

```
elif operation == "MIRROR_Y":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
elif operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

#selection at the end -add back the deselected mirror modifier object
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier ob is the active ob
#mirror_ob.select = 0
#me = bpy.context.selected_objects[0]
#me.data.objects[me.name].select = 1
#me.select = 1
print("Please select exactly two objects, the last one will be the active object")
#me.select = 0
```

A deep dive on BigHPC platform delivery into HPC clusters

Speaker: Samuel Bernardo (LIP) on behalf of BigHPC consortium

- The BigHPC consortium
- BigHPC platform overview
- Development challenges
- Integration Activities
- Takeaways

Austin, USA

UT Austin  **TEXAS**
The University of Texas at Austin



Wavecom

TACC **TACC**



INESC TEC



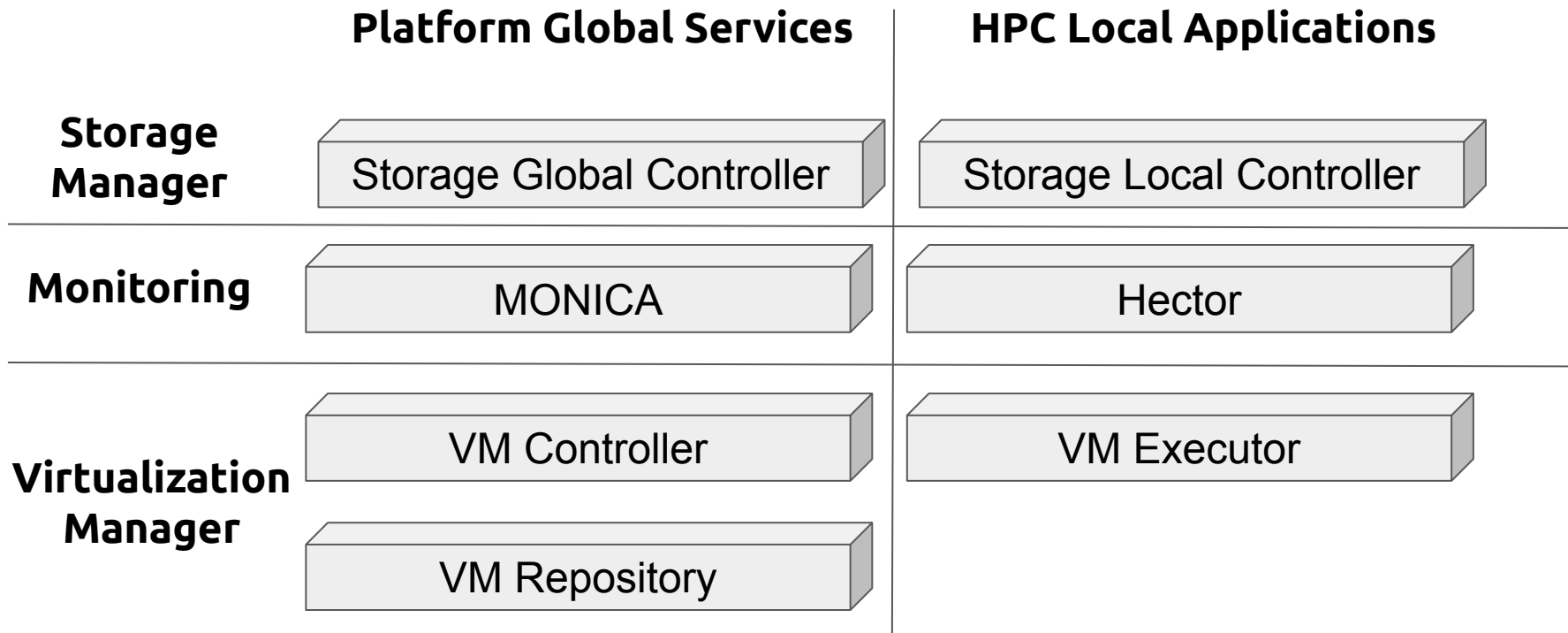
LIP



MACC

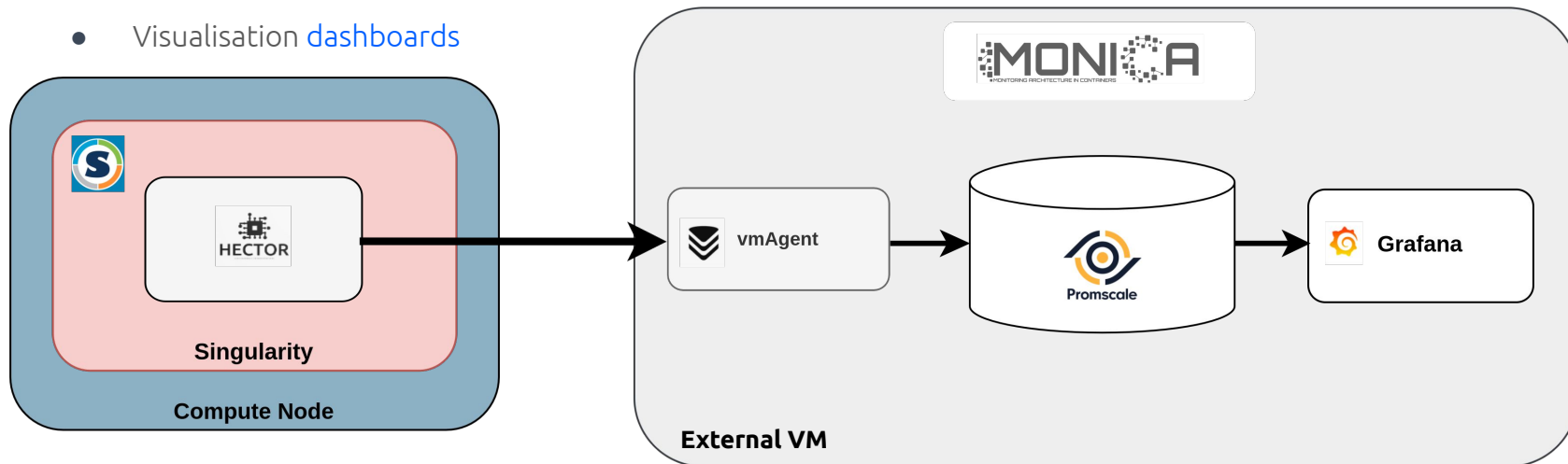
Portugal



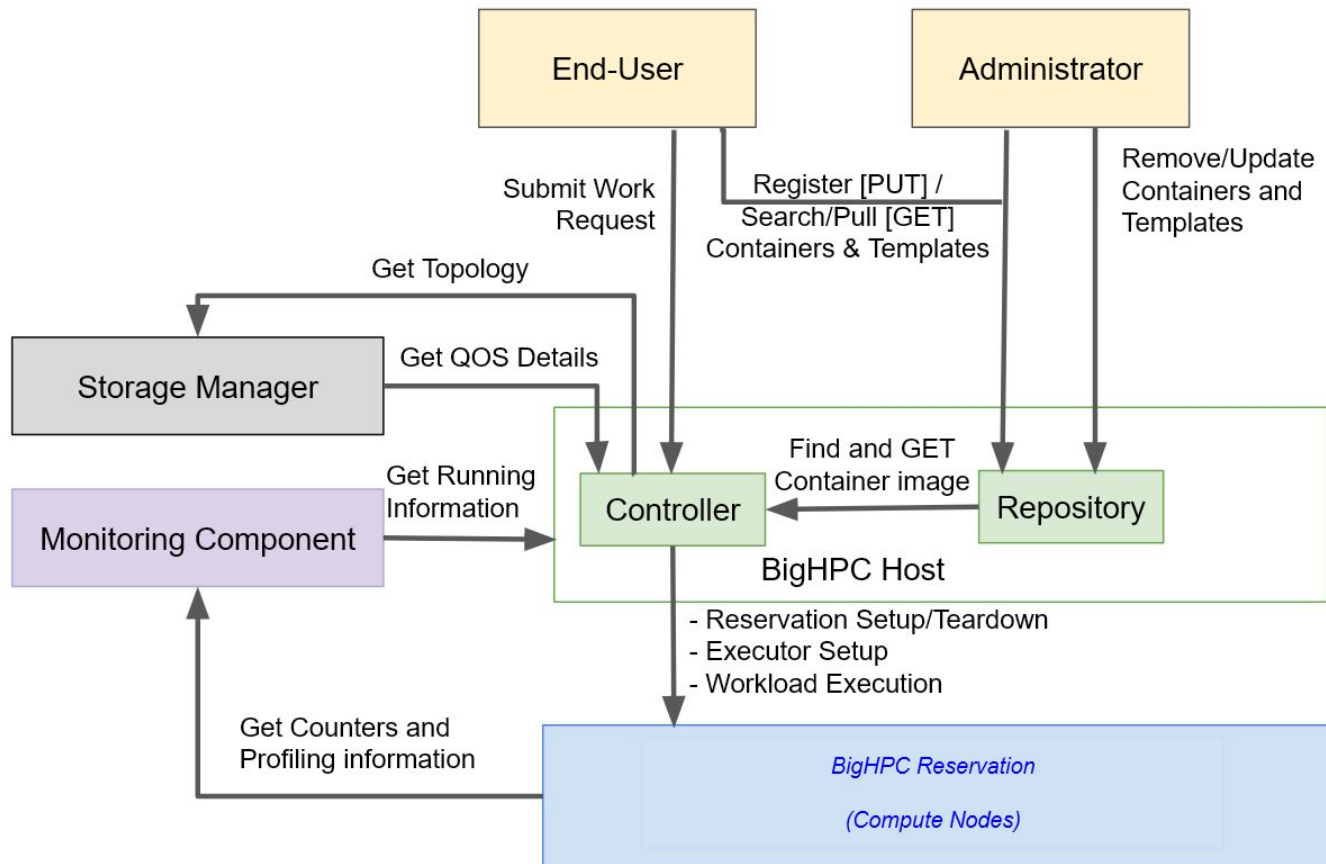


Monitoring backend - Overview

- Monitoring **probe inside compute nodes** responsible for **collecting and pushing metrics**
- **Aggregator** of metrics from **several probes**
- **SQL** based database with support for **PromQL queries**
- Visualisation **dashboards**

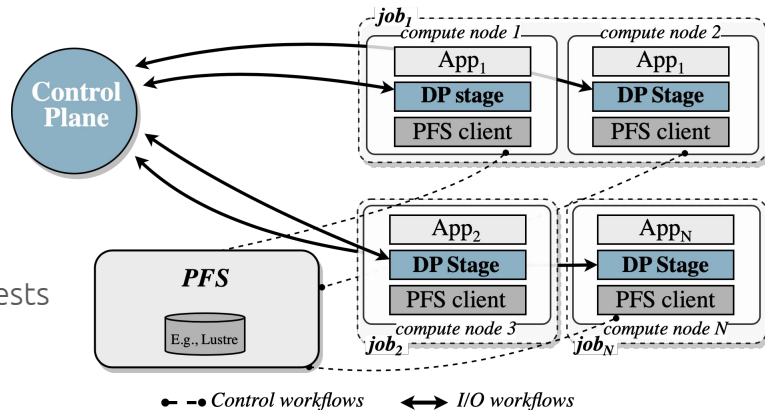


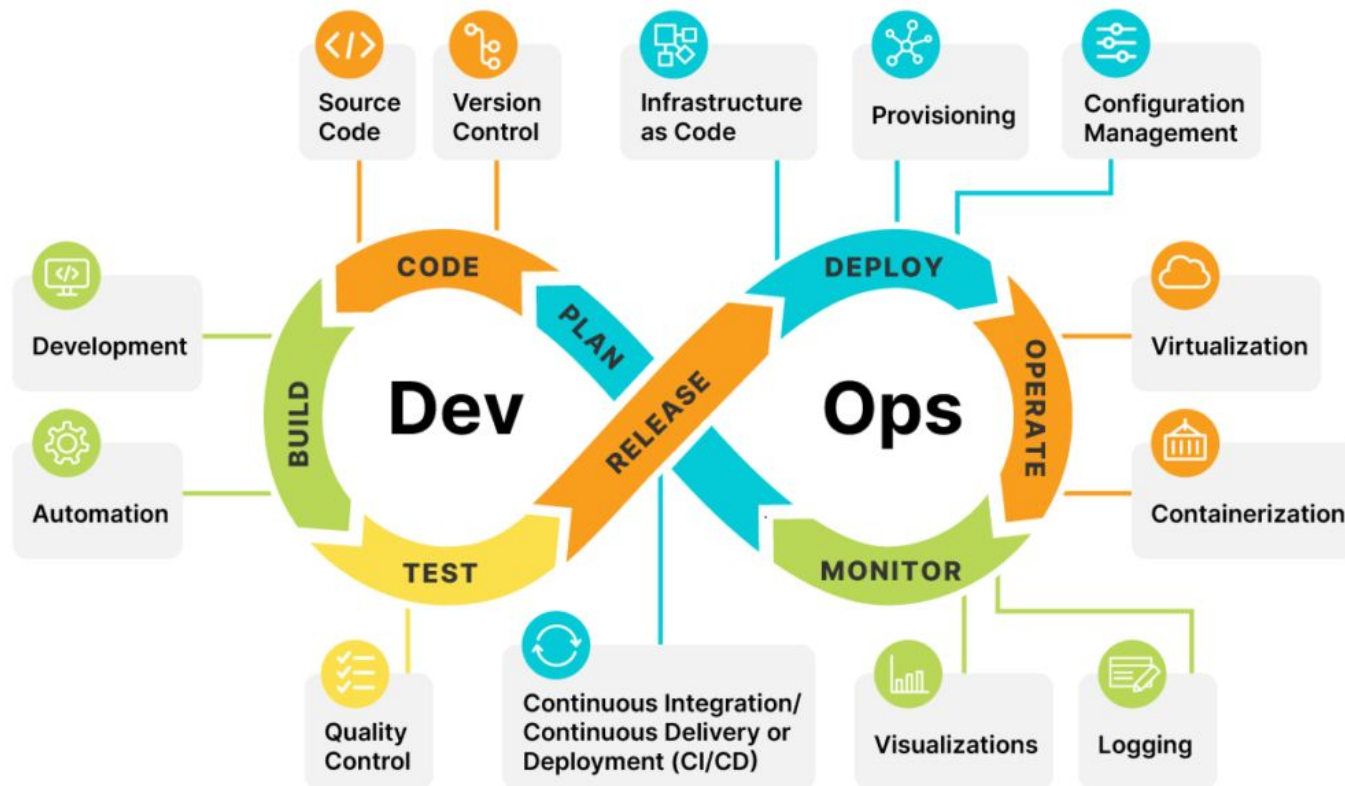
Virtualization Manager - Architecture



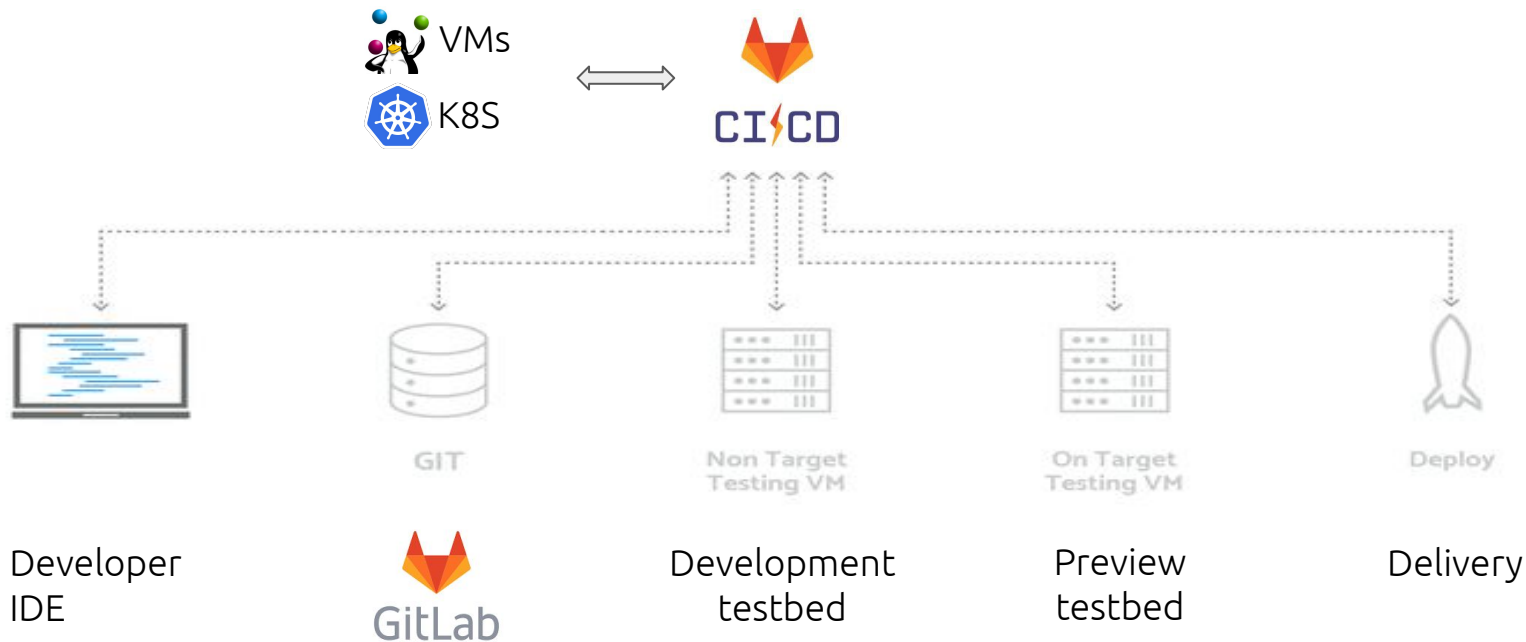
Software-Defined Storage - Overview

- Improve storage performance and management for HPC services
 - Ensure Quality of Service for I/O-intensive jobs
 - Alleviate I/O pressure at the shared parallel file system
- Key design features
 - Target both Big Data and Scientific workloads
 - Transparent to applications
 - Agnostic of the underlying storage resources (e.g., local and shared file systems)
- Based on the Software-Defined Storage paradigm
 - Data plane stages manage per application I/O requests
 - Control plane ensures holistic coordination across data plane stages



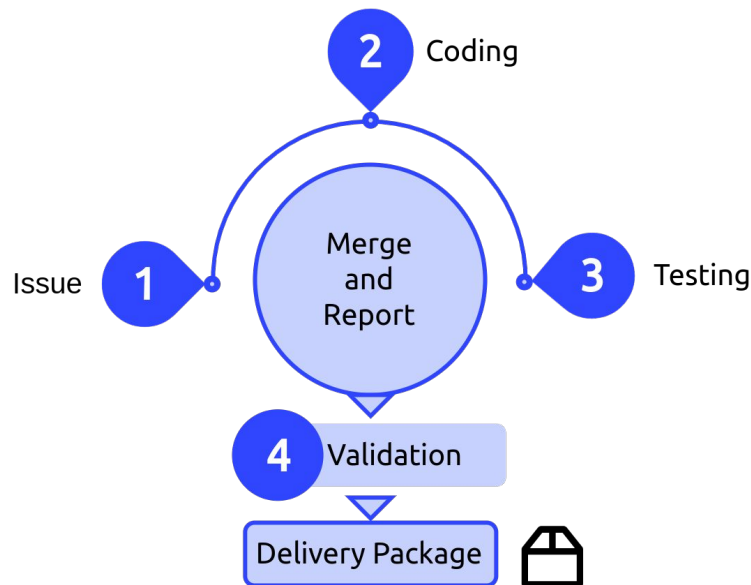


Integration: delivery workflow over testbeds



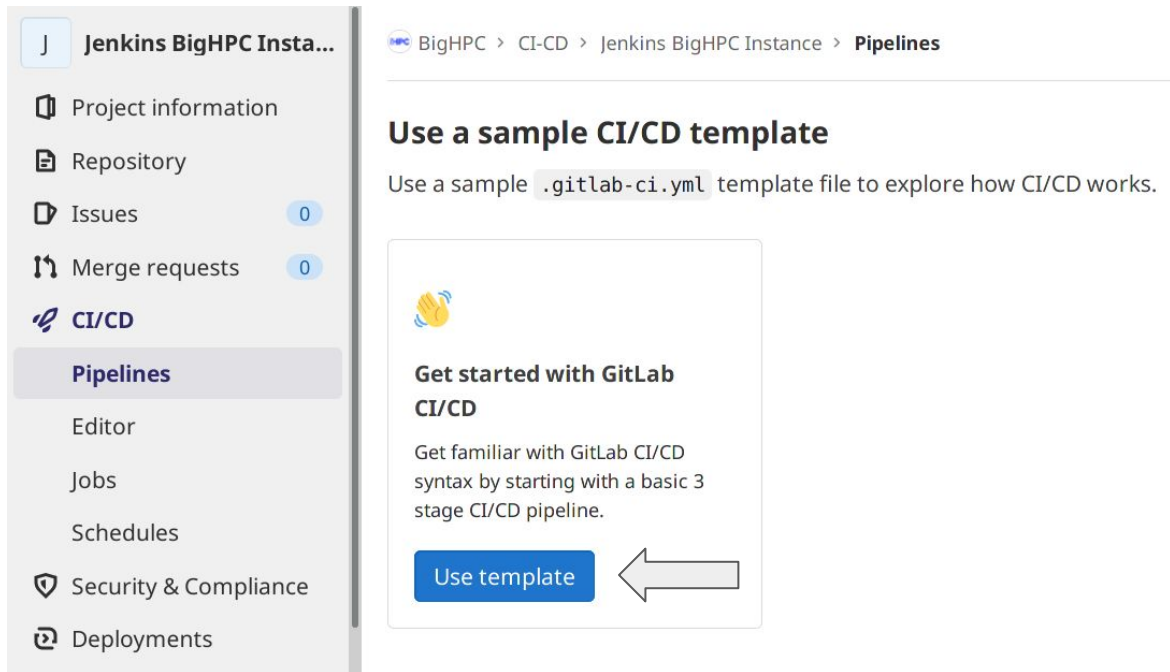
Integration: continuous development

- We use [Gitlab cloud platform](#) to create the pipelines to automate all the procedures between issue and code management.
- All the tests are conducted over Gitlab CI pipelines using the provided tools. The output of those tools are summarized in a report accessible from the pipeline job interface.



Integration: continuous development

- Create a new project in Gitlab
- Add pipeline from template



The screenshot displays the Jenkins BigHPC web interface. On the left is a sidebar with navigation links: 'Jenkins BigHPC Insta...', 'Project information', 'Repository', 'Issues' (with a count of 0), 'Merge requests' (with a count of 0), 'CI/CD', 'Pipelines' (highlighted), 'Editor', 'Jobs', 'Schedules', 'Security & Compliance', and 'Deployments'. The main content area shows a breadcrumb trail: 'BigHPC > CI-CD > Jenkins BigHPC Instance > Pipelines'. Below this is a heading 'Use a sample CI/CD template' followed by the text 'Use a sample `.gitlab-ci.yml` template file to explore how CI/CD works.' A card titled 'Get started with GitLab CI/CD' contains the text 'Get familiar with GitLab CI/CD syntax by starting with a basic 3 stage CI/CD pipeline.' and a blue 'Use template' button. A large white arrow points from the right towards the 'Use template' button.

Integration: continuous development



D

Default

Project information

Repository

Issues

0

Merge requests

0

CI/CD

Pipelines

Editor

Jobs

Schedules

Security & Compliance

Deployments

Monitor

Infrastructure

Packages & Registries

Analytics

<<

Collapse sidebar

8

Preparing environment

00:01

9

Running on runner-ed2dce3a-project-30845538-concurrent-0 via runner-ed2dce3a-srm-1635421744-8d13a1cd...

11

Getting source from Git repository

00:02

12

\$ eval "\$CI_PRE_CLONE_SCRIPT"

13

Fetching changes with git depth set to 50...

14

Initialized empty Git repository in /builds/bighpc/ci-cd/pipeline-templates/default/.git/

15

Created fresh repository.

16

Checking out fab3eb66 as main...

17

Skipping Git submodules setup

19

Executing "step_script" stage of the job script

00:10

20

Using docker image sha256:27d049ce98db4e55ddfaec6cd98c7c9cfd195bc7e994493776959db33522383b for ruby:2.5 with digest ruby@sha256:ecc3e4f5da13d881a415c9692bb52d2b85b090f38f4ad99ae94f932b3598444b ...

21

\$ echo "Linting code... This will take about 10 seconds."

22

Linting code... This will take about 10 seconds.

23

\$ sleep 10

24

\$ echo "No lint issues found."

25

No lint issues found.

27

Cleaning up project directory and file based variables

00:01

29

Job succeeded

lint-test-job

Retry

Duration: 24 seconds

Timeout: 1h (from project) ?

Runner: #380987 (ed2dce3a) shared-runners-manager-6.gitlab.com

Commit fab3eb66

Update README.md

✓ Pipeline #397447084 for main

development

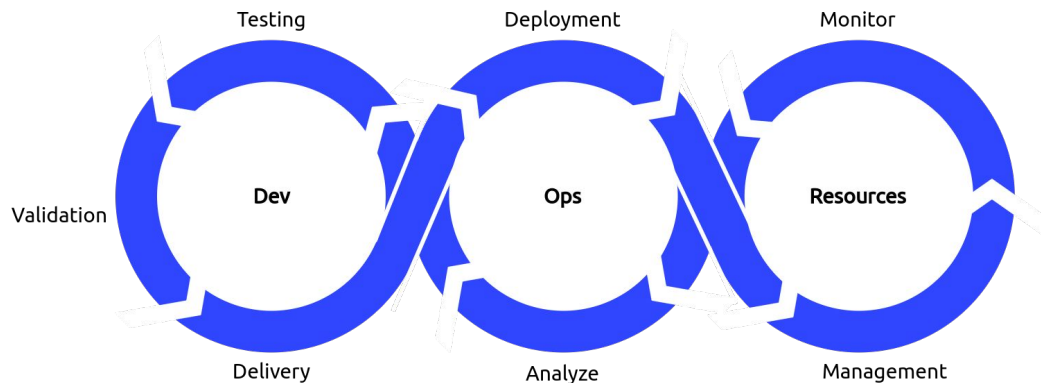
→ lint-test-job

security-dev-job

unit-test-job

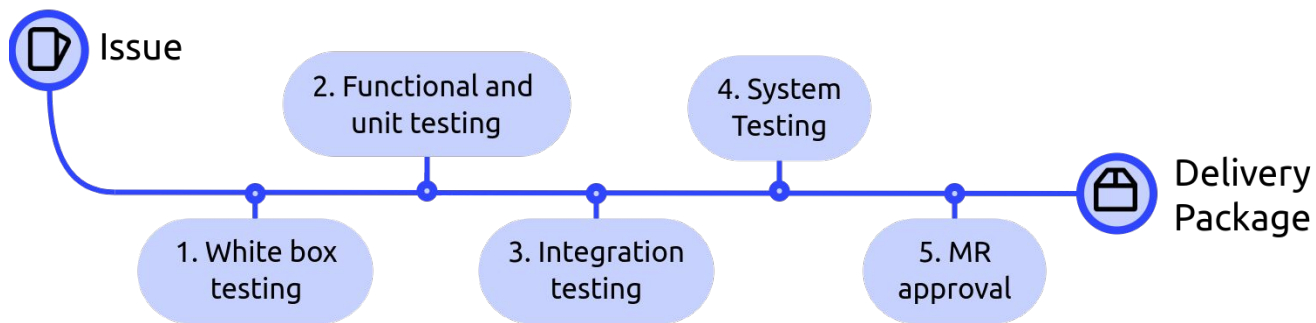
Involved development and operation domains on the continuous loop:

- Testing, Validation and Delivery
- Deployment and Analysis
- Management and Monitoring

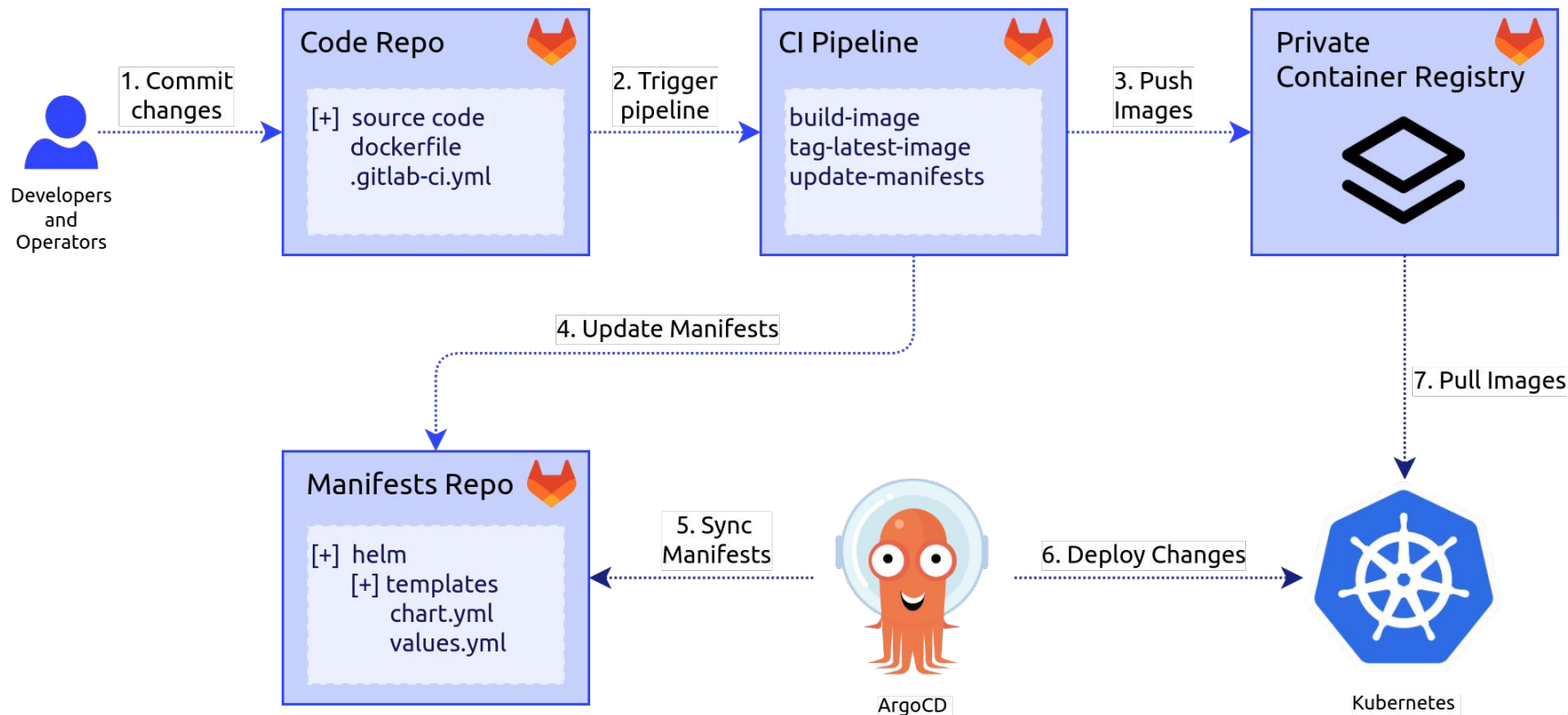


Validation is completed in three steps:

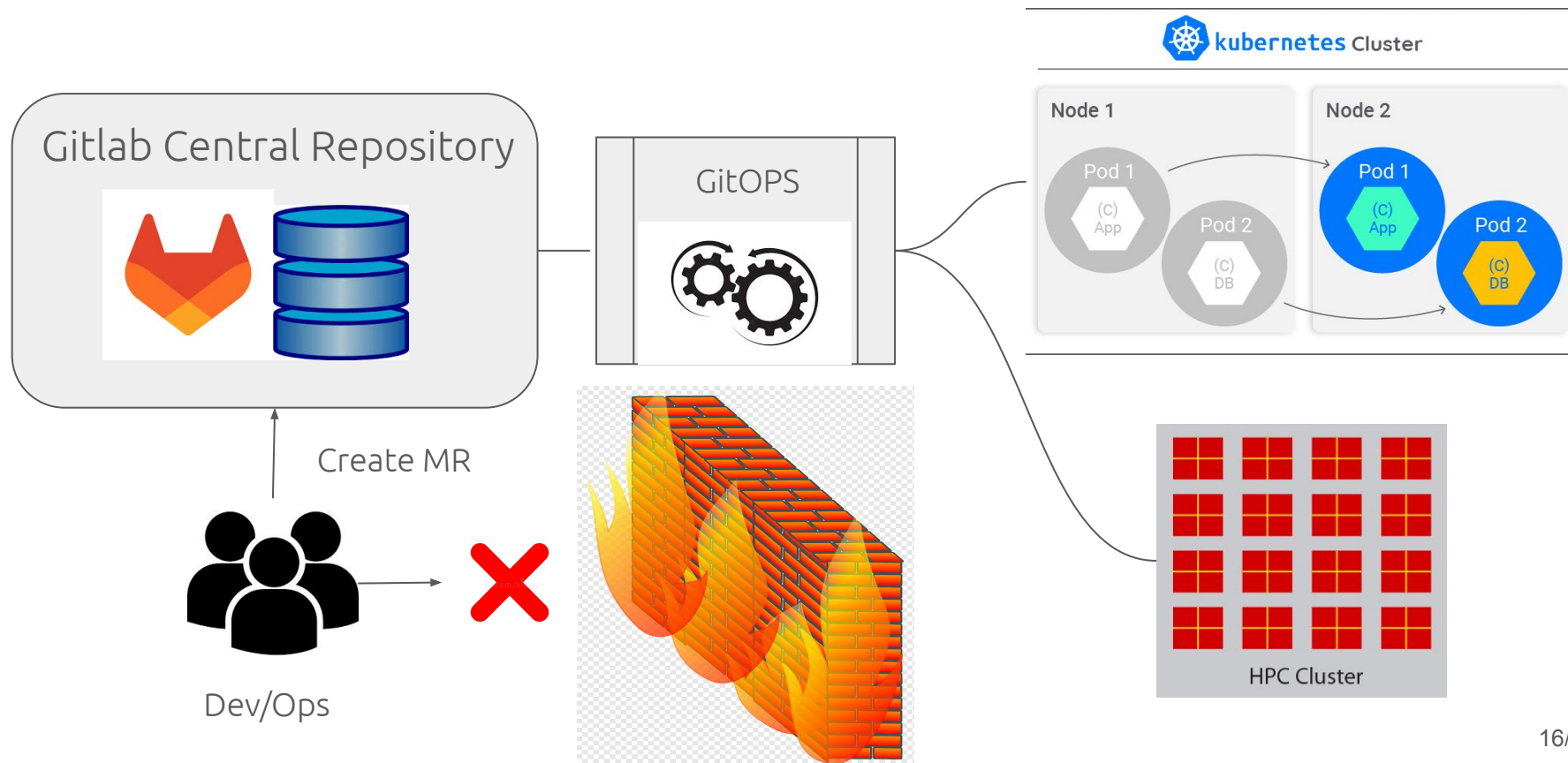
- tools running over the pipeline pass the defined tests
- reports are validated by the developers and operators
- merge requests are approved and a new release delivered



Integration: deployment, analysis and monitor



Integration: pilot overview using GitOps



BigHPC platform interfaces



Container Image Upload

File path: *

Path of file to upload [required]

Application name: *

Name of application that the container is built for.

Container name: *

Unique name for container

Library: *

Library information for the container in json format

Container type: * ⓘ

- ☐ Singularity (SIF)
☐ Docker

CPU instruction set: * ⓘ

- ☐ avx2
☐ avx512f

Accelerator:

Accelerator that container supports

Upload

Clear fields

Project developed by BigHPC Consortium.



Submit a BigHPC job

Container arguments:

Pass through these arguments to singularity

Name: *

Name of the job

Container image id: *

Container Image Id

Time: *

Time in minutes

Nodes: *

Number of nodes

Processes: *

Number of processes

PPN: *

Processes per node

Partition id: *

Cluster id

Input directory: *

Input directory relative to your root directory in bighpc datastore

Command: *

Execute a command in the container

Launch

Clear fields

Project developed by BigHPC Consortium.



Admin: set BigHPC partition

Partition name: *

Name of the partition

CPU: *

CPU architecture

Site: *

TACC

Memory: *

Memory size in GB

Sockets: *

Sockets number

Accelerator:

Sockets number

Scratch directory: *

Temporary input and output storage path

Login node: *

The FQDN of the login node for the partition

Setup path: *

Path in partition where BigHPC components are installed

Global SDS: *

SDS global controller name and port

Create

Clear fields

Project developed by BigHPC Consortium.

- [Unified framework](#) for running parallel and Big Data jobs at HPC centres
- [Container-based solution](#) to ease portability and deployment
- [Storage QoS policies](#) to ensure fairness and avoiding I/O bottlenecks
- [Monitoring and visualization](#) tools to understand cluster's utilization

Thank you for your attention!

More details at: <https://bighpc.wavecom.pt/>

Contact: info.bighpc@wavecom.pt



BIG HPC

HIGH
PERFORMANCE
COMPUTING

Partners:



Funding:

Cofinanciado por:

