

How to **easily** offer your application as a **self-service template** by using OpenShift and GitLab-CI

4th Developers@CERN Forum

Alberto Rodríguez Peón IT-CDA-WF

Running a **service** on the cloud

A **single instance of your application** serving all users



Application maintenance and **server provision is simple** as only a single instance running

All users **connect through the same well-known endpoint**

`https://myapp.cern.ch`



Running a **service** on the cloud

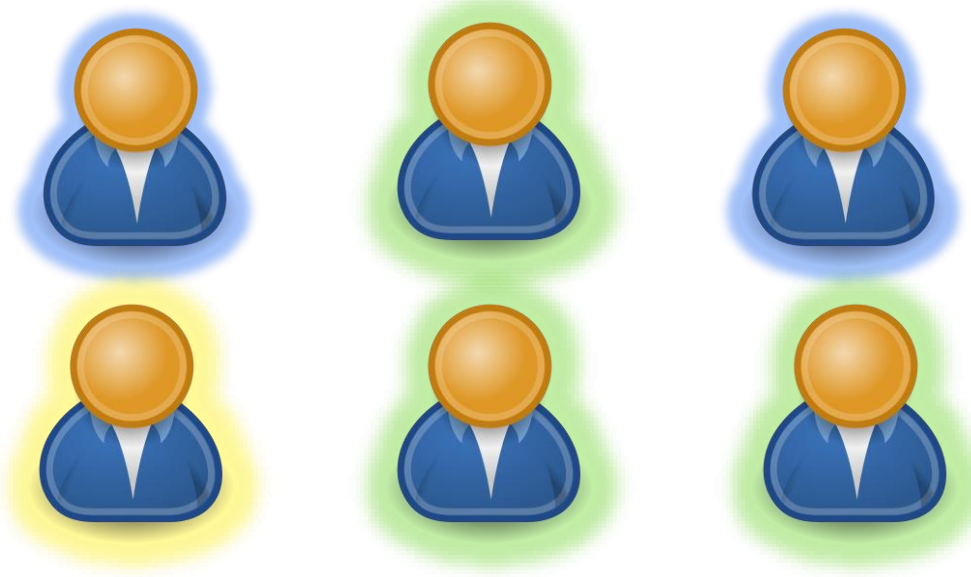
A **single instance of your application** serving all users



Application maintenance and **server provision** is **simple** as only a single instance running

All users **connect through the same well-known endpoint**

`https://myapp.cern.ch`



Running a **service** on the cloud

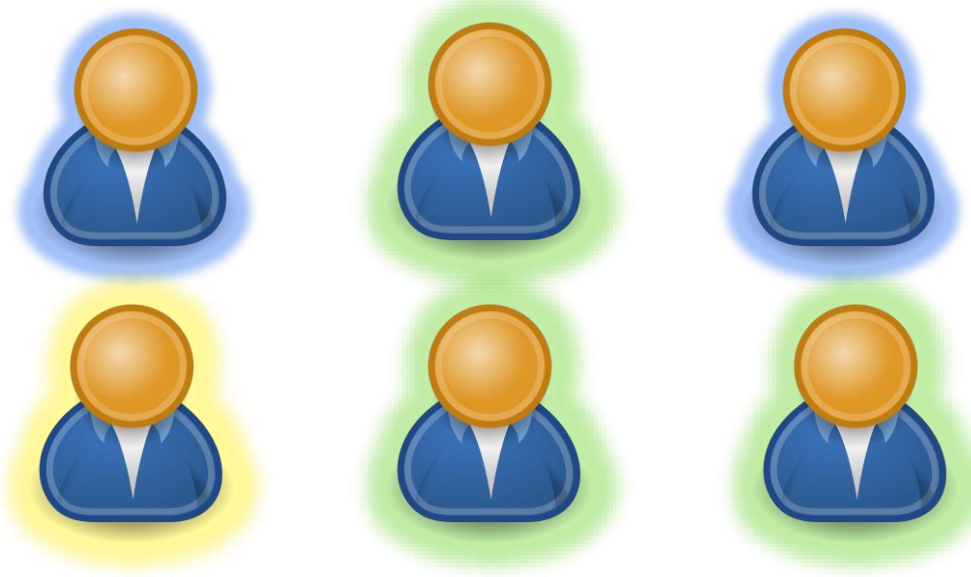
A **single instance of your application** serving all users



Application maintenance and **server provision is simple** as only a single instance running

All users **connect through the same well-known endpoint**

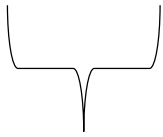
`https://myapp.cern.ch`



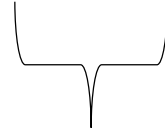
Application needs to support multi-tenancy, ACLs and independent user configuration!

Running **individual instances** per team

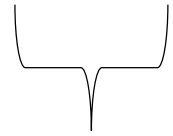
Individual instances solve this but maintenance, provision and configuration efforts get **multiplied!**



`https://myapp-1.cern.ch`



`https://myapp-2.cern.ch`



`https://myapp-3.cern.ch`



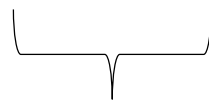
Application offered as a **self-service template**

Application binaries are offered **from a central catalog**, allowing users to pull them and **run their private instance**

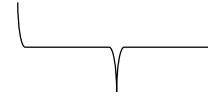


New software releases and **security updates are pushed to the central repository** and **propagated to users' instances**

Instance provision is automatically provided by the platform



`https://myapp-3.cern.ch`



`https://myapp-3.cern.ch`



Instance owners **have full privileges to configure their private copy** of the application



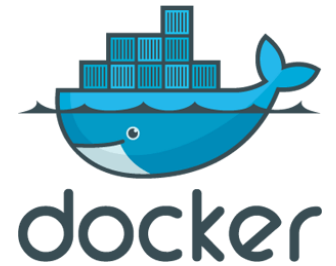
A real-life use case: Jenkins service @ CERN

- **Jenkins** is a clear example of software that works well with **the self-service model**
- As admins, we want to:
 - **Offer a readily available and curated template** that users can instantiate without effort
 - **Automate provisioning of new instances**
 - **Keep** users' instances up to date while minimizing maintenance efforts through automated procedures
 - **Automate build, test and deployment of new releases**
- OpenShift and GitLab-CI provide tools to achieve all this!

Packaging the application in a Docker Image

- By using Docker, we can **encapsulate the Jenkins binaries in a standard and reusable way that users can run**

- **Containers isolate app from host, allowing total delegation to users!**



- ... but **only providing an image is not enough**


- We still need an **orchestrating platform to provide:**

- Routing to the container
 - Instance provisioning
 - Lifecycle of the application
 - Data storage
 - Upgrade strategies



Using OpenShift templates

- By using **OpenShift**, we can use **an existing platform instead of running our orchestrating platform**
 - OpenShift supports adding all the missing pieces with **Templates**
 - Completely **self-service**
 - **Trivial for users to launch!**



jenkins-cern

Jenkins service, with persistent storage and CERN customizations.

Images

jenkins-cern:stable

Parameters

* JENKINS_ADMIN_EGROUP

An e-group that will be initially granted the admin role. To have access to the new instance, make sure you are yourself member of that e-group!

* VOLUME_CAPACITY

Volume space available for Jenkins data, e.g. 2Gi, 5Gi (suffix 'Gi' is required and stands for Gibibyte)

JENKINS_PASSWORD

Please leave blank: a random password will be generated. Jenkins admin password is useless when using SSO.

Keeping users' instances up to date

- OpenShift provides two features to **ensure private Jenkins instances** are up to date with new **software releases** and **security fixes**
 - **ImageStreams** are indirections to Docker images
 - **ImageTriggers** watch a given tag of an ImageStream and **redploy** the application whenever the tag changes
- All Jenkins instances are configured to use the **stable** tag of a **shared ImageStream** and have an **ImageTrigger** for that tag

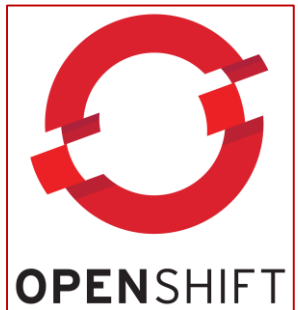
OpenShift ImageStreams at work

Jenkins image in the Docker registry



<code>docker.io/_/jenkins:latest</code>	<code>docker.io/_/jenkins@sha256:a4b585874b21...</code>
<code>docker.io/_/jenkins:2.73.2-1</code>	<code>docker.io/_/jenkins@sha256:714c62c1f62c...</code>

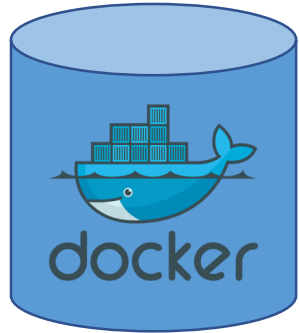
Jenkins ImageStream



<code>openshift/jenkins:stable</code>	<code>docker.io/_/jenkins@sha256:714c62c1f62c...</code>
<code>openshift/jenkins:latest</code>	<code>docker.io/_/jenkins@sha256:a4b585874b21...</code>
<code>openshift/jenkins:2.73.2-1</code>	<code>docker.io/_/jenkins@sha256:714c62c1f62c...</code>

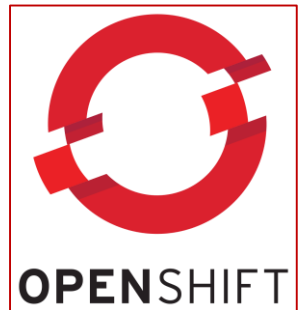
OpenShift ImageStreams at work

Jenkins image in the Docker registry



<code>docker.io/_/jenkins:latest</code>	<code>docker.io/_/jenkins@sha256:a4b585874b21...</code>
<code>docker.io/_/jenkins:2.73.2-1</code>	<code>docker.io/_/jenkins@sha256:714c62c1f62c...</code>
<code>docker.io/_/jenkins:2.73.2-2</code>	<code>docker.io/_/jenkins@sha256:60ea1a543548...</code>

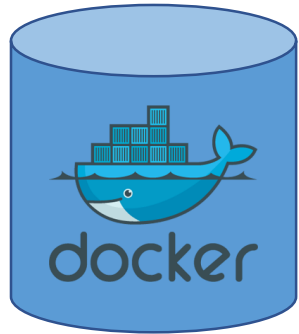
Jenkins ImageStream



<code>openshift/jenkins:stable</code>	<code>docker.io/_/jenkins@sha256:714c62c1f62c...</code>
<code>openshift/jenkins:latest</code>	<code>docker.io/_/jenkins@sha256:a4b585874b21...</code>
<code>openshift/jenkins:2.73.2-1</code>	<code>docker.io/_/jenkins@sha256:714c62c1f62c...</code>

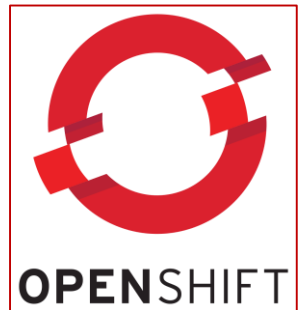
OpenShift ImageStreams at work

Jenkins image in the Docker registry



<code>docker.io/_/jenkins:latest</code>	<code>docker.io/_/jenkins@sha256:a4b585874b21...</code>
<code>docker.io/_/jenkins:2.73.2-1</code>	<code>docker.io/_/jenkins@sha256:714c62c1f62c...</code>
<code>docker.io/_/jenkins:2.73.2-2</code>	<code>docker.io/_/jenkins@sha256:60ea1a543548...</code>

Jenkins ImageStream

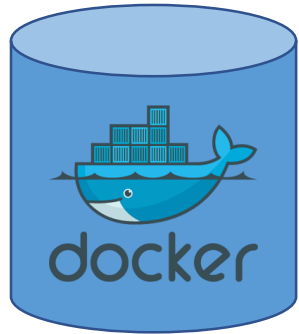


```
$ oc import-image jenkins:2.73.2-2
```

<code>openshift/jenkins:stable</code>	<code>docker.io/_/jenkins@sha256:714c62c1f62c...</code>
<code>openshift/jenkins:latest</code>	<code>docker.io/_/jenkins@sha256:a4b585874b21...</code>
<code>openshift/jenkins:2.73.2-1</code>	<code>docker.io/_/jenkins@sha256:714c62c1f62c...</code>
<code>openshift/jenkins:2.73.2-2</code>	<code>docker.io/_/jenkins@sha256:60ea1a543548...</code>

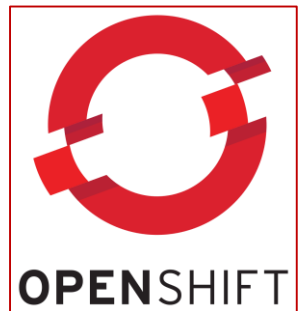
OpenShift ImageStreams at work

Jenkins image in the Docker registry



<code>docker.io/_/jenkins:latest</code>	<code>docker.io/_/jenkins@sha256:a4b585874b21...</code>
<code>docker.io/_/jenkins:2.73.2-1</code>	<code>docker.io/_/jenkins@sha256:714c62c1f62c...</code>
<code>docker.io/_/jenkins:2.73.2-2</code>	<code>docker.io/_/jenkins@sha256:60ea1a543548...</code>

Jenkins ImageStream



```
$ oc tag jenkins:2.73.2-2 jenkins:stable
```

<code>openshift/jenkins:stable</code>	<code>docker.io/_/jenkins@sha256:60ea1a543548...</code>
<code>openshift/jenkins:latest</code>	<code>docker.io/_/jenkins@sha256:a4b585874b21...</code>
<code>openshift/jenkins:2.73.2-1</code>	<code>docker.io/_/jenkins@sha256:714c62c1f62c...</code>
<code>openshift/jenkins:2.73.2-2</code>	<code>docker.io/_/jenkins@sha256:60ea1a543548...</code>

Applications with an ImageTrigger for `stable` will be redeployed!



GitLab-CI Pipelines

- The build and deployment process **is fully managed within GitLab-CI**
 - **Building the Docker image**
 - **Importing it to OpenShift**
 - **Testing it**
 - **Uploading the template**
 - **Redeploying all instances**
- To interact with OpenShift, we use a centrally provided Docker image with the `oc` CLI installed
 - <https://gitlab.cern.ch/paas-tools/openshift-client>

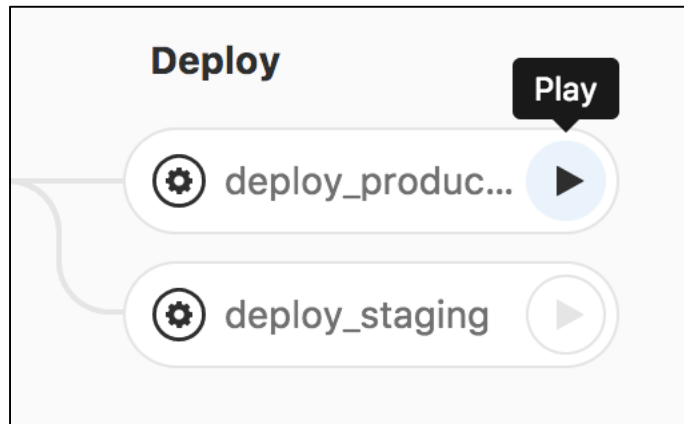


Automatic testing with GitLab-CI

- For verification of newer releases, automated tests are always run after a new image is built
 - Integrates nicely with GitLab-CI as **pipeline can be aborted if test stage fails**
 - Deployment into **OpenShift** easily achieved by using oc Docker image
- For Jenkins, two pre-created OpenShift projects are used to:
 - Create a **brand new instance** with the new image and template
 - **Create an instance with the old image** and template **and redeploy with the new image** immediately after
 - In both cases, a simple Jenkins job is run to verify all is working

Redeploying all instances after a new release

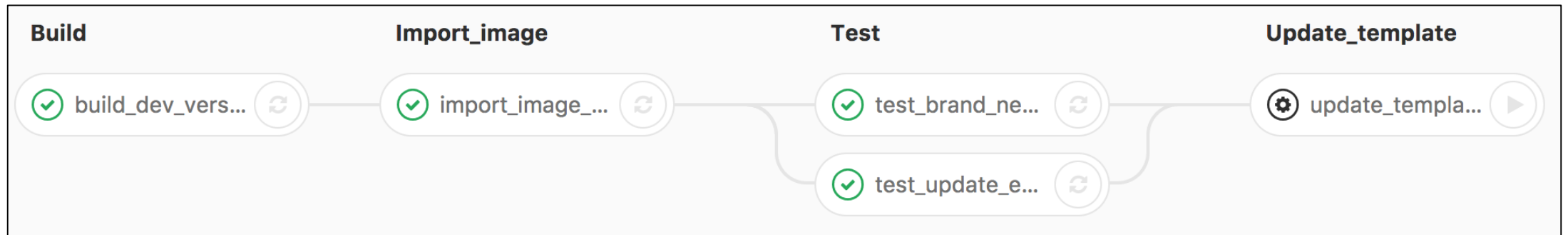
- Redeploying all running Jenkins instances happens whenever the **stable** tag is updated



- As this **potentially affects all instance owners**, redeployment is launched through a GitLab-CI manual trigger
 - After a global announcement and during a well-known **intervention window**

Our deployment workflow - development

- A **development pipeline is launched** by pushing to **any** branch of the repository



The image is **built** with the **latest** tag and pushed to the Docker registry

Import the built image into **the staging environment** in openshift-dev.cern.ch

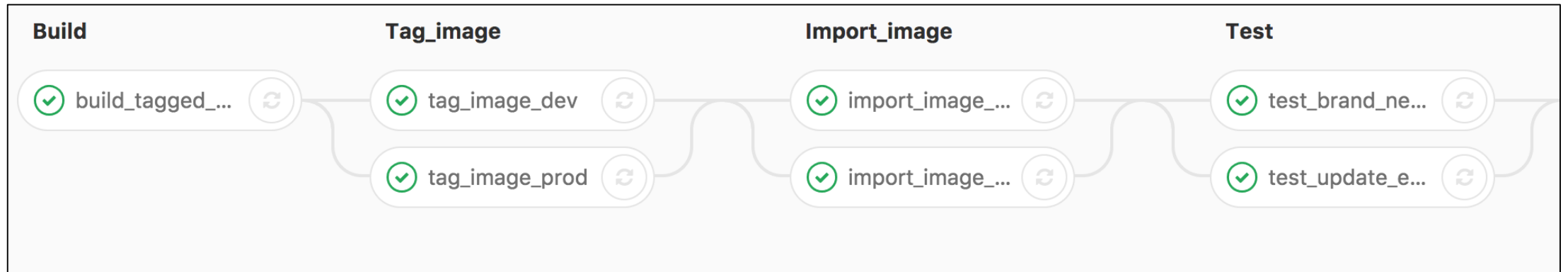
Run automated tests for the new image

Manual trigger to **upload the template** into the **staging environment** in openshift-dev.cern.ch

Only launched when template has been modified

Our deployment workflow - production

- A **production pipeline is run** by pushing a **git tag** (marked with an image release)



The image is **built** with the `<git_tag>` tag and **pushed to** the Docker registry.

The tag represents a software release and have the form of
2.73.2-2

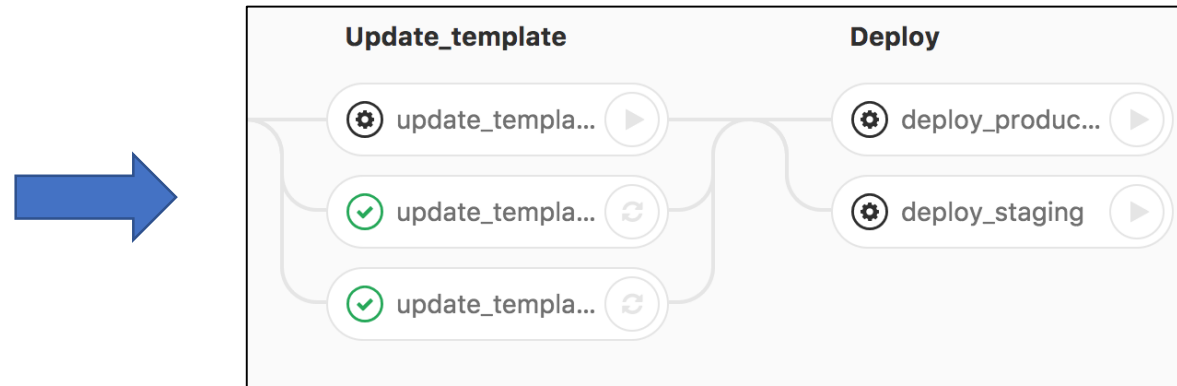
Create tags in the **ImageStream** for the new Docker tag with the version release

Import the built image into **the staging environment** in `openshift-dev.cern.ch` and also into the **production environment** in `openshift.cern.ch`

Run automated tests for the new image

Our deployment workflow – production (II)

- A **production pipeline** is run by pushing a **tag with the image release**



Update the templates in both **staging** (openshift-dev.cern.ch) and **production** (openshift.cern.ch) environments.

A manual trigger can be used to update the **template** in case the tests do not pass

Tag the release image with **stable**, **triggering a redeployment** of all applications

For production, this job is usually run during a **previously announced time window**, letting users know their **application will be restarted**

Summary

- With this workflow we managed to:
 - Offer a **centralized template** for users to launch private instances of Jenkins
 - By running in OpenShift, we get:
 - **An orchestration platform** that we (Jenkins admins) don't need to manage
 - **Image management features** that allow us to **redeploy users' applications** when there are security fixes and software releases
- By using GitLab-CI, we get:
 - We can **automate all build, test and deployment operations**
 - Manual triggers, to **launch jobs that required coordination**

Templates available at the moment



More coming soon!



What about YOUR app?

- This is our **personal experience** with the Jenkins Service
- Does your application fit this model?
 - Or any software you might want to distribute like this?
- Please **reach out** if it is the case
 - We can provide **expertise and help** to write a template for your application and configure the build and deployment process with GitLab-CI
 - https://gitlab.cern.ch/paas-tools/openshift_app_template_example