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, ...
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
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
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 management / OIDC handling.

- The database is PostgreSQL.
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
Integration with Gitlab

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

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

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