

Building Containers in GitLab-CI

Just the first steps

Tigran Mkrtchyan
Hamburg, 28.03.2025

Why We Need a CI?

Consistency, Reproducibility

- Local environment independence
 - Do I know all dependencies when I build locally?
- Reproducibility
 - Can you still build after OS update? After another package is installed?
- Process documentation
 - Do have documentation how to build? Do you keep it up-to-date?
 - Do you need all build dependencies at runtime?
- Testability
 - Do you regularly test your software? Can others do it as well?
- How I get your software

Why We Need a CI?

Consistency, Reproducibility

The **CI** gives a common framework to get stuff done with a **predictable and reproducible result.**

Gitlab-CI on One Slide

All you need to know :)

- Jobs defines an action at a particular stage
 - All jobs run a script action
 - Jobs are executed by *runners*
- A job belongs to a single stage
 - Stage defines when job should run
- Stages might consist out of multiple jobs
 - All jobs in a stage must succeed to before next stage can start
- Stages run sequentially
- Jobs in a single stage run parallel
- Jobs can pass artifacts to each other
- Jobs can depend on each other
- One or more stages build a *pipeline*
- *Pipeline* triggered by:
 - Event: push, merge, ...
 - Scheduler
 - Manually
- Default stages:
 - *build*
 - *test*
 - *deploy*

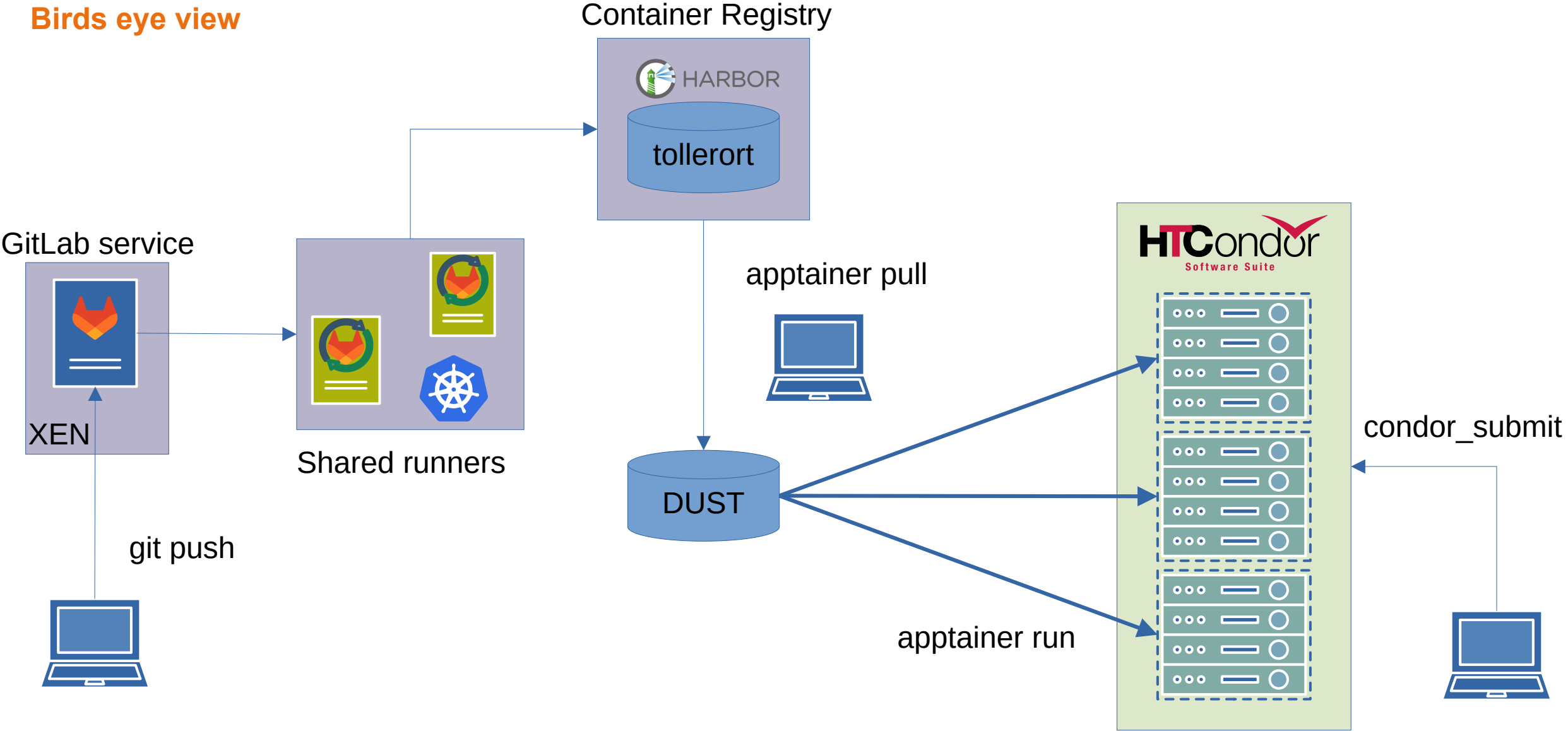
The Actors

Lot of moving parts

- GitLab CI
 - Code repository and CI
 - Store intermediate artifacts
- Kubernetes
 - To execute GitLab runners (to execute pipeline)
- Container registry
 - To publish/share container images
- Workgroup server
 - To prepare job environments
 - Submit the jobs
 - Get the results
- NAF
 - Access to batch resources
 - Access to data (DUST, dCache, CVMFS)

Container CI Workflow with NAF

Birds eye view



Harbor (tollerort.desy.de) integration

Yeah!

H **Harbor** ✓ Active

After the Harbor integration is activated, global variables `$HARBOR_USERNAME`, `$HARBOR_HOST`, `$HARBOR_OCI`, `$HARBOR_PASSWORD`, `$HARBOR_URL` and `$HARBOR_PROJECT` will be created for CI/CD use.

Enable integration
☒ Active

Harbor URL

Base URL of the Harbor instance.

Harbor project name

The name of the project in Harbor.

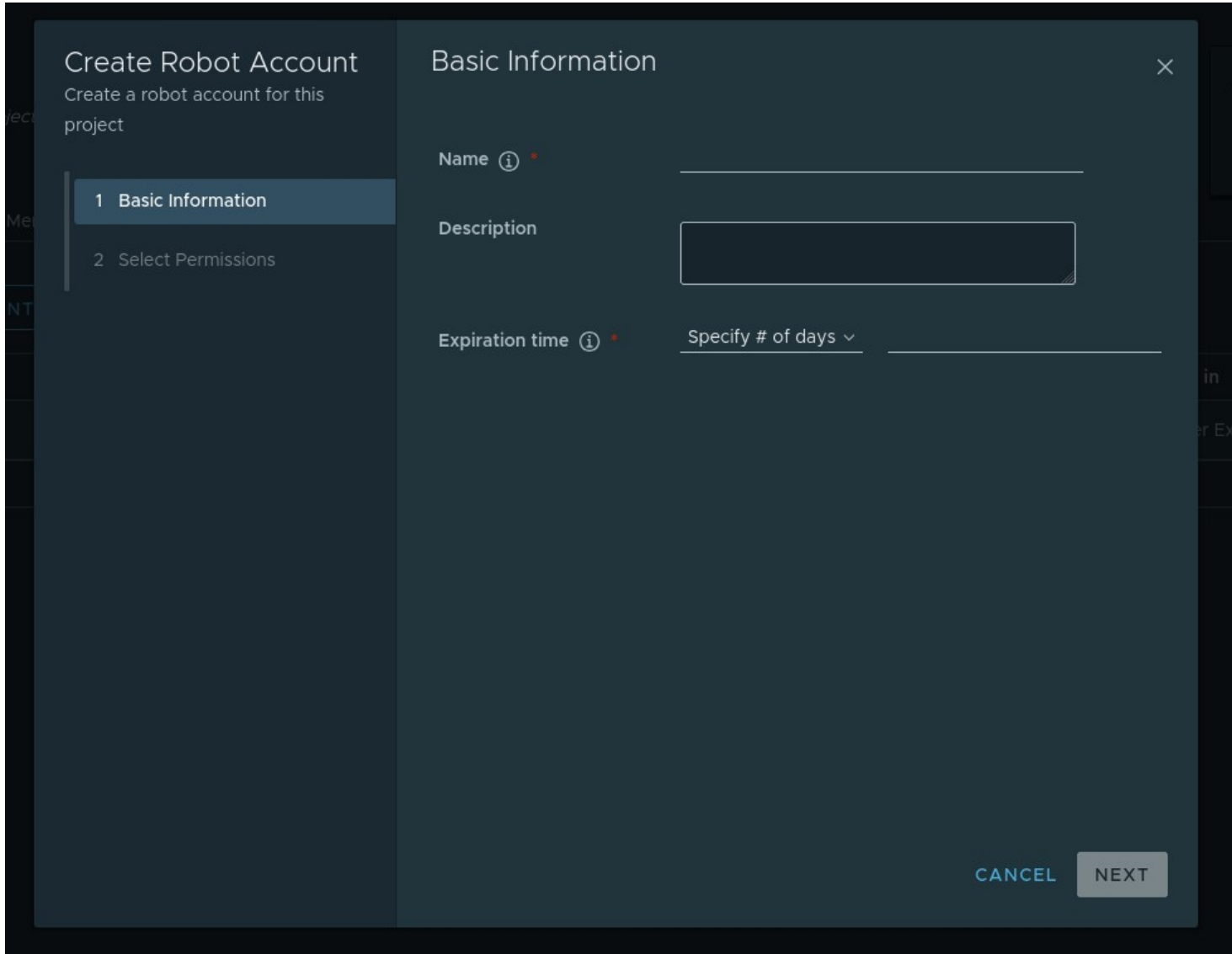
Harbor username

Enter new Harbor password

Leave blank to use your current password.

Harbor (tollerort.desy.de) integration

Yeah!



The screenshot shows a 'Create Robot Account' dialog box with a dark theme. On the left, a sidebar contains the title 'Create Robot Account' and the subtitle 'Create a robot account for this project'. Below this, there are two steps: '1 Basic Information' (highlighted) and '2 Select Permissions'. The main area is titled 'Basic Information' and contains three fields: 'Name' (with an info icon and a red asterisk), 'Description' (with a text area), and 'Expiration time' (with an info icon, a red asterisk, and a dropdown menu labeled 'Specify # of days'). At the bottom right, there are 'CANCEL' and 'NEXT' buttons.

Create Robot Account
Create a robot account for this project

1 Basic Information
2 Select Permissions

Basic Information

Name ⓘ *

Description

Expiration time ⓘ * Specify # of days ▾

CANCEL NEXT

Some Do and Don'ts

Use your power wisely!

NAF/HTCondor is a powerful compute facility, so...

- **Don't pull/push images directly from/to container registry as a part of your job.**
 - Pull images manually before hand.
- **Don't pull images to AFS.**
 - Use DUST.
 - Defined environment variables.
- **Don't put data into containers.**
 - dCache/DUST accessible at runtime.



Hands-on

Hello World Example

```
# Dockerfile
FROM almalinux:9-minimal
RUN microdnf install -y epel-release
RUN microdnf install -y figlet
CMD [ "figlet", "Hello", "World" ]
```

```
# helloworld.def
Bootstrap: docker
From: almalinux:9-minimal
%post
    microdnf install -y epel-release
    microdnf install -y figlet
%runscript
    figlet "Hello Apptainer"
```

Build Apptainer Container

Build Apptainer container:

stage: build

image: almalinux:9

script:

- dnf install -y epel-release
- dnf install -y apptainer
- apptainer build --fakeroot helloworld.sif container/helloworld.def
- apptainer registry login --username "\$HARBOR_USERNAME" \
- password "\$HARBOR_PASSWORD" docker://\$HARBOR_HOST
- apptainer push \$CI_PROJECT_NAME.sif \
- oras://\$HARBOR_HOST/\$HARBOR_PROJECT/\$CI_PROJECT_NAME:apptainer-
- \$CI_COMMIT_SHORT_SHA

Build OCI/Docker Container

Build OCI/Docker container:

```
stage: build
image: gcr.io/kaniko-project/executor:debug
script:
  - cp $CI_PROJECT_DIR/container/* .
  - mkdir -p /kaniko/.docker
  - echo "{\"auths\":{\"$CI_REGISTRY\":{\"username\":\"$CI_REGISTRY_USER\",
    \"password\":\"$CI_REGISTRY_PASSWORD\"}}}" > /kaniko/.docker/config.json
  - >
    /kaniko/executor
    --context $CI_PROJECT_DIR
    --dockerfile $CI_PROJECT_DIR/Dockerfile
    --destination $CI_REGISTRY_IMAGE:$CI_COMMIT_SHORT_SHA
```

Run OCI Container in Pipeline

Run the container:

```
stage: test
image: $CI_REGISTRY_IMAGE:$CI_COMMIT_SHORT_SHA
script:
  - figlet "Hello GitLab!"
```

Run Containers

Run OCI/Docker container:

```
stage: test
image: $CI_REGISTRY_IMAGE:$CI_COMMIT_SHORT_SHA
script:
  - figlet "Hello GitLab!"
```

Run Apptainer container:

```
stage: test
image: almalinux:9
script:
  - dnf install -y epel-release
  - dnf install -y apptainer
  - apptainer run helloworld.sif # assumes artifacts is configured
```

Convert Docker → Apptainer , option

Convert OCI to Apptainer format:

stage: convert

image: almalinux:9

script:

- dnf install -y epel-release
- dnf install -y apptainer
- apptainer registry login --username "\$CI_REGISTRY_USER" --password "\$CI_REGISTRY_PASSWORD" docker://\$CI_REGISTRY
- apptainer pull docker://\$CI_REGISTRY_IMAGE:\$CI_COMMIT_SHORT_SHA
- mv \${CI_REGISTRY_IMAGE##*/}_\$CI_COMMIT_SHORT_SHA.sif \$CI_PROJECT_NAME.sif

artifacts:

paths:

- \$CI_PROJECT_NAME.sif

expire_in: "1 days"

Convert Docker → Apptainer , option 2

Convert docker-archive to Apptainer:

```
stage: convert
image: almalinux:9
dependencies:
  - Build OCI/Docker tar container
script:
  - dnf install -y epel-release
  - dnf install -y apptainer
  - apptainer build $CI_PROJECT_NAME.sif docker-archive:$CI_PROJECT_NAME-
$CI_COMMIT_SHORT_SHA.tar
artifacts:
  paths:
    - $CI_PROJECT_NAME.sif
  expire_in: "1 days"
```

More examples:

`https://gitlab.desy.de/fh-sustainability-forum/sustainable-coding-tutorial/ci4naf`

Thank you

Contact

Deutsches Elektronen-
Synchrotron DESY

Tigran Mkrtchyan
IT, Scientific Computing

www.desy.de