OOM journey from Gerrit / Jenkins to Gitlab

Sylvain Desbureaux
Agenda

• OOM and gerrit’s use
  – Jenkins
  – Gating via Gitlab CI

• Job migration to gitlab
  – Sensitive tasks
  – Linting
  – Gating

• Next steps
Jenkins limitations

• Jenkins from LFN uses VM to perform the jobs
  – Therefore, dependencies are shared
  – Complicated to test code against different versions of same executable
  – JJB are thought for maven and (a bit) docker and not for helm

• For OOM committers, Jenkins adds more issues than it helps
• LF rule enforcement is done via git hooks on gerrit
  – CLA signature check
  – “Issue-ID” and “Signed-by” being set check
• Jenkins performs “linting”
  – Helm linting
  – Bash script linting
  – No whitespaces / tabs linting
  – Image being released linting
  – Commit message format linting
  – Documentation linting (doc8, link check, …)
• Gitlab CI (+ some micro services) performs gating
  – Each gerrit review sent to OOM (and integration and core SO) are deployed
  – We then check that everything works when deployed
OOM and current Gitlab CI use

- gerrit
- gerrit2mqtt
- mqtt2gerrit
- MQTT
- Gating controller
- Gating Worker X
- Gitlab CI

Push code

Send to μs

Send to new code topic

“You’re number XX in queue”

“Gating started”

“Gating finished, here’s the results”

Add review in queue

Remove review from queue

Check if current gate has finished

Check if current gate has finished

Start gating

Check if current gate has finished

Check if current gate has finished
Job migration to gitlab: sensitive tasks

• Some tasks are sensitive because they need to have access to sensitive data

• CLA check needs to have list of all users with signed CLA

• Helm upload needs to have credentials to push into ONAP nexus
Job migration to gitlab: sensitive tasks

• CLA check: use "external validation"

CLA check will
• Update users lists if too old (once every hour)
• Take the list of all code authors from the merge request
• Validate they’re present in one of user lists
• Approve / disapprove according to the result

1: https://gitlab.com/Orange-OpenSource/Ifn/onap/gating-tools/external-approval-listener
2: https://gitlab.com/Orange-OpenSource/Ifn/onap/gating-tools/cla-checker
Job migration to gitlab: sensitive tasks

- Helm upload: (will) use protected / hidden environment variables
  - Creds will be set into protected and hidden variables
    - They can be used only on protected branches (master + ‘stable/*’)
    - Script using them won’t show them
    - Can’t be retrieved out of OOM repo
Job migration to gitlab: linting

- All linting tasks will be performed into dedicated steps using
  - Dedicated docker container(s) (helm linter will use a matrix of container in order to check against needed versions)
  - Simple scripts

- Tasks will be triggered only if needed:
  - Doc task will be triggered only if “doc” folder is touched
  - Helm / gating / linting will be triggered only if “Kubernetes” folder is triggered
Job migration to gitlab: launching gates

- Launching gates has 2 issues to solve:
  - We need to be able to work in "gerrit triggered" mode and in "gitlab triggered" mode
  - We have a limited number of systems for the gate so the queue system must remain
Job migration to gitlab: launching gates

- gerrit
- gerrit2mqtt
- mqtt2gerrit
- MQTT
- Gating controller
- Gating Worker X
- Gitlab CI

Push code: External approver call → Send to new request topic → "You're number XX in queue" → "Gating started" → "Gating finished, here's the results" → Check if current gate has finished → Start gating.

External approver call: Check if current gate has finished → Check if everything is ready → Send to new code topic → Add review in queue → Remove review from queue.

"You're number XX in queue": Check if current gate has finished → Start gating.

"Gating started": Check if current gate has finished → Check if everything is ready → Send to new code topic → Add review in queue → Remove review from queue.

"Gating finished, here's the results": Check if current gate has finished → Start gating.

Check if everything is ready: Send to new code topic → Add review in queue → Remove review from queue.

Send to new request topic: "You're number XX in queue" → "Gating started" → "Gating finished, here's the results" → Check if current gate has finished → Start gating.

Add review in queue: Check if current gate has finished → Start gating.

Remove review from queue: Check if current gate has finished → Start gating.

Gitlab CI: Check if current gate has finished → Start gating.
Job migration to gitlab: launching gates

- Gating launcher¹ will:
  - Check that Kubernetes folder has changed
  - Check that helm linting is OK
  - If yes, ask for a gate

- Gitlab2mqtt is the same tool as CLA check
  External Validation Frontend²

- MQTT2Gitlab³ is a new μs

1: https://gitlab.com/Orange-OpenSource/lfn/onap/gating-tools/gating-launcher
2: https://gitlab.com/Orange-OpenSource/lfn/onap/gating-tools/external-approval-listener
Next steps

- All µs are deployed on ONAP Azure tooling zone (on Kubernetes)
- Tests have been made and it works
- We can migrate “when we want”
- Decide to move to Gitlab instead of Gerrit / Jenkins; decision will be taken by the next OOM PTL
User journey to push code

Fork project

Project name:
oom

Project URL:
https://gitlab.com
Select a namespace:

Project description (optional):
Introduces the ONAP Platform OOM (ONAP Operations Manager) to efficiently Deploy, Manage, Operate the ONAP platform and its components (e.g. MSO, DCAE, SDC, etc.) and infrastructure (VMs, Containers).

Visibility level:
- Private: Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.
- Internal: This project can be accessed by any logged in user.
- Public: The project can be accessed without any authentication.

Fork project
User journey to push code

Create a branch (via UI or git cli)

Click on create Merge Request
User journey to push code

Verify that the target branch is onap/oom/oom

<table>
<thead>
<tr>
<th>Source branch</th>
<th>Target branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdesbure/oom</td>
<td>onap/oom/oom</td>
</tr>
<tr>
<td>newFeature</td>
<td>master</td>
</tr>
</tbody>
</table>

Compare branches and continue