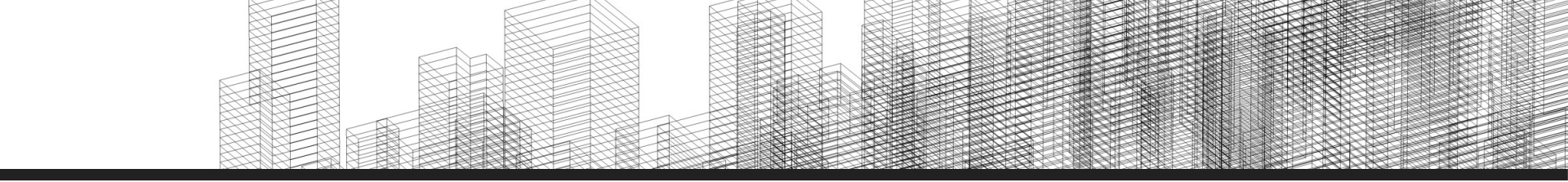


Pagure CI based on Zuul

Fabien Boucher

Topics

- Zuul, main concepts and features
- Zuul and Pagure
- Proof of concept
- How Fedora could benefit from Zuul

Zuul, main concepts and features

Keeping code branch healthy is hard

- Validating a patch can take a long time or can require a complex environment
- The velocity of a patch submission can be high
- Merging code is usually an human process, prone to side effect

A simple gating strategy

- Project’s maintainers should merge patch only if the patch pass the test suite
- Code Review system such as Github allow to run test suites on PR

Simple gating is flawed

- Potential side effects as pre-merge testing may occur on an outdated version of the code
- Potential side effects increase in case of interdependent repositories

A proposal for a better gating system

- Approved patches must always be tested on top of the latest version of the code JUST BEFORE they are merged
- Multi-repo and un-merged patches dependencies support
- Patches merging order must match the patches approval order
- The gating must be automated
- The CI system must be scalable

Main concepts and some numbers

Built for OpenStack testing needs

- multi-repository and patches dependency management
- CI and parallel co-gating
- Scaling
- Compatible with Gerrit, Github and Pagure

Ballpark statistics from OpenStack CI

- 1500 git repos
- 2K jobs per hour
- 10K patches merged by month

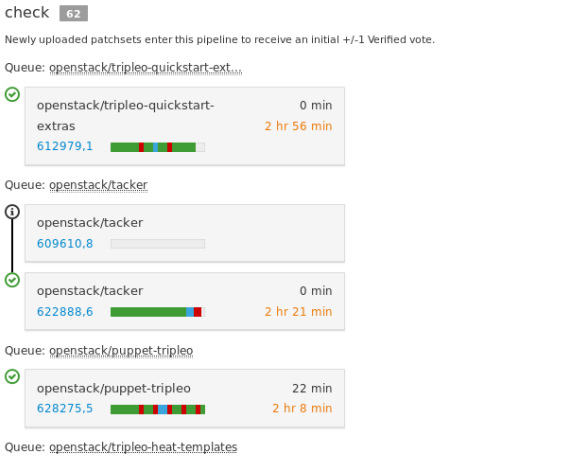
Zuul is generic and not tied only to OpenStack

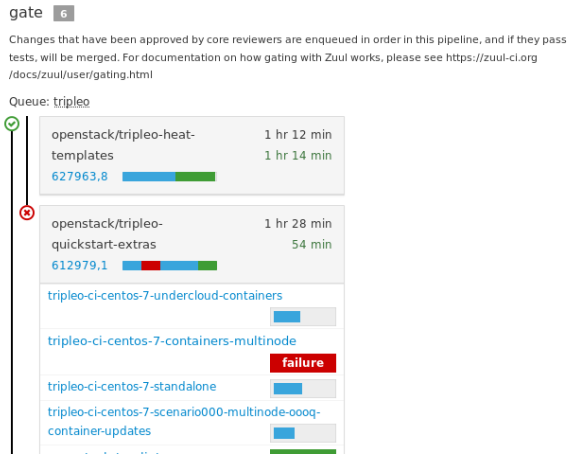
Zuul - More concepts and features

- Event-driven pipelines
- CI-as-code
- Ansible
- Support for jobs inheritance, jobs dependencies, jobs chaining with artifacts sharing
- Multi-nodes jobs
- Resources lifecycle management (Nodepool) and reproducible job environments
- Secrets management
- Multi-tenancy

Event pipelines

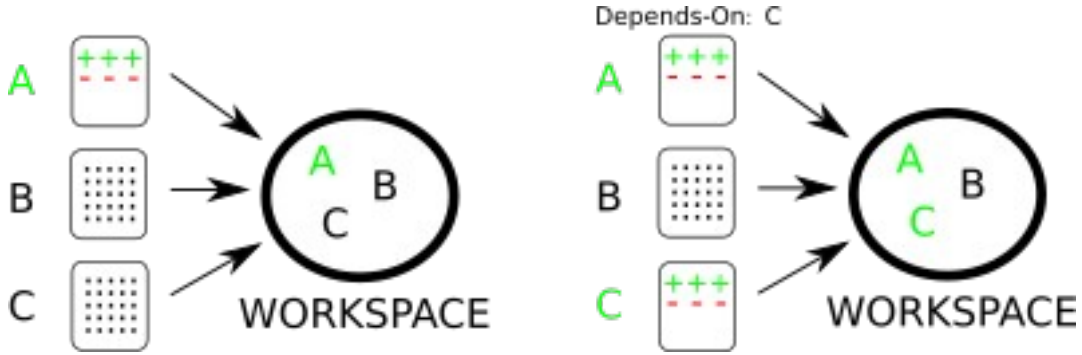
- Pipelines are run depending on events on the Code Review system:
  - CHECK: when a patch is created or updated
  - GATE: when a patch is approved for merging
- Pipelines can also be defined on git events



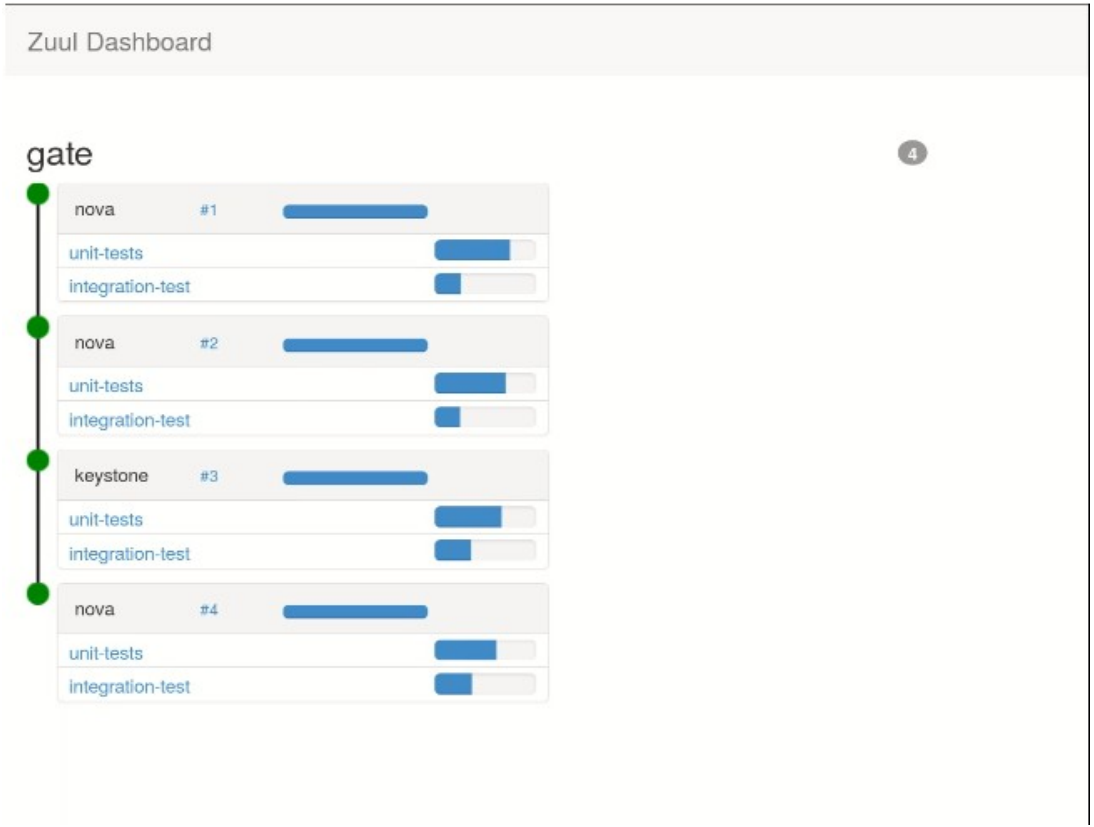


Cross-repositories dependencies

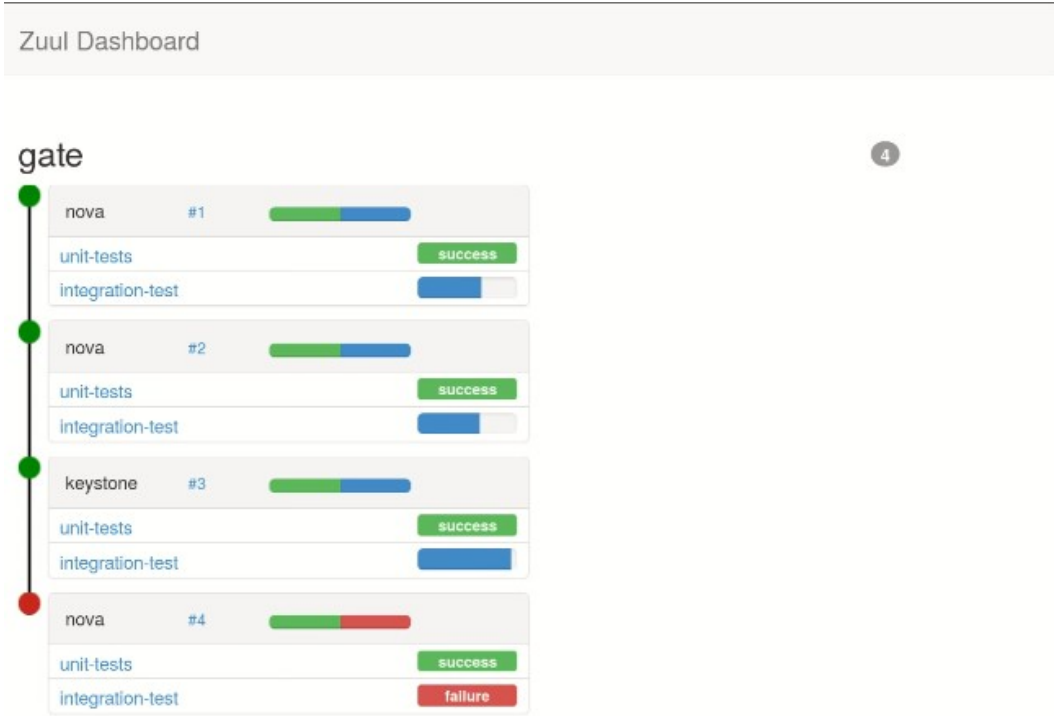
- When starting a job for a specific repository, Zuul pulls every repository defined with a dependency relationship as well into the job's workspace
- By default, dependencies are fetched at the tip of the branch
- To fetch a non merged patch instead, use the "Depends-On" keyword and the URL of the patch in the commit message
- Zuul won't merge a patch that depends on other patches until all the dependencies have been merged



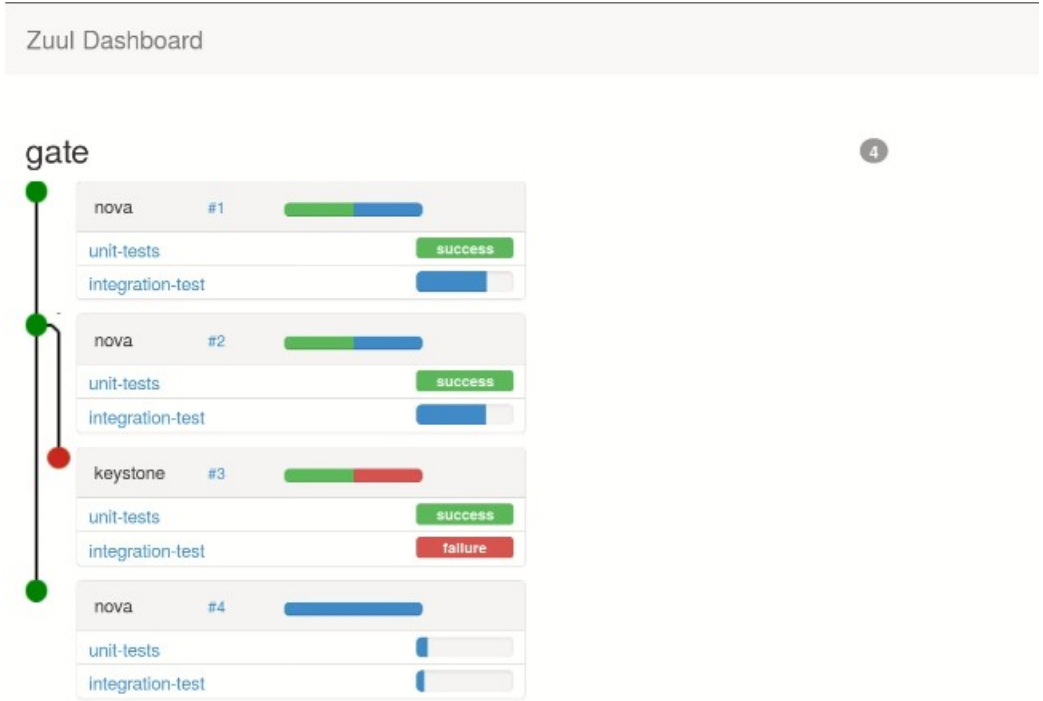
Gate pipeline workflow



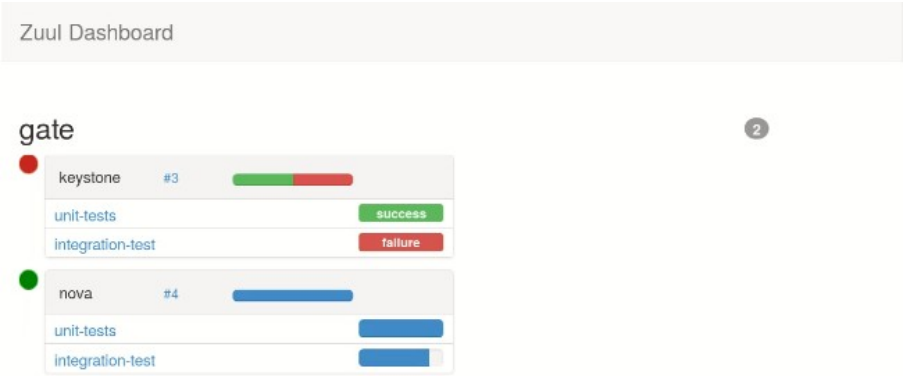
Gate pipeline workflow

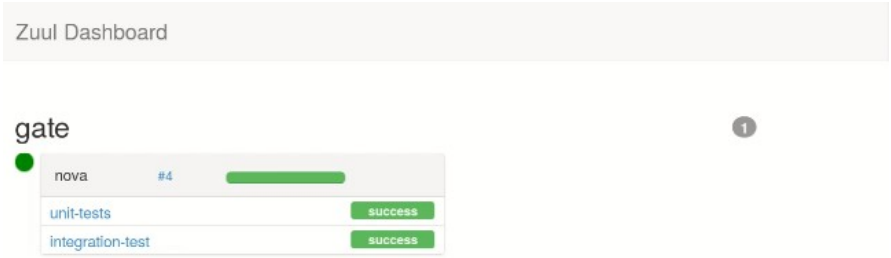


Gate pipeline workflow



Gate pipeline workflow



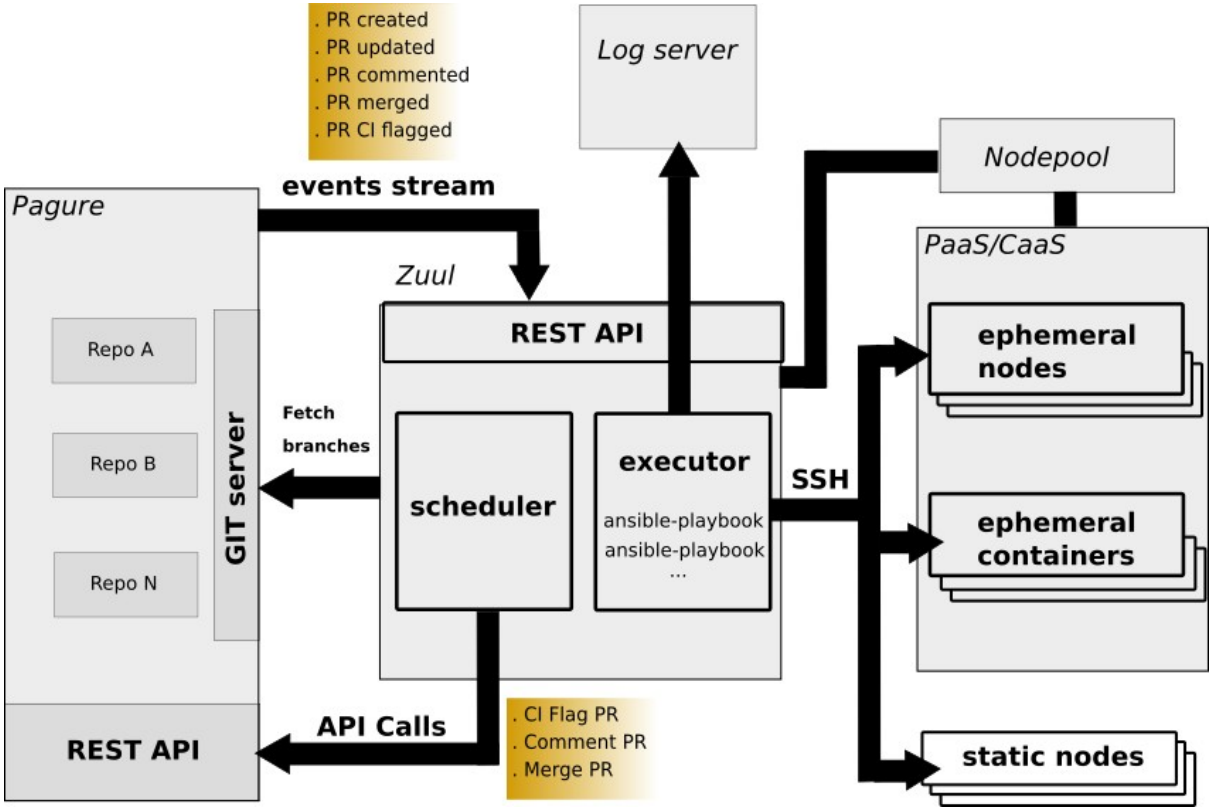


Integration between Zuul and Pagure

Zuul drivers for Code Review system

- Current Code Review drivers:
  - Gerrit
  - Github and Github enterprise
  - **Pagure**
- A driver should be able to
  - listen to events
  - read PR status (approval, flags, mergeability, …)
  - Act on PR (report back CI status, merge code)

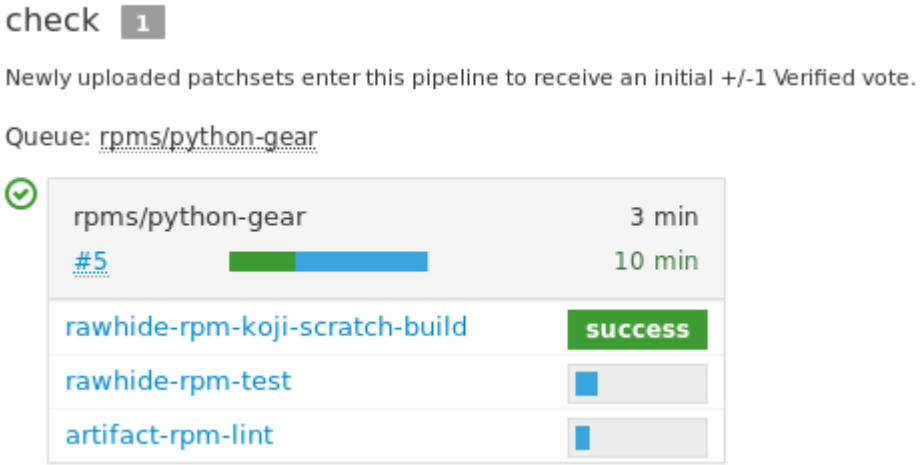
How Zuul interact with Pagure



Proof of concept

Artifact sharing with child jobs

```
- project:
    name: rpms/python-gear
    check:
      jobs:
        - rawhide-rpm-koji-scratch-build
        - rawhide-rpm-test:
            dependencies:
              - rawhide-rpm-koji-scratch-build
        - artifact-rpm-lint:
            dependencies:
              - rawhide-rpm-koji-scratch-build
```



The job definition

```
- job:
    name: rawhide-rpm-koji-scratch-build
    description: RPM scratch build for rawhide target
    roles:
      - zuul: zuul-distro-jobs
    provides: repo
    run: playbooks/koji/build.yaml
    post-run: playbooks/koji/fetch.yaml
    secrets:
      - krb_keytab
    nodeset:
      nodes:
        - name: mock-host
          label: cloud-fedora
    vars:
      mock_config: fedora-rawhide-x86_64
      target: rawhide
      scratch_build: true
```

The post-run playbook tasks

```yaml
- name: Upload logs for all builds
  synchronize:
    src: '{{ ansible_user_dir }}/repo/'
    dest: '{{ zuul.executor.log_root }}/buildset/'
    mode: pull

- name: Return repo url
  zuul_return:
    data:
      zuul:
        artifacts:
          - name: repo
            url: buildset
```

How child jobs reuse the artifact

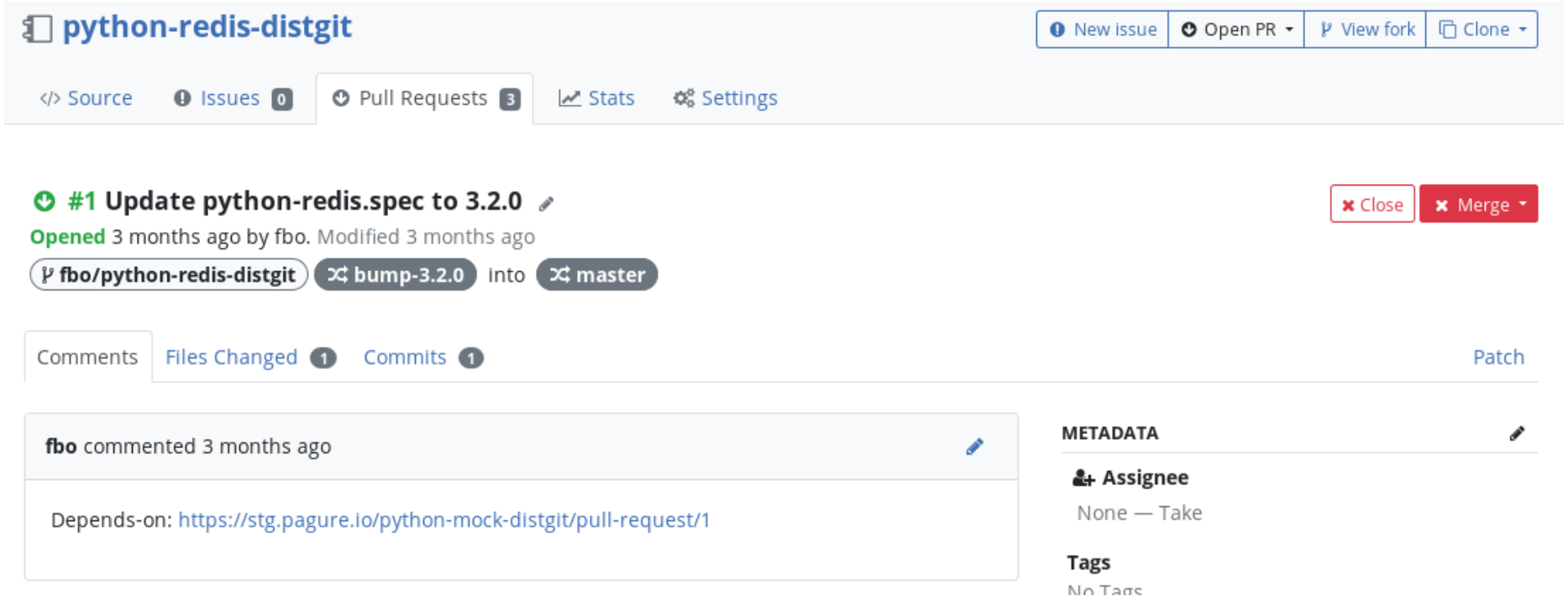
From the job's inventory

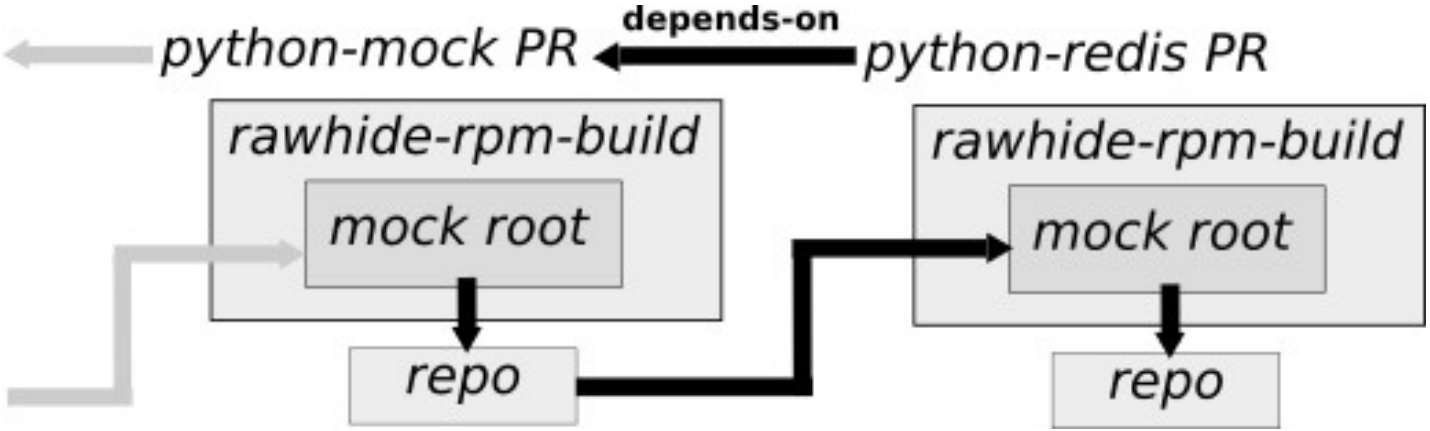
```
artifacts:
- change: '5'
  job: rawhide-rpm-koji-scratch-build
  name: repo
  patchset: 0f16a5cda21602dd1a662c8d40d00380460b9f8f
  project: rpms/python-gear
  url: https://fedora.softwarefactory-project.io/logs/5/5/0f16a5cda21602dd1a662c8d40d00380460b9f8f/check/rawhide-rpm-koji-scratch-build/56c8414/buildset
```

```
- name: Add intermediary repositories from changes ahead
  become: yes
  copy:
    dest: "/etc/yum.repos.d/zuul-build.repo"
    content: |
      {% for artifact in zuul.artifacts|default([]) %}{% if artifact.name == 'repo' %}
      [{{ artifact.project | regex_replace('/', '_') }}-{{ artifact.change }}-{{ artifact.patchset }}]
      name={{ artifact.project }}-{{ artifact.change }}-{{ artifact.patchset }}
      enabled=1
      baseurl={{ artifact.url }}
      gpgcheck=0

      {% endif %}{% endfor %}
```

PR dependency and RPM BuildRequire





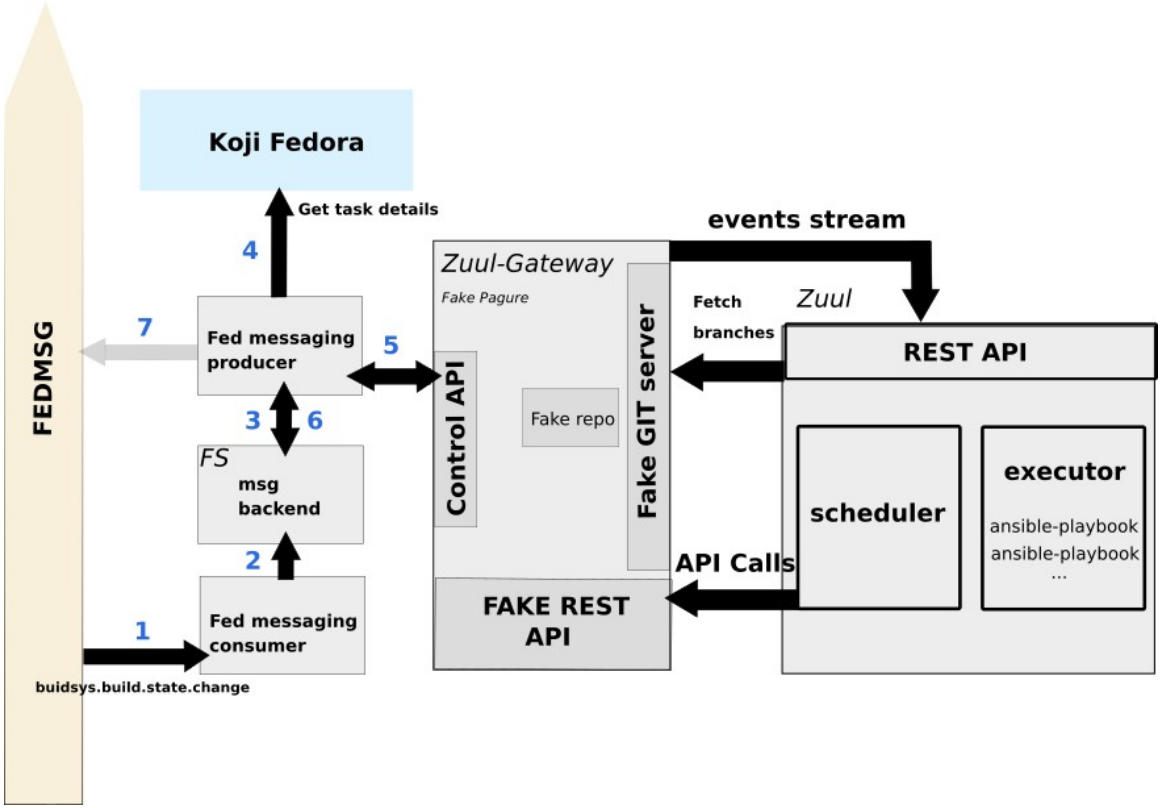
From the job's inventory

```
artifacts:
- change: '1'
  job: rawhide-rpm-build
  name: repo
  patchset: e37b8f2ebbc328eb7c22aa3c879cc4bcda843d73
  project: python-mock
  url: https://fedora.softwarefactory-project.io/logs/1/1/e37b8f2ebbc328eb7c22aa3c879cc4bcda843d73/check/rawhide-rpm-build/b0bc434/buildset
```

From the job' logs

```
node | 2019-05-20 13:43:59,889 INFO:dlrn-build-mock:DEBUG:  python3-py              noarch1.8.0-1.fc31          fedora                                                          453 k
node | 2019-05-20 13:43:59,890 INFO:dlrn-build-mock:DEBUG:  python3-pytest          noarch4.4.1-1.fc31          fedora                                                          1.6 M
node | 2019-05-20 13:43:59,891 INFO:dlrn-build-mock:DEBUG:  redis                   x86_645.0.5-1.fc31          fedora                                                          665 k
node | 2019-05-20 13:43:59,894 INFO:dlrn-build-mock:DEBUG:  python2-mock           noarch2.0.0-14.fc31         python-mock-distgit-1-e37b8f2ebbc328eb7c22aa3c879cc4bcda843d73  128 k
node | 2019-05-20 13:43:59,895 INFO:dlrn-build-mock:DEBUG:  python3-mock           noarch2.0.0-14.fc31         python-mock-distgit-1-e37b8f2ebbc328eb7c22aa3c879cc4bcda843d73  130 k
```

Run Zuul jobs from Fedmsg events



How Fedora could benefit from Zuul ?

Main advantages of Zuul

- Multi-repository and depends-on
- Co-Gating
- Cross-provider Gating (Pagure/Github)
- Zuul job + Ansible
- CI configuration as code

Resources

More info on the POC: https://fedoraproject.org/wiki/Zuul-based-ci

How to spawn a Zuul sandbox:

- https://zuul-ci.org/docs/zuul/admin/quick-start.html
- https://www.softwarefactory-project.io/docs/3.3/operator/quickstart.html

Software Factory used for the POC - https://fedora.softwarefactory-project.io/zuul

Questions / Comments ?