

```
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
name = bpy.context.selected_objects[0]
bpy.data.objects[name].select = 1
```

BigHPC: Management Framework for Consolidated Big Data and High-Performance Computing

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

- The BigHPC consortium
- Overview of BigHPC platform and development challenges
- Why GitOPS and how to get it in practice
- GitOPS applied to BigHPC
- Next steps

Austin, USA

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

 wavecom

TACC **TACC**

 INESC TEC

 LIP

LIP

 **MACC**
Minho
Advanced
Computing
Center

MACC

Portugal

Wavecom

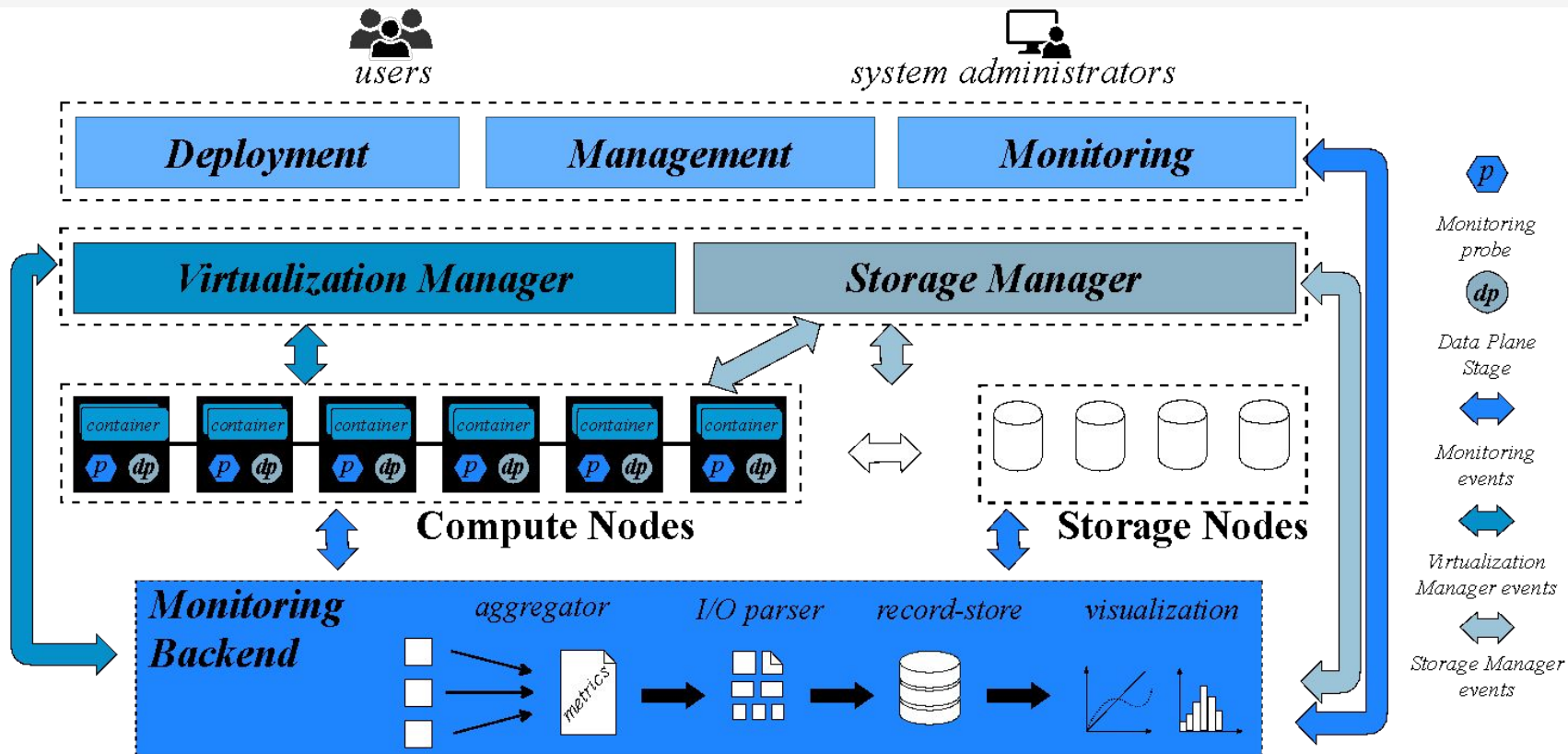
INESC TEC

LIP

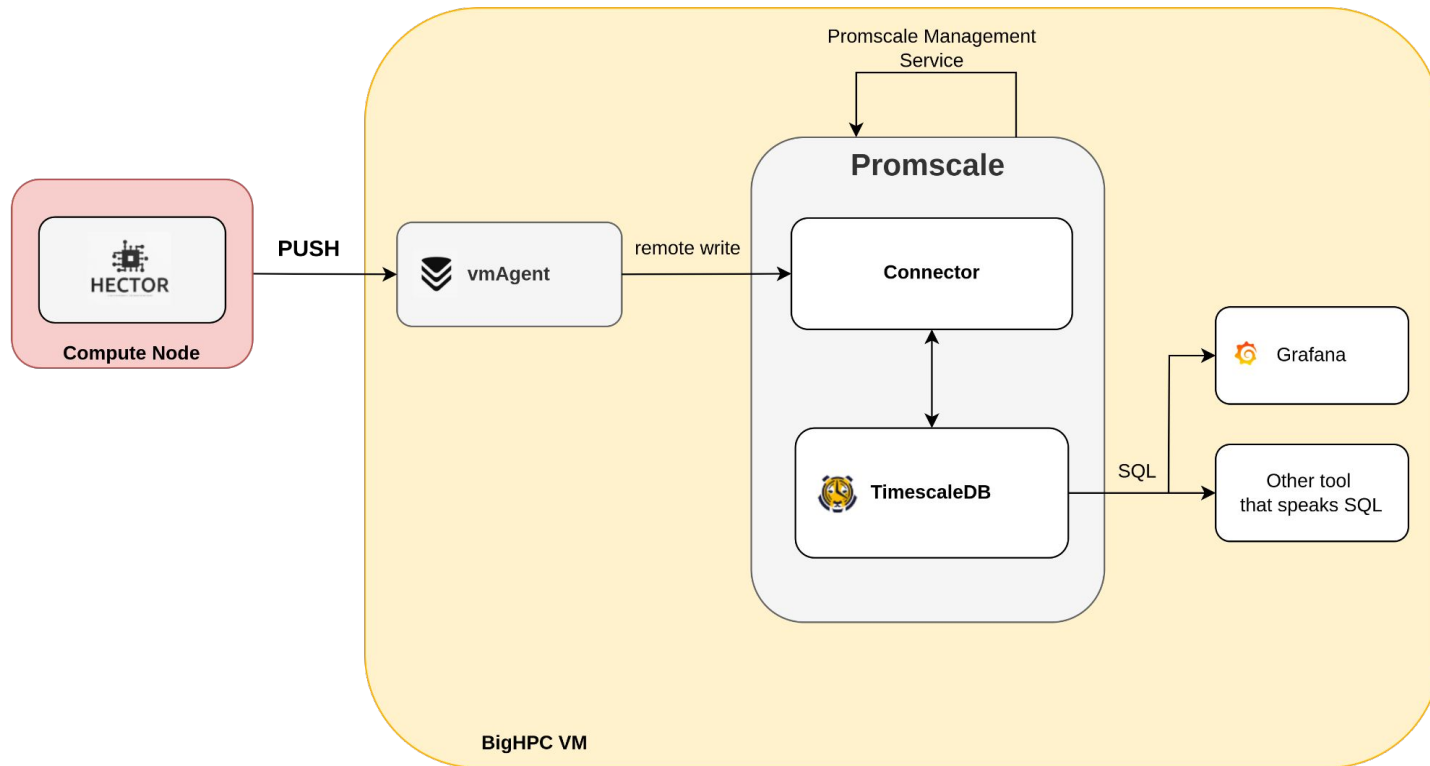
MACC



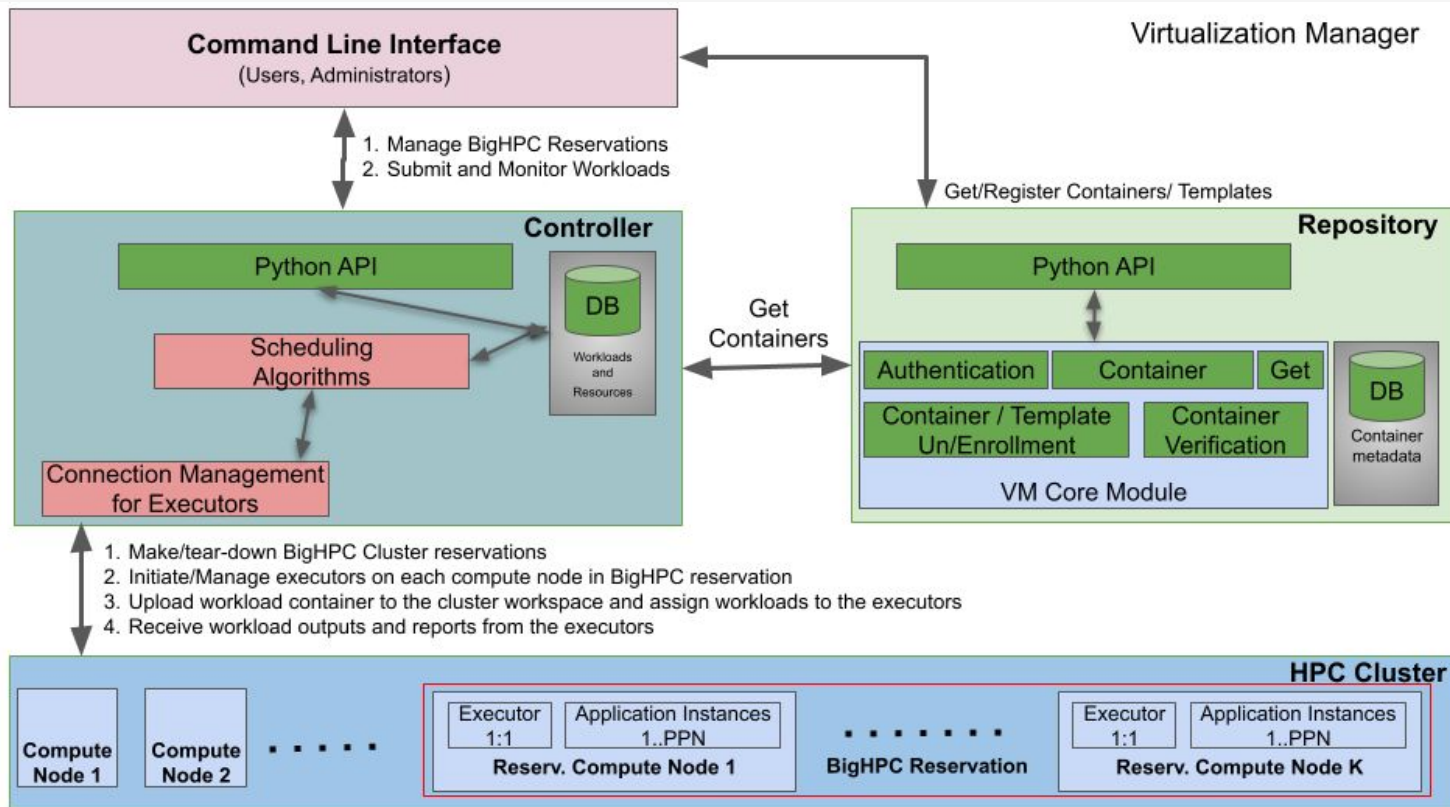
BigHPC platform overview



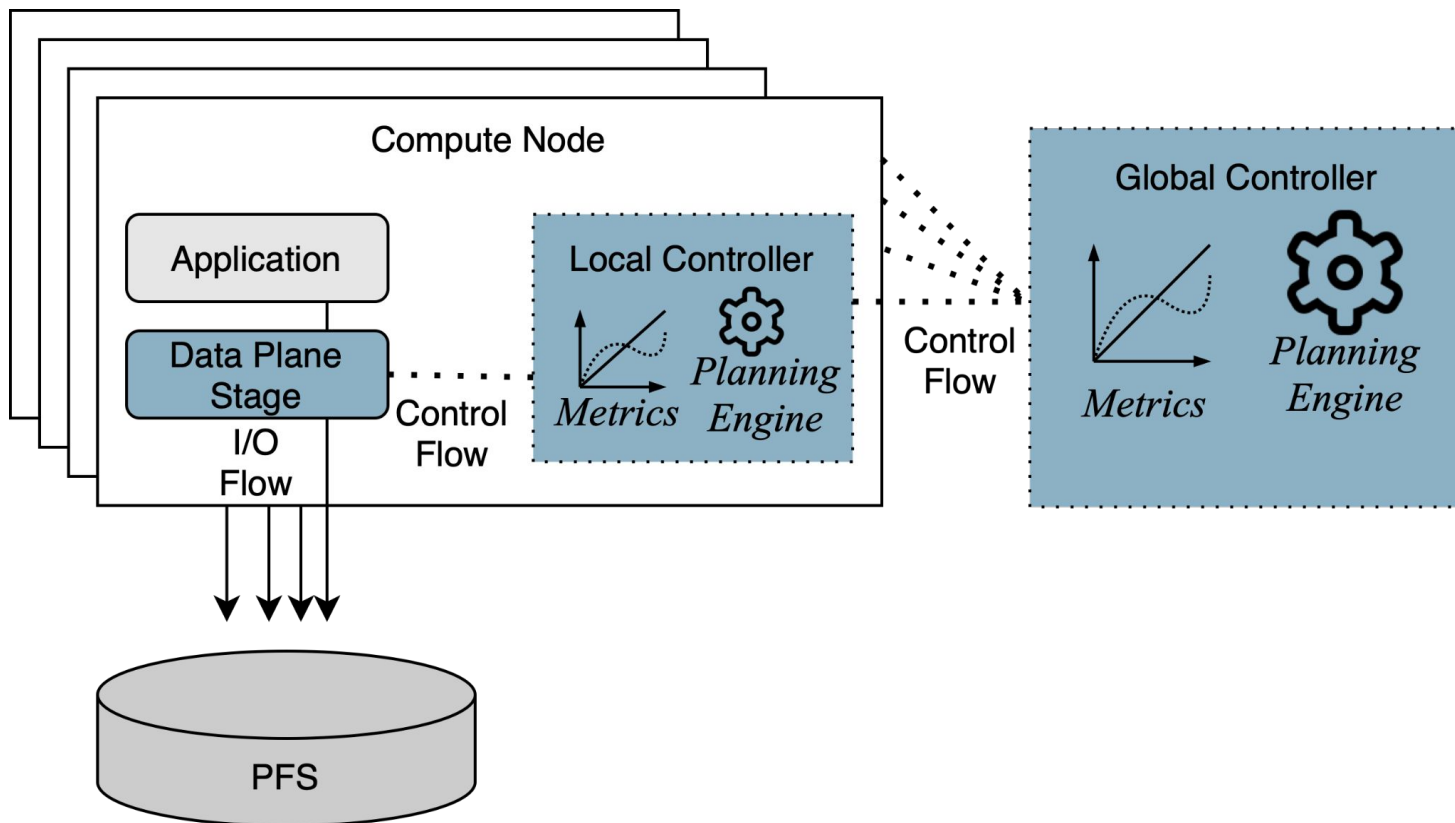
BigHPC platform: Monitoring Backend



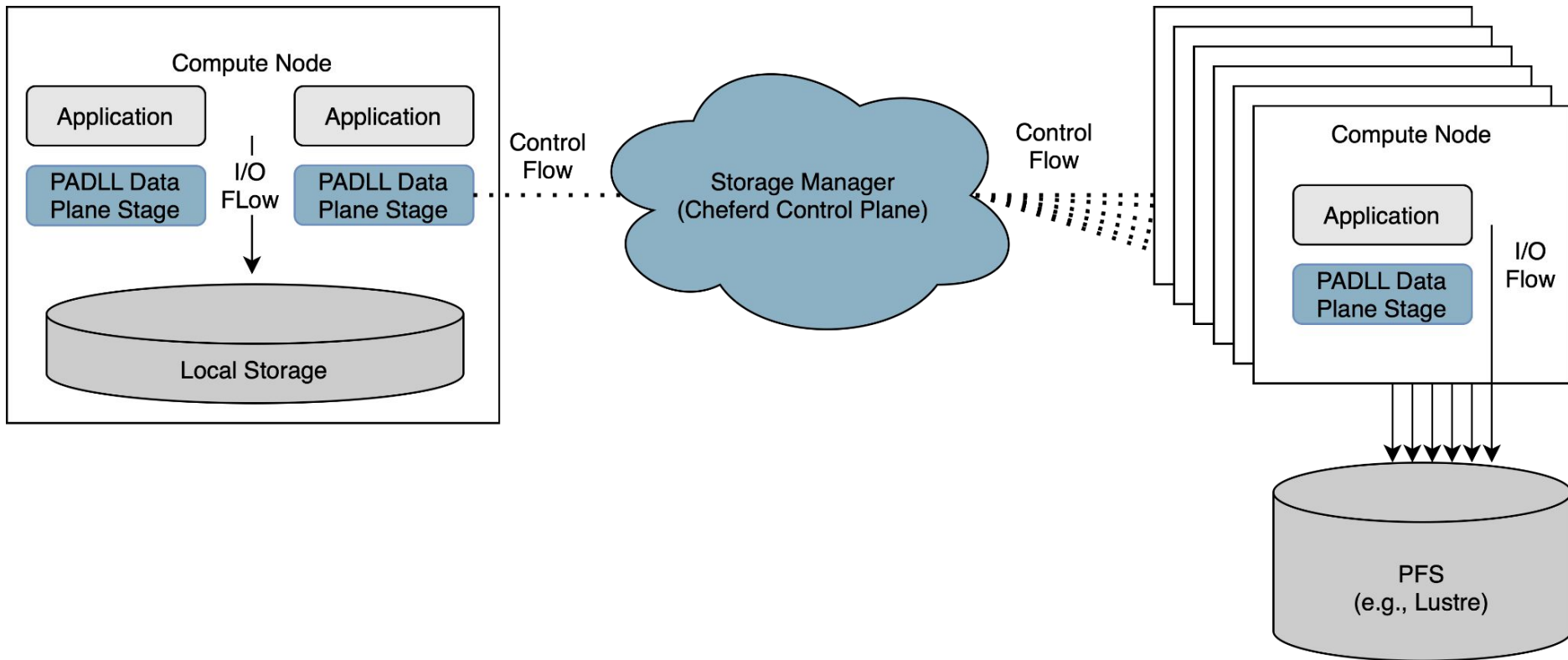
BigHPC platform: Virtualization Manager



BigHPC platform: Storage Manager Control Plane

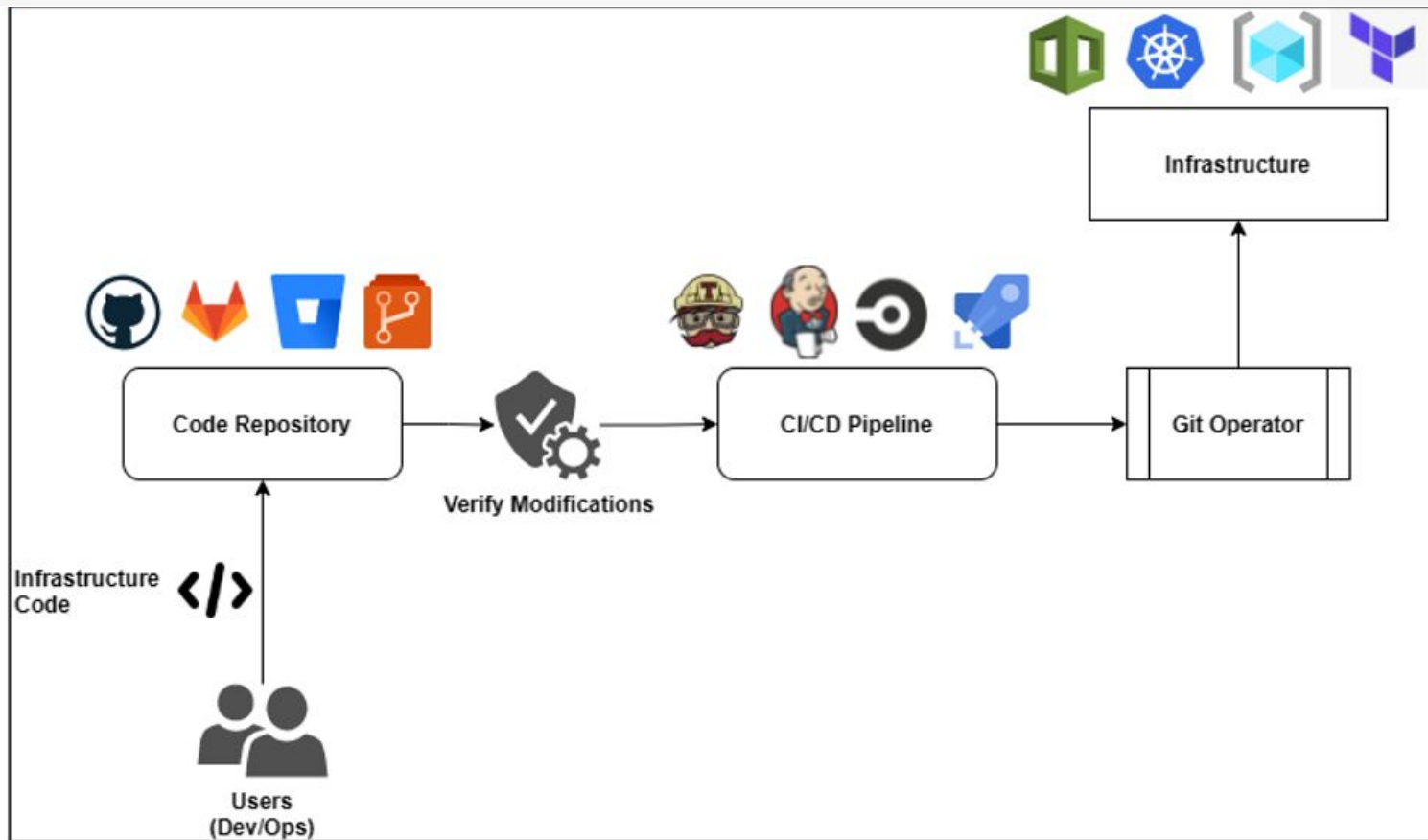


BigHPC platform: Storage Manager Data Plane

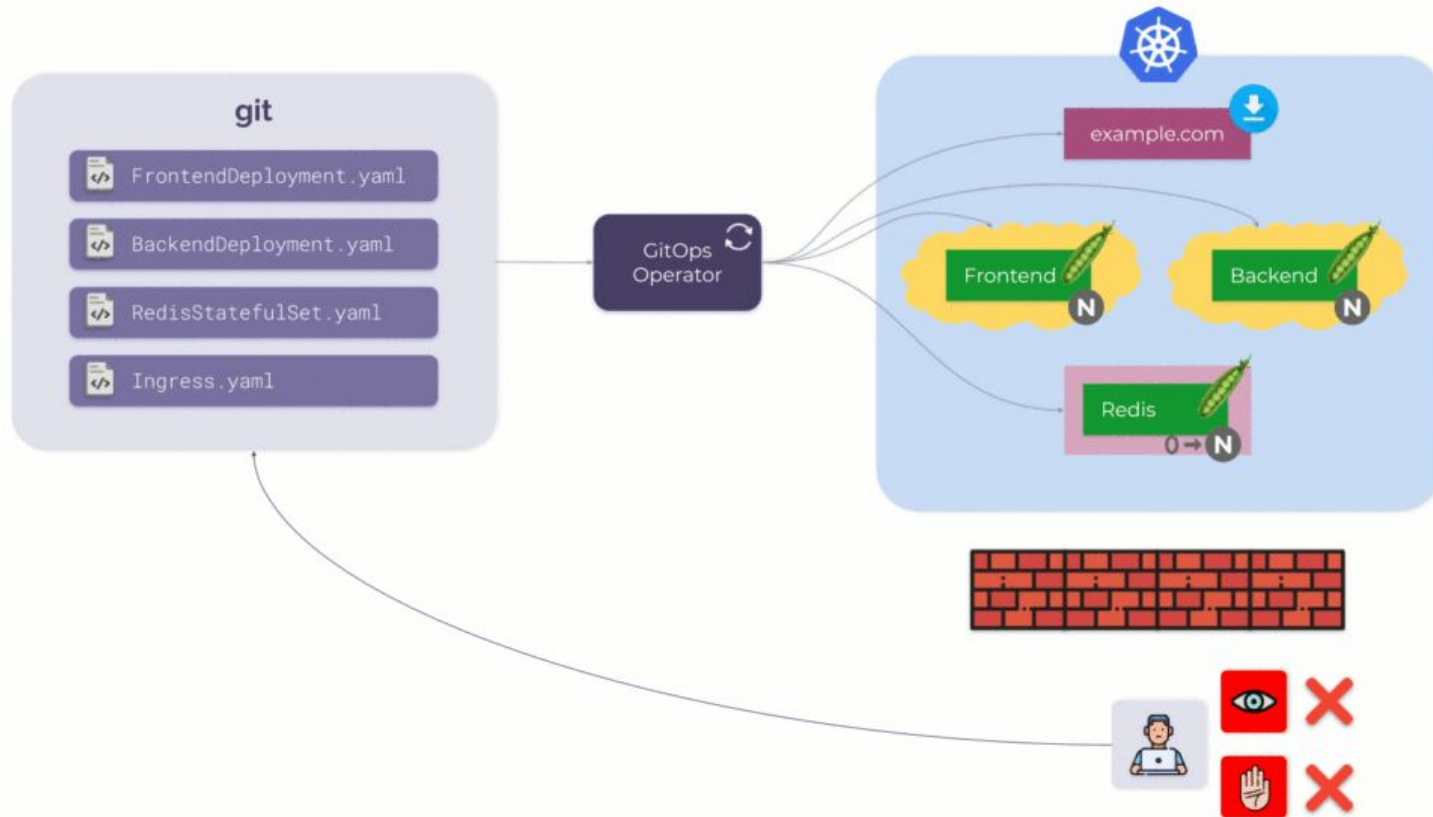


- Software quality
- Components integration
- Deploy more often
- Easy error recovery
- Keep teams together in same direction
- Share knowledge between teams
- Documented deliveries with complete changes history

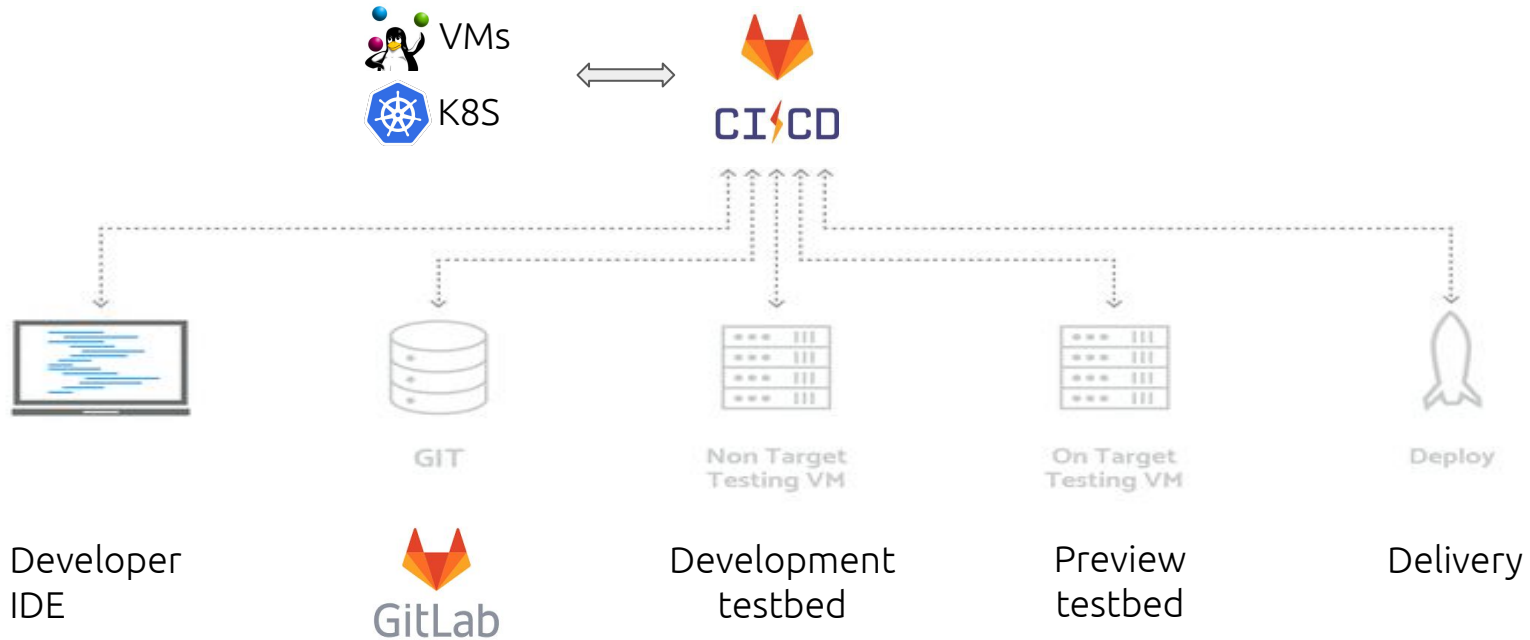
GitOps workflow



Why GitOps



Current status: implemented workflow



Review Job log directly from web interface



- 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

```
ruby:2.5 with digest ruby@sha256:ecc3e4f5da13d881a415c9692bb52d2b85b090f38f4ad99ae94f932b359844
4b ...
8 Preparing environment 00:01
9 Running on runner-ed2dce3a-project-30845538-concurrent-0 via runner-ed2dce3a-srm-1635421744-8d
13a1cd...
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:ecc3e4f5da13d881a415c9692bb52d2b85b090f38f4ad99ae94f932b359844
4b ...
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

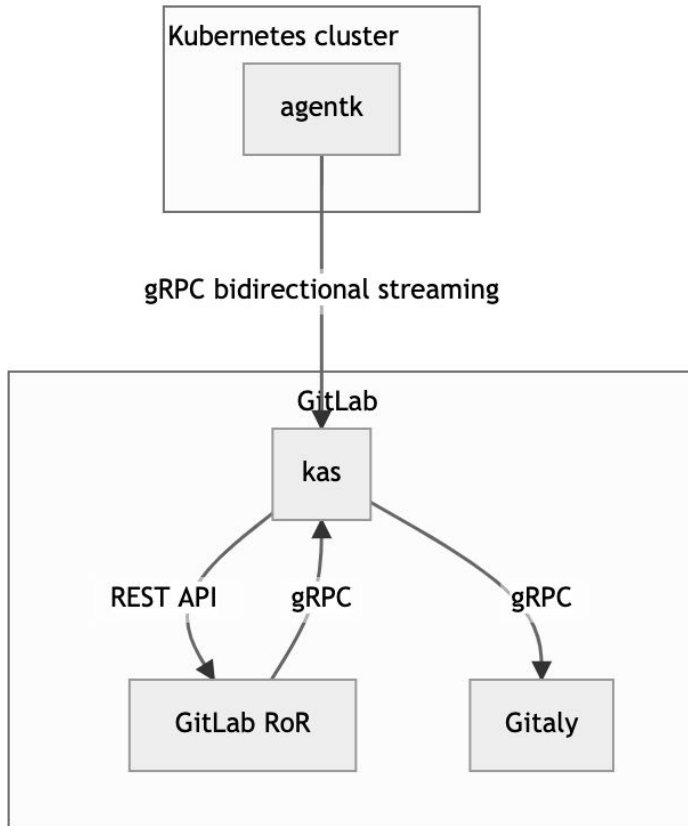
Commit fab3eb66 ⓘ
Update README.md

✓ **Pipeline** #397447084 for main ⓘ

development ▼

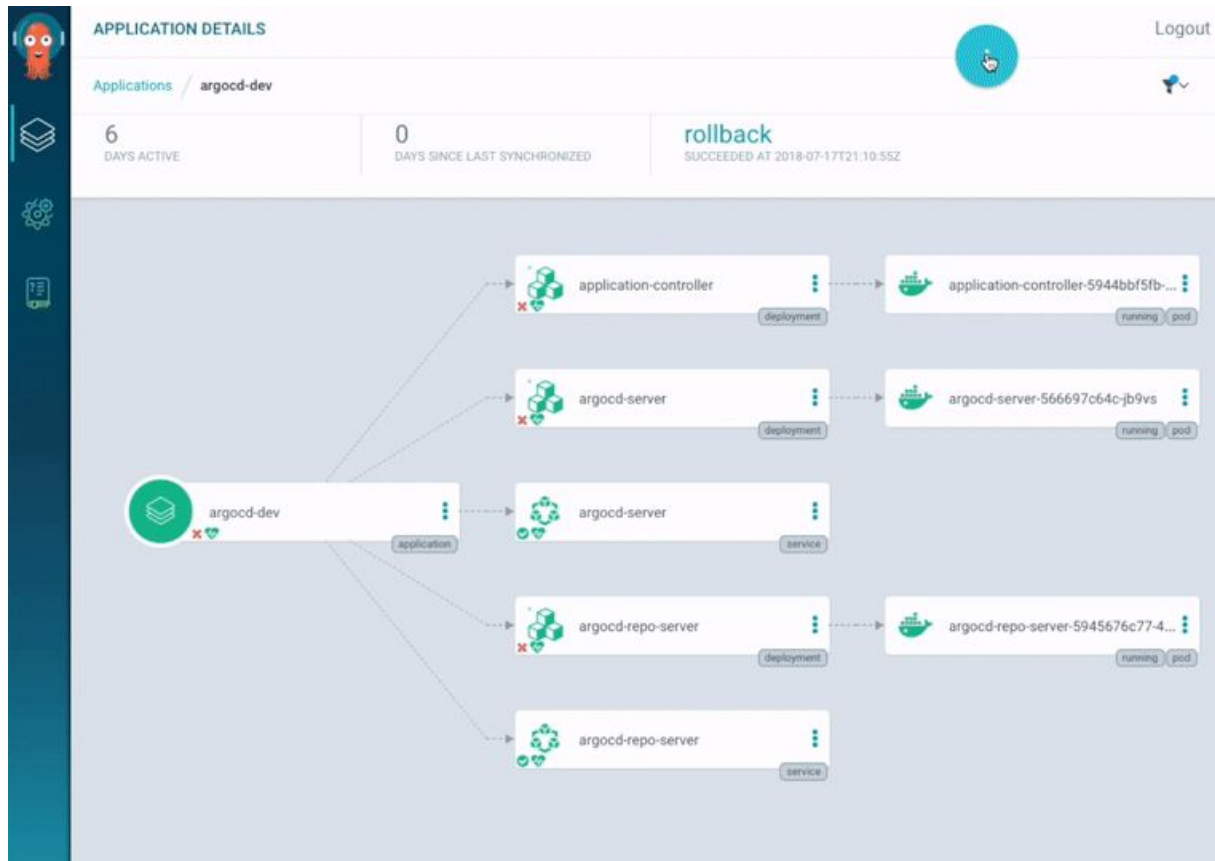
- ✓ lint-test-job
- ✓ security-dev-job
- ✓ unit-test-job

Attach the Environment with the Infrastructure



- GitOps push based strategy is implemented using Gitlab Agent
- agentk communicates to the Gitlab Agent Server (KAS) to perform GitOPS operations
- Changes applied to Kubernetes cluster anytime developer changes a manifest file managed within Gitlab
- This approach keeps Gitlab platform isolated from real infrastructure

GitOps aware of infrastructure state



- Argo CD provides a web user interface that allows to check application real time activity
- Health status analysis of application resources

- GitOPS insight
 - Improve GitOPS implementation, extending Gitlab CI/CD agent with kubernetes
 - Improve team collaboration using Gitlab platform
 - Test deployment error recovery taking the advantage of git
- Pilot
 - Test platform components with real use cases
 - Identify application performance issues

Thank you for your attention!

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

Contact: info.bighpc@wavecom.pt



BIG HPC HIGH PERFORMANCE COMPUTING

Partners:



Funding:

Co-financiado por:



User and administrator interfaces

Jupyterhub

Jupyter | admin's server On

Files **1** Running

Home / **2** **3** Upload New ↕

Select items to perform actions on them.

<input type="checkbox"/>	Name ↕	Status	Last Modified ↕
<input type="checkbox"/>	@Recycle		3 months ago
<input type="checkbox"/>	jupyter_example		21 days ago

Jupyterhub

Jupyter

Admin

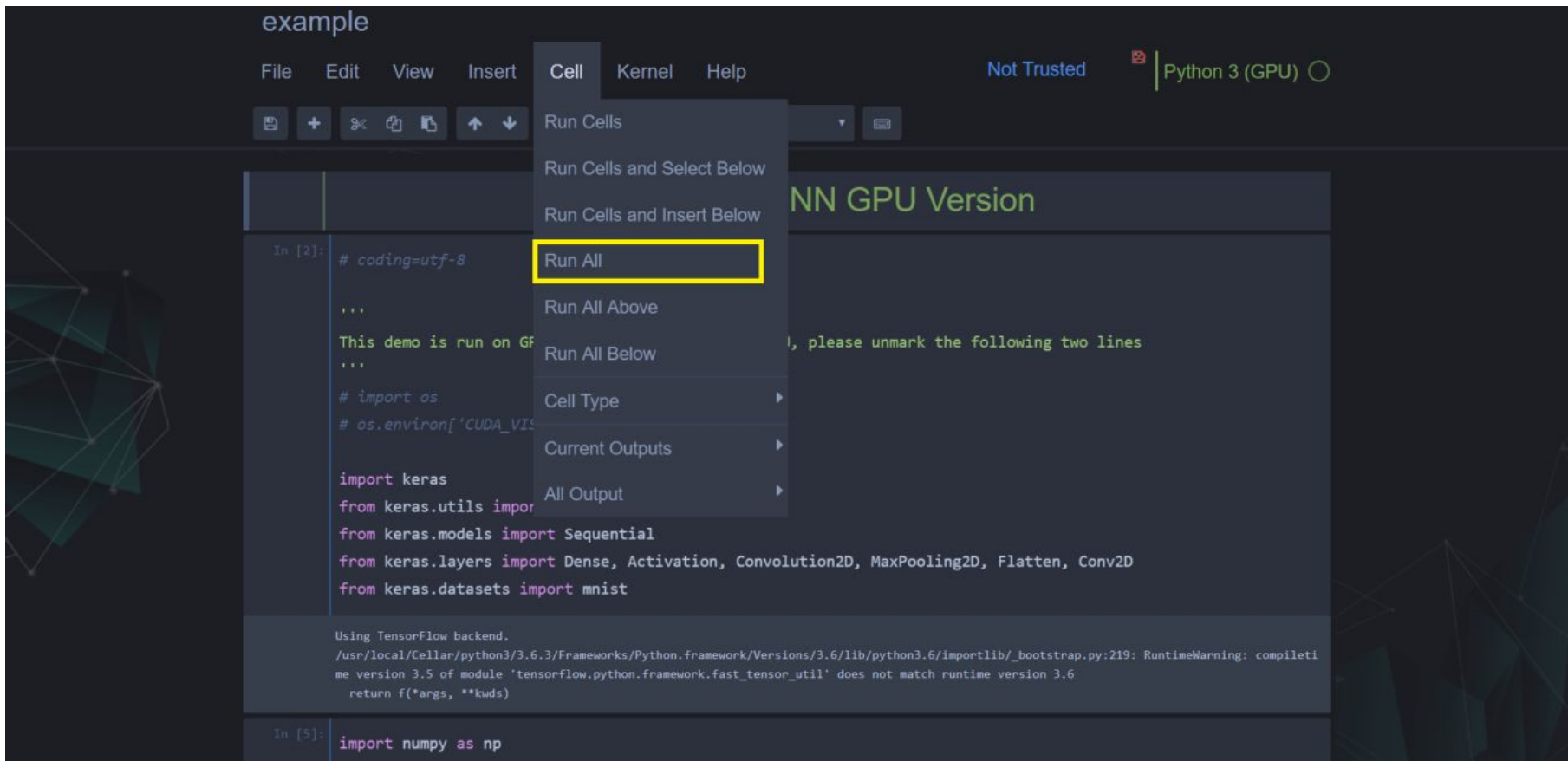
Server Management

Jupyterhub created a server for each NAS user. You can view the user list and manage their server status.

Stop all servers | All Server : 5

Username ↕	Description	Last Seen ↕	Server Status ↕	Action
admin	Admin	a few seconds ago	<input checked="" type="checkbox"/> On	Access Server
user	User	7 hours ago	<input type="checkbox"/> Off	

Manage code and do debugging



The screenshot shows a Jupyter Notebook interface with a dark theme. The title of the notebook is "example". The top menu bar includes "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". The status bar on the right indicates "Not Trusted" and "Python 3 (GPU)".

The "Cell" menu is open, showing the following options:

- Run Cells
- Run Cells and Select Below
- Run Cells and Insert Below
- Run All** (highlighted with a yellow box)
- Run All Above
- Run All Below
- Cell Type
- Current Outputs
- All Output

The code cell contains the following Python code:

```
In [2]: # coding=utf-8
...
This demo is run on GPU, please unmark the following two lines
...
# import os
# os.environ['CUDA_VISIBLE_DEVICES'] = '0'

import keras
from keras.utils import plot_model
from keras.models import Sequential
from keras.layers import Dense, Activation, Convolution2D, MaxPooling2D, Flatten, Conv2D
from keras.datasets import mnist
```

The output of the cell shows:

```
Using TensorFlow backend.
/usr/local/Cellar/python3/3.6.3/Frameworks/Python.framework/Versions/3.6/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: compiletime version 3.5 of module 'tensorflow.python.framework.fast_tensor_util' does not match runtime version 3.6
  return f(*args, **kwargs)
```

The next code cell is partially visible:

```
In [5]: import numpy as np
```

OnDemand: extend access to applications



OSC OnDemand Files Jobs Clusters Desktops Desktop Apps



Ohio Supercomputer Center

An OH·TECH Consortium Member

OnDemand provides an integrated, single access point for all of your HPC resources.

of your HPC

- Interactive HPC
 - ANSYS Workbench
 - Abaqus/CAE
 - COMSOL Multiphysics
 - Jupyter
 - MATLAB
 - RStudio Server
- Virtual Desktop Interface
 - Paraview

OSC OnDemand Files Jobs Clusters Interactive Apps My Interactive Sessions All Apps

- >_Owens Shell Access
- >_Pitzer Shell Access
- >_Ruby Shell Access
- System Status

Ohio Supercomputer Center

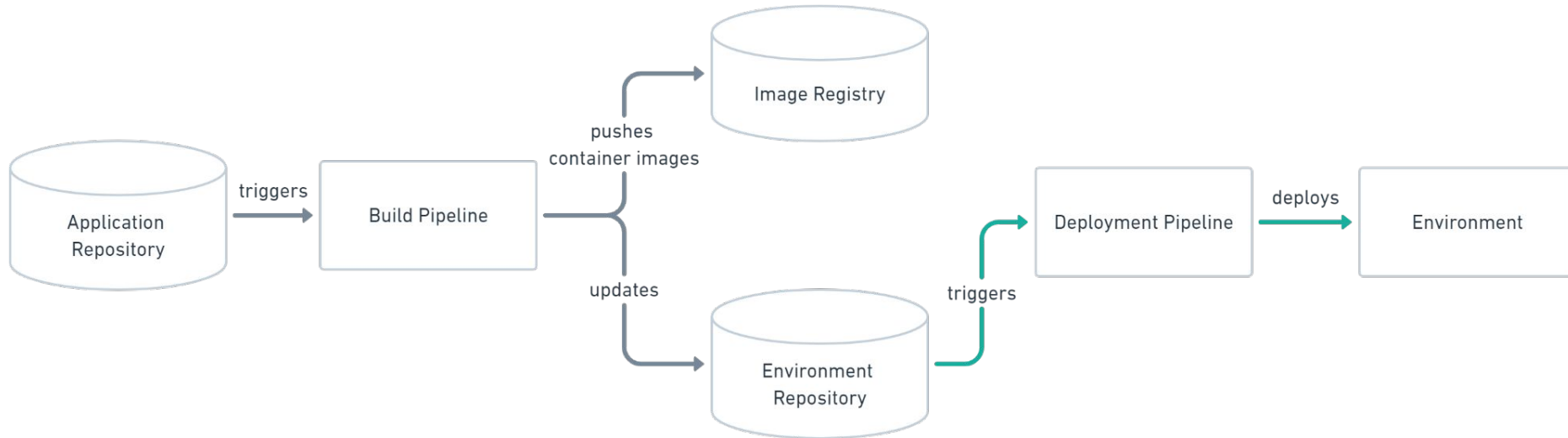
An OH·TECH Consortium Member

OnDemand provides an integrated, single access point for all of your HPC resources.

- Its inception in 2017 by Weaveworks
- Uses developer common tools:
 - git version control system to track code changes
 - continuous deployment tools
- Have a git repository that contains infrastructure declarative description
- Production environment match the described state in the repository
- Deployment and updates means push changes to a git repository

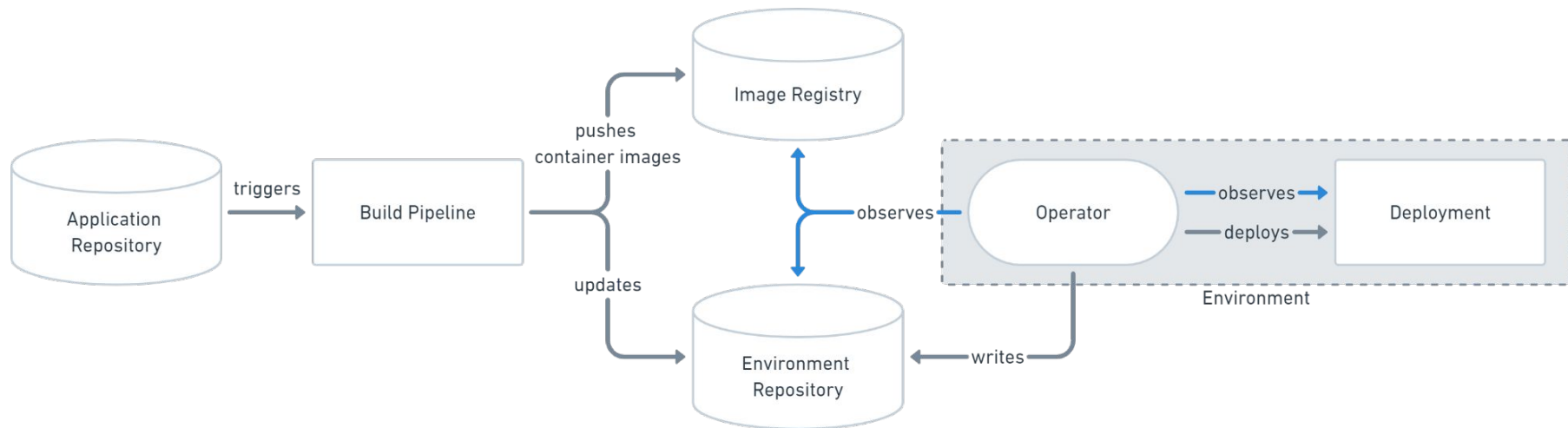
- Always use same procedure to deploy applications without the burden of switching between required tools
- Lighter learning curve since everything happens on git with less effort
- Error recovery as easy as git revert
- Easier credential management since only required access to git repository
- No need to give developers direct access to the deployed endpoints
- Complete description of what is deployed
 - every changes goes through the git repository
 - complete history of every change made to the system
- Share knowledge with great commit messages where everybody can reproduce and find examples how to set up new systems

Deployment strategy: push based



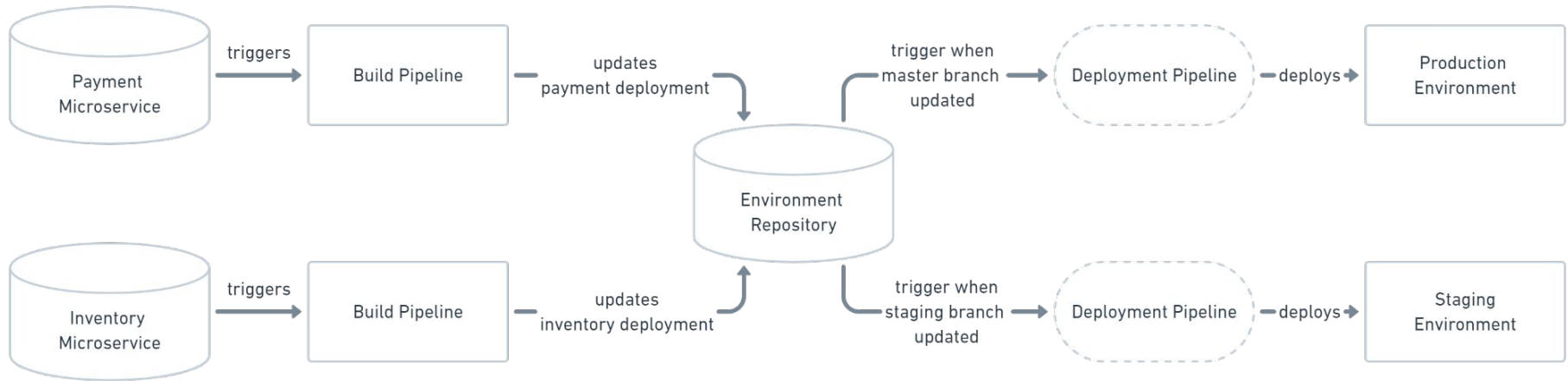
- Implemented by Gitlab CI/CD, Jenkins, Circle CI or Travis CI
- Code updates trigger pipeline execution (external event)
- Environment configuration repository is updated with new deployment descriptors
- misses automatic check of environment desired state

Deployment strategy: pull based



- Operator is introduced comparing with previous strategy
- Implemented by Argo CD, Flux CD and Gitlab starting from v15
- Operator checks consistency between deployed infrastructure, environment repository and image registry

Multiple Applications and Environments



- GitOPS also support multiple build pipelines from different repositories
- Changes in each project generate an update to environment repository
- Multiple environments with GitOPS means separate branches in environment repository
- Operator will react on changes and run the associated environment pipeline

- Adopting GitOPS using Gitlab platform
 - Implement continuous deployment applying software quality best practices
 - Well documented deliveries with complete history of system changes
 - Promote shared knowledge and get teams together
 - Easy to reproduce the thought process of changing infrastructure
 - Get the examples to setup new systems

- Testbeds and Pilot
 - Review the hardware profiles and application requirements
 - Define execution environments for the required testbeds
 - Gather applications that covers the demanded use case requirements
 - Prepare the pilot to test platform components with collected applications over HPC
 - Collect the metrics and get the report from jobs submission

Current status: team collaboration



CI-CD

- Subgroup information
- Issues** 3
 - List
 - Board**
 - Milestones
 - Merge requests 0
 - Security
 - Kubernetes
 - Packages & Registries
 - Settings

BigHPC > CI-CD > **Issue Boards**

Development Search Show labels Edit board Create list

> **Open** 3 +

- Pilot testbed review**
critical discussion documentation
highpc/ci-cd/task-5-3#3
- Deployment, configuration and usability**
discussion documentation
highpc/ci-cd/task-5-3#2
- Platform evaluation using real applications**
discussion documentation
highpc/ci-cd/task-5-3#1

> **Closed** 0

« Collapse sidebar

Current status: team collaboration



C CI-CD

Subgroup information

Activity

Labels

Members

Issues 3

Merge requests 0

Security

Kubernetes

Packages & Registries

Settings

BigHPC > CI-CD > Activity

All Push events Merge events Issue events Comments Wiki Designs Team



Samuel Bernardo @samuellip

Commented on issue #3 "Pilot testbed review" at [BigHPC / CI-CD / Usability ar Development:...](#)



Samuel Bernardo @samuellip

Commented on issue #1 "Platform evaluation using real applications" at [BigHI MPI multinode job...](#)



Samuel Bernardo @samuellip

Commented on issue #1 "Platform evaluation using real applications" at [BigHI submit workload request to the scheduler..](#)

Gitlab stages for the project

- Development testbed
- Preview testbed
- Delivery to production (create a release)

Build

build-job

Development

unit-test-job

lint-test-job

security-dev-job

Preview

integration-test-job

functional-test-job

security-prev-job

Production

deploy-job

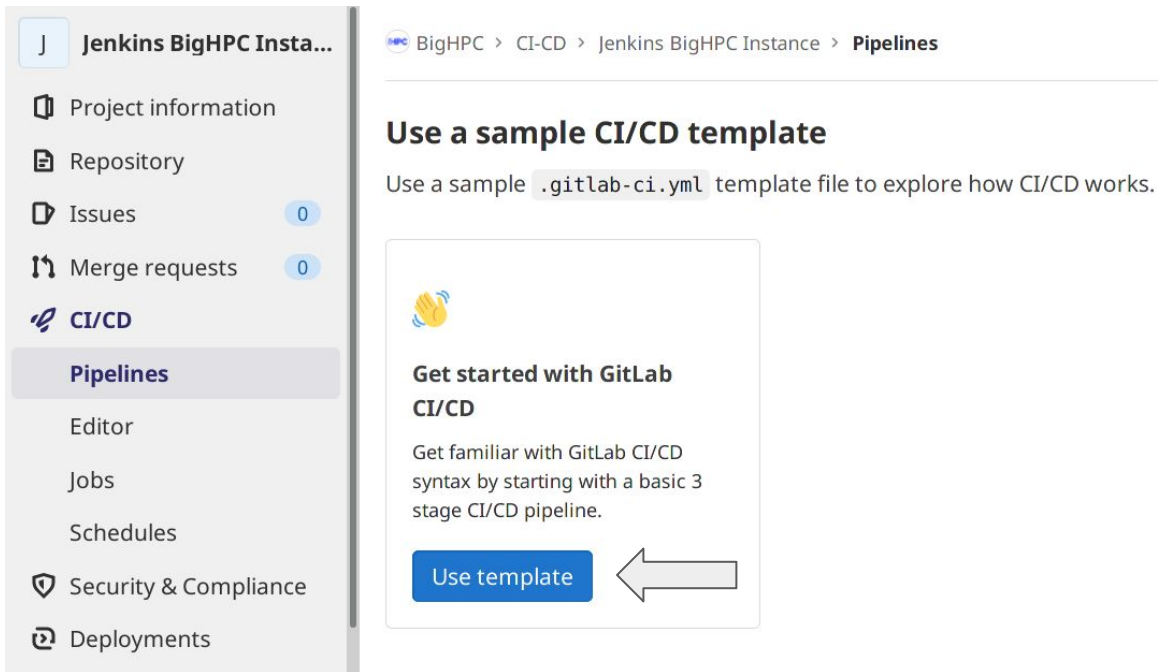
Create the pipeline configuration



The screenshot shows the GitLab web interface for editing a pipeline configuration. The left sidebar contains navigation options: Default, Project information, Repository, Issues (0), Merge requests (0), CI/CD (selected), Pipelines, Editor (highlighted), Jobs, Schedules, Security & Compliance, Deployments, Monitor, Infrastructure, and Collapse sidebar. The main editor area displays a YAML file with the following content:

```
14 # For more information, see: https://docs.gitlab.com/ee/ci/yaml/index.html#stages
15
16 stages:           # List of stages for jobs, and their order of execution
17 | - build
18 | - development
19 | - preview
20 | - production
21
22
23 #####
24 # Build stage #
25 #####
26
27 build-job:
28 |   stage: build
29 |   script:
30 |     - echo "Compiling the code..."
31 |     - echo "Compile complete."
32
33
34 #####
35 # Development stage #
36 #####
37
38 unit-test-job:
```


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




J Jenkins BigHPC Insta...

- Project information
- Repository
- Issues 0
- Merge requests 0
- CI/CD
- Pipelines**
- Editor
- Jobs
- Schedules
- Security & Compliance
- Deployments

BigHPC > CI-CD > Jenkins BigHPC Instance > Pipelines

Use a sample CI/CD template

Use a sample `.gitlab-ci.yml` template file to explore how CI/CD works.

 **Get started with GitLab CI/CD**

Get familiar with GitLab CI/CD syntax by starting with a basic 3 stage CI/CD pipeline.

[Use template](#) ←

Check pipeline results

Click over status of pipeline after passing the tests

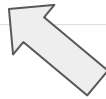
BigHPC > Pipeline Templates > Default > Pipelines

All 4 Finished Branches Tags

Clear runner caches CI lint Run pipeline

Filter pipelines Show Pipeline ID ▾

Status	Pipeline ID	Triggerer	Commit	Stages	Duration	
passed	#397447084 latest		main ↪ fab3eb66 Update README.md		🕒 00:04:02 🕒 7 minutes ago	
passed	#397446186		main ↪ 3695b7e1 Update .gitlab-ci.yml f...		🕒 00:04:04 🕒 8 minutes ago	



Look into pipeline stages

D **Default**

Project information

Repository

Issues 0

Merge requests 0

CI/CD

Pipelines

Editor

Jobs

Schedules

Security & Compliance

Deployments

Monitor

Infrastructure

Packages & Registries

Analytics

🕒 8 jobs for `main` in 4 minutes and 2 seconds

🚩 `latest`

🔗 [fab3eb66](#) 📄

🔗 No related merge requests found.

Pipeline Needs Jobs 8 Tests 0

Build

✅ build-job 🔄

Development

✅ lint-test-job 🔄

✅ security-de... 🔄

✅ unit-test-job 🔄

Preview

✅ functional-t... 🔄

✅ integration-... 🔄

✅ security-pre... 🔄

Production

✅ deploy-job 🔄

Check jobs state

- 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

Pipeline Needs **Jobs 8** Tests 0

Status	Name	Job ID	Coverage
Build			
	build-job	#1725216939	00:00:11 13 minutes ago
Development			
	lint-test-job	#1725216943	00:00:24 13 minutes ago
	security-dev-job	#1725216944	00:01:12 12 minutes ago
	unit-test-job	#1725216940	00:01:24 12 minutes ago
Preview			
	functional-test-job	#1725216947	00:01:13 11 minutes ago

- Wiki
- Snippets
- Settings
 - General
 - Integrations
 - Webhooks
 - Repository
 - CI/CD**
 - Monitor
 - Pages
 - Packages & Registries
 - Usage Quotas
- « Collapse sidebar

- Gitlab Runners are agents that run CI/CD jobs from Gitlab
- Gitlab Runner implements executors used to run builds in different environments
- Environments comprehends different Operating Systems and available tools, such as docker and kubernetes
- Each Gitlab Runner can have different access policies and can be limited to pipelines asking for specific tag


Group runners


These runners are shared across projects in this group.

Group runners can be managed with the [Runner API](#).

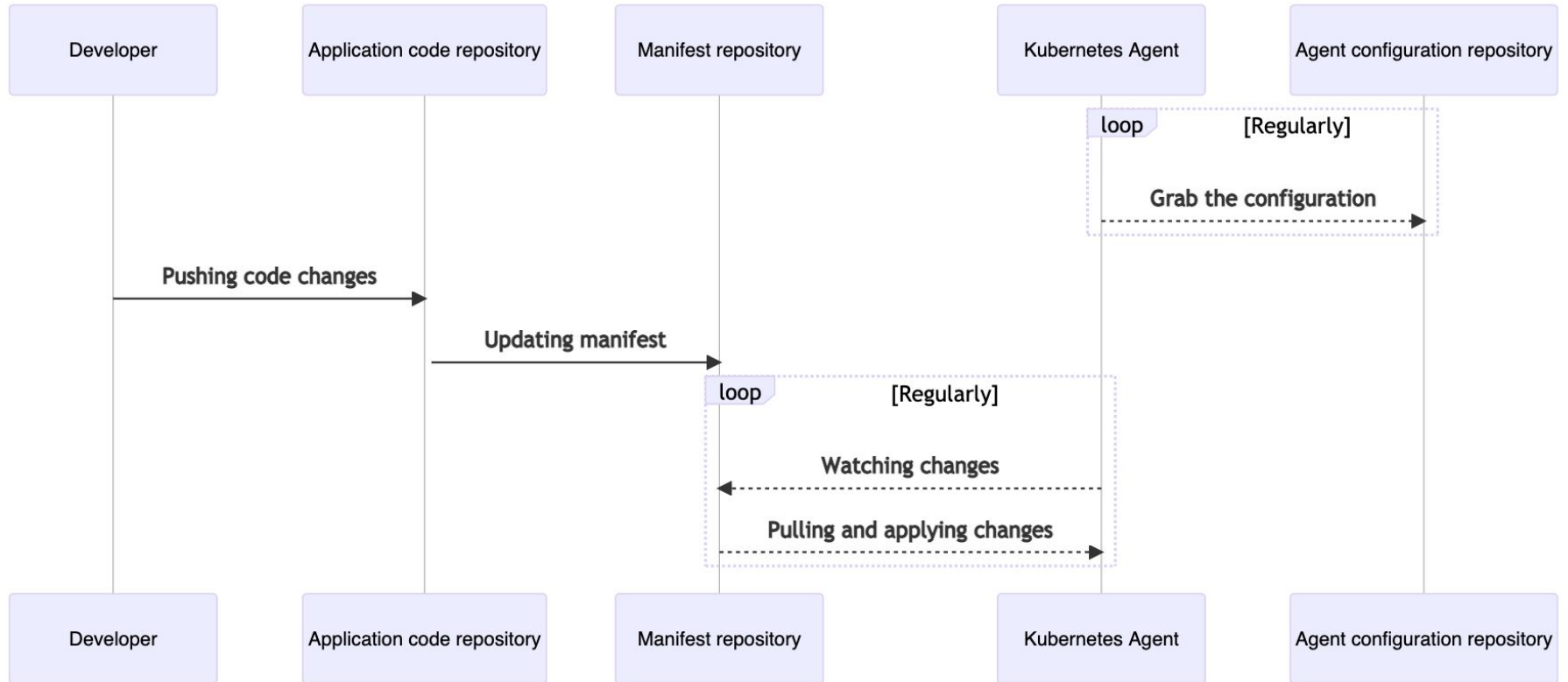
Disable group runners for this project

Available group runners: 2

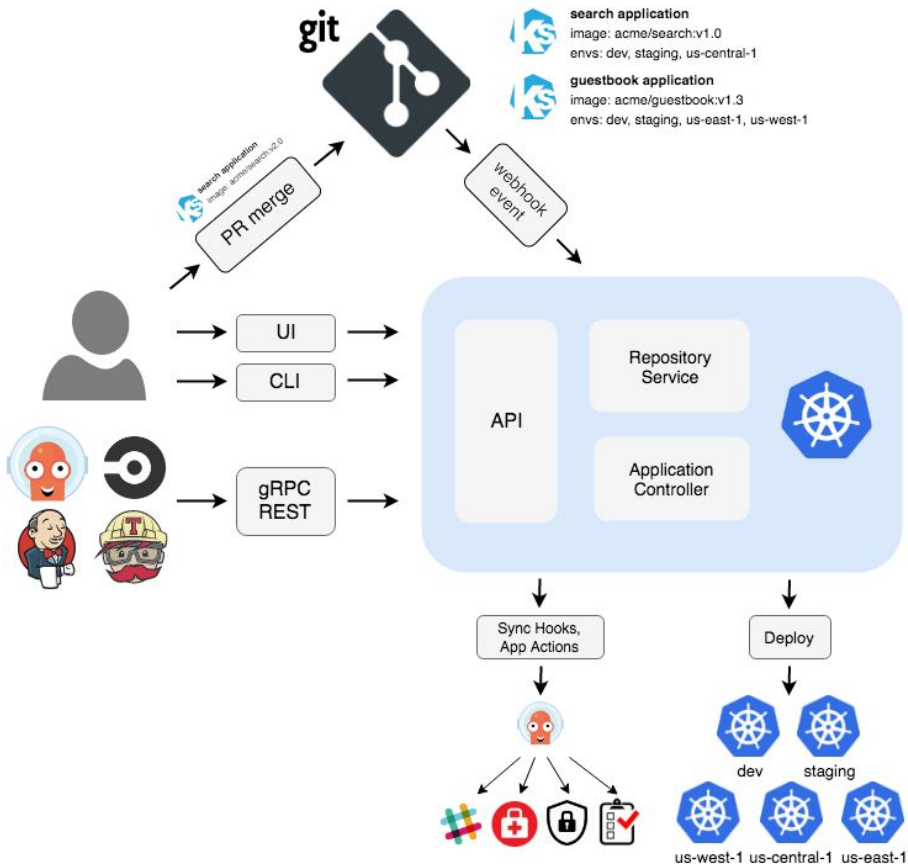
● #13935088 (UJU3fxiw) 
bighpc runner on bighpc vm via docker
[bighpc-runner-02](#)

● #13934510 (VL9E8yys) 
Runner installed on bighpc vm via rpm
[bighpc-runner-01](#)

Attach the Environment with the Infrastructure



GitOps aware of infrastructure state



- Argo CD implements the GitOps pull based strategy
- This not only follows the repositories activity, but also the deployment state in the infrastructure
- Webhook integration with Gitlab
- Support of multiple config management and templating tools (Kustomize, Helm, ...)

- Develop Gitlab CI/CD configuration for the required pipelines
- Create a Gitlab project and submit the code for the defined testbeds
- Connect Gitlab project with Gitlab runner instance, providing the reports in Gitlab
- Create docker images to pack the required code
- Do the required experimental validation checking the automated results over the testbeds (development and preview)
- Deploy the release to the production infrastructure
- Present the Pilot for further validation

