Adapting GitOps to manage Helmholtz Cloud services at DESY

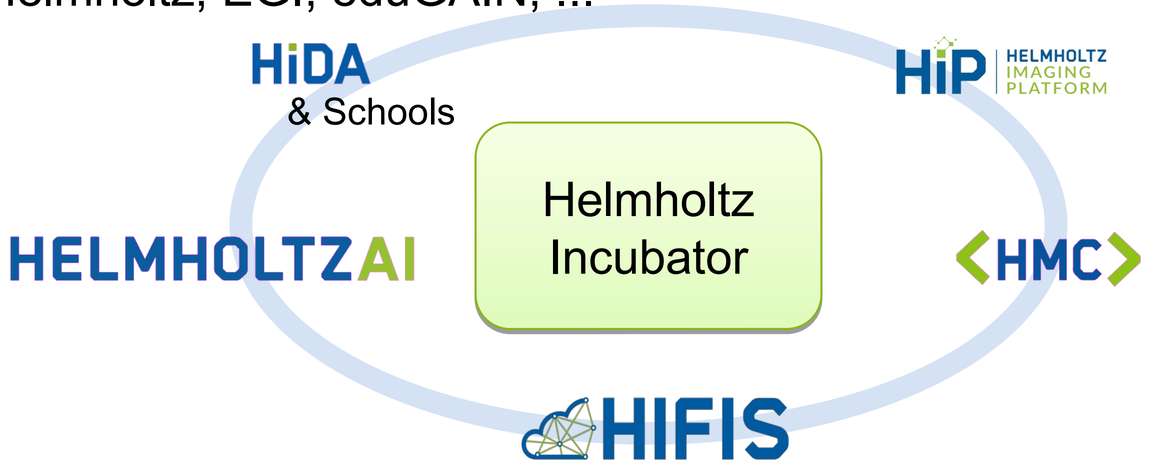
Thomas Beermann, Sebastian Wagner, Tim Wetzel, Johannes Reppin, Michael Schuh, Patrick Fuhrmann

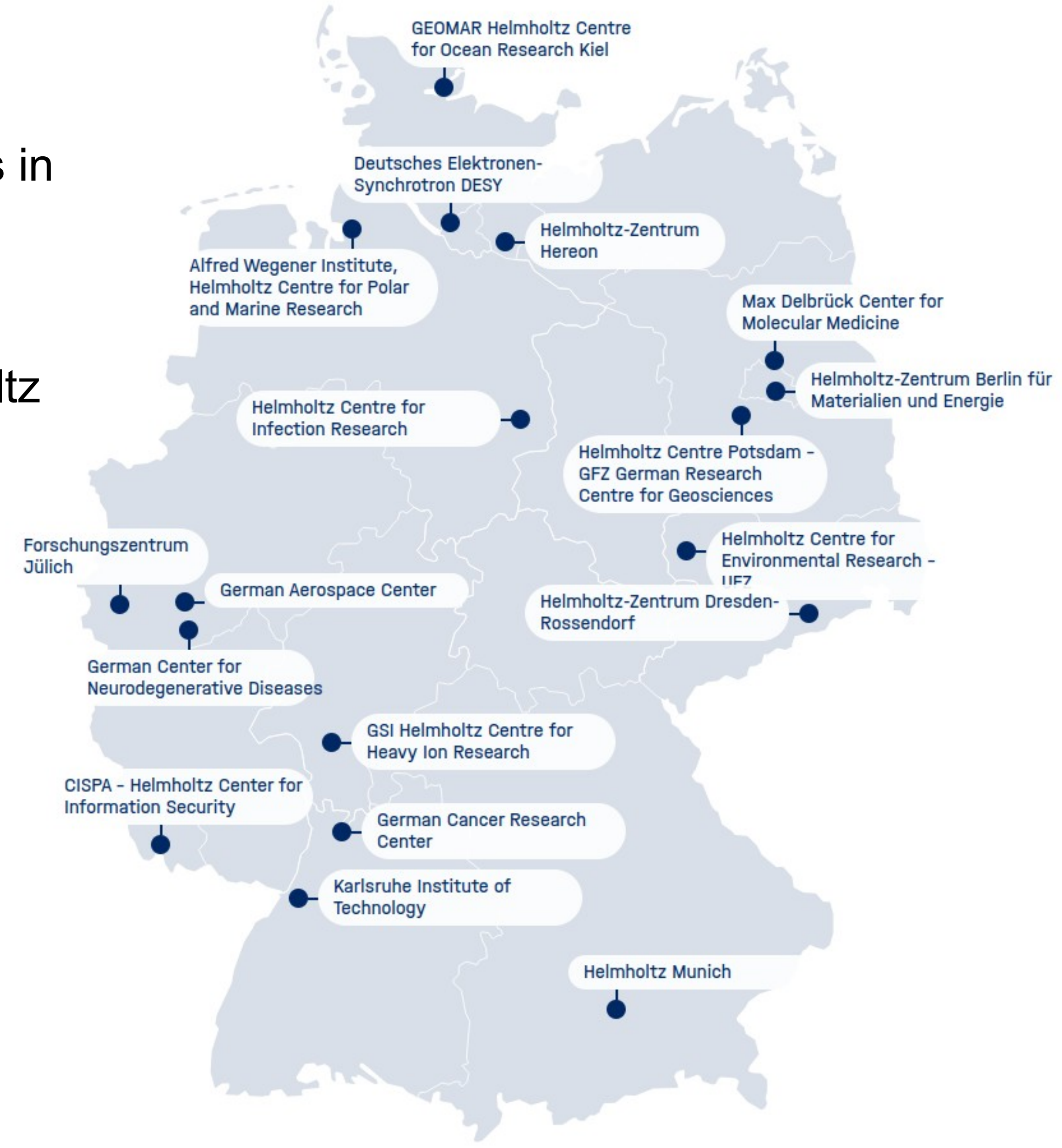
CHEP 2023, Norfolk, VA, USA, 2023-05-11

What is HIFIS?

Helmholtz federated IT services

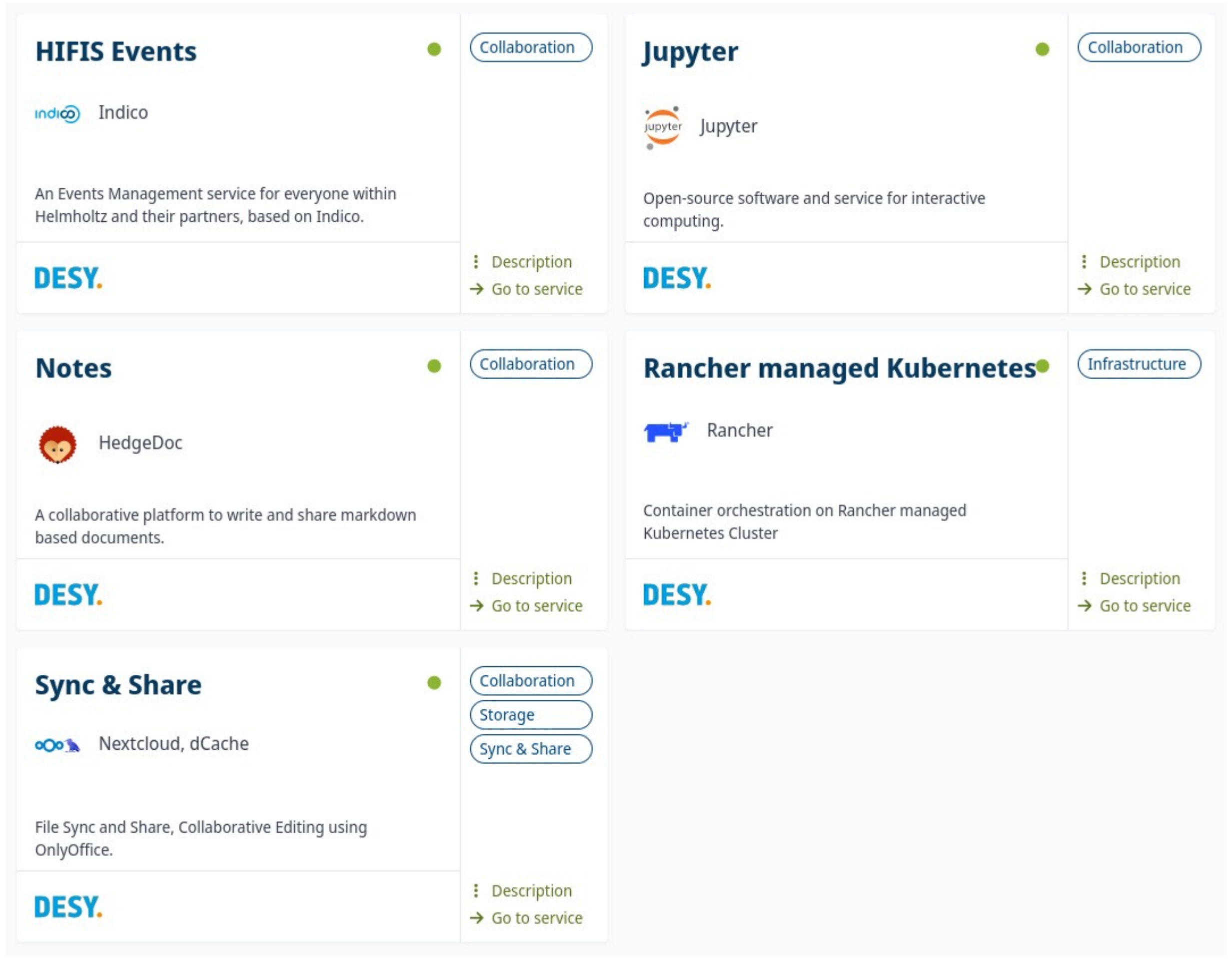
- Helmholtz Association with 18 autonomous research centres in Germany
- Incubator platforms for better collaboration between centres
  - Using synergies is key!
- HIFIS is the central IT service federation platform in Helmholtz
- Very good review last year from international experts
- Centres make web services and resources available for all other Helmholtz members
- Central AAI with community attributes and integration
  - Helmholtz, EGI, eduGAIN, ...




GEOMAR Helmholtz Centre for Ocean Research Kiel
Deutsches Elektronen-Synchrotron DESY
Helmholtz-Zentrum Hereon
Alfred Wegener Institute, Helmholtz Centre for Polar and Marine Research
Max Delbrück Center for Molecular Medicine
Helmholtz-Zentrum Berlin für Materialien und Energie
Helmholtz Centre for Infection Research
Helmholtz Centre Potsdam - GFZ German Research Centre for Geosciences
Forschungszentrum Jülich
Helmholtz Centre for Environmental Research - UFZ
German Aerospace Center
Helmholtz-Zentrum Dresden-Rossendorf
German Center for Neurodegenerative Diseases
GSI Helmholtz Centre for Heavy Ion Research
CISPA - Helmholtz Center for Information Security
German Cancer Research Center
Karlsruhe Institute of Technology
Helmholtz Munich

DESY Services for HIFIS

- DESY is one of the providers that hosts several services for the Helmholtz Cloud
- Among those services is a Rancher managed Kubernetes cluster
- Some services like Jupyter and Notes (HedgeDoc) and the Helmholtz Cloud Portal are deployed on Kubernetes
- The Portal is a catalogue of all HIFIS services and is developed at DESY



HIFIS Events — Collaboration
Indico
An Events Management service for everyone within Helmholtz and their partners, based on Indico.
DESY. — Description — Go to service

Jupyter — Collaboration
Jupyter
Open-source software and service for interactive computing.
DESY. — Description — Go to service

Notes — Collaboration
HedgeDoc
A collaborative platform to write and share markdown based documents.
DESY. — Description — Go to service

Rancher managed Kubernetes — Infrastructure
Rancher
Container orchestration on Rancher managed Kubernetes Cluster
DESY. — Description — Go to service

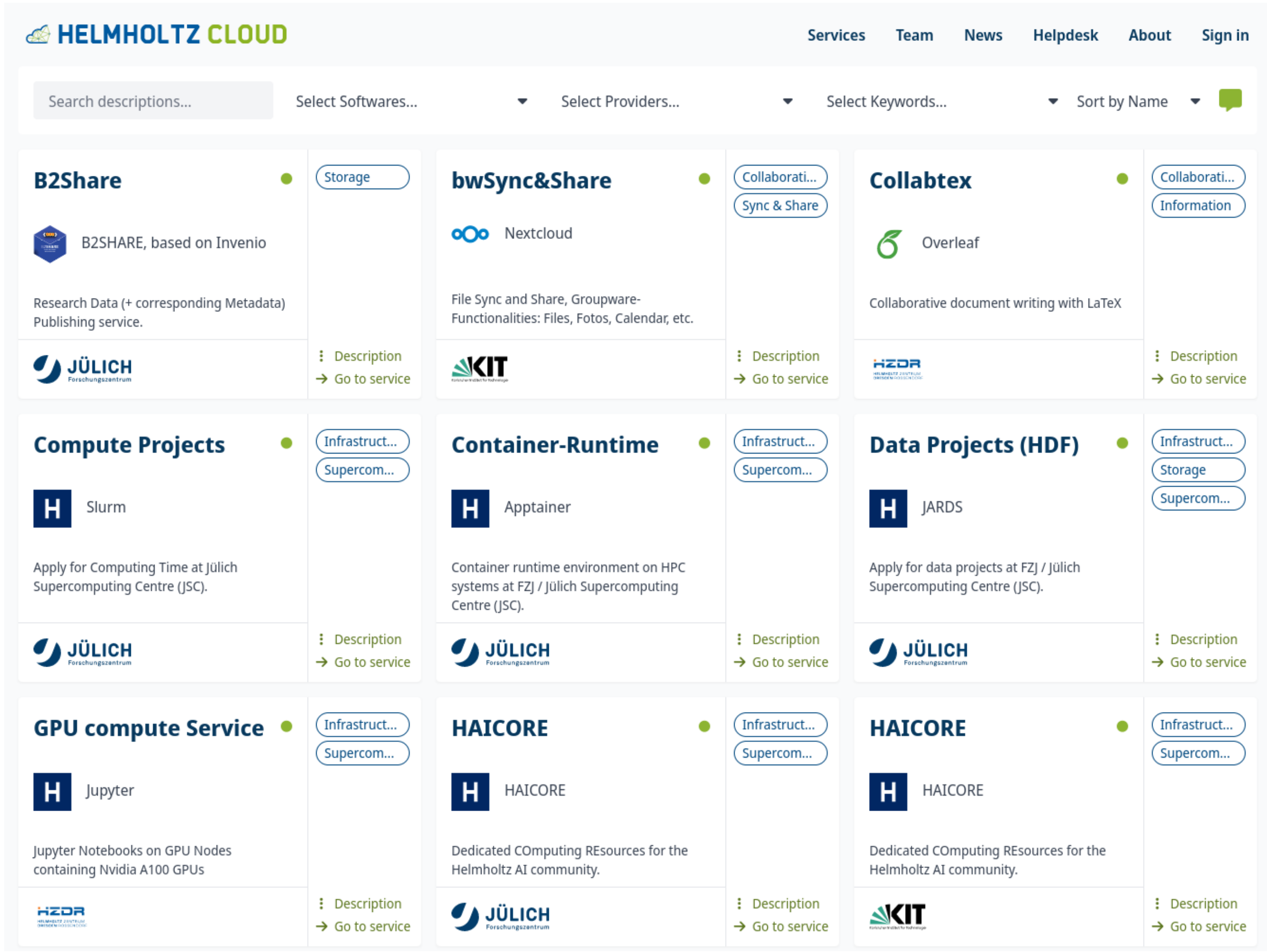
Sync & Share — Collaboration, Storage, Sync & Share
Nextcloud, dCache
File Sync and Share, Collaborative Editing using OnlyOffice.
DESY. — Description — Go to service

Infrastructure

- Our deployments are fully-containerized
- We use Openstack for our underlying cloud infrastructure
- On top we deploy Kubernetes clusters with Rancher
- The applications are installed on Kubernetes using Helm charts
- The Helm releases are managed by FluxCD and the configuration is stored centrally in GitLab
- For the Helmholtz Cloud Portal we use Gitlab CI/CD pipelines to build, test and deploy our code
- The Gitlab runners are also deployed with Flux on Kubernetes

Helmholtz Cloud Portal

- The Helmholtz Cloud Portal is the central entrypoint for users and provides a catalogue of all available services
- It is under active development and it will provide many more features to manage the Helmholtz Cloud in the future, e.g., a central resource booking interface
- It is a Python application split in two parts:
  - A core application based on standard Python tools like pydantic, alembic and asyncpg that handles the business logic
  - A web application based on Django and Vue.js for the frontend and session mangement / OIDC handling
- The database is PostgreSQL


HELMHOLTZ CLOUD

Services | Team | News | Helpdesk | About | Sign in

Search descriptions... | Select Softwares... | Select Providers... | Select Keywords... | Sort by Name

B2Share — Storage
B2SHARE, based on Invenio
Research Data (+ corresponding Metadata) Publishing service.
Description
Go to service

bwSync&Share — Collaborati... | Sync & Share
Nextcloud
File Sync and Share, Groupware-Functionalities: Files, Fotos, Calendar, etc.
Description
Go to service

Collabtex — Collaborati... | Information
Overleaf
Collaborative document writing with LaTeX
Description
Go to service

Compute Projects — Infrastruct... | Supercom...
Slurm
Apply for Computing Time at Jülich Supercomputing Centre (JSC).
Description
Go to service

Container-Runtime — Infrastruct... | Supercom...
Apptainer
Container runtime environment on HPC systems at FZJ / Jülich Supercomputing Centre (JSC).
Description
Go to service

Data Projects (HDF) — Infrastruct... | Storage | Supercom...
JARDS
Apply for data projects at FZJ / Jülich Supercomputing Centre (JSC).
Description
Go to service

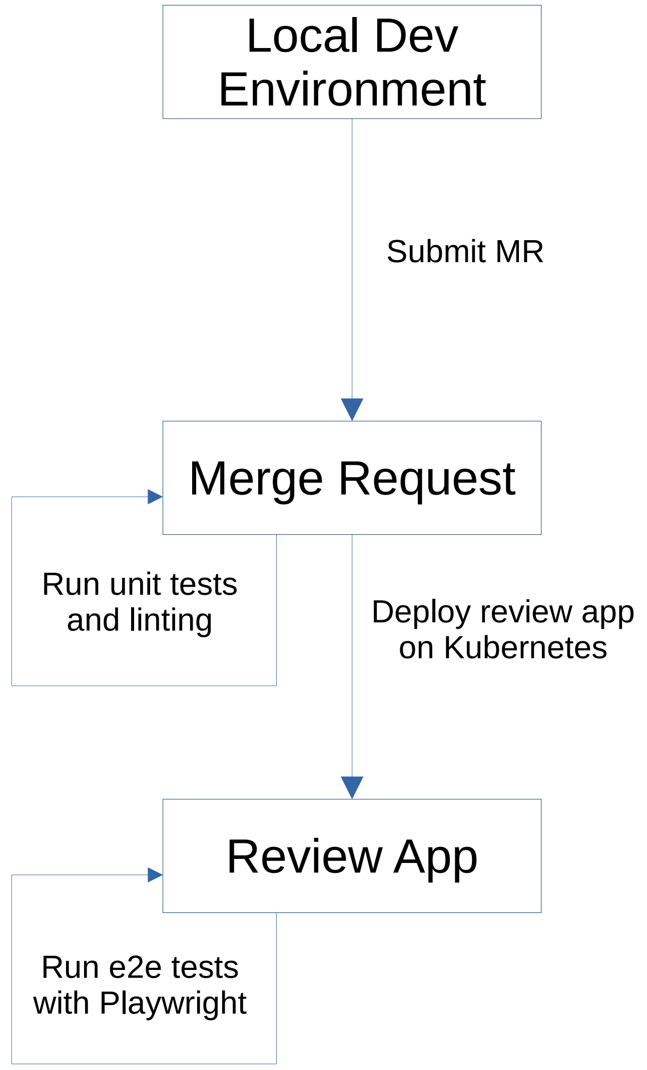
GPU compute Service — Infrastruct... | Supercom...
Jupyter
Jupyter Notebooks on GPU Nodes containing Nvidia A100 GPUs
Description
Go to service

HAICORE — Infrastruct... | Supercom...
HAICORE
Dedicated COmputing REsources for the Helmholtz AI community.
Description
Go to service

HAICORE — Infrastruct... | Supercom...
HAICORE
Dedicated COmputing REsources for the Helmholtz AI community.
Description
Go to service

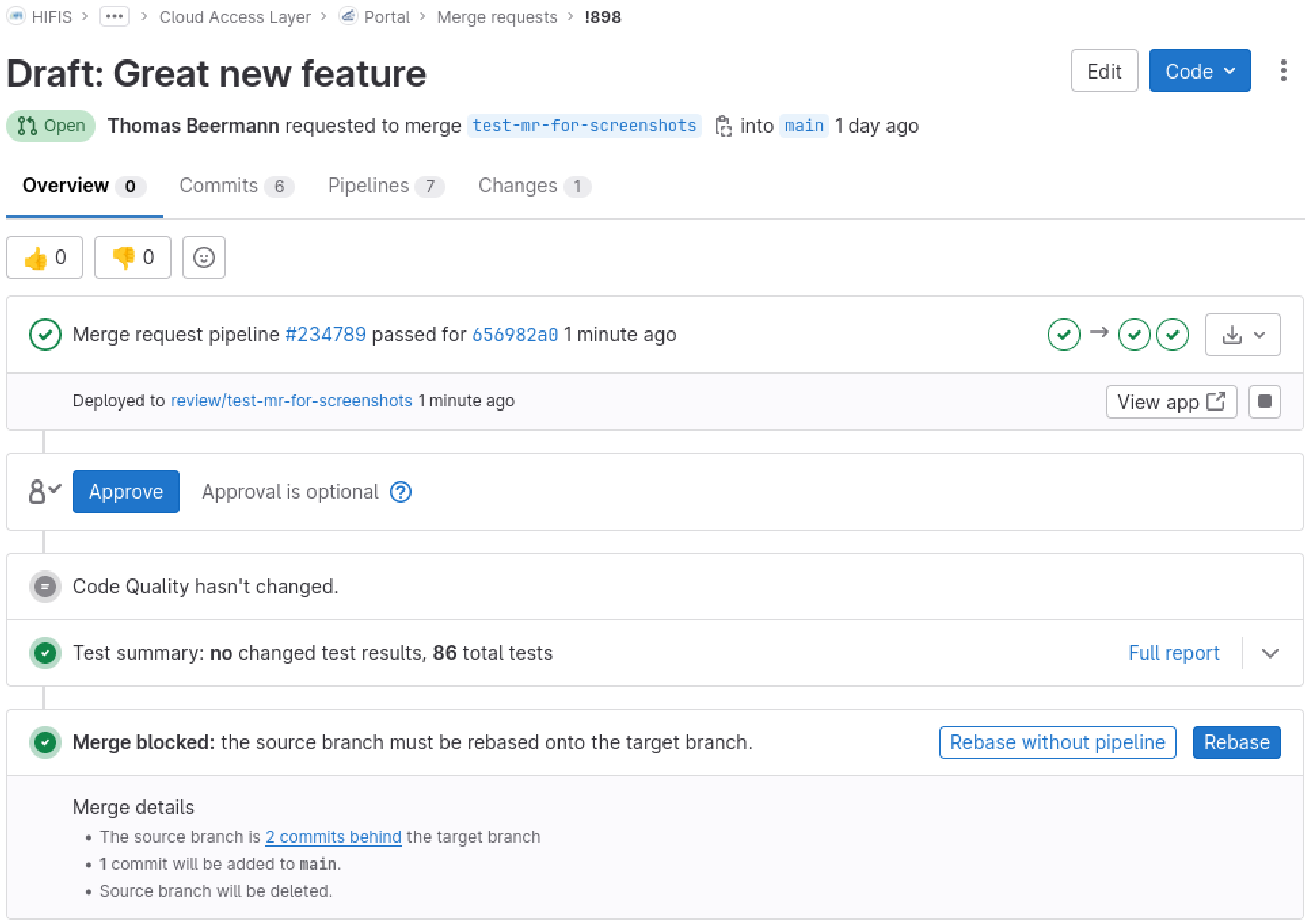
Development of Cloud Portal

- Local development is done using dev containers with Podman in VSCode
- Changes are submitted as merge requests in Gitlab that trigger CI/CD pipelines
- The pipelines first run several code checks (ruff, black, bandit, mypy, vue-tsc) and unit tests
- In parallel, deployment containers for the specific MR are built
- If the tests are successful a deployment of a review environment is triggered
- For the review environment a new namespace is created on the Kubernetes cluster
- Inside the namespace a full Cloud Portal with a separate DB is installed using Helm
- After the deployment is finished Playwright is used to do end-to-end test on the review app


Local Dev Environment
Submit MR
Merge Request
Run unit tests and linting
Deploy review app on Kubernetes
Review App
Run e2e tests with Playwright

Integration with Gitlab

- The whole development cycle is fully integrated with Gitlab
- Pipeline status overview can be checked from the merge request page



Draft: Great new feature

Thomas Beermann requested to merge `test-mr-for-screenshots` into `main` 1 day ago

Merge request pipeline #234789 passed for `656982a0` 1 minute ago

Deployed to review/test-mr-for-screenshots 1 minute ago

Approval is optional

Code Quality hasn't changed.

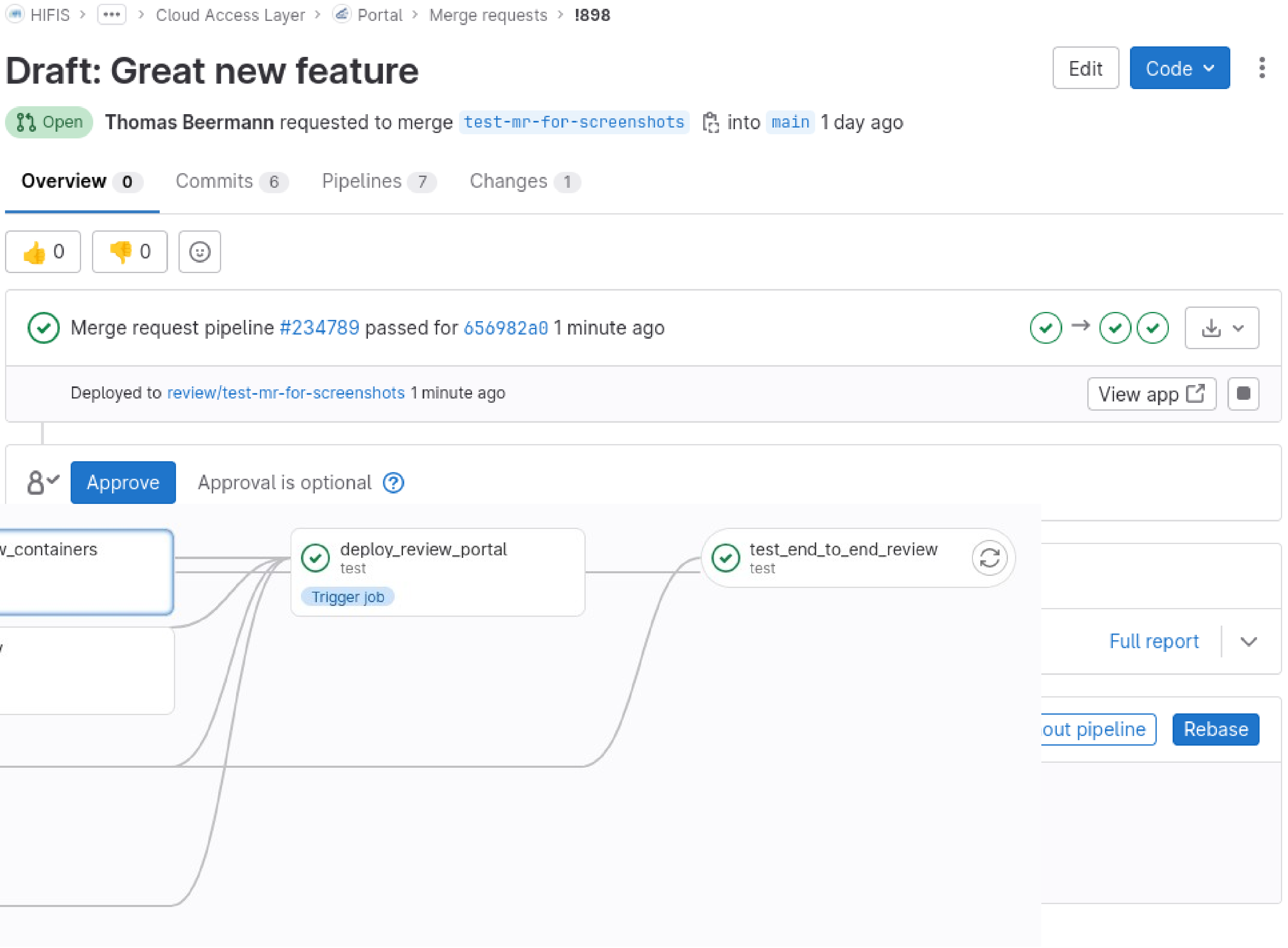
Test summary: **no** changed test results, **86** total tests

Merge blocked: the source branch must be rebased onto the target branch.

Merge details
- The source branch is 2 commits behind the target branch
- 1 commit will be added to `main`.
- Source branch will be deleted.

Integration with Gitlab

- The whole development cycle is fully integrated with Gitlab
- Pipeline status overview can be checked from the merge request page



HIFIS > ... > Cloud Access Layer > Portal > Merge requests > !898

Draft: Great new feature

Open Thomas Beermann requested to merge `test-mr-for-screenshots` into `main` 1 day ago

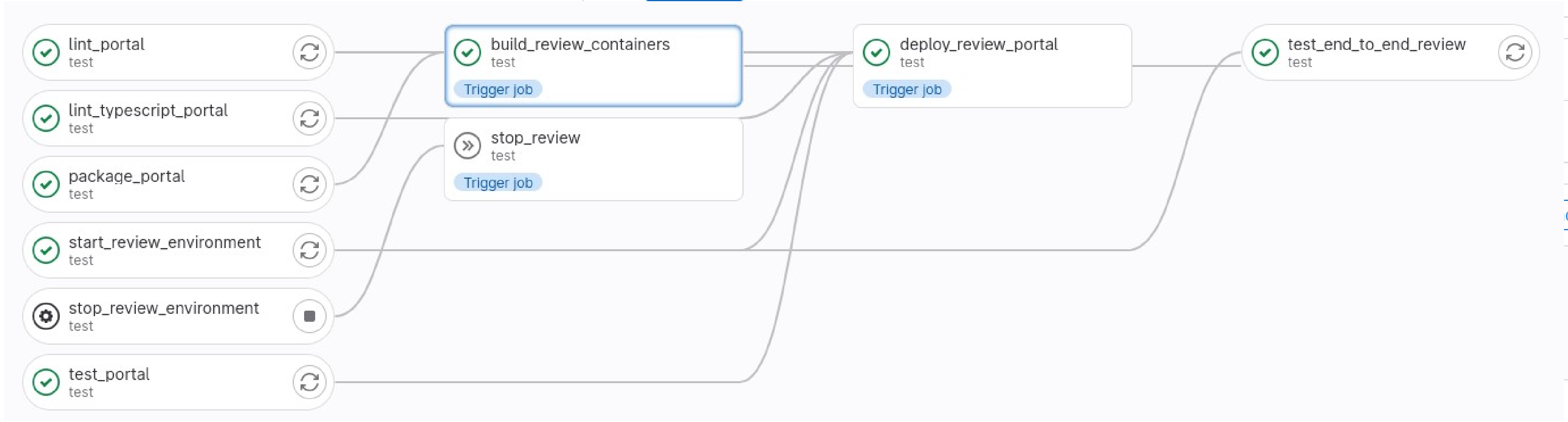
Overview 0 | Commits 6 | Pipelines 7 | Changes 1

👍 0 👎 0

Merge request pipeline #234789 passed for `656982a0` 1 minute ago

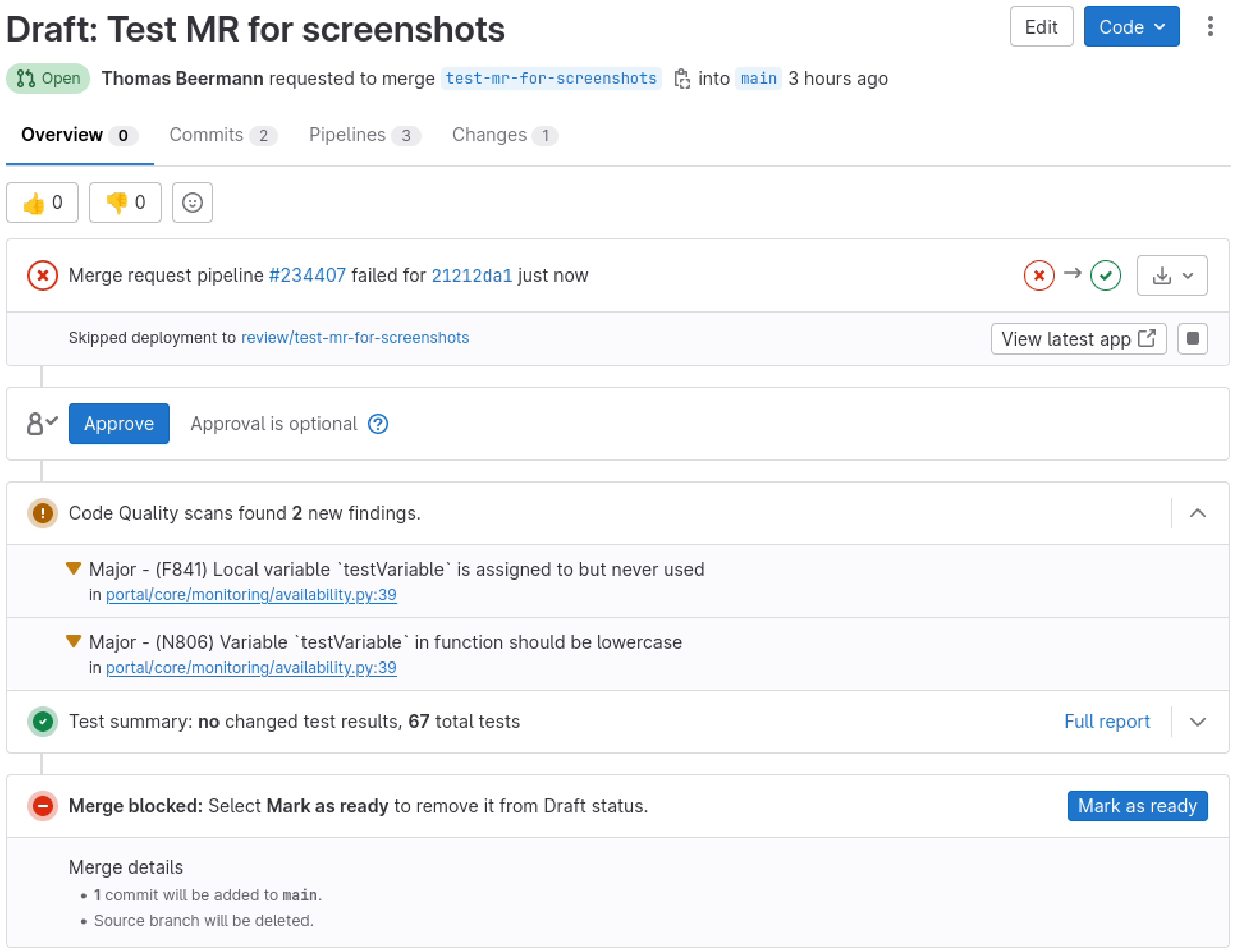
Deployed to review/test-mr-for-screenshots 1 minute ago

Approve Approval is optional



Integration with Gitlab

- The whole development cycle is fully integrated with Gitlab
- Test failures are also reported directly in the merge request



Draft: Test MR for screenshots

Open Thomas Beermann requested to merge `test-mr-for-screenshots` into `main` 3 hours ago

Merge request pipeline #234407 failed for 21212da1 just now

Skipped deployment to review/test-mr-for-screenshots

Approval is optional

Code Quality scans found **2** new findings.

Major - (F841) Local variable `testVariable` is assigned to but never used
in portal/core/monitoring/availability.py:39

Major - (N806) Variable `testVariable` in function should be lowercase
in portal/core/monitoring/availability.py:39

Test summary: **no** changed test results, **67** total tests

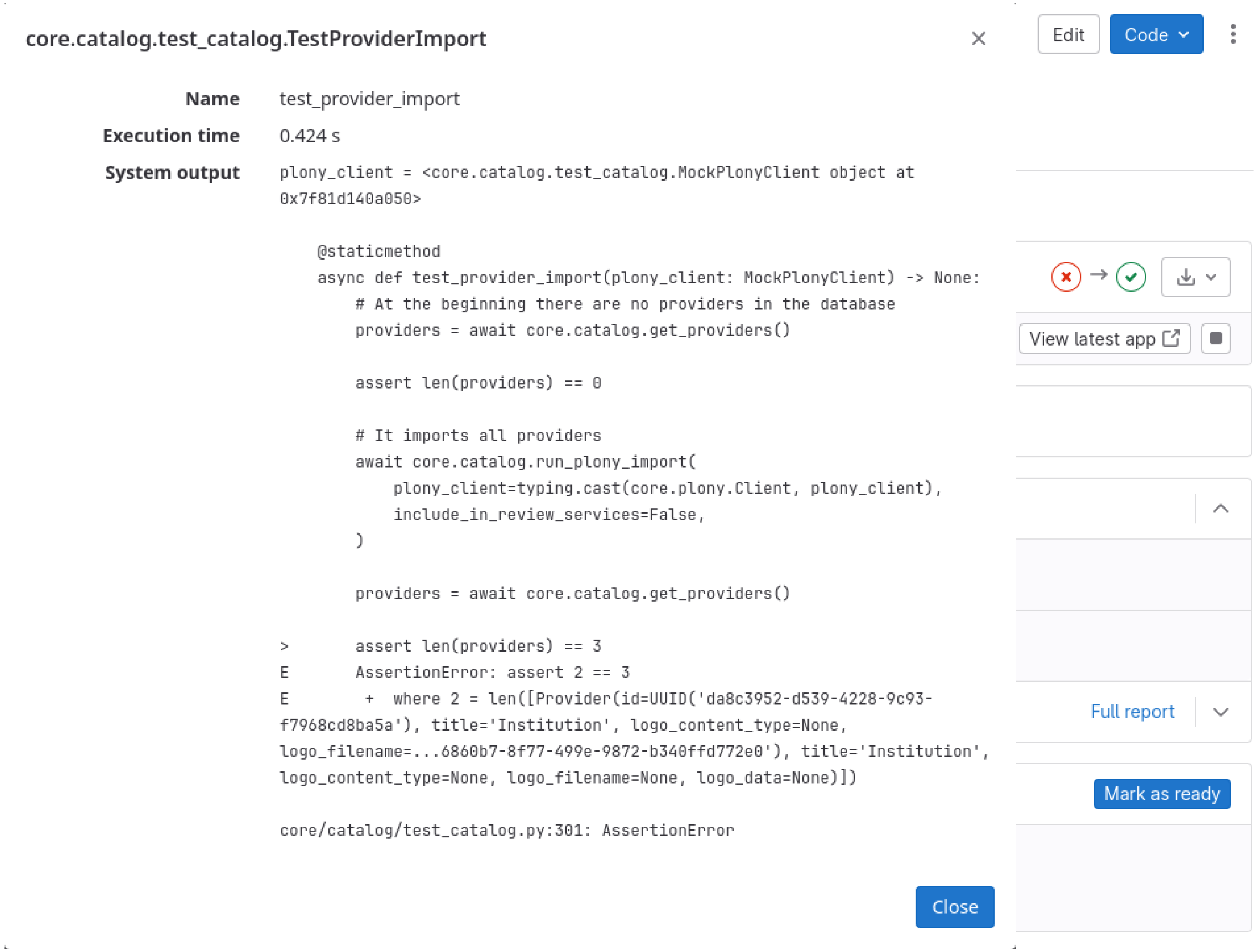
Merge blocked: Select **Mark as ready** to remove it from Draft status.

Merge details

- 1 commit will be added to `main`.
- Source branch will be deleted.

Integration with Gitlab

- The whole development cycle is fully integrated with Gitlab
- Test failure are also reported directly in the merge request also with specific error reports.



core.catalog.test_catalog.TestProviderImport

Name test_provider_import

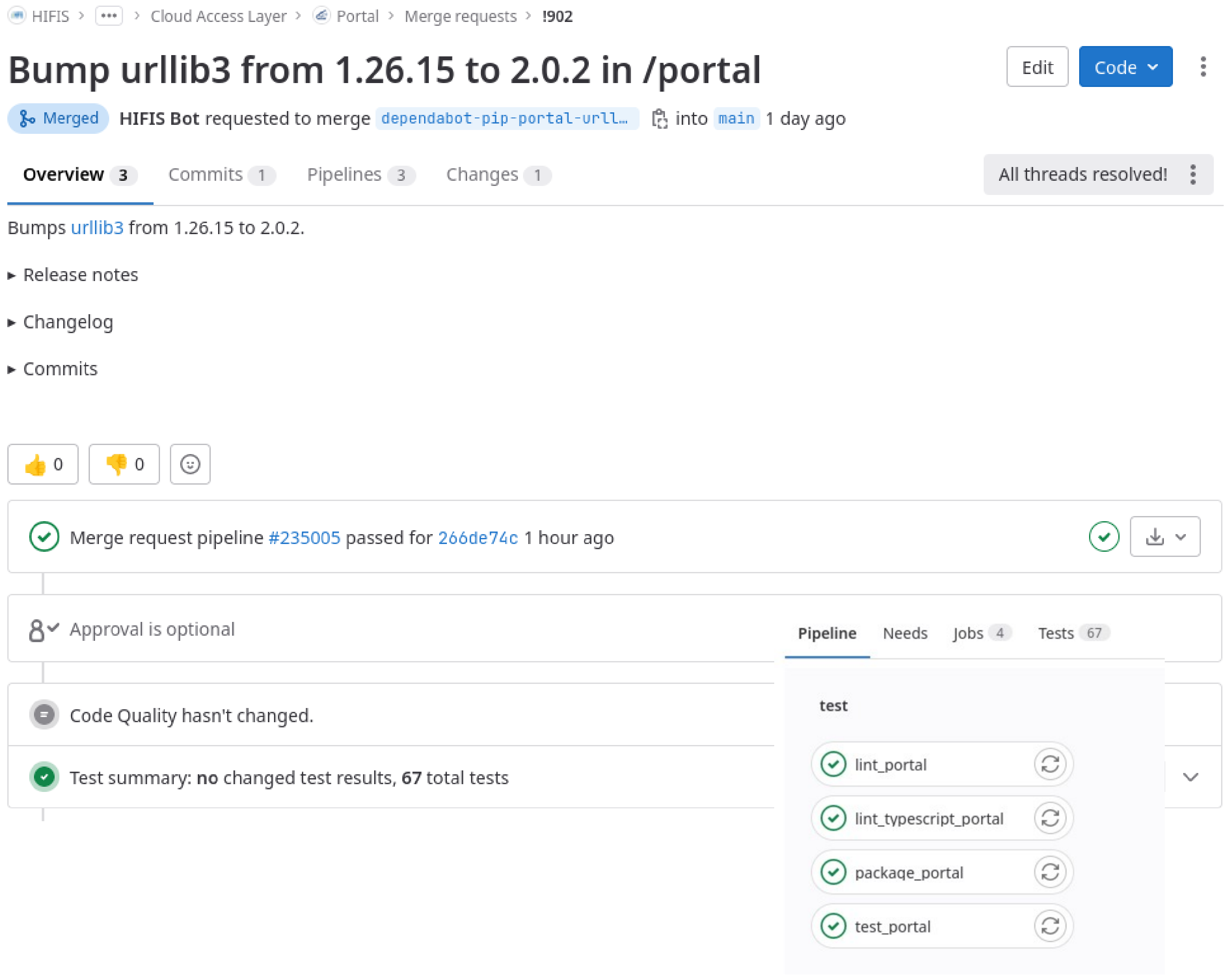
Execution time 0.424 s

System output

```
plony_client = <core.catalog.test_catalog.MockPlonyClient object at
0x7f81d140a050>

    @staticmethod
    async def test_provider_import(plony_client: MockPlonyClient) -> None:
        # At the beginning there are no providers in the database
        providers = await core.catalog.get_providers()

        assert len(providers) == 0

        # It imports all providers
        await core.catalog.run_plony_import(
            plony_client=typing.cast(core.plony.Client, plony_client),
            include_in_review_services=False,
        )

        providers = await core.catalog.get_providers()

>       assert len(providers) == 3
E       AssertionError: assert 2 == 3
E        +  where 2 = len([Provider(id=UUID('da8c3952-d539-4228-9c93-
f7968cd8ba5a'), title='Institution', logo_content_type=None,
logo_filename=...6860b7-8f77-499e-9872-b340ffd772e0'), title='Institution',
logo_content_type=None, logo_filename=None, logo_data=None)])

core/catalog/test_catalog.py:301: AssertionError
```

Dependabot

- We also use dependabot to automatically keep our dependencies up to date
- It checks for updates to any Python, Node or Docker image dependency and creates merge requests for each update
- Only runs a limited pipeline with linting and test and does not deploy a review app for each dependency update
- The merge requests are checked daily and then merged to the main branch
- The main branch is deployed to an integration environment that is used for further testing before moving to production


HIFIS > ··· > Cloud Access Layer > Portal > Merge requests > !902

Bump urllib3 from 1.26.15 to 2.0.2 in /portal

Merged HIFIS Bot requested to merge dependabot-pip-portal-urll… into main 1 day ago

Overview 3 Commits 1 Pipelines 3 Changes 1 All threads resolved!

Bumps urllib3 from 1.26.15 to 2.0.2.

▸ Release notes

▸ Changelog

▸ Commits

👍 0 👎 0

Merge request pipeline #235005 passed for 266de74c 1 hour ago

Approval is optional

Code Quality hasn't changed.

Test summary: no changed test results, 67 total tests

Pipeline Needs Jobs 4 Tests 67

test

- lint_portal
- lint_typescript_portal
- package_portal
- test_portal

Production Deployments

- The production deployments for the Helmholtz Cloud Portal, the Jupyter service and HedgeDoc are fully managed with Flux
- The Helm configuration is stored in separate Gitlab repositories for each service
- Configuration changes are made using merge requests with approvals
- For the Cloud Portal the release management is fully automated:
  - Whenever we decide to do a release we create a tag in the Cloud Portal source repository
  - A Gitlab pipeline then automatically creates a release and builds the corresponding container images
  - A Flux image automation checks regularly for new images and automatically updates the Helm release in the repository
  - Flux will then apply this new Helm release and deploys the new images

Summary

- Using GitOps helped us a lot to manage our services
- Our thorough testing pipelines make sure that we don’t easily introduce bugs or vulnerabilities in our code
- Dependabot helps us keeping our dependencies up-to-date to quickly mitigate any security issues
- The deployment of review apps help us to further evaluate any changes to our application and Playwright makes sure that our workflow keep working
- Flux is very useful to centrally manage our configurations and to add traceability for any changes

Thank you