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

Yannick Christian Thomas, Johannes Schnatterer
Cloudogu GmbH

Slides

in/yannickchristhomas in/jschnatterer @schnatterer@floss.social

Version: 202406191530-4e286bb

Agenda

1. Intro
2. Meet GOP
3. Exercises, Getting Started

cloudogu

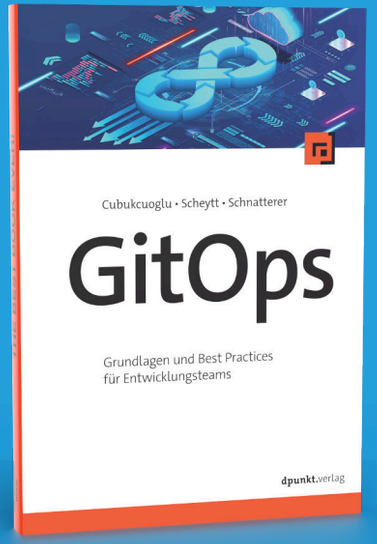
Yannick Christian Thomas

Cloud Engineer

Johannes Schnatterer

Technical Lead

Consulting + Infrastructure Team



🙋 What is your profession?

🤓 Software Engineer / Developer

🙋 What is your profession?

🛠️ Platform Engineer / Ops person

 What is your profession?

 None of the above

🙋 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.

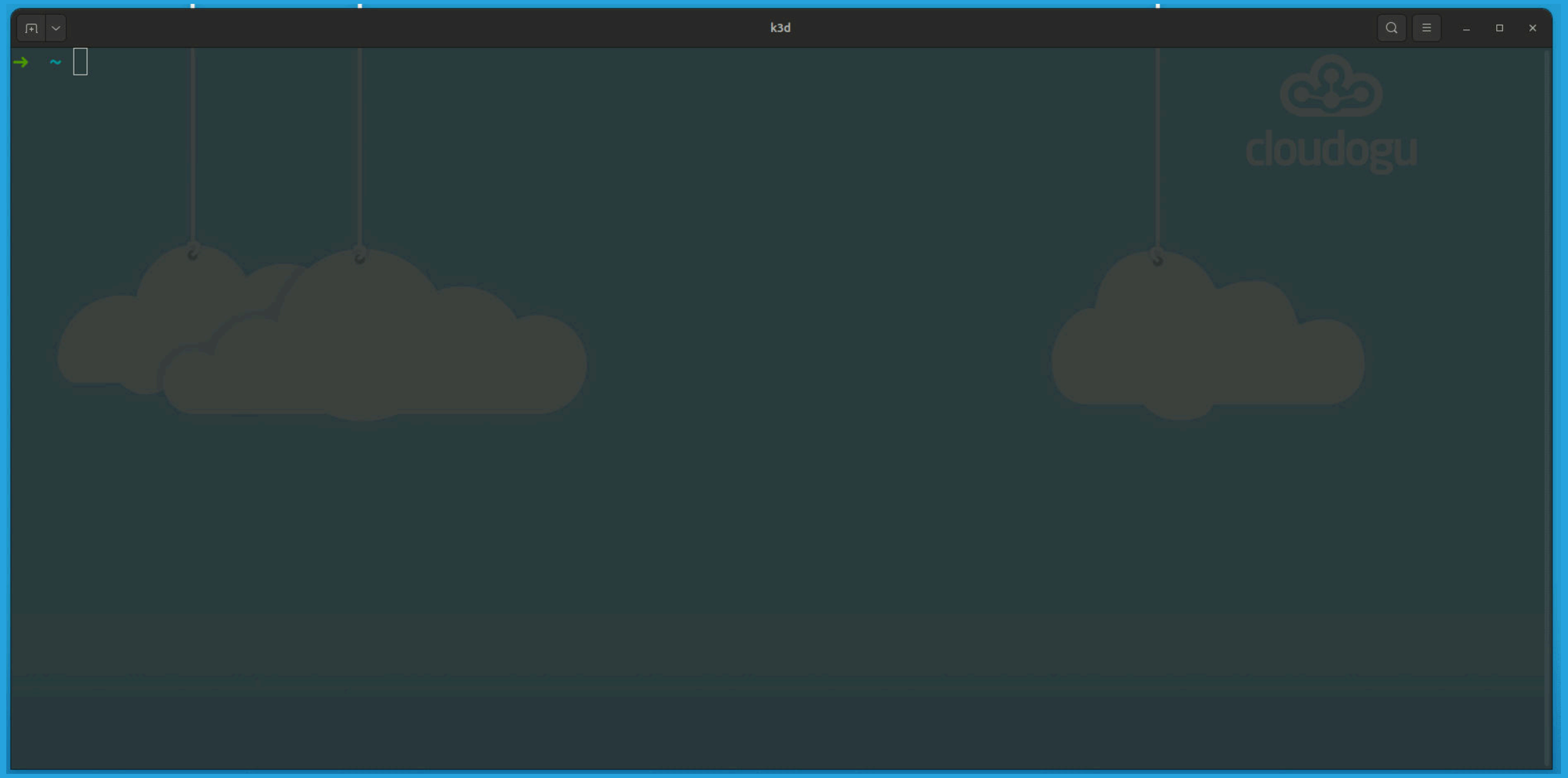
10:04 PM · Nov 27, 2017

---

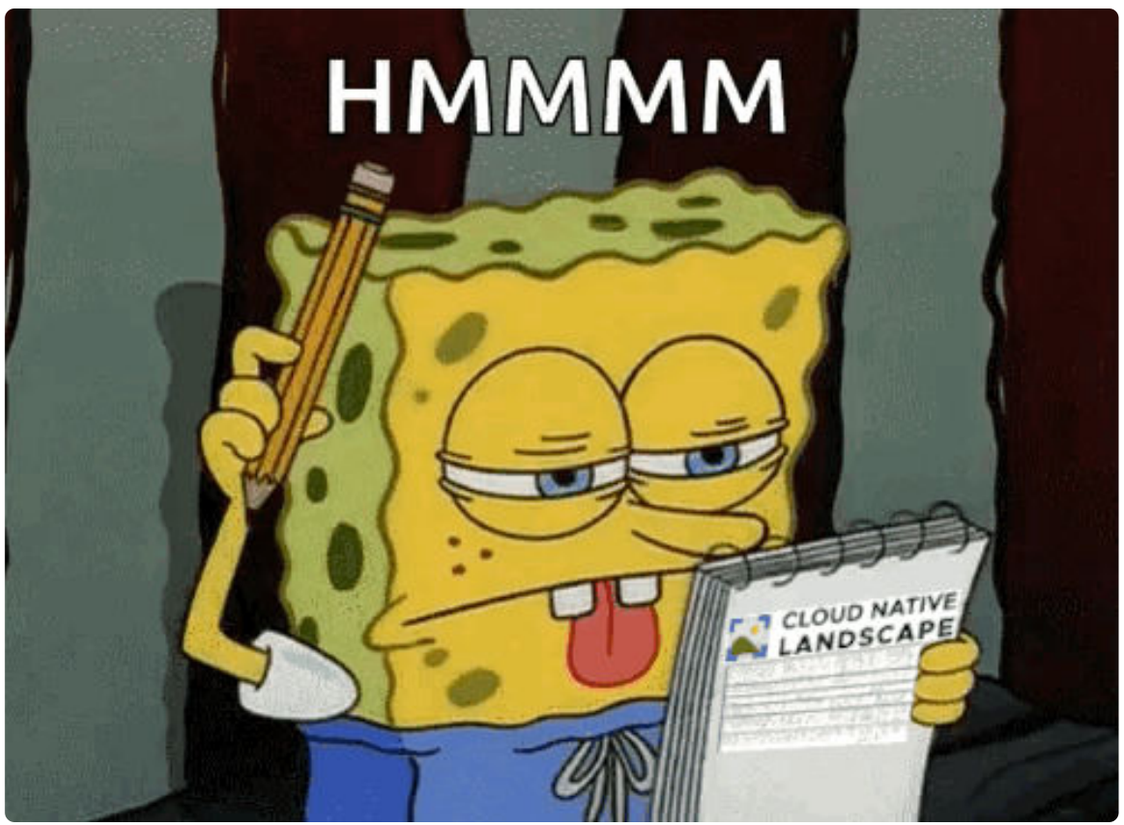
237 Reposts **44** Quotes **748** Likes **22** Bookmarks

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...

a98089f [redacted] / scripts / apply.sh ↑ Top

Code Blame Raw

```
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)

Slides

```
VERSION='0.3.0'
bash <(curl -s \
  "https://raw.githubusercontent.com/cloudogu/gitops-playground/$VERSION/scripts/init-cluster.sh") \
   && docker run --rm -t -u $(id -u) \
    -v ~/.config/k3d/kubeconfig-gitops-playground.yaml:/home/.kube/config \
    --net=host \
    ghcr.io/cloudogu/gitops-playground:$VERSION --yes --base-url=http://localhost --ingress-nginx \
      --argocd --monitoring --vault=dev --mailhog
```

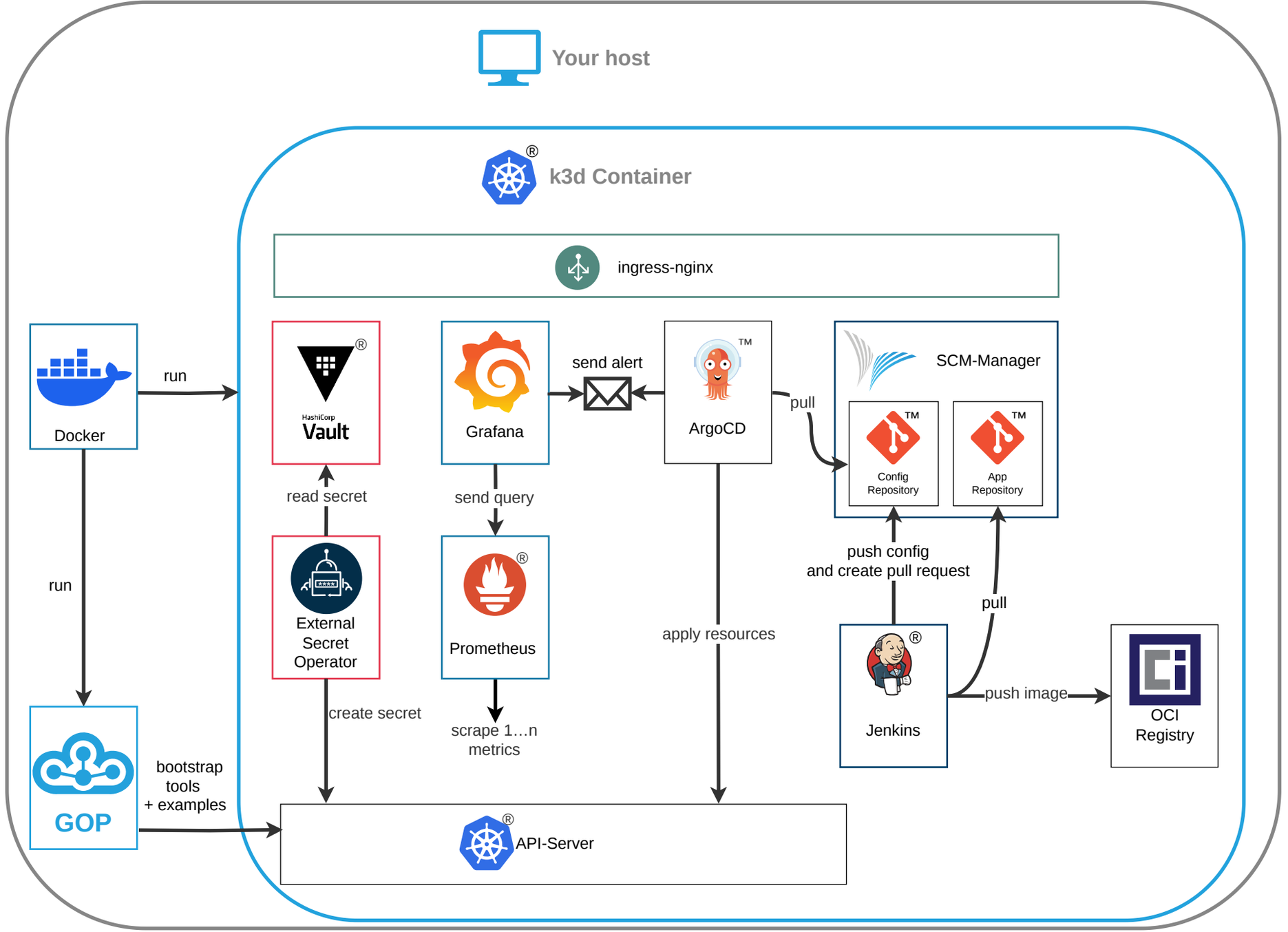
cloudogu/gitops-playground

💡 k3d hints

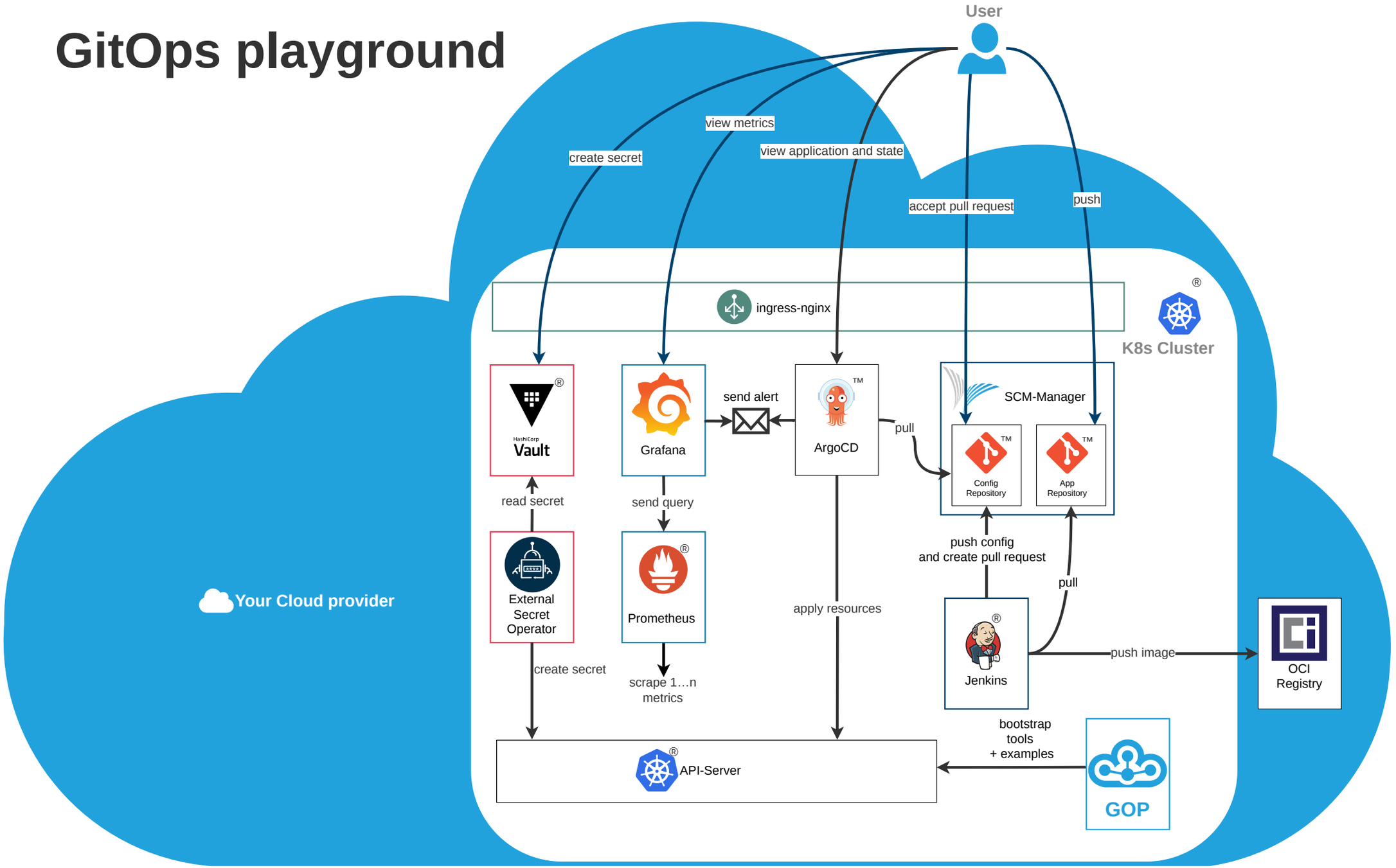
```
# Get an overview
kubectl get ingress -A # --context k3d-gitops-playground
# Also possible without installing kubectl
docker exec  k3d-gitops-playground-server-0 kubectl get ingress -A
```

```
# Pause to save resources
k3d cluster stop gitops-playground
# Continue later
k3d cluster start gitops-playground
```

```
# Cleanup
k3d cluster rm gitops-playground
# Or
docker rm $(docker ps -a -q --filter "name=^k3d-gitops-playground")
```



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
```

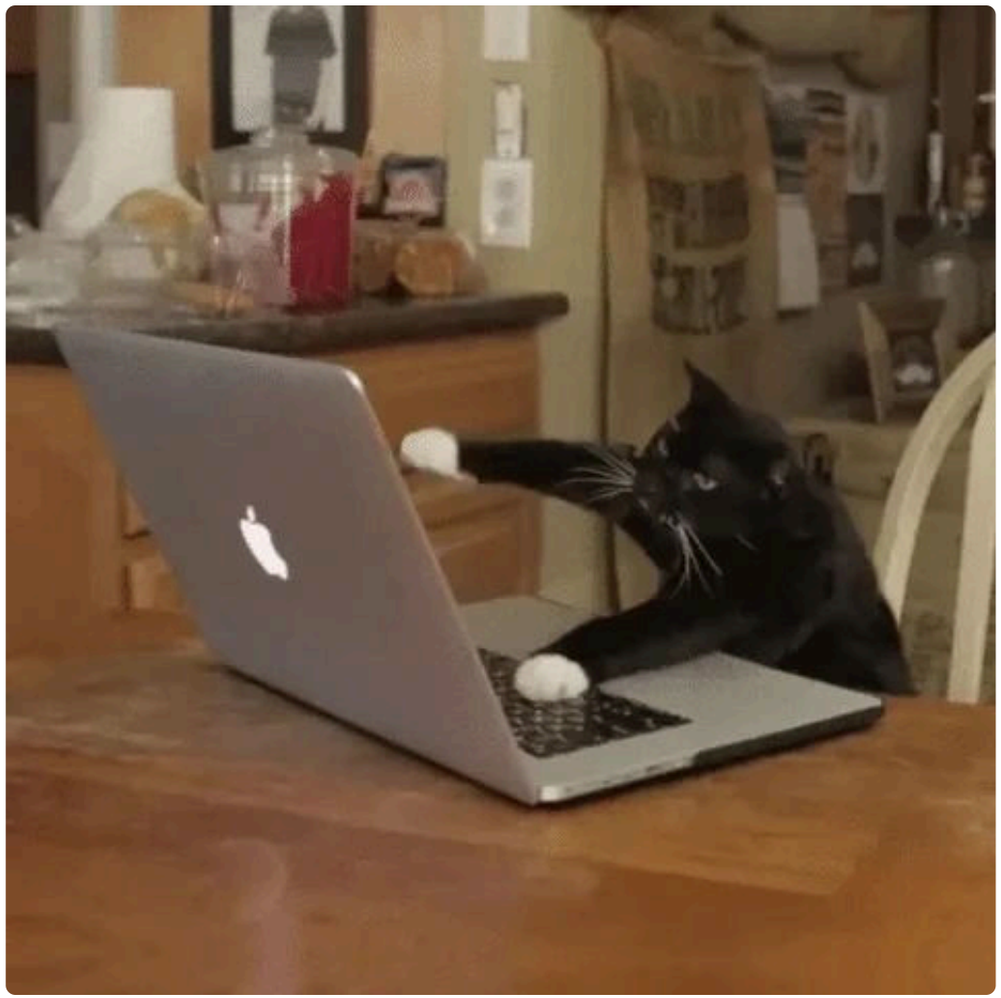
docker run ...

```
docker run
  # Remove container after running, keeping your device clean
  # (remove in case of error to preserve logs)
  --rm
  # Colorful output, please
  -t
  # Mount kubeconfig for k3d
  -v ~/.config/k3d/kubeconfig-gitops-playground.yaml:/home/.kube/config \
  # Run as current user to avoid  permission issues with kubeconfig
  -u $(id -u) \
  # Make k3d cluster available on 0.0.0.0 as described in kubeconfig
  --net=host \
# Image, pin for reproducibility
ghcr.io/cloudogu/gitops-playground:$VERSION \
  #Params for gop:
  --yes --base-url=http://localhost --ingress-nginx --argocd --monitoring --vault=dev --mailhog
```

`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/JJX9t2j0ZTN9S

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**

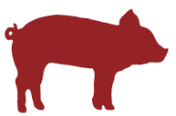
  Deploy a Grafana dashboard for an app using GitOps
- **Secrets Management**

  Integrate secrets into app, propagate updates automatically
- ~~Progressive Delivery~~ Reach out if interested

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

 Getting started

- Login: `admin` / `admin`
-  scmm.localhost
-  argocd.localhost
-  grafana.localhost  skip changing password on first login
-  vault.localhost
-  mailhog.localhost
-  jenkins.localhost

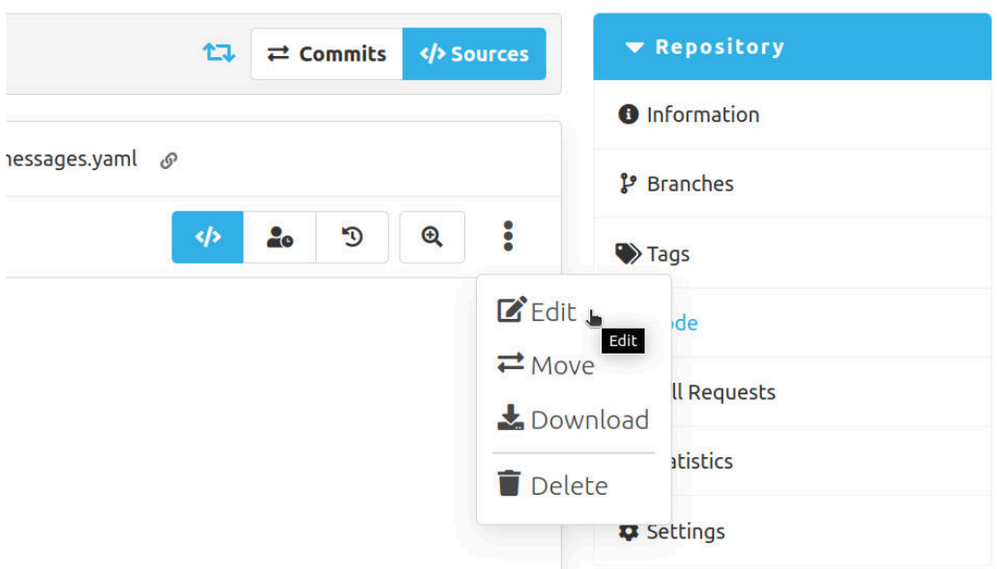
🏋️ Exercise: GitOps+Alerting  

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

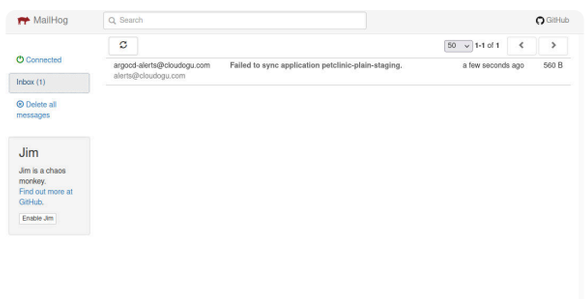
- Go to  argocd.localhost, click + NEW APP
- Enter `Name`: `broken`
- Click on `Project Name`, choose `example apps`
- Click on `Repository URL`, choose the `broken-application` repo
- Enter `Path`: `.`
- Click on `Cluster URL`, choose `https://kubernetes.default.svc`
- Enter `Namespace`: `example-apps-staging`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

3. Deploy, receive alert and fix app

- Paste content here:

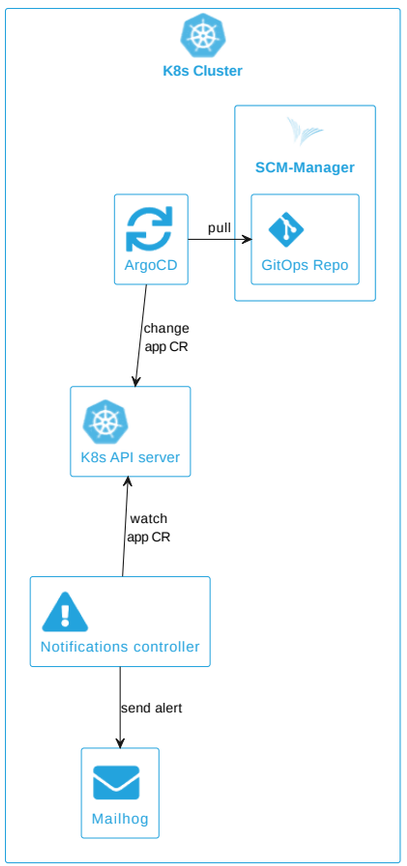
  scmm.localhost/scm/repo/argocd/example-apps/code/sourceext/create/main/argocd
- Enter `Filename`: `broken.yaml`, and commit message, then click Commit
- Go to argocd.localhost/applications/argocd/broken, click SYNC
- Check email in mailhog.localhost
- Follow link to ArgoCD-UI, analyse error
- Fix error in repo:

  scmm.localhost/scm/repo/argocd/example-apps/code/sources/main/argocd/broken.yaml
- Go to argocd.localhost/applications/argocd/broken, click SYNC
- Follow `ingress` link to open application in browser



Then have a closer look at the concepts behind this 👇

Alerting in GOP



Argo CD config:

github.com/cloudogu/gitops-playground/blob/0.3.0/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

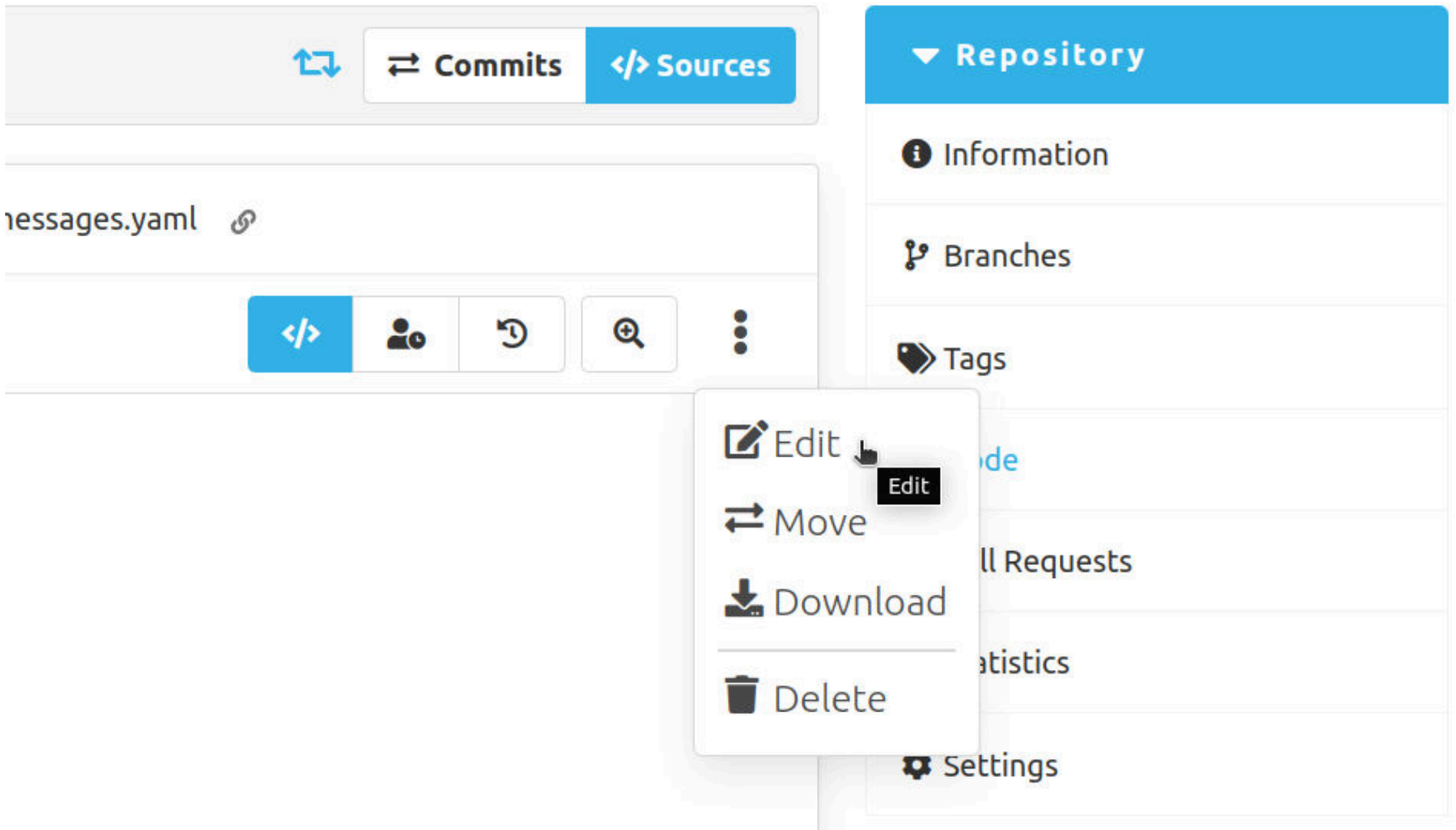
4. Press 

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 pull request, deploy production

1. Change app, build image, deploy staging

1. Open Argo CD `Application` for staging:

   argocd.localhost/applications/example-apps-staging/petclinic-plain
2. Follow `ingress` link to open application in browser
3. Change `welcome` message in app repo

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

   jenkins.localhost/job/example-apps/job/petclinic-plain/job/main
5. Press REFRESH in ArgoCD UI ➡ Watch GitOps deployment
6. 🔄 Reload app in Browser ➡ Shows message in staging 🥳

2. Accept pull request, deploy production

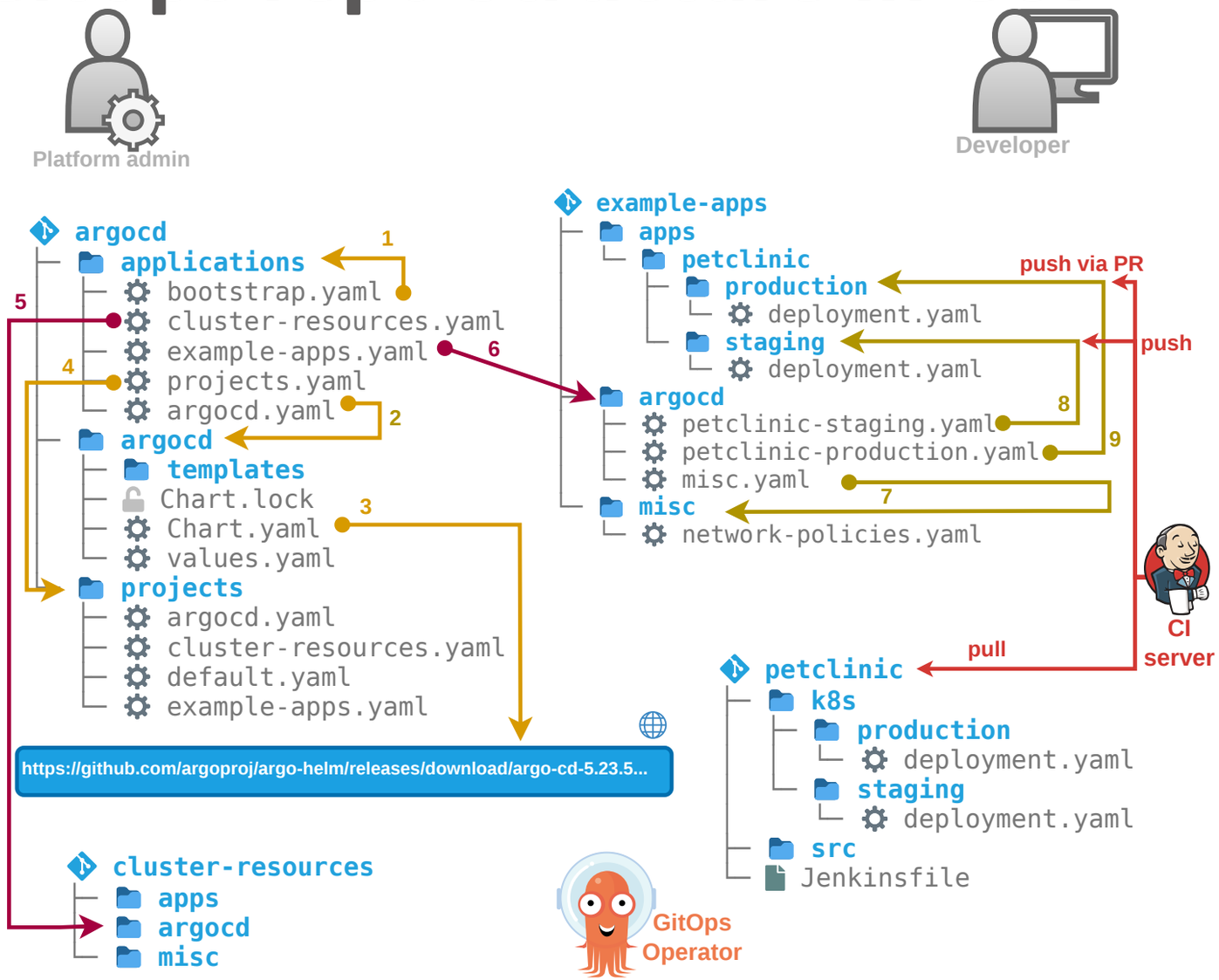
1. Open Argo CD `Application` for production:

    argocd.localhost/applications/example-apps-production/petclinic-plain
2. Follow  `ingress`  link to open application in browser
3. Accept pull request for `petclinic-plain`

    scmm.localhost/scm/repo/argocd/example-apps/pull-requests
4. Press  in ArgoCD UI  Watch GitOps deployment
5.  Reload app in Browser  Shows message in production 

Have a closer look at the concepts behind this 

GitOps repo structure in GOP



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: Monitoring  

- Deploy a Grafana dashboard for an app using GitOps 🚀
- (Expose and visualize metrics of Spring Boot app 🏆)

Deploy a Grafana dashboard for an app using GitOps

1. Expose metrics
2. Create specific Grafana dashboard JSON
3. Deploy dashboard via GitOps
4. Watch metrics

1. Expose metrics

- Enable metrics export on nginx via GitOps

scmm.localhost/scm/repo/argocd/example-apps/code/sourceext/edit/main/apps/nginx-helm-umbrella/values.yaml

```
nginx:
  metrics:
    enabled: true
    serviceMonitor:
      enabled: true
```

- Go to argocd.localhost/applications/example-apps-production/nginx-helm-umbrella, click SYNC
- Check if `servicemonitor` was created

2. Create specific Grafana dashboard JSON

- grafana.localhost/dashboard/import
- Paste content from here 👇 and click Load

  github.com/nginxinc/nginx-prometheus-exporter/blob/v1.2.0/grafana/dashboard.json
- `Name`: `nginx-helm-umbrella`
- Click `Select a Prometheus data source`: `Prometheus`
- Click Import

3. Deploy dashboard via GitOps

- Copy JSON from grafana.localhost/d/MsjffzSZz?editview=dashboard_json
- to scmm.localhost/scm/repo/argocd/example-apps/code/sourceext/create/main/apps/nginx-helm-umbrella
  - `Path`: **Add** `/files`
  - Enter `Filename`: `dashboard.json` + commit message, click **Commit**
- Add another file

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-helm-umbrella-dashboard
  labels:
    grafana_dashboard: "1"
data:
  dashboard.json: |-
{{ .Files.Get "files/dashboard.json" | indent 4 }}
```

- `Path`: **Add** `/templates`
- Enter `Filename`: `dashboard.yaml` + commit message, click **Commit**
- Go to argocd.localhost/applications/example-apps-production/nginx-helm-umbrella, click SYNC
- Check if `configmap` was created

4. Watch metrics

- Follow 🔀 `ingress` 🔗 link to open app in browser
- Generate traffic by 🔄 reloading
- Enjoy your dashboard

   grafana.localhost/d/MsjffzSZz 🥳

Expose and visualize metrics of Spring Boot app 🏆

- Expose container port by name:

  scmm.localhost/scm/repo/argocd/petclinic-plain/code/sources/main/k8s/staging/deployment.yaml

```
ports:
  # ..
  - containerPort: 9080
    name: actuator
```

- Expose prometheus metrics from app:

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

```
<dependency>
  <groupId>io.micrometer</groupId>
  <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

- Wait for build and deployment to staging

Create service for metrics port

```yaml
apiVersion: v1
kind: Service
metadata:
  name: spring-petclinic-plain-monitor
  namespace: example-apps-staging
  labels:
    app: spring-petclinic-plain
    type: metrics
spec:
  ports:
    - name: metrics
      port: 9080
      protocol: TCP
      targetPort: actuator
  selector:
    app: spring-petclinic-plain
```

💡 Use `kubectl` for faster iteration. GitOps can come later.

Create service monitor

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: spring-petclinic-plain-monitor
  namespace: example-apps-staging
spec:
  endpoints:
    - interval: 15s
      path: /actuator/prometheus
      port: actuator
  namespaceSelector:
    matchNames:
      - example-apps-staging
  selector:
    matchLabels:
      app: spring-petclinic-plain
      type: metrics
```

Find a suitable JVM / spring / micrometer dashboard and import it to Grafana

 grafana.com/grafana/dashboards 🥳

🏋️ Exercise: Secrets Management 

Integrate secrets into app, propagate updates automatically

- Warmup 🤓
- Mount secret into app 🚀

Warmup 🤓

- Secret exposed via HTTP 😅

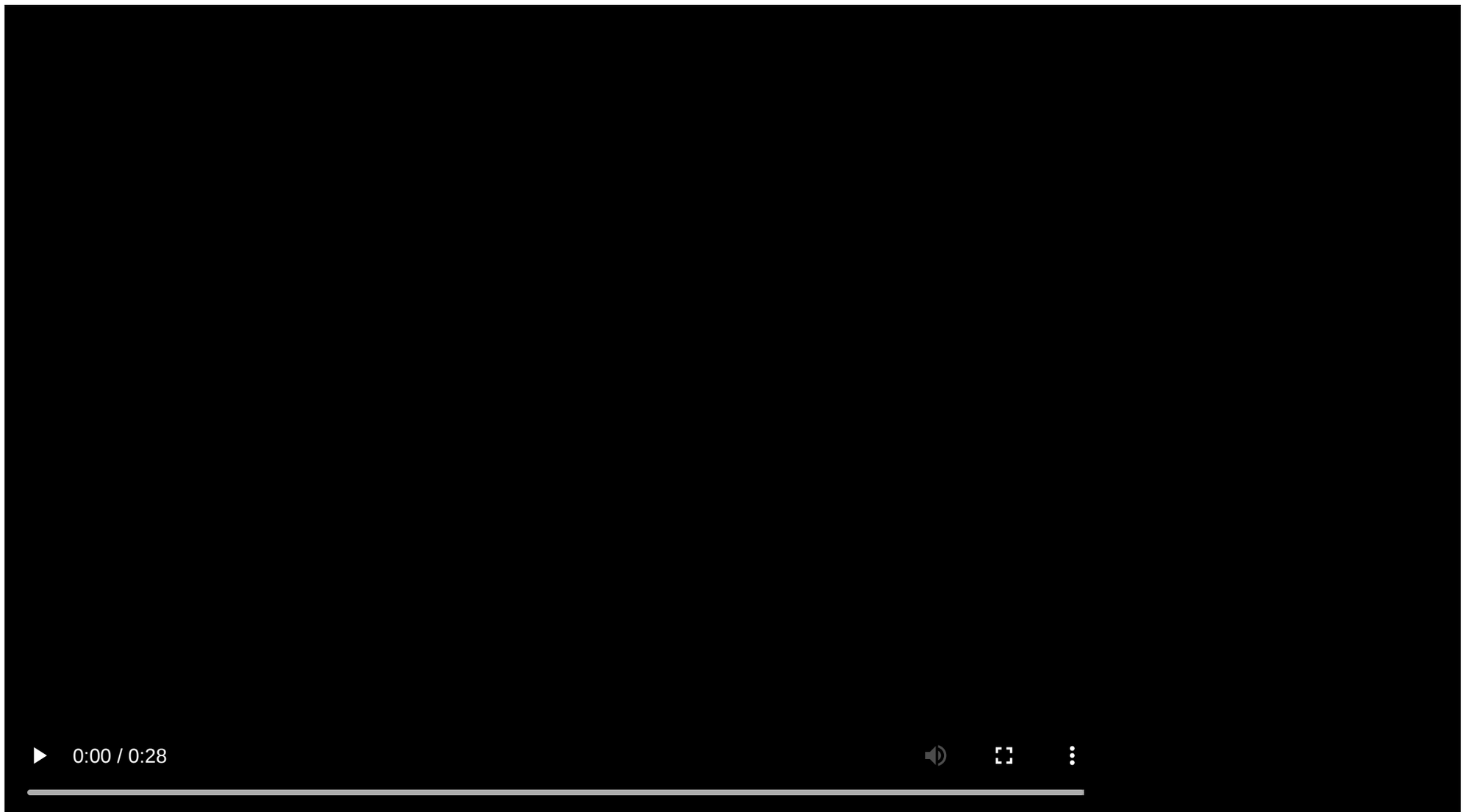
  🌐 staging.nginx-helm.nginx.localhost/secret
- Change in Vault:

  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



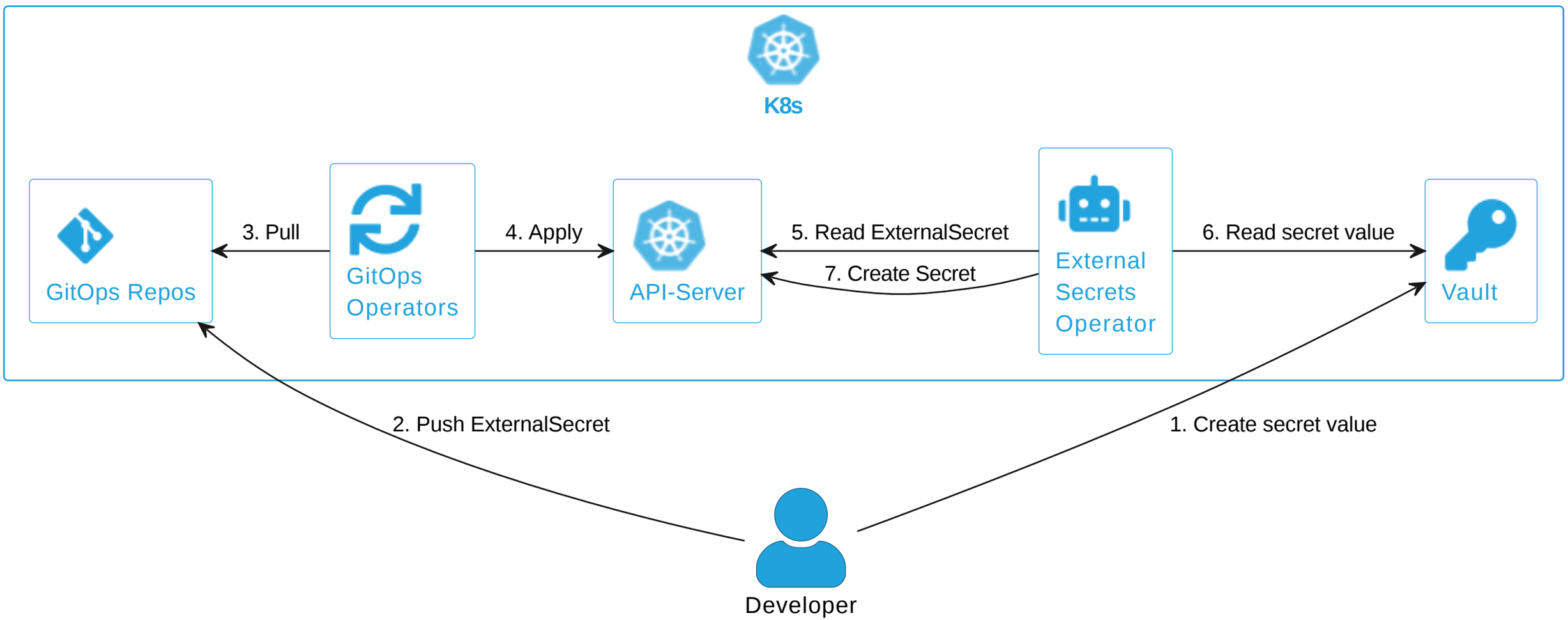
Mount secret into app 🚀

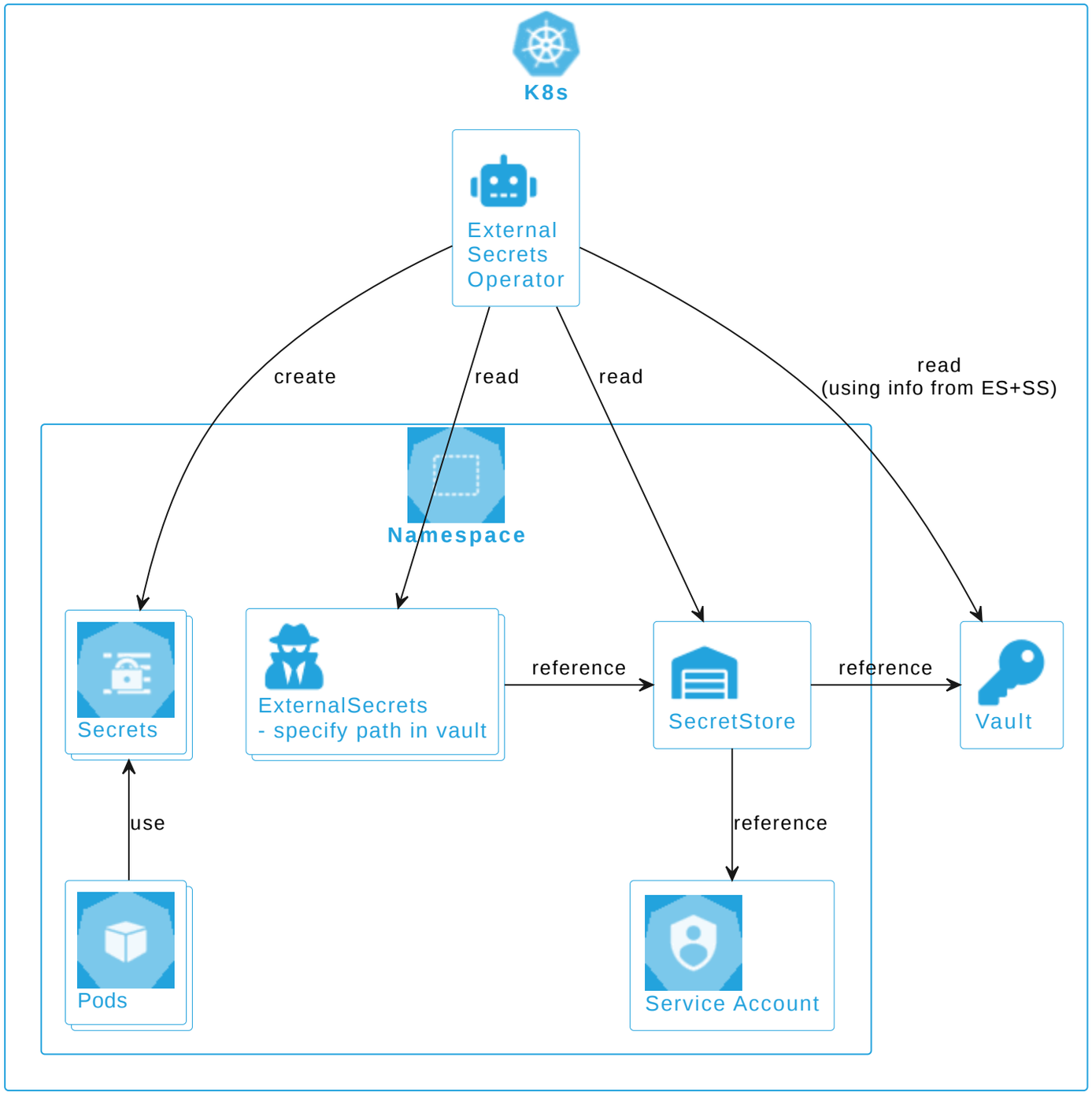
Create a new secret in Vault and mount it into an app

1. Create secret in Vault and sync into cluster via `ExternalSecret`
2. Use secret in app

Let's start with some basics 👇

External Secrets Operator (ESO) 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 `ExternalSecret`:

  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

1. Create secret in Vault and sync into cluster via `ExternalSecret`

1. Create secret in Vault
   - vault.localhost/ui/vault/secrets/secret
   - Click Create secret +
   - `Path for this secret`: `production/nginx-helm-umbrella`
   - `key`: `my-secret`, value: choose any
2. Deploy `ExternalSecret` via GitOps (💡 example on previous slide)

   scmm.localhost/scm/repo/argocd/example-apps/code/sourceext/create/main/apps/nginx-helm-umbrella
   - `Path`: **Add** `/templates`
   - Enter `Filename`: `secret.yaml` + commit message, click Commit
3. Go to argocd.localhost/applications/example-apps-production/nginx-helm-umbrella, click SYNC
4. Check if `secret` was created

2. Use secret in app

4. Mount `secret` into NGINX (💡 example on previous slide):
   - scmm.localhost/scm/repo/argocd/example-apps/code/sourceext/edit/main/apps/nginx-helm-umbrella/values.yaml
   - 💡 Hint: Add one `nginx.extraVolumes` and one `nginx.extraVolumeMounts`
5. Go to argocd.localhost/applications/argocd/broken, click SYNC
6. Follow `ingress` link to open application in browser
7. Add path `/secret` 🥳
8. Optional: Change the secret in Vault and wait for sync as in Warmup 🤓

⚠️ Secret in vault is transient, i.e. gone after restart (dev mode)

Please take a few moments to answer 5 short questions about GOP

Thanks for helping us improve 🙏

Yannick Christian Thomas
Johannes Schnatterer
Cloudogu GmbH

cloudogu

Please reach out for all questions or feedback!

yannick.thomas@cloudogu.com
johannes.schnatterer@cloudogu.com

in/yannickchristhomas
in/jschnatterer
@schnatterer@floss.social

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

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