RCS-SIS GitOps

Setup and use cases

By Benjamin Bergia

RCS-SIS

The CERN Scientific Information Service aims at efficiently managing, preserving and disseminating scientific information to make it openly accessible and reusable to CERN and the worldwide High-Energy Physics community.

All about collaborations:

➔ Inside the CERN community (CERN Analysis Preservation and CERN Academic Training).

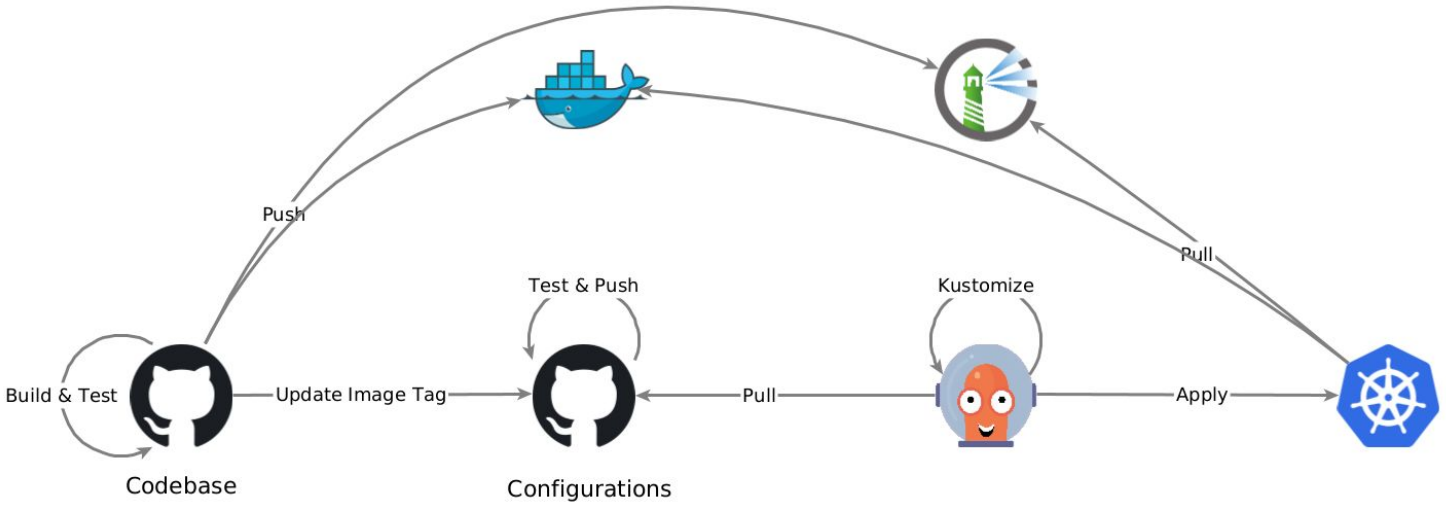
➔ With other institutions in the field (InspireHEP and HEPData).

➔ With the scholarly community as a whole (SCOAP3, SciPost and arXiv).

Requirements

- Codebases should be publicly available on Github.
- Docker images should also be publicly available.
- Some external, non-CERN, developers.
- Different release cycles.
- Production traffic 24x7.
- QA and Prod environment for each project.

Overview


Push
Test & Push
Kustomize
Pull
Build & Test
Update Image Tag
Pull
Apply
Codebase
Configurations

Codebases

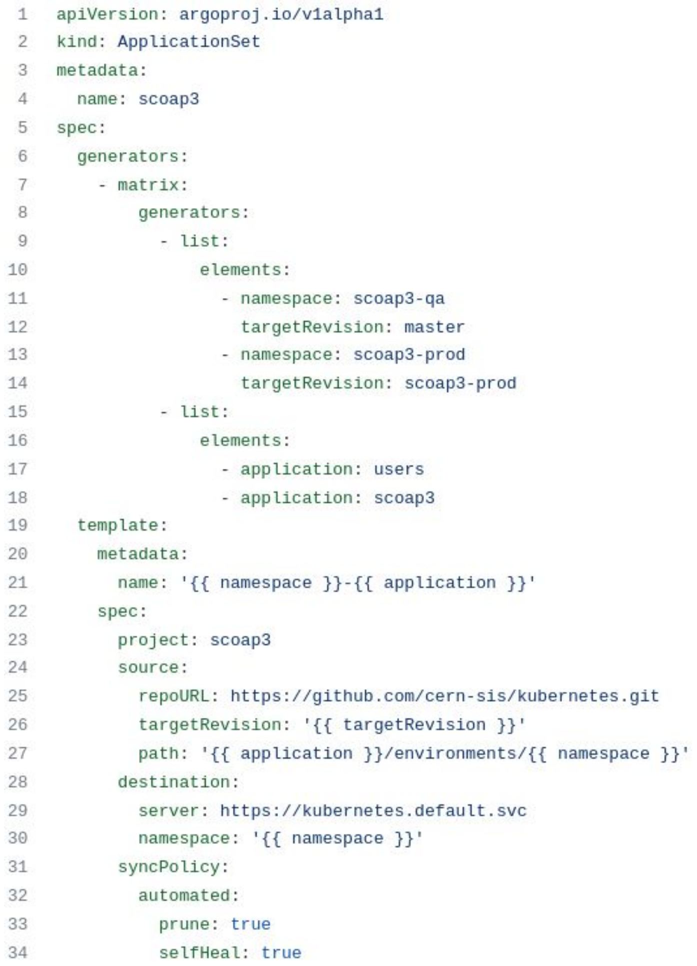
- Github public repository
- Currently python only
- Github actions:
  - Build docker images
  - Run test
  - Push to DockerHub or CERN Registry
  - Trigger events on the configuration repository

Configurations

- Github private repository
- Flat YAML files + Kustomizations
- Github actions:
  - Test
  - Push to production branches
  - Update image tags
  - Call ArgoCD Webhook

ArgoCD

- In-Cluster
- Different projects
- Pull from the Configurations repository
- Run Kustomize
- Apply result
- Auto Sync & Self Heal
- 61 Apps
- ApplicationSet



```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: scoap3
spec:
  generators:
    - matrix:
        generators:
          - list:
              elements:
                - namespace: scoap3-qa
                  targetRevision: master
                - namespace: scoap3-prod
                  targetRevision: scoap3-prod
          - list:
              elements:
                - application: users
                - application: scoap3
  template:
    metadata:
      name: '{{ namespace }}-{{ application }}'
    spec:
      project: scoap3
      source:
        repoURL: https://github.com/cern-sis/kubernetes.git
        targetRevision: '{{ targetRevision }}'
        path: '{{ application }}/environments/{{ namespace }}'
      destination:
        server: https://kubernetes.default.svc
        namespace: '{{ namespace }}'
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
```

Kustomize

- Flat YAML files
- Base resources
- Copy
- Transform
- You can layer transformations
- Doesn’t enforce any file structure

Basically prototype-based (think JS)

Pros:

- YAML all the way
- High Level (get a lot done quickly)
- Generators
- Remote resources

Cons:

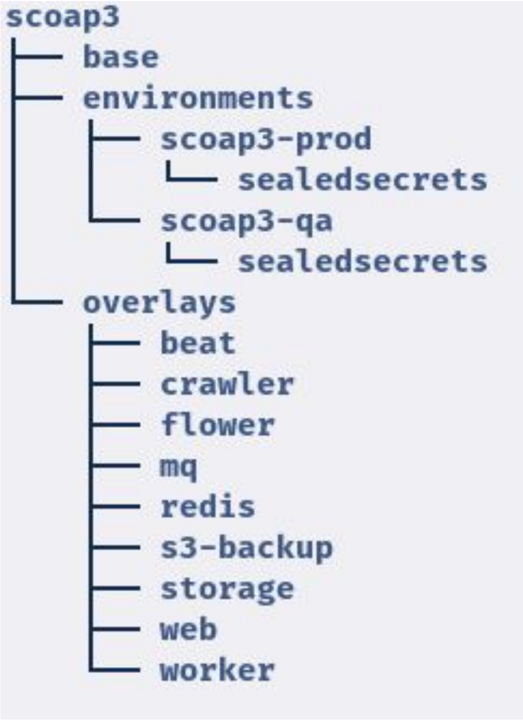
- New features
- Team responsiveness
- Arbitrary limitations (opinionated)

Configurations structure

Base resources

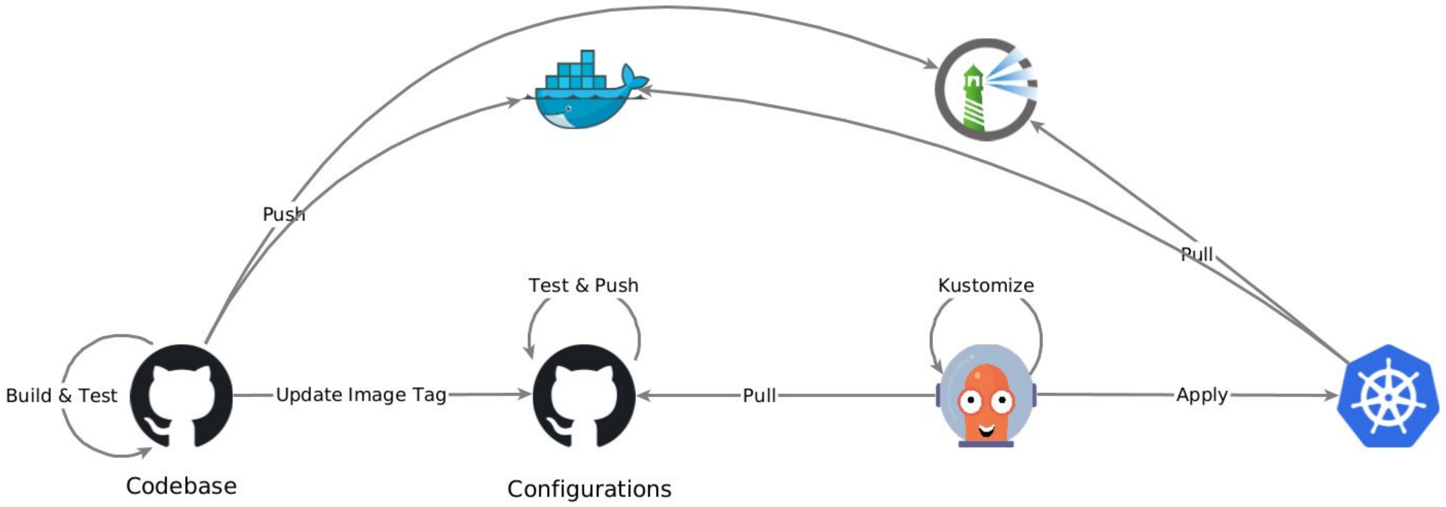
Overlays that build on top of the base

One environment for each namespace that include all the overlays needed.

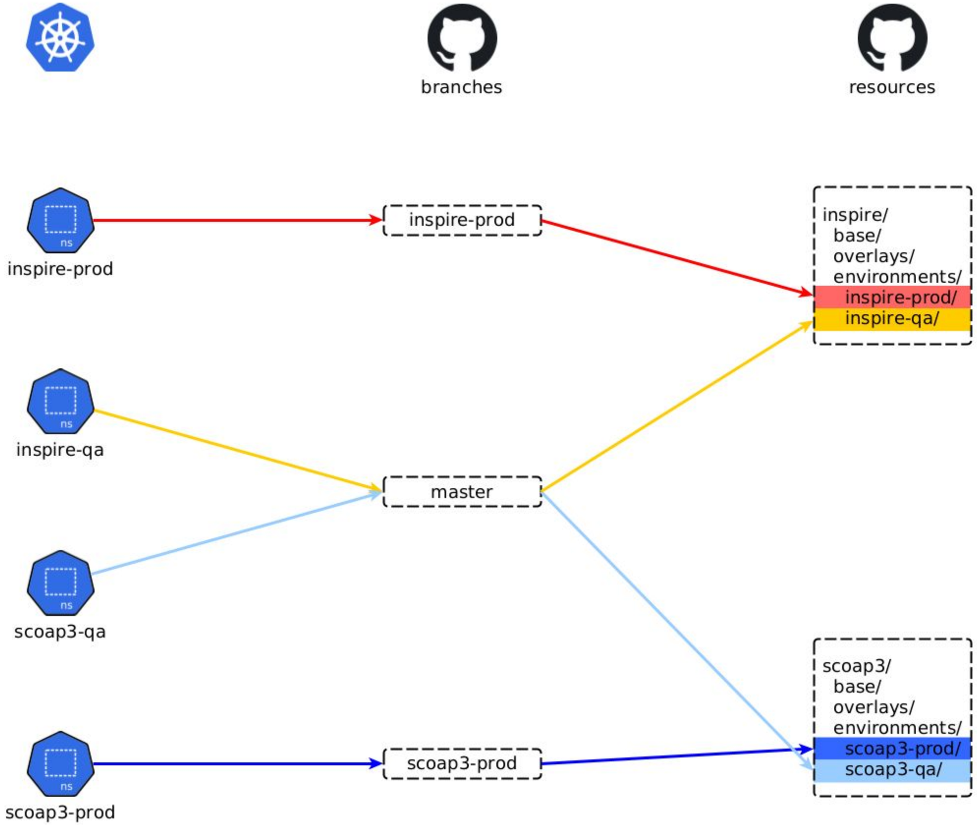


```
scoap3
├── base
├── environments
│   ├── scoap3-prod
│   │   └── sealedsecrets
│   └── scoap3-qa
│       └── sealedsecrets
└── overlays
    ├── beat
    ├── crawler
    ├── flower
    ├── mq
    ├── redis
    ├── s3-backup
    ├── storage
    ├── web
    └── worker
```

Overview


Push
Pull
Test & Push
Kustomize
Build & Test
Update Image Tag
Pull
Apply
Codebase
Configurations

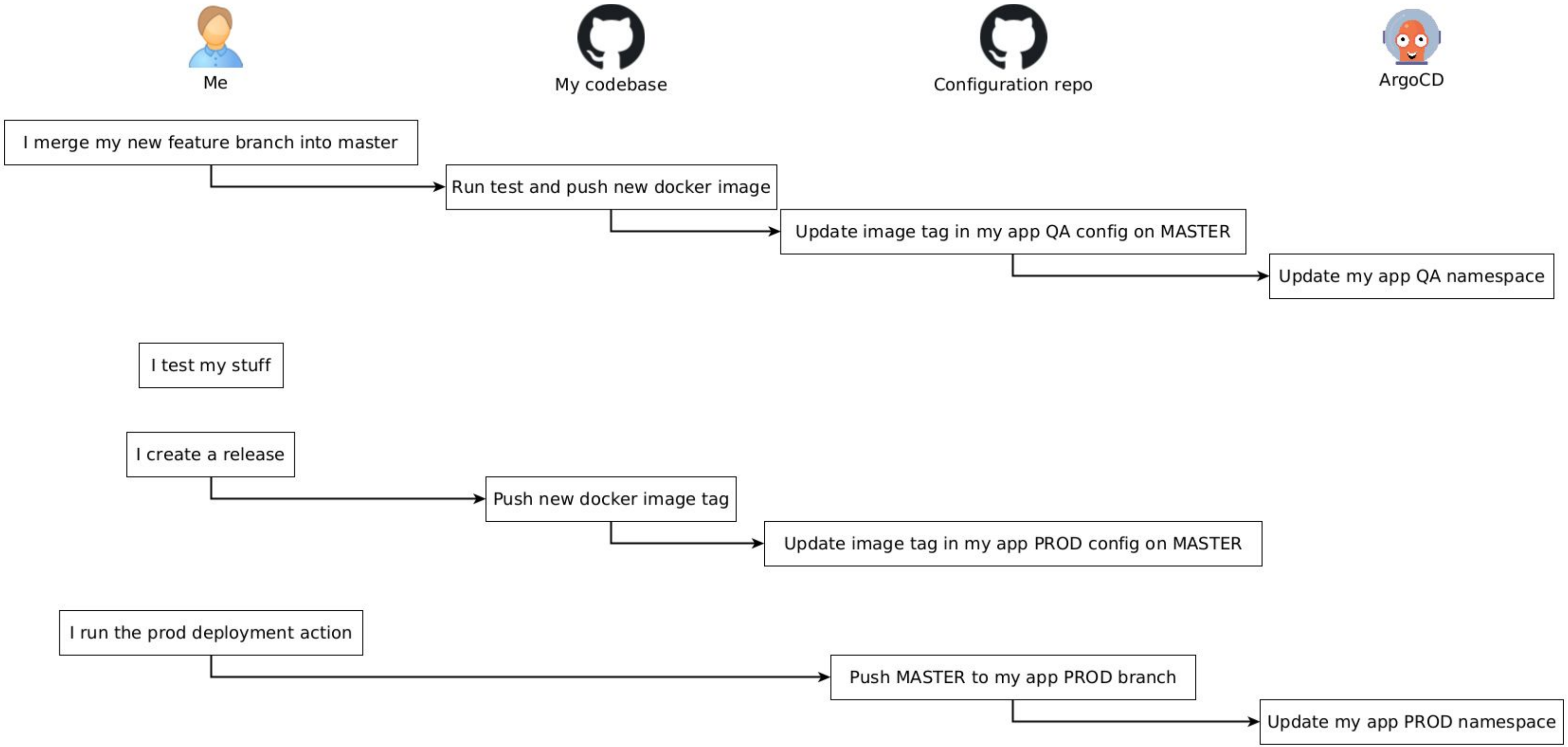
Branching & environments



WHY?!

- Everything on master goes to all QAs
- Prod committed on master
- Each project advance its prod branch when needed

Deployment process



Me

My codebase

Configuration repo

ArgoCD

I merge my new feature branch into master

Run test and push new docker image

Update image tag in my app QA config on MASTER

Update my app QA namespace

I test my stuff

I create a release

Push new docker image tag

Update image tag in my app PROD config on MASTER

I run the prod deployment action

Push MASTER to my app PROD branch

Update my app PROD namespace

Why doing all of this

- Straightforward: what is on the repo is on the cluster.
- Simple rollbacks.
- Single source of truth.
- Releases don’t impact other projects.
- The image used to run the codebase test is the one going on prod.

Where we struggle

- Local Dev environments
- Manual interventions
- Git workflow on the Configuration repo
- Checking changes before deployment
- Testing before pushing to master

Future improvements

- Store generated YAML on the Prod branches (Github Page like)
- Add more policies to conftest
- Branchless git workflow?

Questions