// CODE ➡ CLUSTER: BOOSTING DEVELOPMENT WITH A LOCAL KUBERNETES OPS PLATFORM

Johannes Schnatterer, Cloudogu GmbH

Slides

@schnatterer@floss.social in/jschnatterer

Version: 202406050933-a7d678d

🏋

🙋 What is your profession?

🤓 Software Engineer / Developer

🙋 What is your profession?

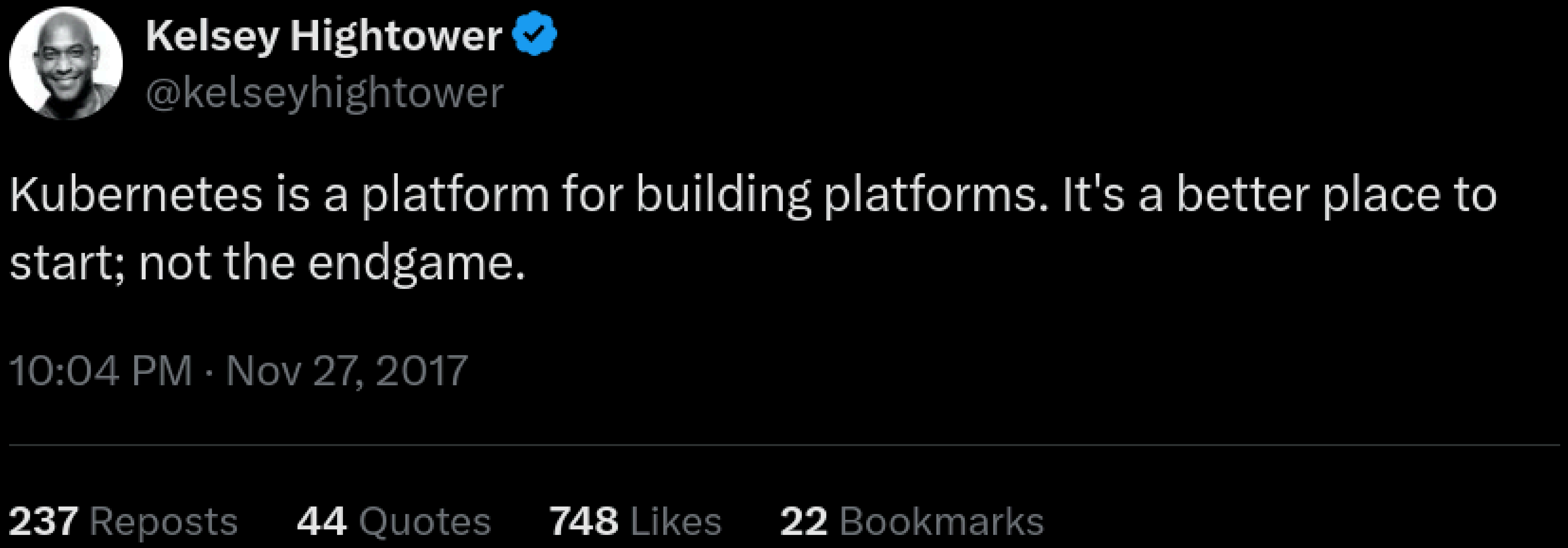
🛠️ Platform Engineer / Ops person

🙋 Who uses Kubernetes for local development?

k3d Minkube Microk8s k3s KIND

Docker Desktop k0s

Rancher Desktop



Kelsey Hightower
@kelseyhightower

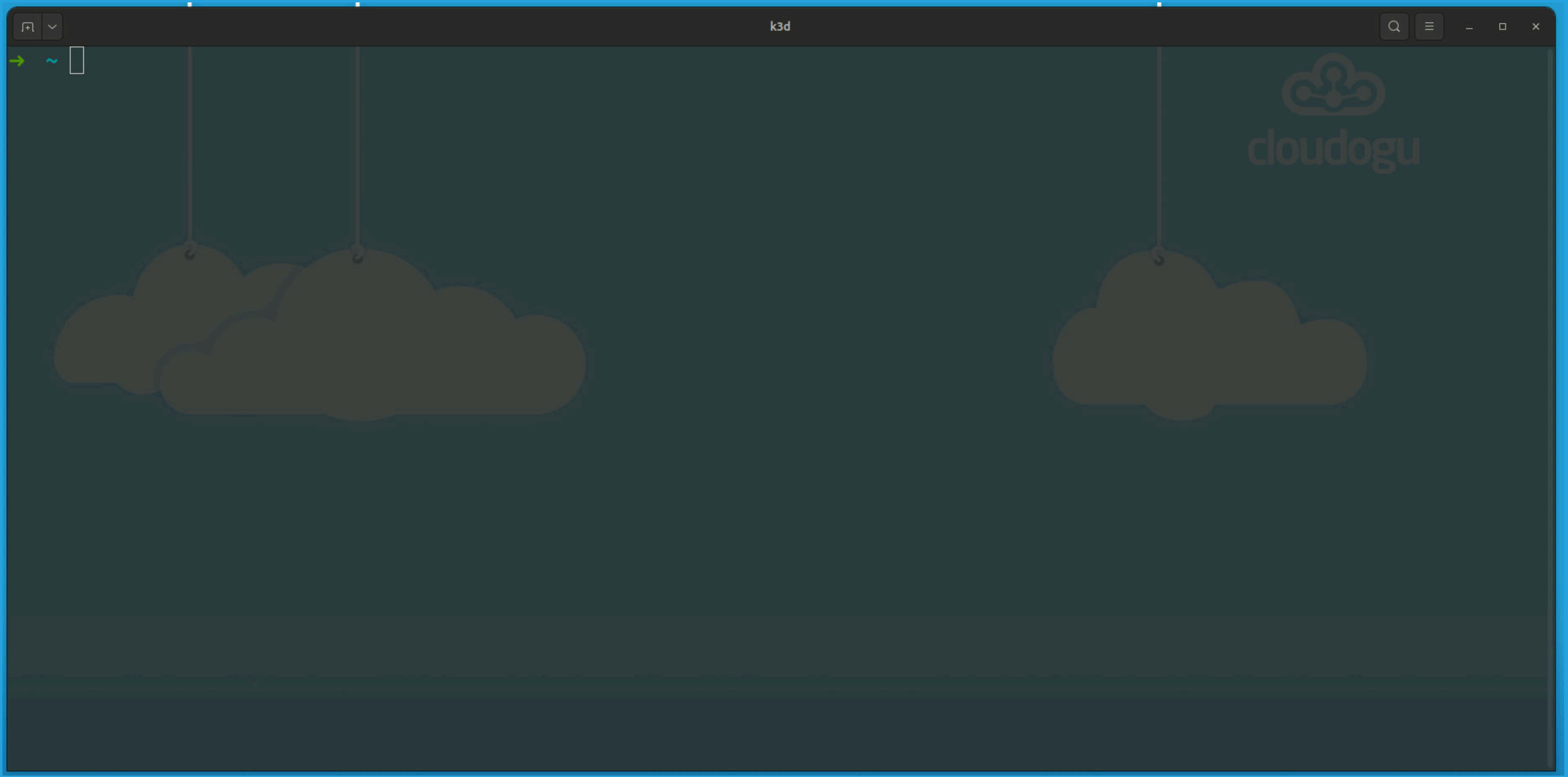
Kubernetes is a platform for building platforms. It's a better place to start; not the endgame.

10:04 PM · Nov 27, 2017

237 Reposts 44 Quotes 748 Likes 22 Bookmarks

twitter.com/kelseyhightower/status/935252923721793536

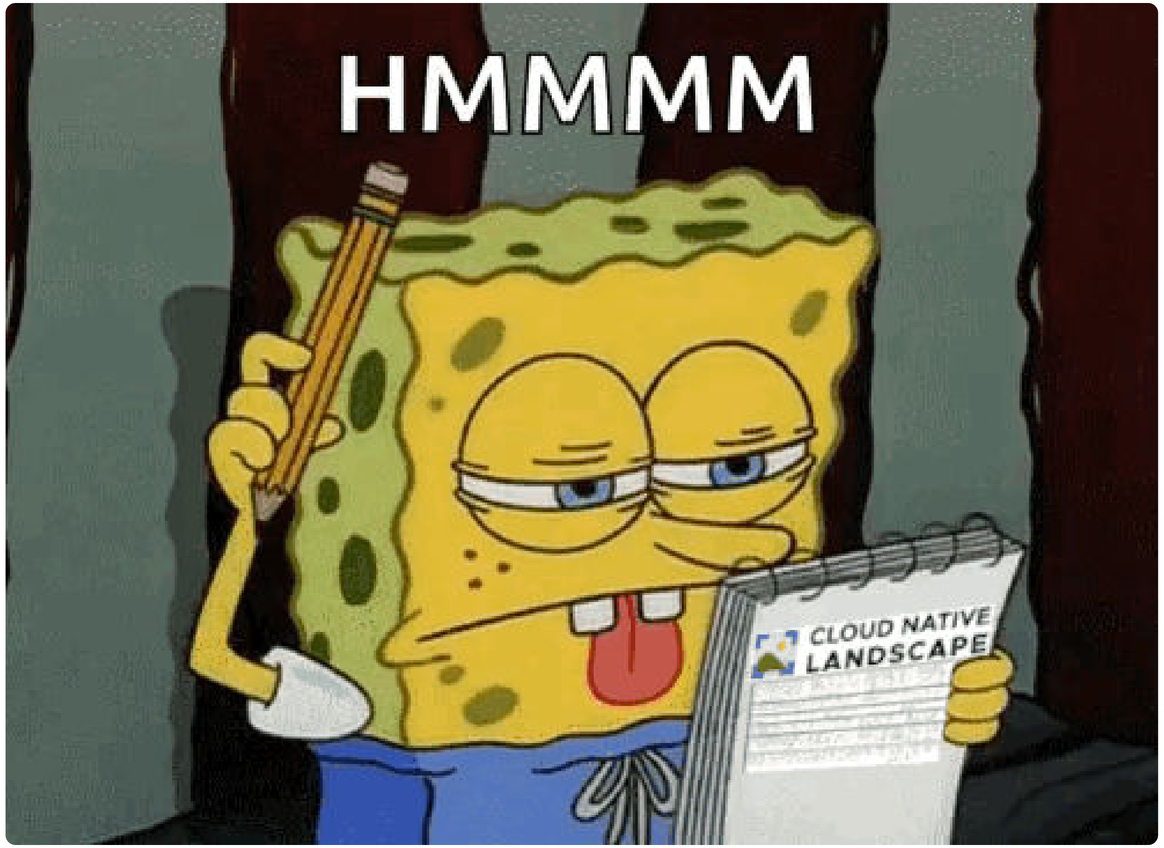
Start a local k8s cluster with one command



```
➜  ~ 
```

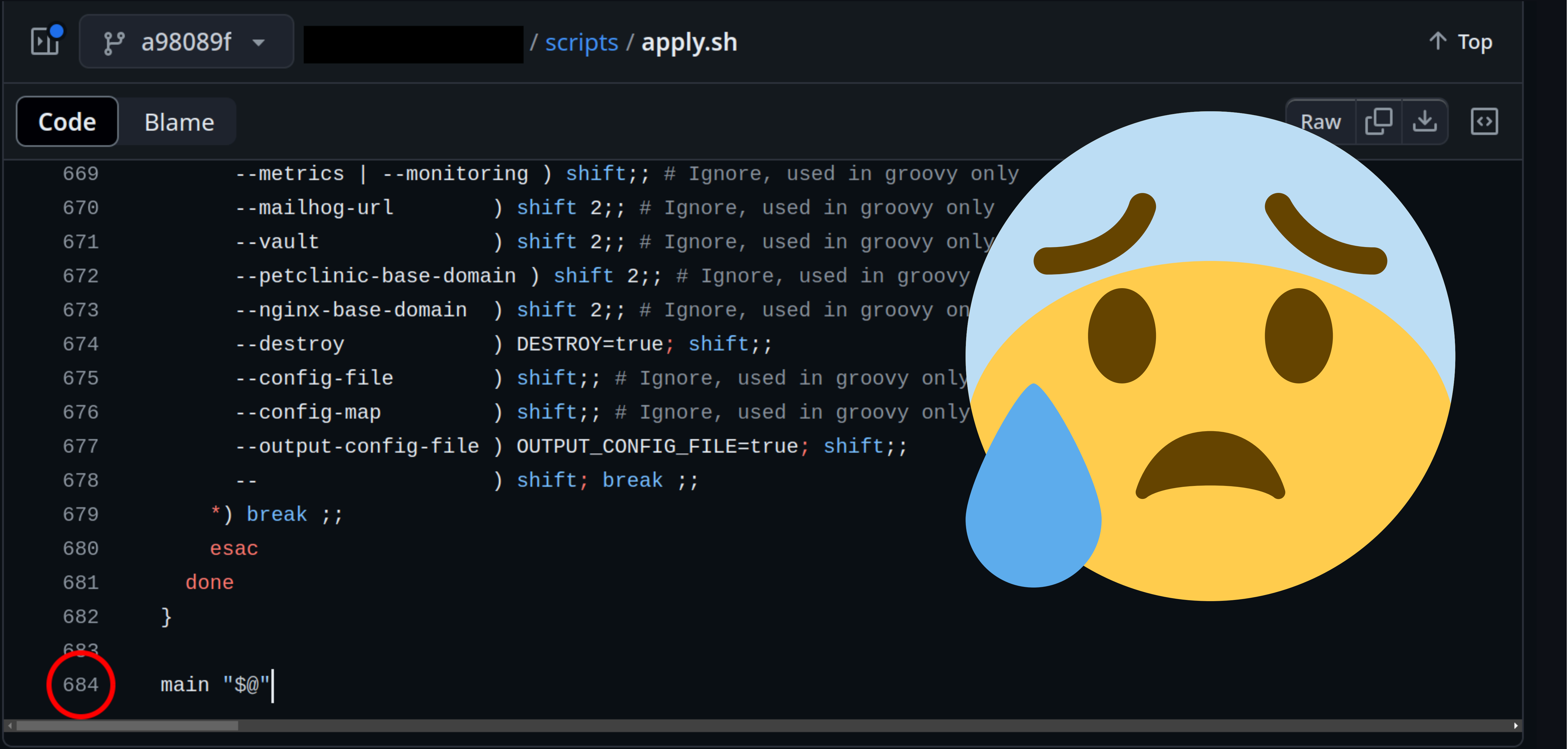
Next, start the platform

...



🌐 tenor.com/view/spongebob-squarepants-spongebob-think-thinking-gif-4280214517394111861

So, let’s write a *little* script…



[redacted] / scripts / apply.sh

```
669             --metrics | --monitoring ) shift;; # Ignore, used in groovy only
670             --mailhog-url        ) shift 2;; # Ignore, used in groovy only
671             --vault              ) shift 2;; # Ignore, used in groovy only
672             --petclinic-base-domain ) shift 2;; # Ignore, used in groovy
673             --nginx-base-domain  ) shift 2;; # Ignore, used in groovy on
674             --destroy            ) DESTROY=true; shift;;
675             --config-file        ) shift;; # Ignore, used in groovy only
676             --config-map         ) shift;; # Ignore, used in groovy only
677             --output-config-file ) OUTPUT_CONFIG_FILE=true; shift;;
678             --                   ) shift; break ;;
679           *) break ;;
680           esac
681         done
682     }
683
684     main "$@"
```

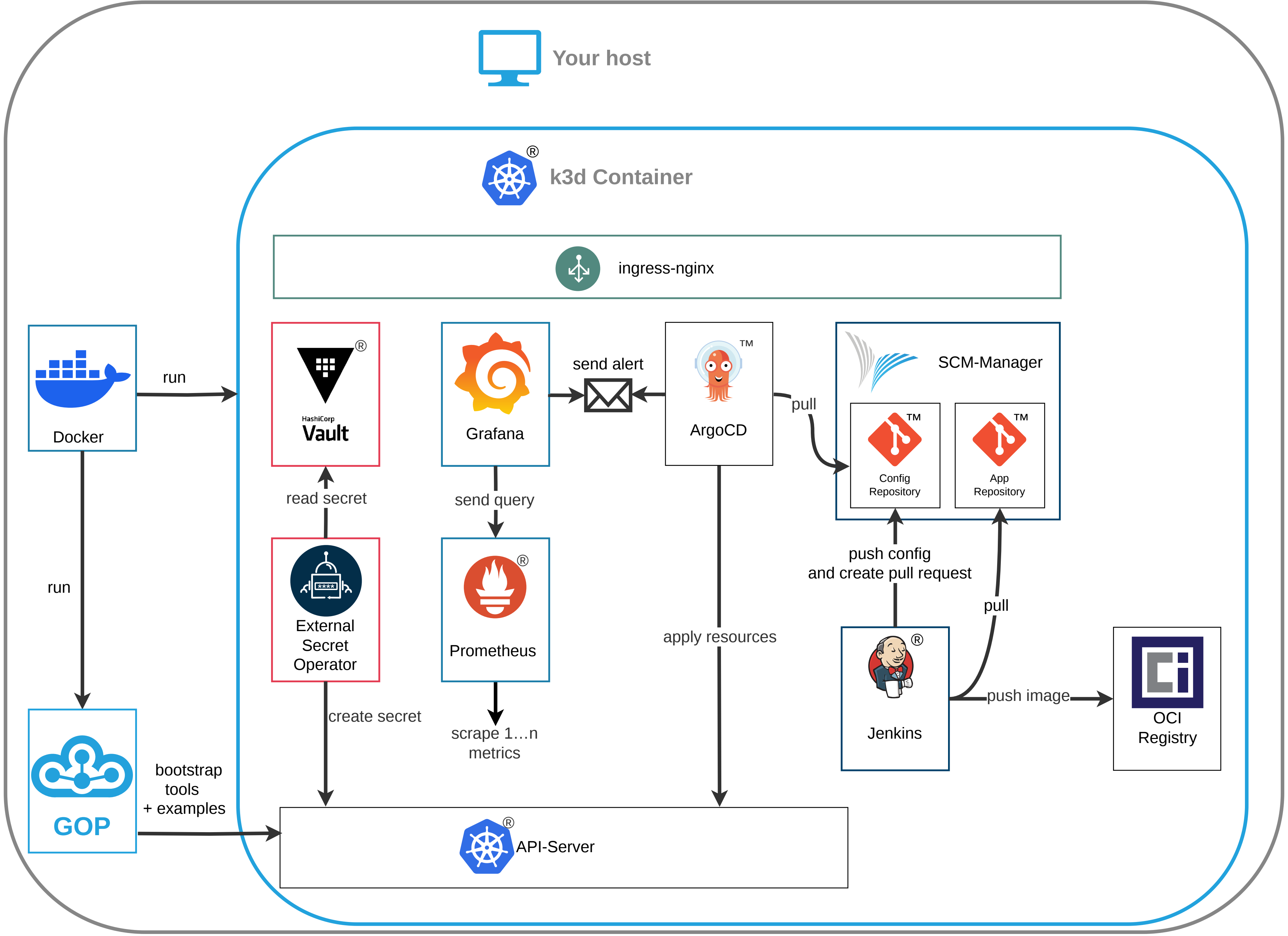
Why not start the platform with one command?

Meet GOP

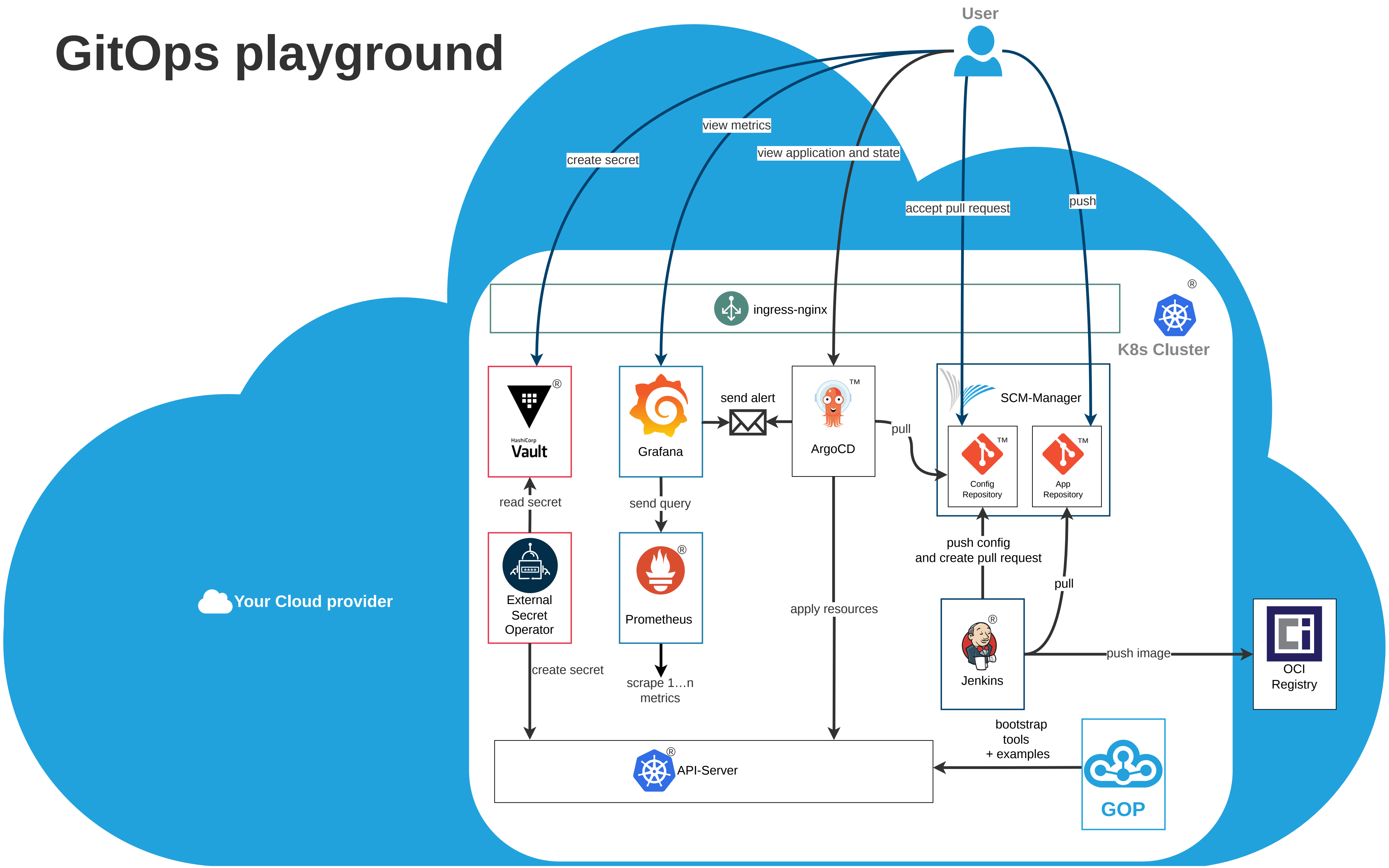
a GitOps-based operational stack (platform)

```
COMMIT='42e5f80'
bash <(curl -s \
  "https://raw.githubusercontent.com/cloudogu/gitops-playground/$COMMIT/scripts/init-cluster.sh") \
  --bind-ingress-port=80 \
   && docker run --rm -t -u $(id -u) \
    -v ~/.config/k3d/kubeconfig-gitops-playground.yaml:/home/.kube/config \
    --net=host \
    ghcr.io/cloudogu/gitops-playground:$COMMIT --yes --base-url=http://localhost --ingress-nginx \
      --argocd --monitoring --vault=dev --mailhog
```

cloudogu/gitops-playground


Your host
k3d Container
ingress-nginx
Docker
run
run
GOP
bootstrap
tools
+ examples
HashiCorp
Vault
read secret
External
Secret
Operator
create secret
Grafana
send alert
send query
Prometheus
scrape 1…n
metrics
ArgoCD
pull
apply resources
SCM-Manager
Config
Repository
App
Repository
push config
and create pull request
pull
Jenkins
push image
OCI
Registry
API-Server

GitOps playground



scripts/init-cluster.sh

```
k3d cluster create gitops-playground \
  # Mount port for ingress
  -p 80:80@server:0:direct \
  # Pin image for reproducibility
  --image=rancher/k3s:v1.29.1-k3s2 \
  # Disable built-in ingress controller, because we want to use the same one locally and in prod
  --k3s-arg=--disable=traefik@server:0 \
  # Allow node ports < 30000
  --k3s-arg=--kube-apiserver-arg=service-node-port-range=8010-65535@server:0 \
  # Hacks to make Docker available in Jenkins
  -v /var/run/docker.sock:/var/run/docker.sock@server:0 \
  -v /etc/group:/etc/group@server:0 -v /tmp:/tmp@server:0 \
  -p 30000:30000@server:0:direct

# Write kubeconfig to ~/.config/k3d/kubeconfig-gitops-playground.yaml
k3d kubeconfig write gitops-playground
```

docker run

```
docker run
  # Remove container after running, keeping your device clean
  # (remove in case of error to preserve logs)
  --rm
  # Colorful output, please
  -t
  # Run as current user, so files written to /tmp are accessible for you
  -u $(id -u) \
  # Mount kubeconfig for k3d
  -v ~/.config/k3d/kubeconfig-gitops-playground.yaml:/home/.kube/config \
  # Make k3d cluster available
  --net=host \
ghcr.io/cloudogu/gitops-playground:$COMMIT #Params for image go here
```

`ghcr.io/cloudogu/gitops-playground`

- OCI image
- Contains logic to install and configure the tools
- App written in Groovy (and bash 😰)
- Additional resources to run e.g. in air-gapped envs

Your turn

🌐 giphy.com/gifs/JIX9t2j0ZTN9S

Exercises 🏋️

- **GitOps+Alerting**

  Deploy broken app via GitOps, get alerted and fix problem
- **GitOps process**

  Promote a change in code all the way to production using GitOps
- **Monitoring**

  Deploy a Grafana dashboard for an app using GitOps
- **Secrets Management**

  Integrate secrets into app, propagate updates automatically
- ~~Progressive Delivery~~

  ~~Watch a canary release live with argo rollouts~~

Getting started

- Login: 👤 `admin` / 🔓 `admin`
- scmm.localhost
- argocd.localhost
- grafana.localhost
- vault.localhost
- mailhog.localhost
- jenkins.localhost

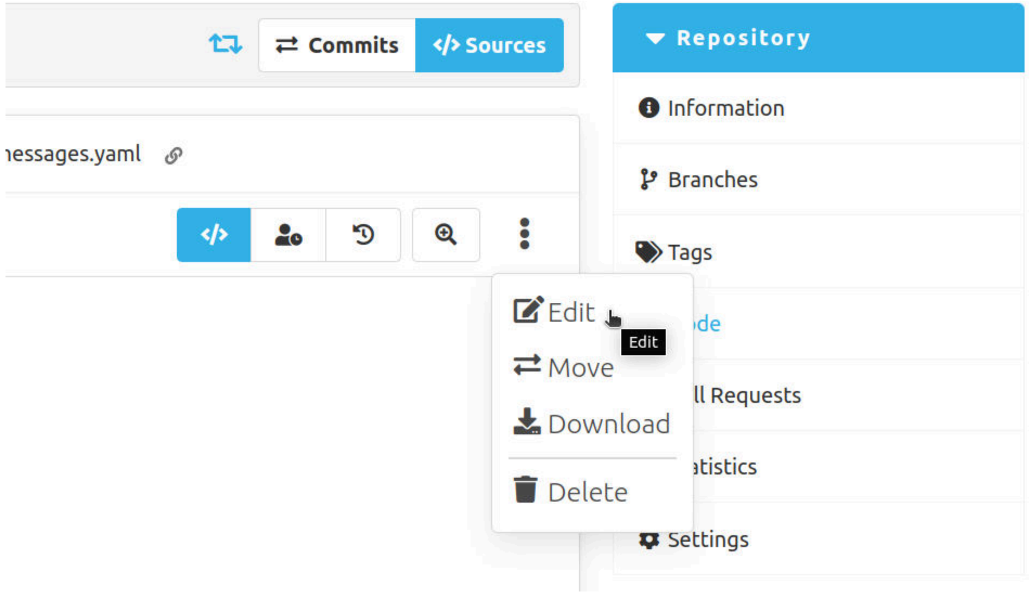
🏋️ Exercise: GitOps+Alerting 🐖

Deploy broken app via GitOps, get alerted and fix problem

Exercise: Alerting

1. Add repo to Argo CD
2. Create Argo CD `Application` YAML
3. Deploy, receive alert and fix app

💡 Hint: Edit file in SCM-Manager



1. Add repo to Argo CD

Add this repo to Argo CD (via GitOps):

`http://scmm-scm-manager.default.svc.cluster.local/scm/repo/exercises/broken-application`

1. Add to `repositories` in Argo CD’s config:
   - scmm.localhost/scm/repo/argocd/argocd/code/sources/main/argocd/values.yaml
2. Authorize `Project` to use the repo by adding it to `sourceRepos` here:
   - scmm.localhost/scm/repo/argocd/argocd/code/sources/main/projects/example-apps.yaml

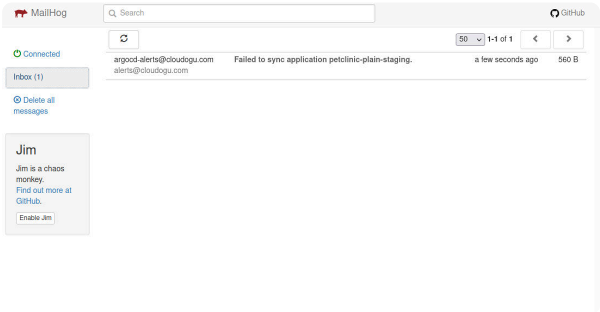
2. Create Argo CD `Application` YAML

- Go to argocd.localhost, click + NEW APP
- Enter `Name`: `broken`
- Click on `Project Name`, choose `example apps`
- Click on `Repository URL`, choose the `broken-application` repo
- Enter `Path`: `.`
- Click on `Cluster URL`, choose `https://kubernetes.default.svc`
- Enter `Namespace`: `example-apps-staging`
- At the top, click EDIT AS YAML and copy content

3. Deploy, receive alert and fix app

- Paste content here:

  scmm.localhost/scm/repo/argocd/example-apps/code/sourceext/create/main/argocd
- Enter `Filename`: `broken.yaml`, and Commit Message, then click Commit
- Go to argocd.localhost/applications/argocd/broken, click SYNC
- Check email in mailhog.localhost



argocd-alerts@cloudogu.com alerts@cloudogu.com **Failed to sync application petclinic-plain-staging.** a few seconds ago 560 B

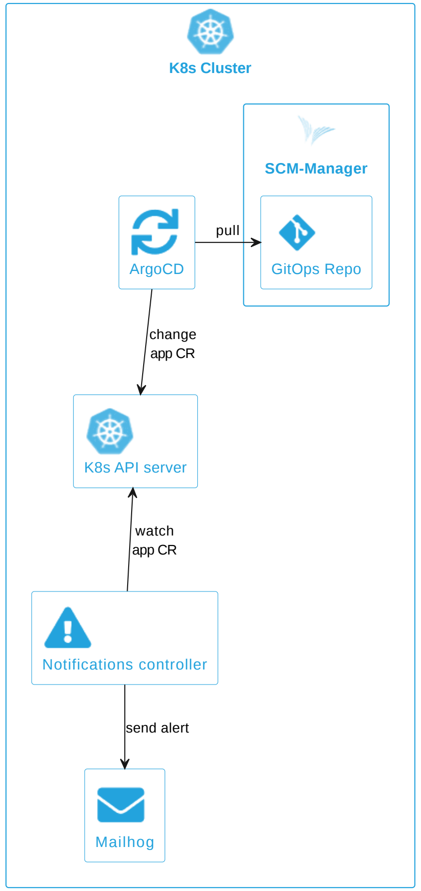
Jim

Jim is a chaos monkey. Find out more at GitHub.

- Follow link to ArgoCD-UI, analyse error
- Fix error in repo:

  scmm.localhost/scm/repo/argocd/example-apps/code/sources/main/argocd/broken.yaml
- Go to argocd.localhost/applications/argocd/broken, click SYNC
- Open application in Browser 🥳
- Have a closer look at the concepts behind this 👇

Alerting in GOP



Argo CD config:

github.com/cloudogu/gitops-playground/blob/0.2.0/argocd/argocd/argocd/values.ftl.yaml

scmm.localhost/scm/repo/argocd/argocd/code/sources/main/argocd/values.yaml

See also

- GitOps repo structure in GOP
- 🏋 Exercise: GitOps process

🏋️ Exercise: GitOps process

Promote a change in code all the way to production using GitOps

- Warmup 🤓
- GitOps with CI server and promotion 🚀

Warmup 🤓

1. Open Argo CD `Application`:

   argocd.localhost/applications/example-apps-staging/petclinic-plain

2. Open app in Browser:

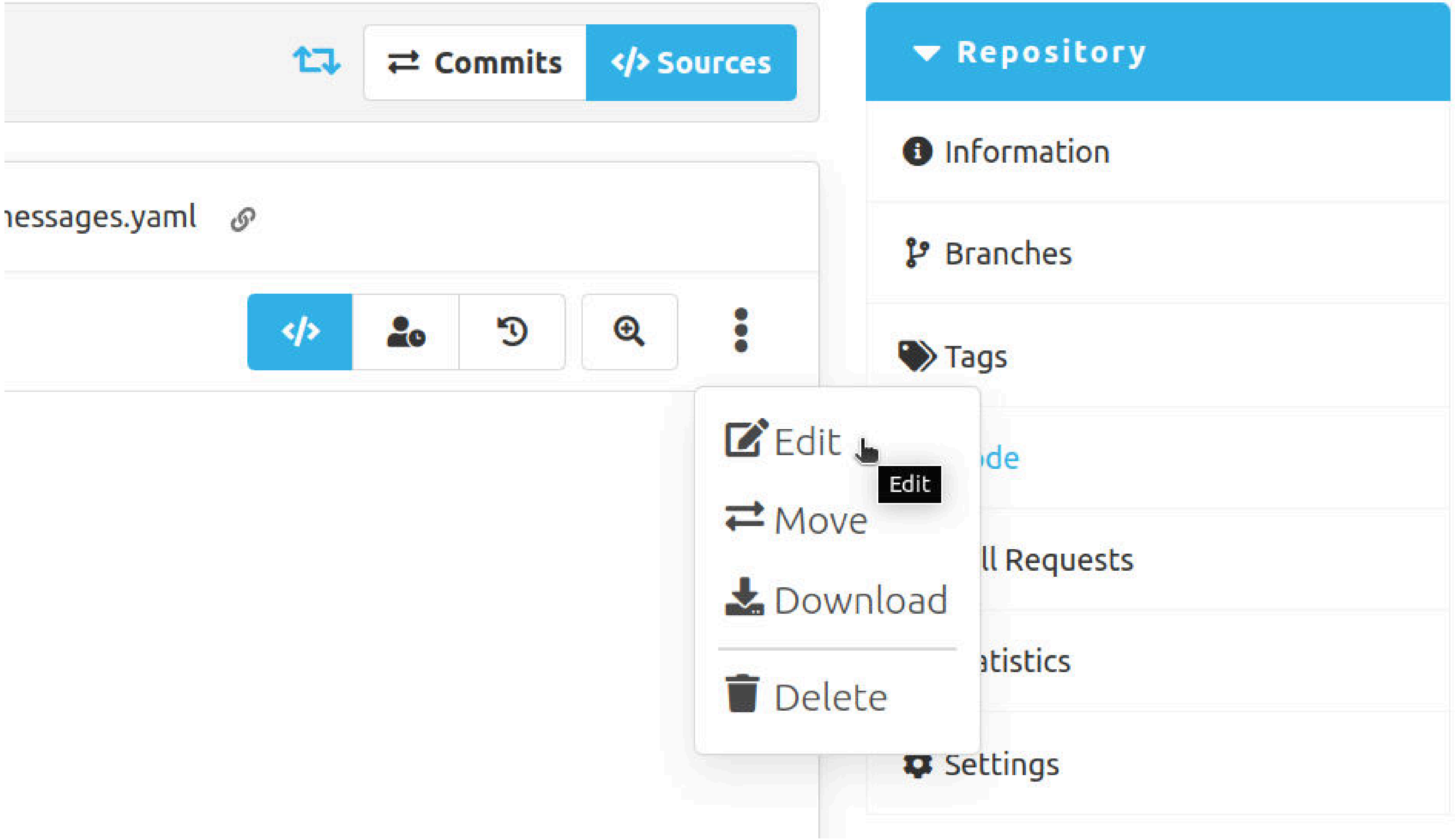
   🌐 staging.petclinic-plain.petclinic.localhost

3. Change `welcome` message in config repo:

   scmm.localhost/scm/repo/argocd/example-apps/code/sources/main/apps/spring-petclinic-plain/staging/generatedResources/messages.yaml

4. Press ⟳ REFRESH in ArgoCD UI
5. `Restart` `deploy` in ArgoCD UI ➡️ Watch GitOps deployment
6. 🔄 Reload app in Browser ➡️ Shows new message 🥳

💡 Hint: Edit file in SCM-Manager



GitOps with CI server and promotion 🚀

First:

Accept Pull Request for `petclinic-plain` to deploy prod

◆ scmm.localhost/scm/repo/argocd/example-apps/pull-requests

Then:

1. Change app, build image, deploy staging
2. Accept PR, deploy production

1. Change app, build image, deploy staging

1. Open staging app in Browser:

   🌐 staging.petclinic-plain.petclinic.localhost

2. Change `welcome` message in app repo

   scmm.localhost/scm/repo/argocd/petclinic-plain/code/sources/main/src/main/resources/messages/messages.properties

3. Wait for Build

   jenkins.localhost/job/example-apps/job/petclinic-plain/job/main

4. Open Argo CD `Application` for staging:

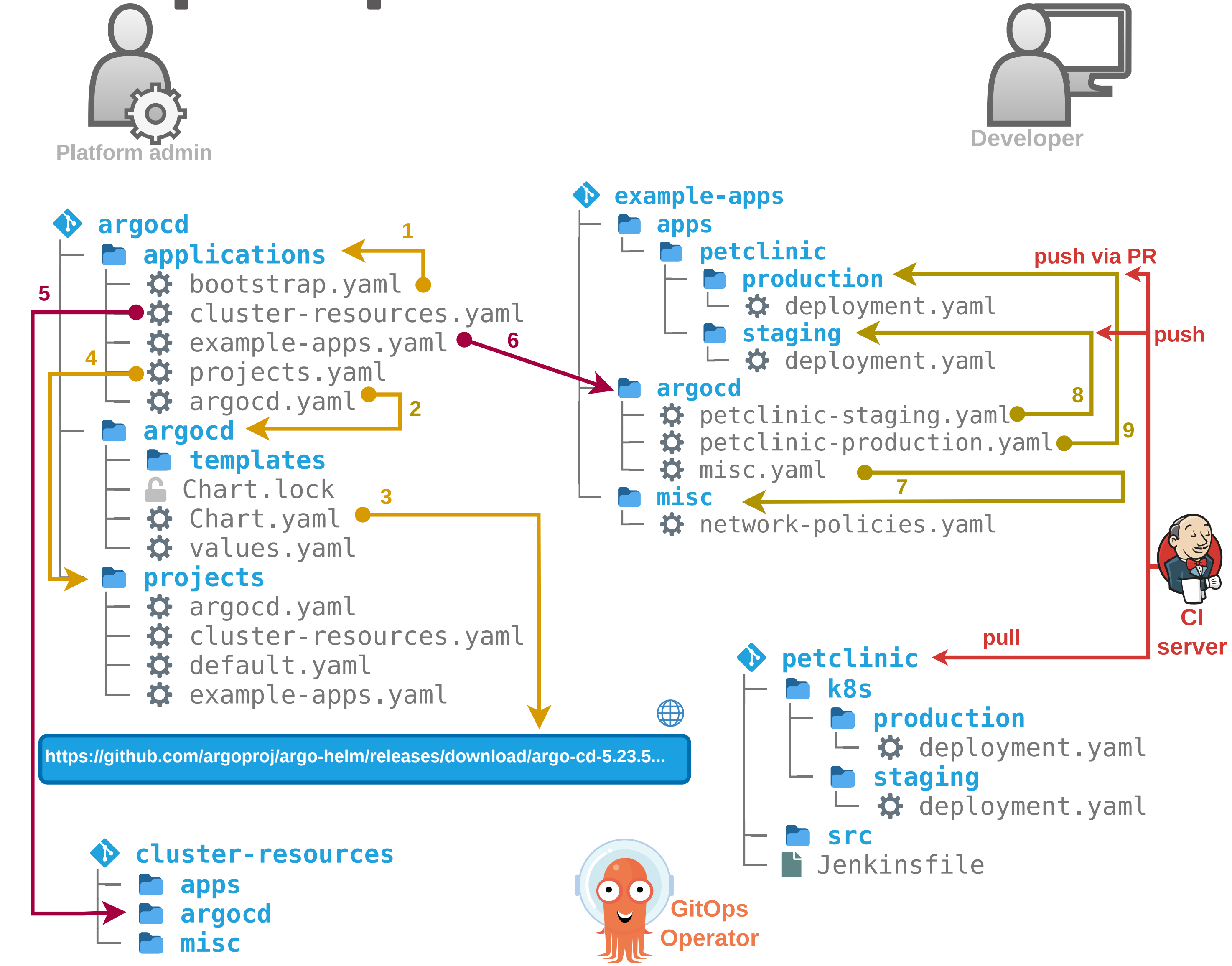
   argocd.localhost/applications/example-apps-staging/petclinic-plain

5. Press ↻ REFRESH in ArgoCD UI ➡️ Watch GitOps deployment
6. 🔄 Reload app in Browser ➡️ Shows message in staging 🥳

2. Accept PR, deploy production

1. Open production app in Browser:
   🌐 production.petclinic-plain.petclinic.localhost
2. Open Argo CD `Application` for production:
   argocd.localhost/applications/example-apps-production/petclinic-plain
3. Accept Pull Request for `petclinic-plain`
   scmm.localhost/scm/repo/argocd/example-apps/pull-requests
4. Press ⟳ REFRESH in ArgoCD UI ➡️ Watch GitOps deployment
5. 🔄 Reload app in Browser ➡️ Shows message in production 🥳🥳
6. Have a closer look at the concepts behind this 👇

GitOps repo structure in GOP



```
argocd
├─ applications
│  ├─ bootstrap.yaml
│  ├─ cluster-resources.yaml
│  ├─ example-apps.yaml
│  ├─ projects.yaml
│  └─ argocd.yaml
├─ argocd
│  ├─ templates
│  ├─ Chart.lock
│  ├─ Chart.yaml
│  └─ values.yaml
└─ projects
   ├─ argocd.yaml
   ├─ cluster-resources.yaml
   ├─ default.yaml
   └─ example-apps.yaml
```

https://github.com/argoproj/argo-helm/releases/download/argo-cd-5.23.5...

```
cluster-resources
├─ apps
├─ argocd
└─ misc
```

```
example-apps
├─ apps
│  └─ petclinic
│     ├─ production
│     │  └─ deployment.yaml
│     └─ staging
│        └─ deployment.yaml
├─ argocd
│  ├─ petclinic-staging.yaml
│  ├─ petclinic-production.yaml
│  └─ misc.yaml
└─ misc
   └─ network-policies.yaml
```

```
petclinic
├─ k8s
│  ├─ production
│  │  └─ deployment.yaml
│  └─ staging
│     └─ deployment.yaml
├─ src
└─ Jenkinsfile
```

scmm.localhost/scm/repo/argocd/petclinic-plain/code/sources/main/Jenkinsfile

uses

github.com/cloudogu/gitops-build-lib

cloudogu.com/blog/gitops-repository-patterns-part-6-examples

🏋️ Exercise: Secrets Management

Integrate secrets into app, propagate updates automatically

- Warmup 🤓
- Mount secret into app 🚀

Warmup 🤓

- Secret exposed via HTTP 😅

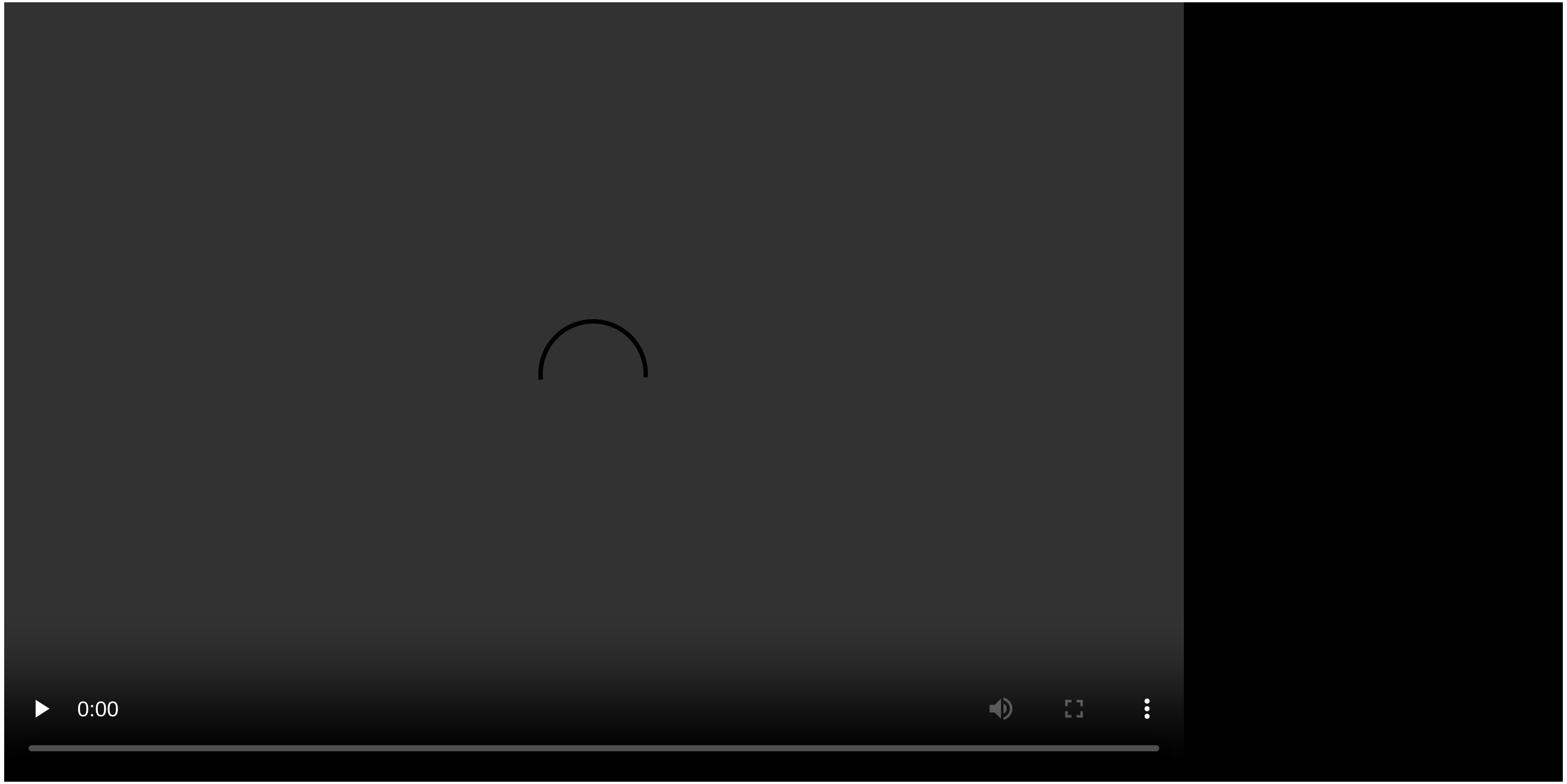
  🌐 staging.nginx-helm.nginx.localhost/secret
- Change in Vault:

  vault.localhost/ui/vault/secrets/secret/edit/staging/nginx-helm-jenkins
- Watch it propagate automatically (<2 min)

  Either reload Browser or:

```
while ; do  echo -n "$(date '+%Y-%m-%d %H:%M:%S'):  " ; \
   curl staging.nginx-helm.nginx.localhost/secret/  ; echo; sleep 1; done
```

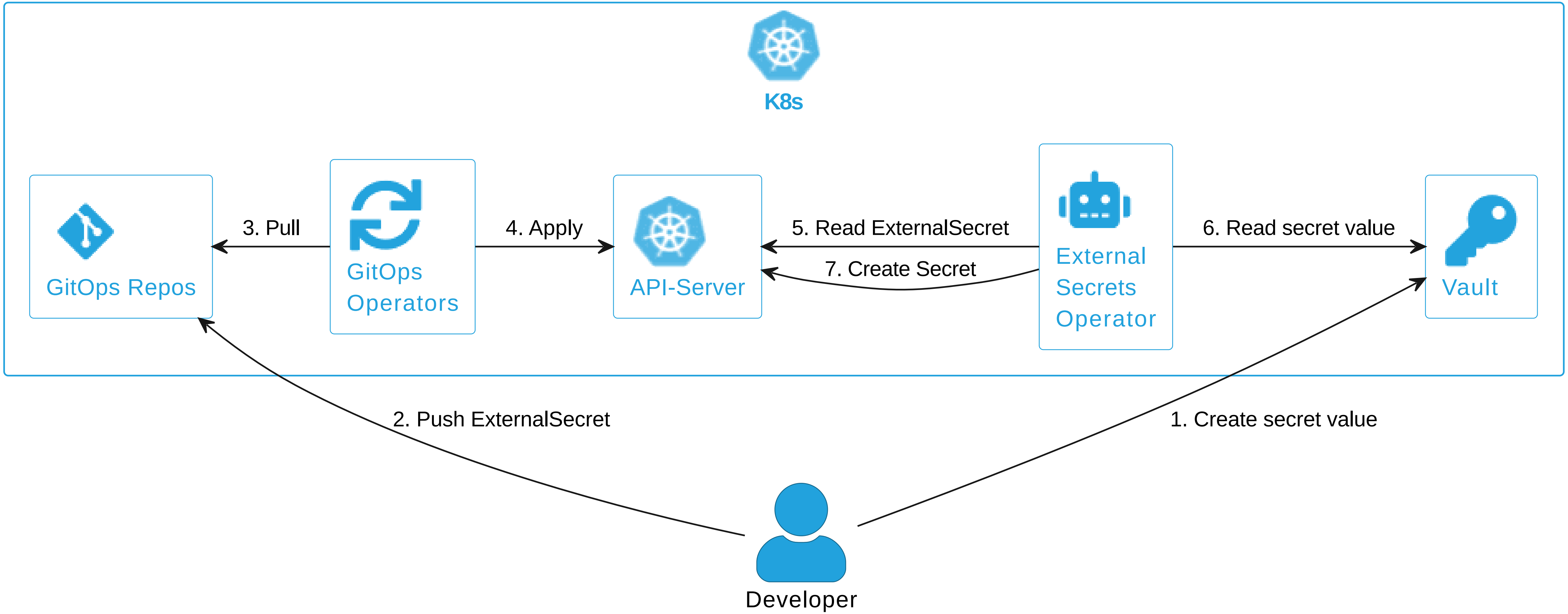
📽️ Warmup in time-lapse

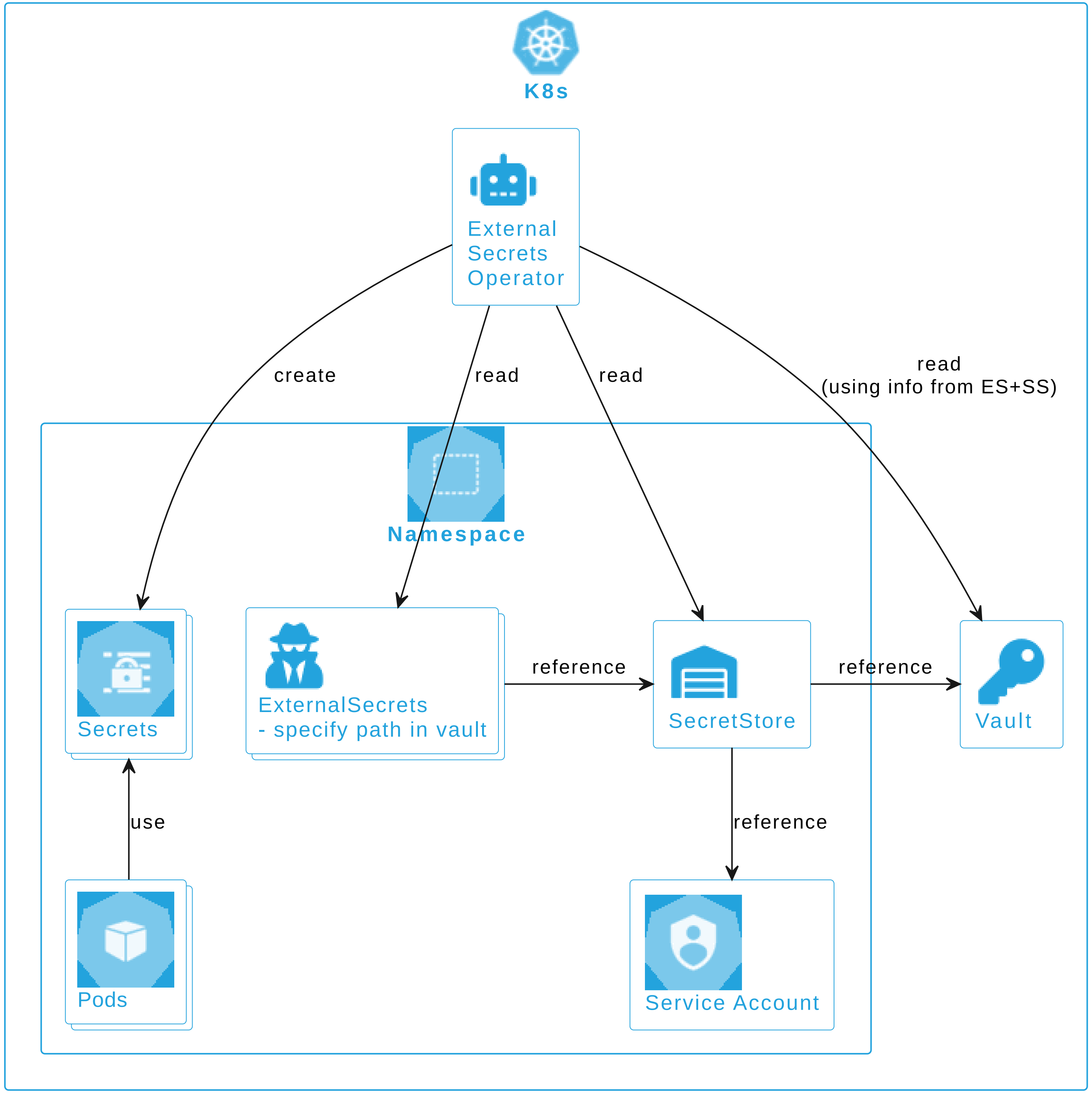


Mount secret into app 🚀

TODO summarize and link steps

External Secrets Operator with Vault




K8s
External Secrets Operator
create
read
read
read
(using info from ES+SS)
Namespace
Secrets
ExternalSecrets
- specify path in vault
reference
SecretStore
reference
Vault
use
reference
Pods
Service Account

ESO+Vault config in GOP

- SecretStore per Namespace:

  scmm.localhost/scm/repo/argocd/cluster-resources/code/sources/main/misc/secrets/secret-store-staging.yaml

- Example External Secret:

  scmm.localhost/scm/repo/argocd/nginx-helm-jenkins/code/sources/main/k8s/staging/external-secret.yaml

- Mounted into app:

  scmm.localhost/scm/repo/argocd/nginx-helm-jenkins/code/sources/main/k8s/values-shared.yaml

TODO

Johannes Schnatterer, Cloudogu GmbH

Slides

💪 Join my team: cloudogu.com/join/cloud-engineer

@schnatterer@floss.social in/jschnatterer @jschnatterer

Legal

Vault is a registered trademark of Hashicorp

Docker is a registered trademark of Docker Inc

Kubernetes is a registered trademark of the Linux Foundation

Git is a registered trademark of Software Freedom Conservancy

Grafana is a registered trademark of Grafana Labs The Grafana Labs Marks are trademarks of Grafana Labs, and are used with Grafana Labs' permission. We are not affiliated with, endorsed or sponsored by Grafana Labs or its affiliat