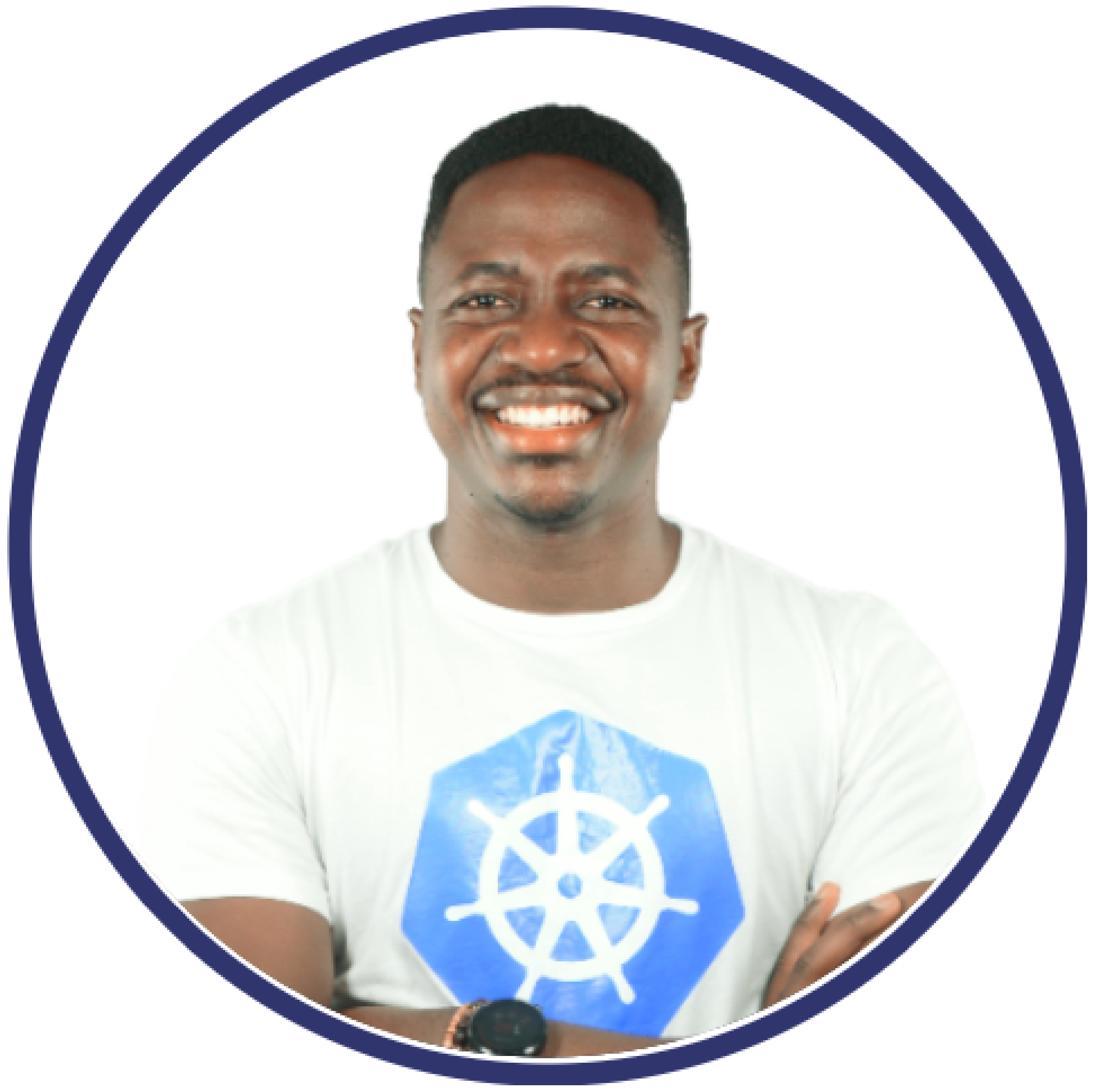
A Confidential Story of Well-Kept *Se***ts*

Lukonde Mwila | @Luke9ine

Lukonde Mwila

Senior Developer Advocate at AWS | CNCF Ambassador

@Luke9ine



So what's your story?

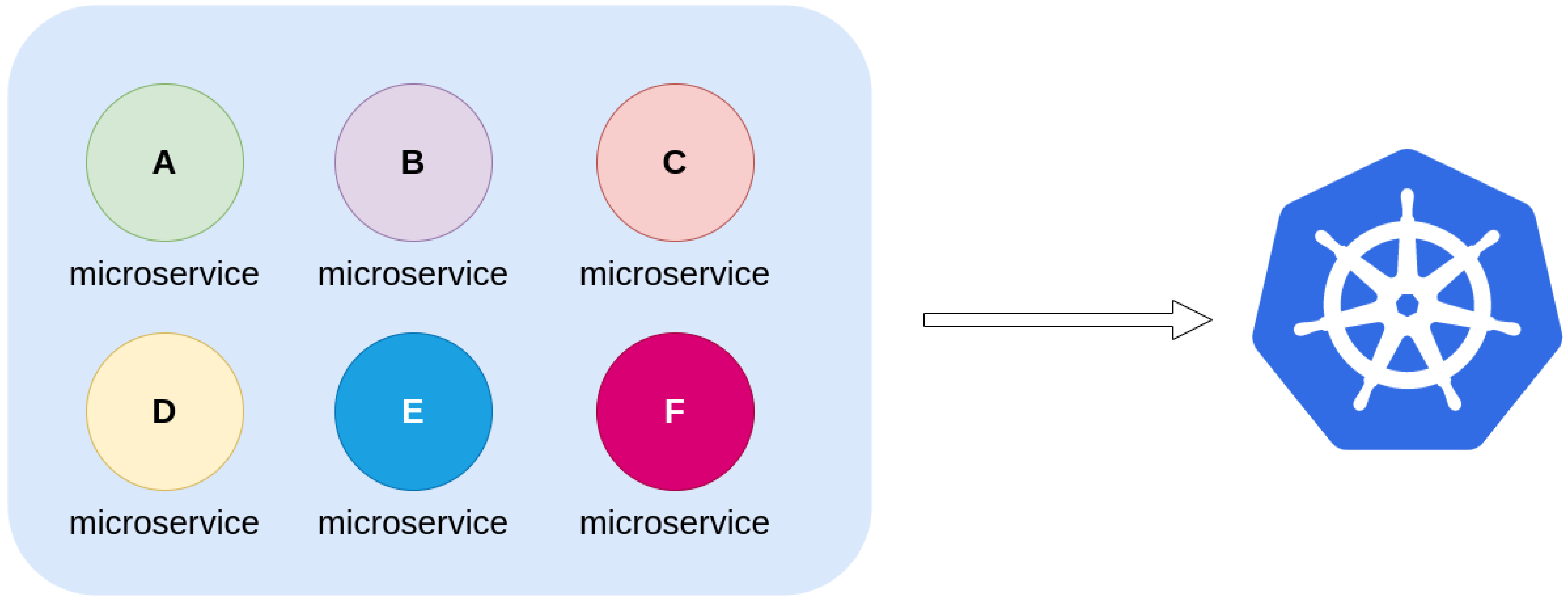
i-chose-k8s

they-chose-k8s-for-me

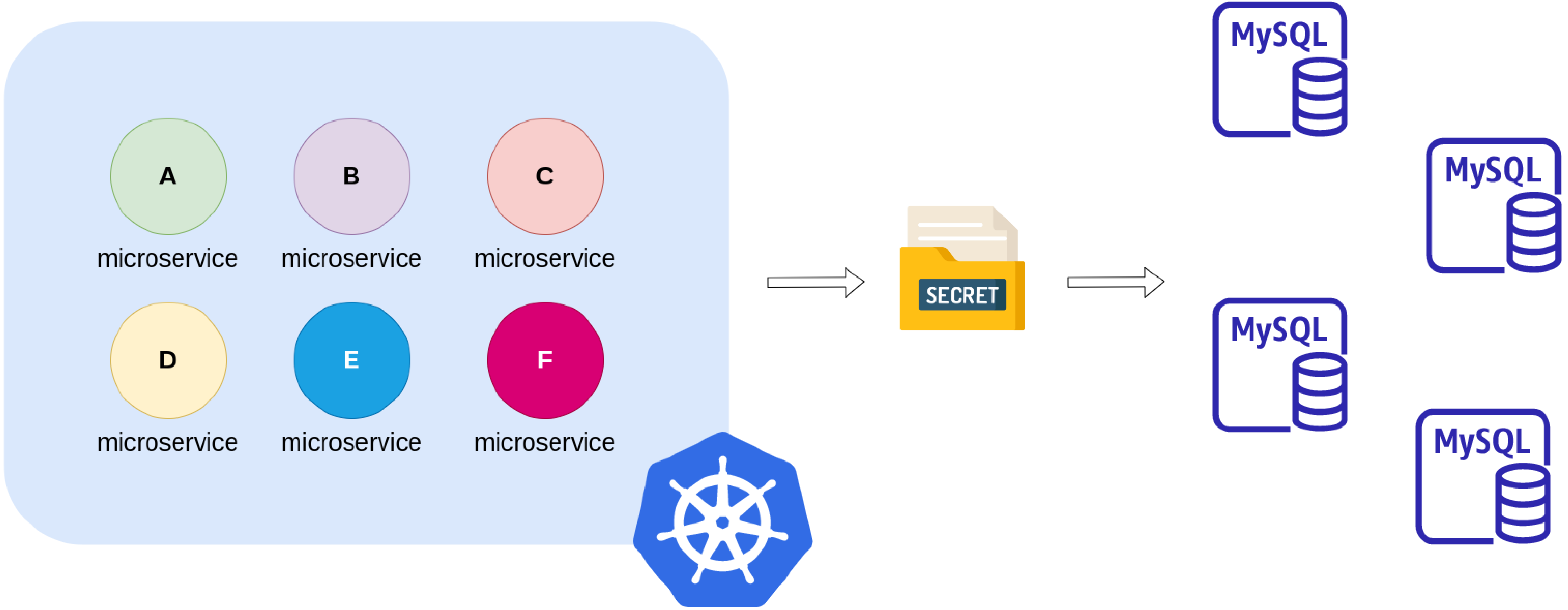
k8s-chose-me

KUBERNETES IS SIMPLE
ALL YOU NEED IS YAML
imgflip.com

Workload Migration



Workload Migration



What is a secret?

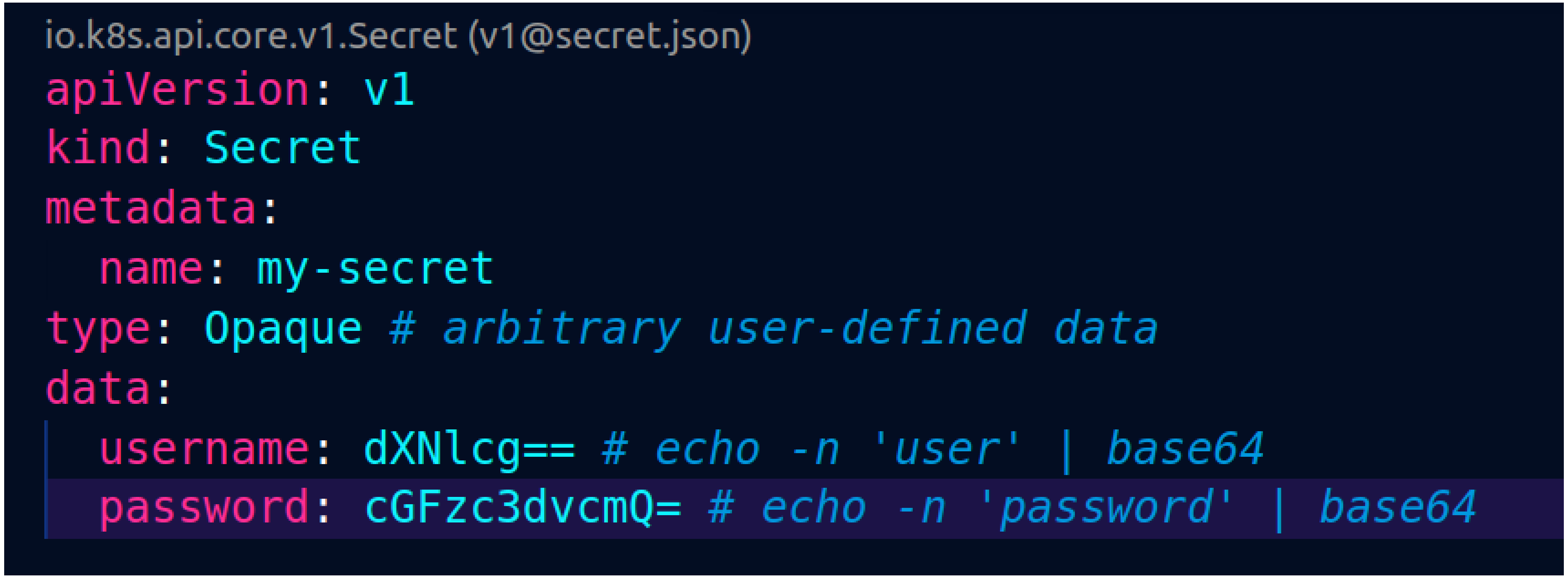
What is a secret?

A K8s resource that is used for storing configuration data.

Stores small pieces of sensitive information:

- Credentials
- TLS certificates
- OAuth tokens
- SSH keys

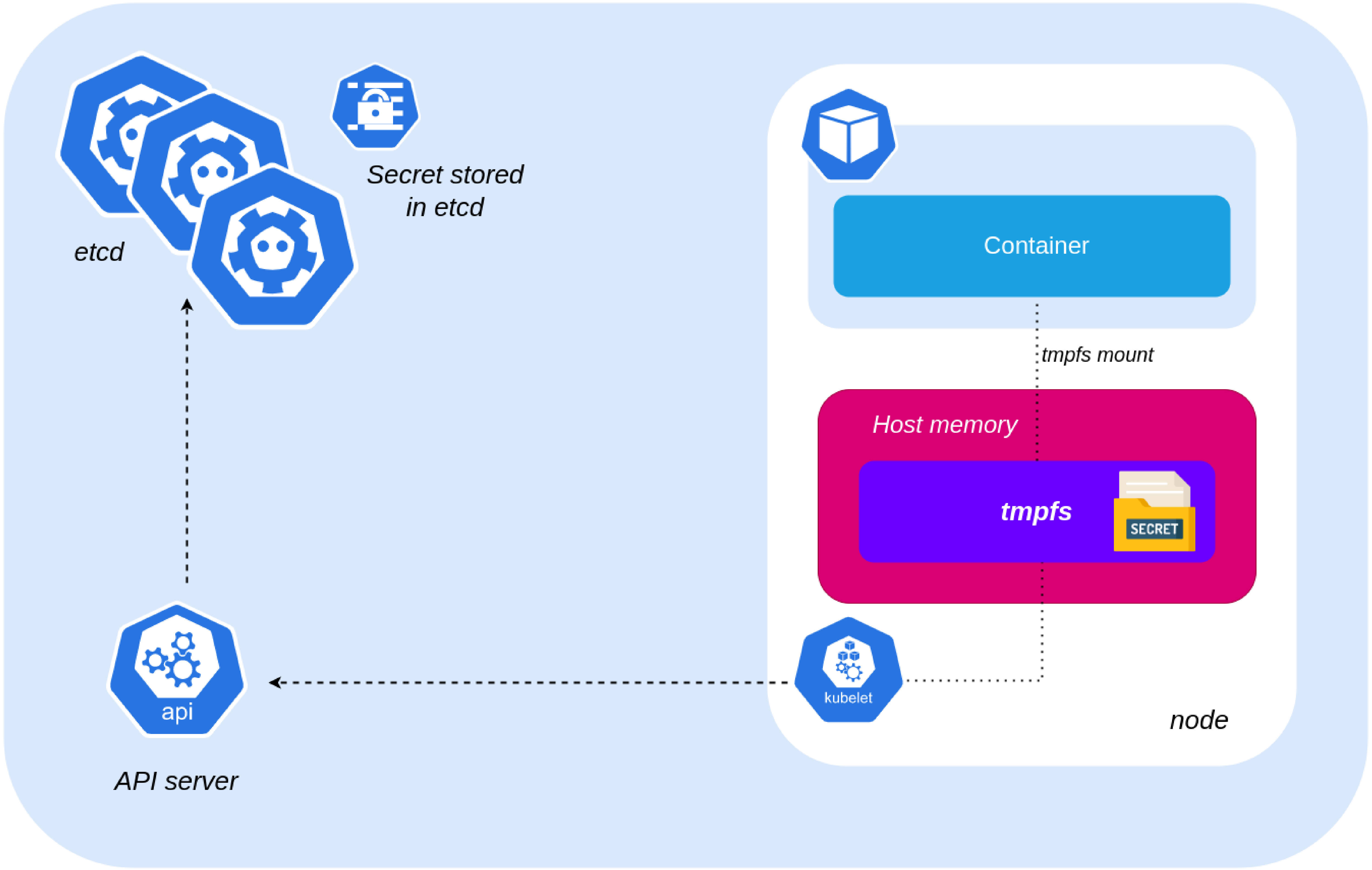
Secrets manifest



```
io.k8s.api.core.v1.Secret (v1@secret.json)
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
type: Opaque # arbitrary user-defined data
data:
  username: dXNlcg== # echo -n 'user' | base64
  password: cGFzc3dvcmQ= # echo -n 'password' | base64
```

So we've decided to store the db credentials in Kubernetes secrets.
You do know secrets don't encrypt the values, right?
imgflip.com

How are secrets mounted?



What are some main risks & vulnerabilities?

Red flags

Some of our primary concerns:

- Non-encrypted data in *etcd*
- Secrets manifest files in git repos
- Mounting secrets as env vars
- Mounting secrets as volumes
- Root user exploitation

ONE DOES NOT SIMPLY
CREATE SECRETS IN K8S
imgflip.com

Overcoming red flags

- Where is the secret stored?
- Who needs to know about the secret?
- How is the secret shared?
- How is it consumed?
- How do you prevent the secret from being easily interpreted?
- How do you create guardrails?

Where is the secret kept?

I'VE KEPT IT SAFE
SO WHY
SHOULDN'T I KEEP IT?
imgflip.com

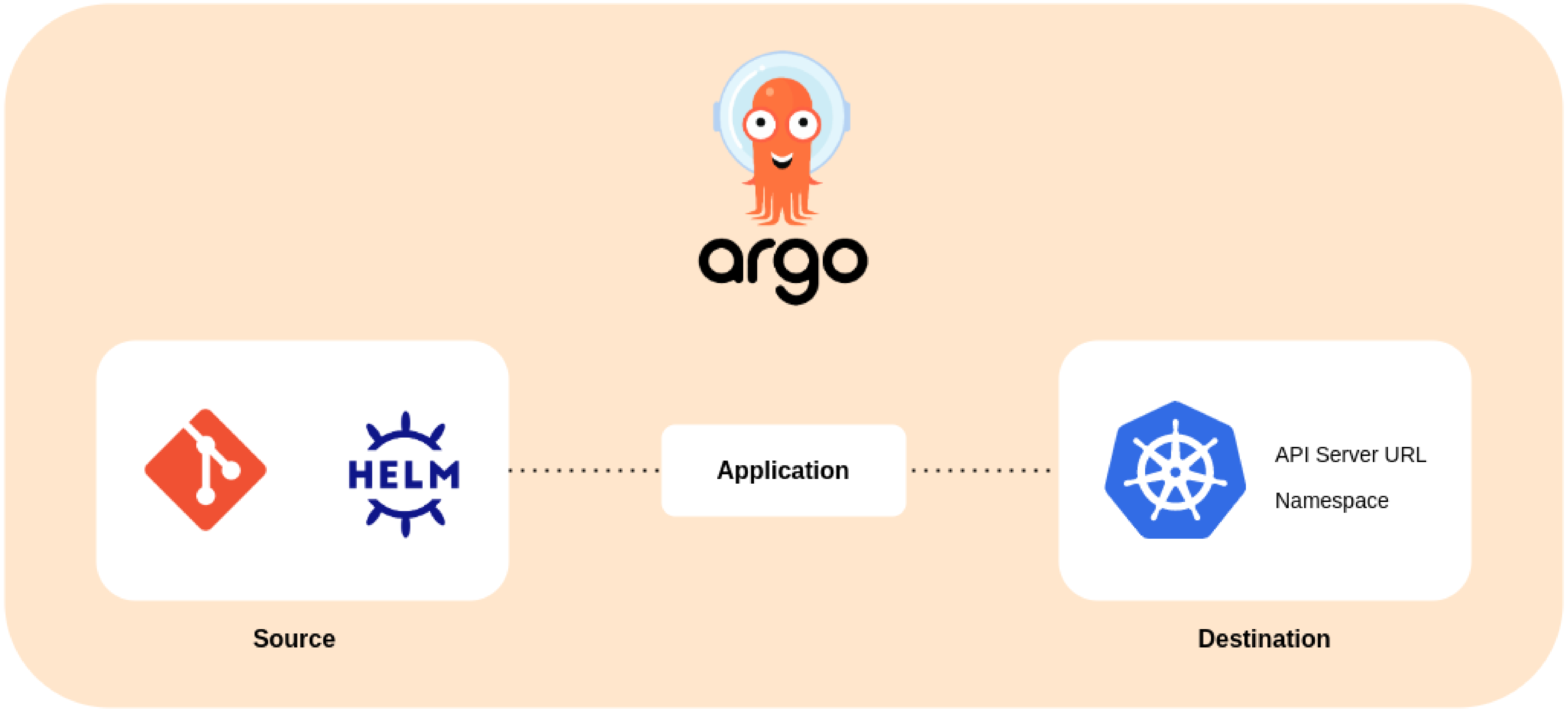
How will it fit with GitOps?

GitOps is a model that combines git and DevOps workflows.

Git works as the source of truth for the live state of your infrastructure.

In Kubernetes, a GitOps operator watches the git or helm repos as the desired state.

How will it fit with GitOps?

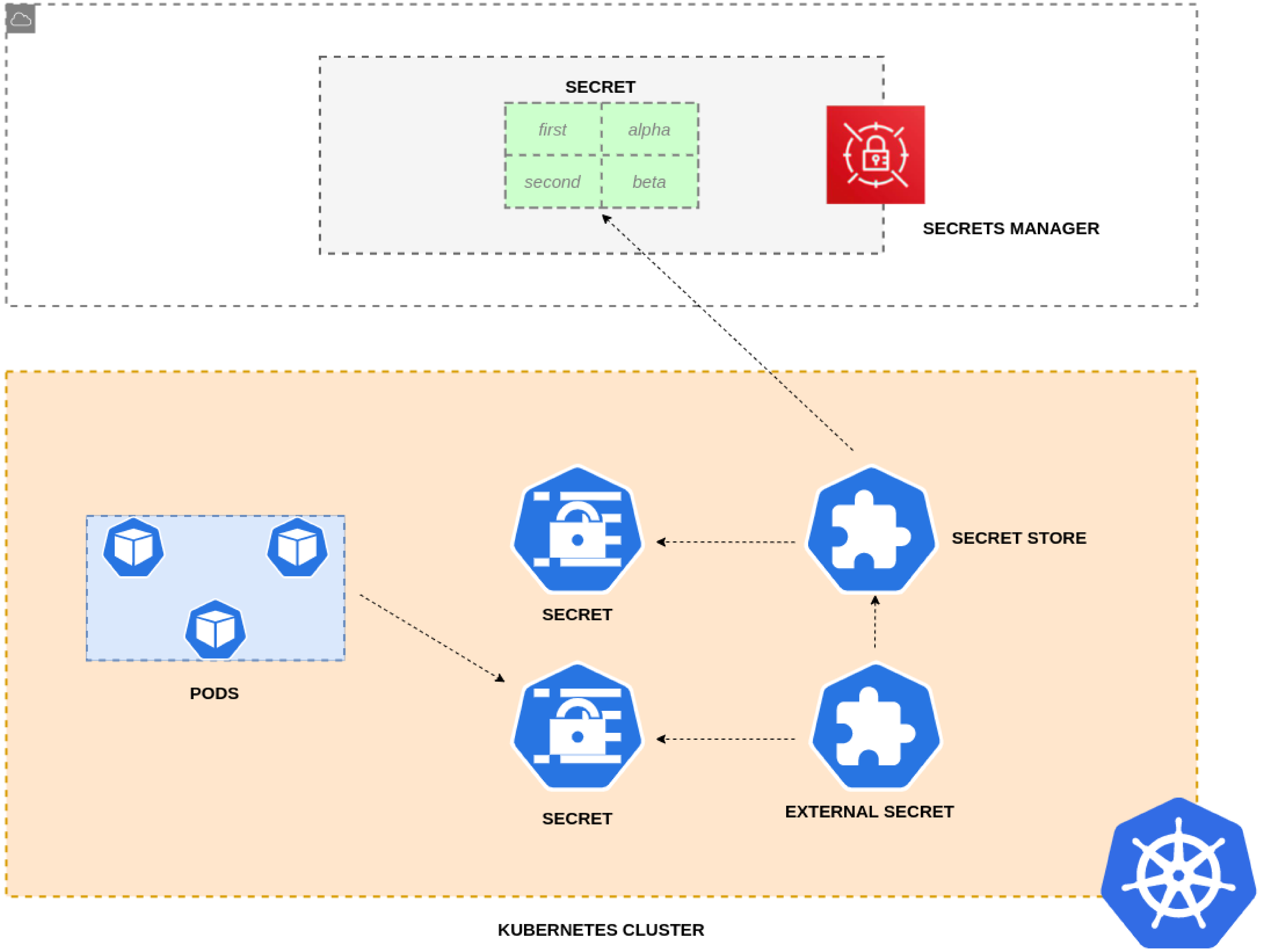


Secrets and git

- Secrets aren't encrypted
- Git repos are collaborative
- Can't apply fine-grained access-control
- Commit history

External Secrets Operator (ESO)

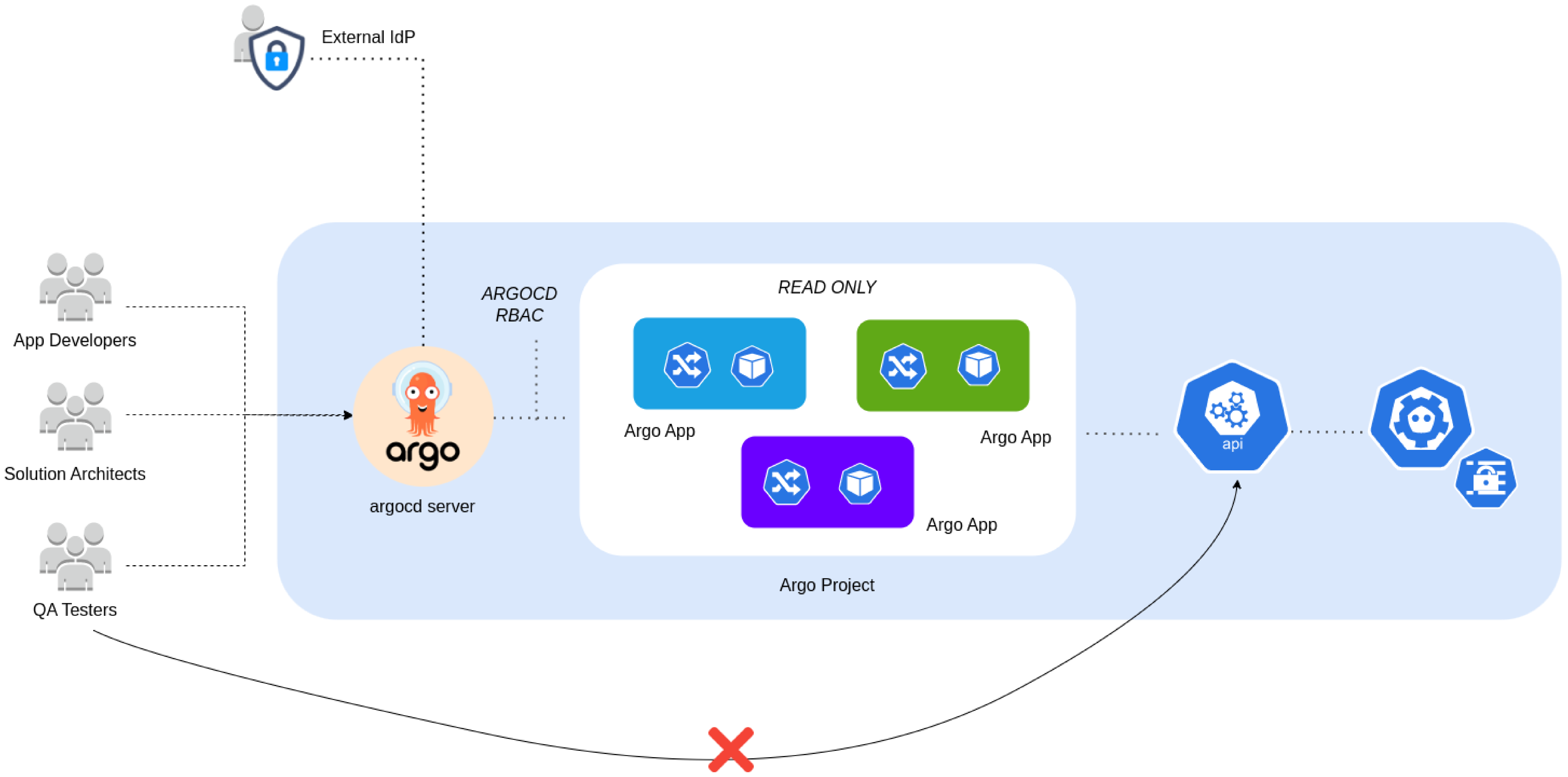
A K8s operator that is used to fetch values from external secrets managers and expose them as secrets in your cluster.


SECRET
first
alpha
second
beta
SECRETS MANAGER
SECRET
SECRET STORE
PODS
SECRET
EXTERNAL SECRET
KUBERNETES CLUSTER

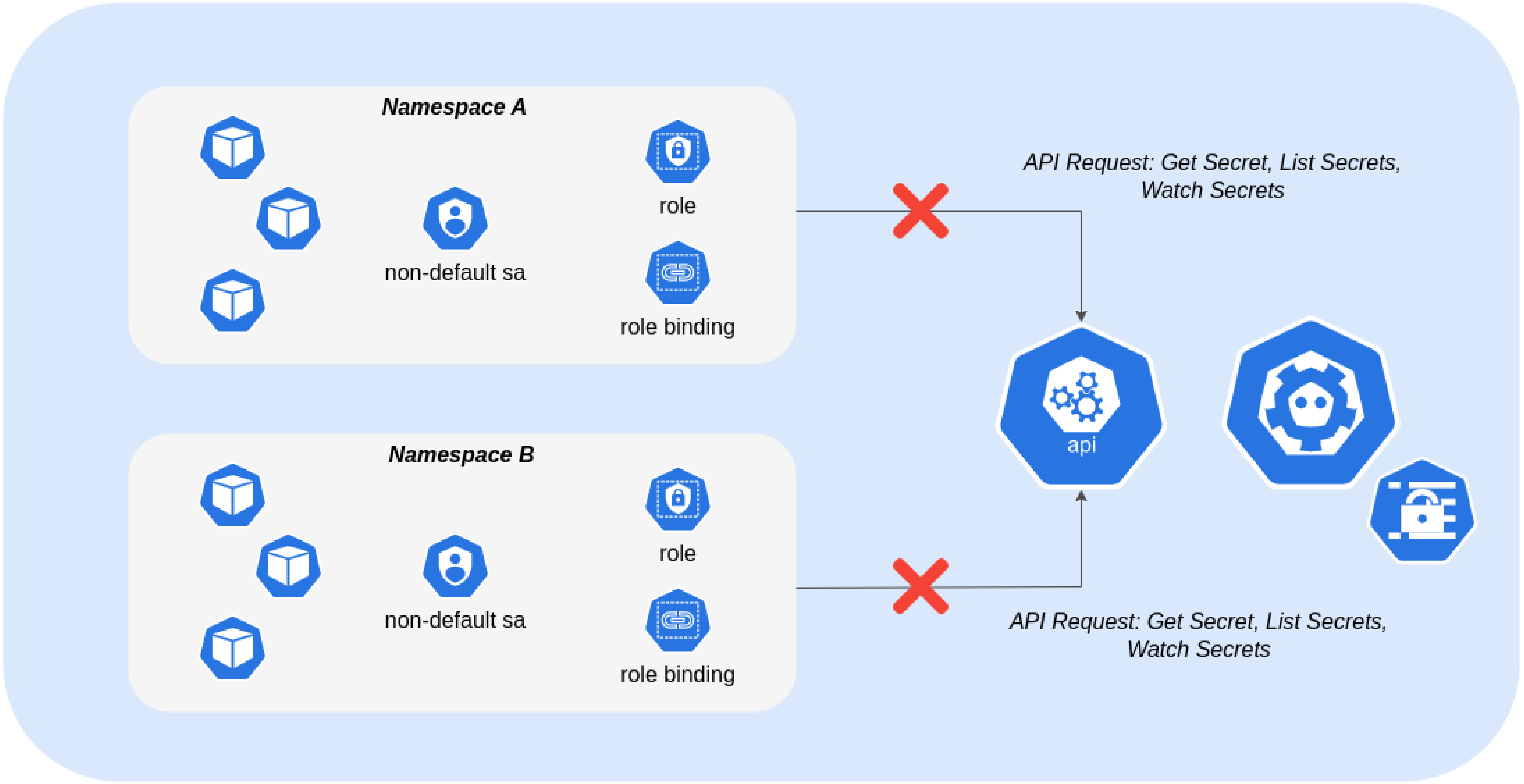
What about the target secret in *etcd*?

Who needs to know about the secret?

Protecting secrets from users

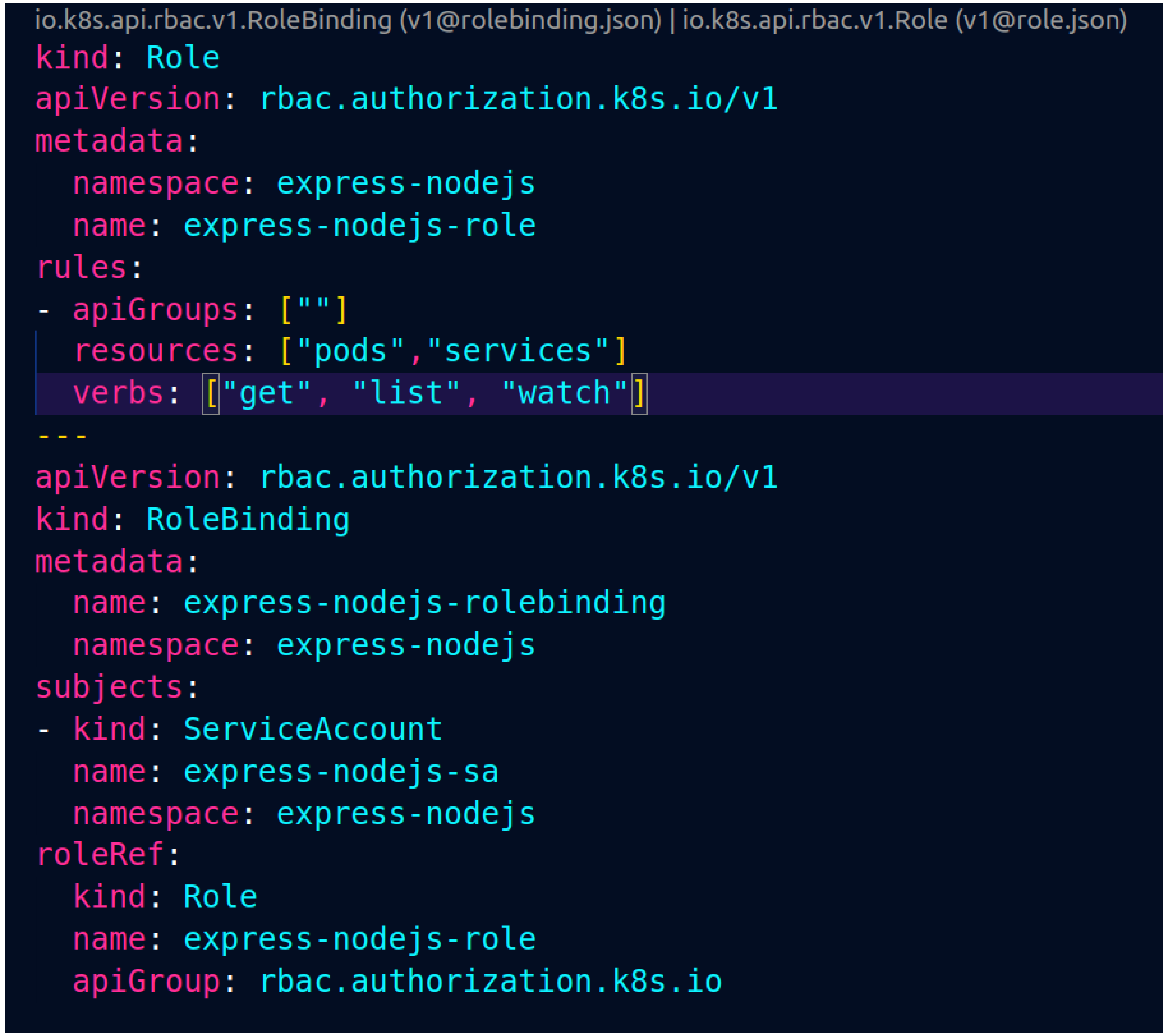


Protecting secrets from workloads



Namespace A
role
non-default sa
role binding
Namespace B
role
non-default sa
role binding
API Request: Get Secret, List Secrets, Watch Secrets
api
API Request: Get Secret, List Secrets, Watch Secrets

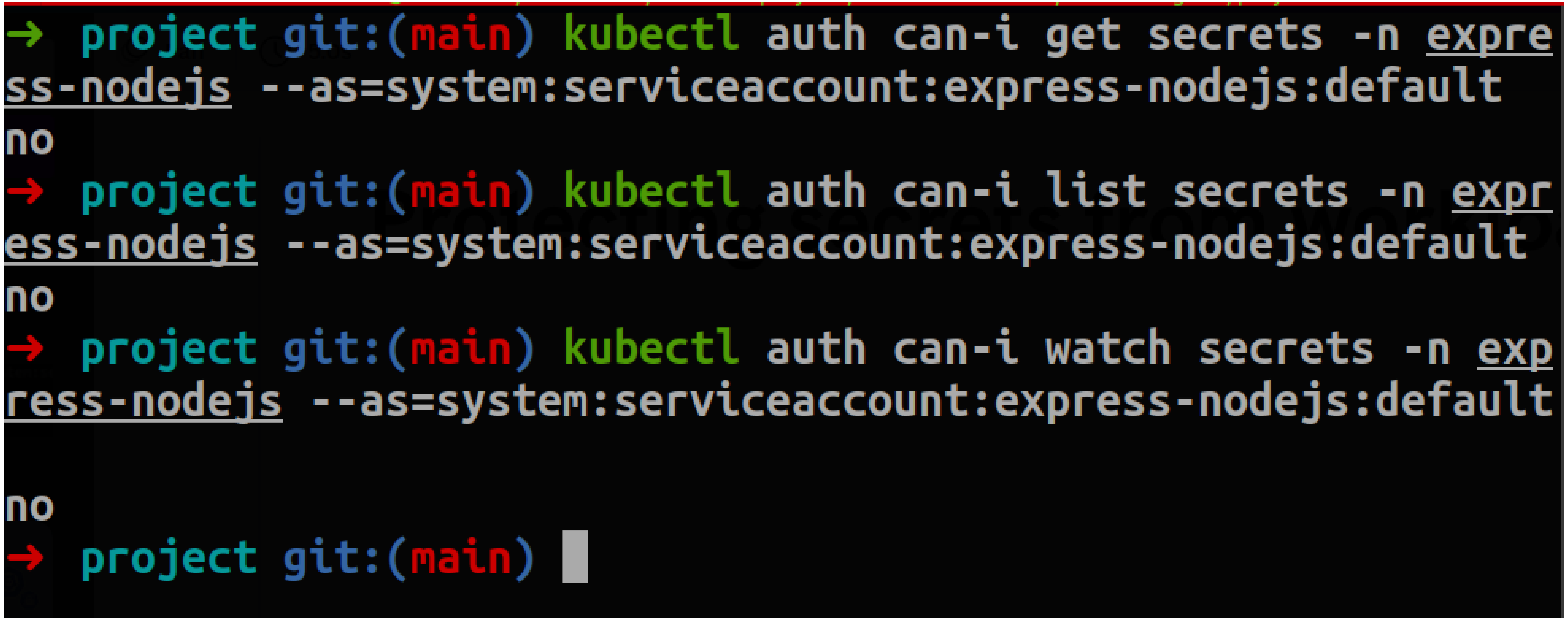
Assigning SAs and using RBAC



io.k8s.api.rbac.v1.RoleBinding (v1@rolebinding.json) | io.k8s.api.rbac.v1.Role (v1@role.json)

```yaml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: express-nodejs
  name: express-nodejs-role
rules:
- apiGroups: [""]
  resources: ["pods","services"]
  verbs: ["get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: express-nodejs-rolebinding
  namespace: express-nodejs
subjects:
- kind: ServiceAccount
  name: express-nodejs-sa
  namespace: express-nodejs
roleRef:
  kind: Role
  name: express-nodejs-role
  apiGroup: rbac.authorization.k8s.io
```

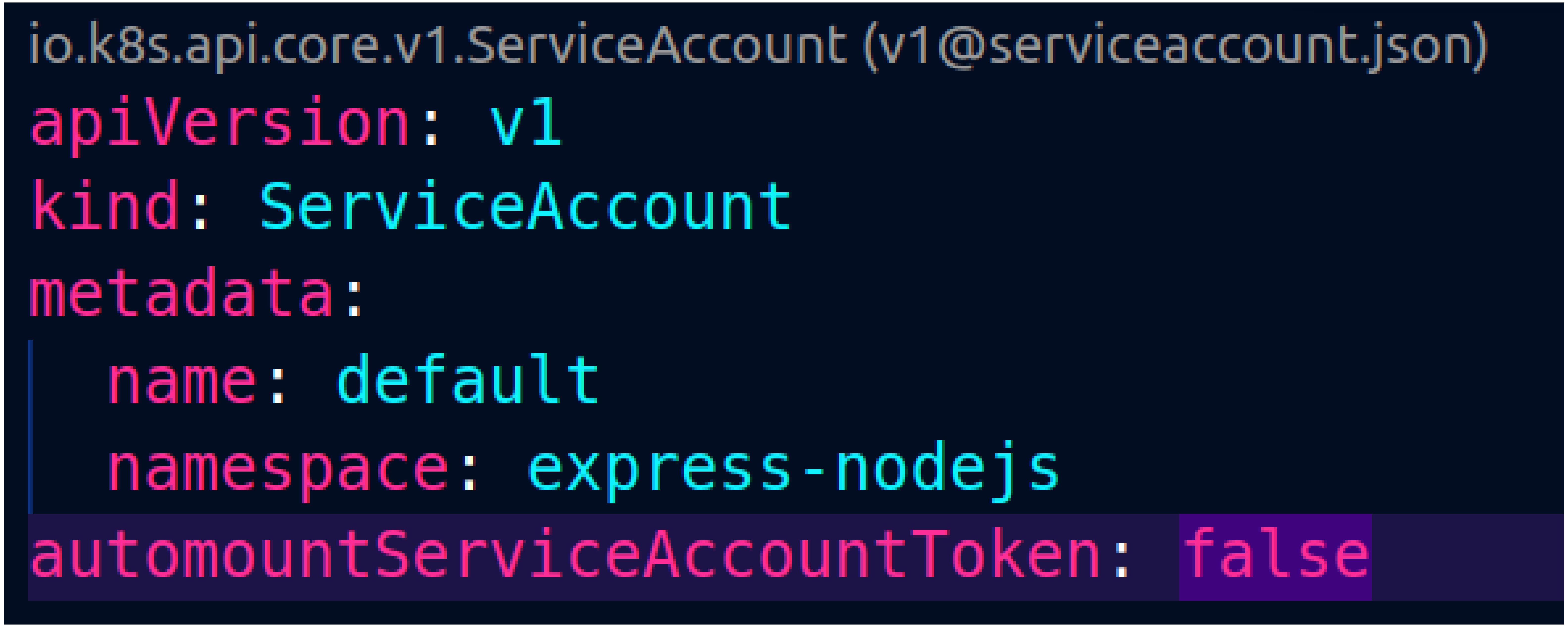
Protecting secrets from workloads



```
➜  project git:(main) kubectl auth can-i get secrets -n expre
ss-nodejs --as=system:serviceaccount:express-nodejs:default
no
➜  project git:(main) kubectl auth can-i list secrets -n expr
ess-nodejs --as=system:serviceaccount:express-nodejs:default
no
➜  project git:(main) kubectl auth can-i watch secrets -n exp
ress-nodejs --as=system:serviceaccount:express-nodejs:default

no
➜  project git:(main) 
```

Protecting secrets from workloads

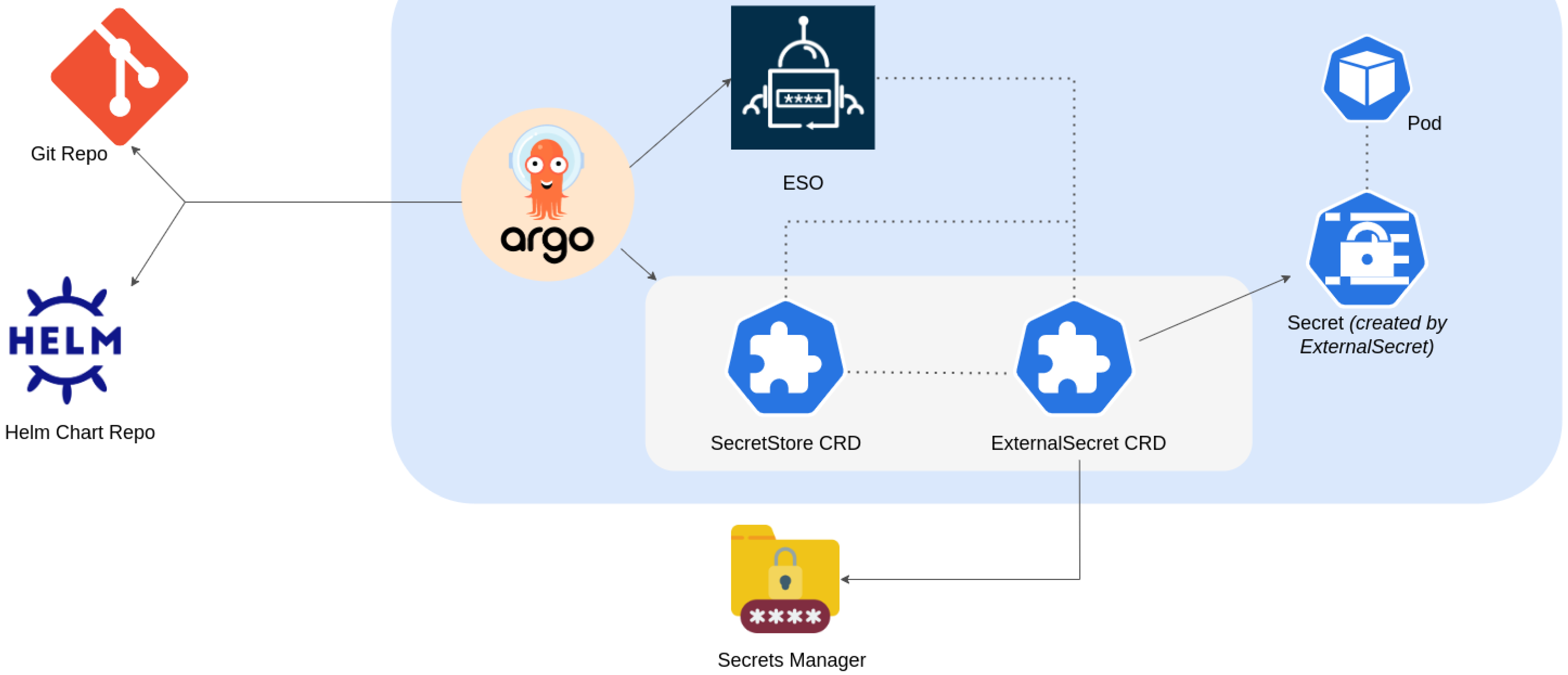


io.k8s.api.core.v1.ServiceAccount (v1@serviceaccount.json)

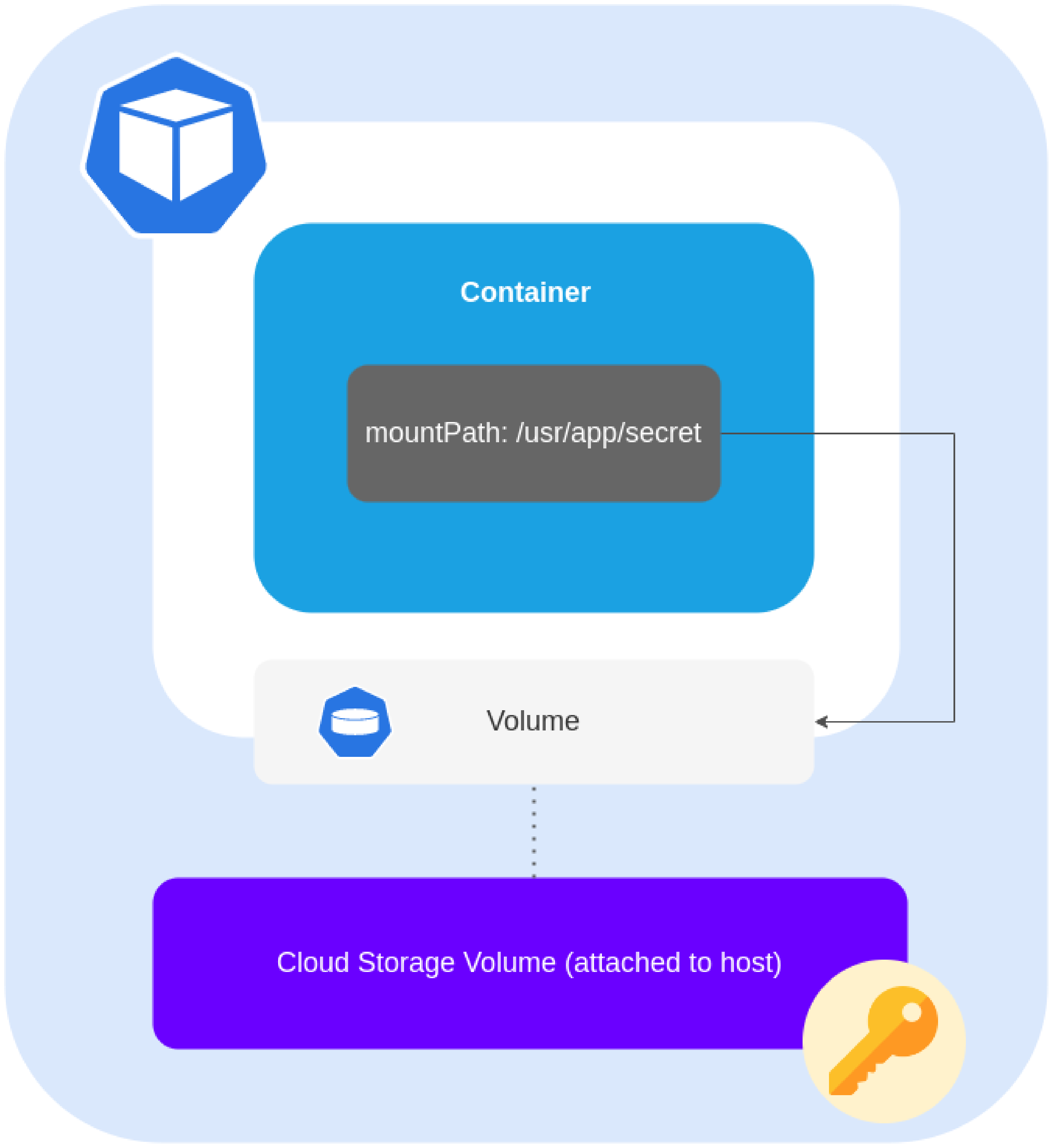
```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: default
  namespace: express-nodejs
automountServiceAccountToken: false
```

How is the secret shared and consumed?

Sharing the secret

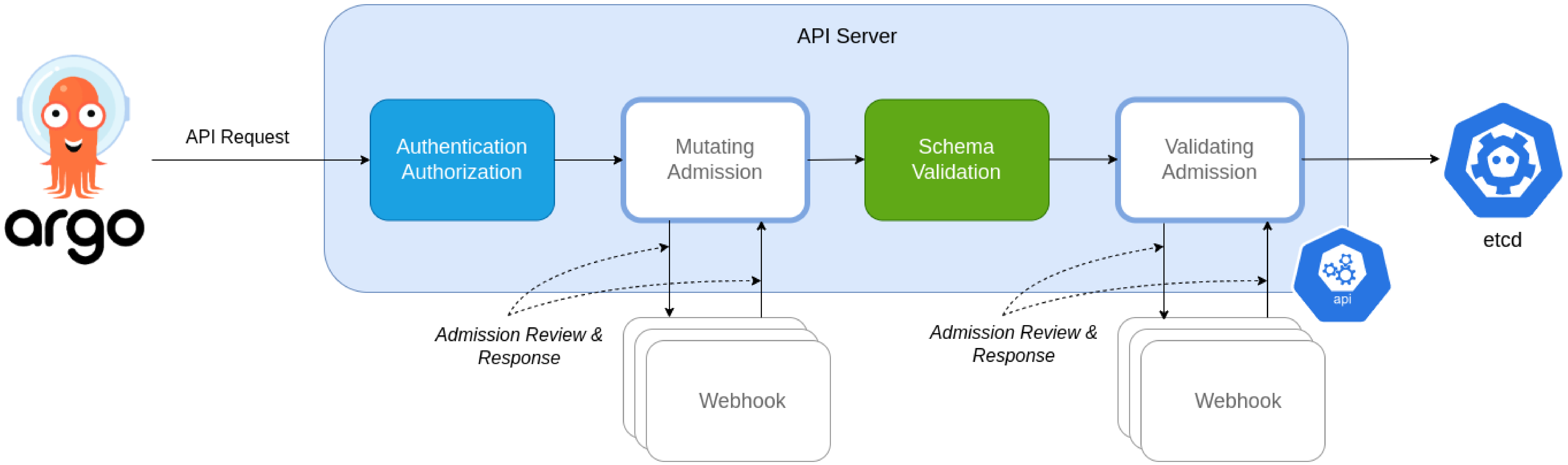


Consuming the secret

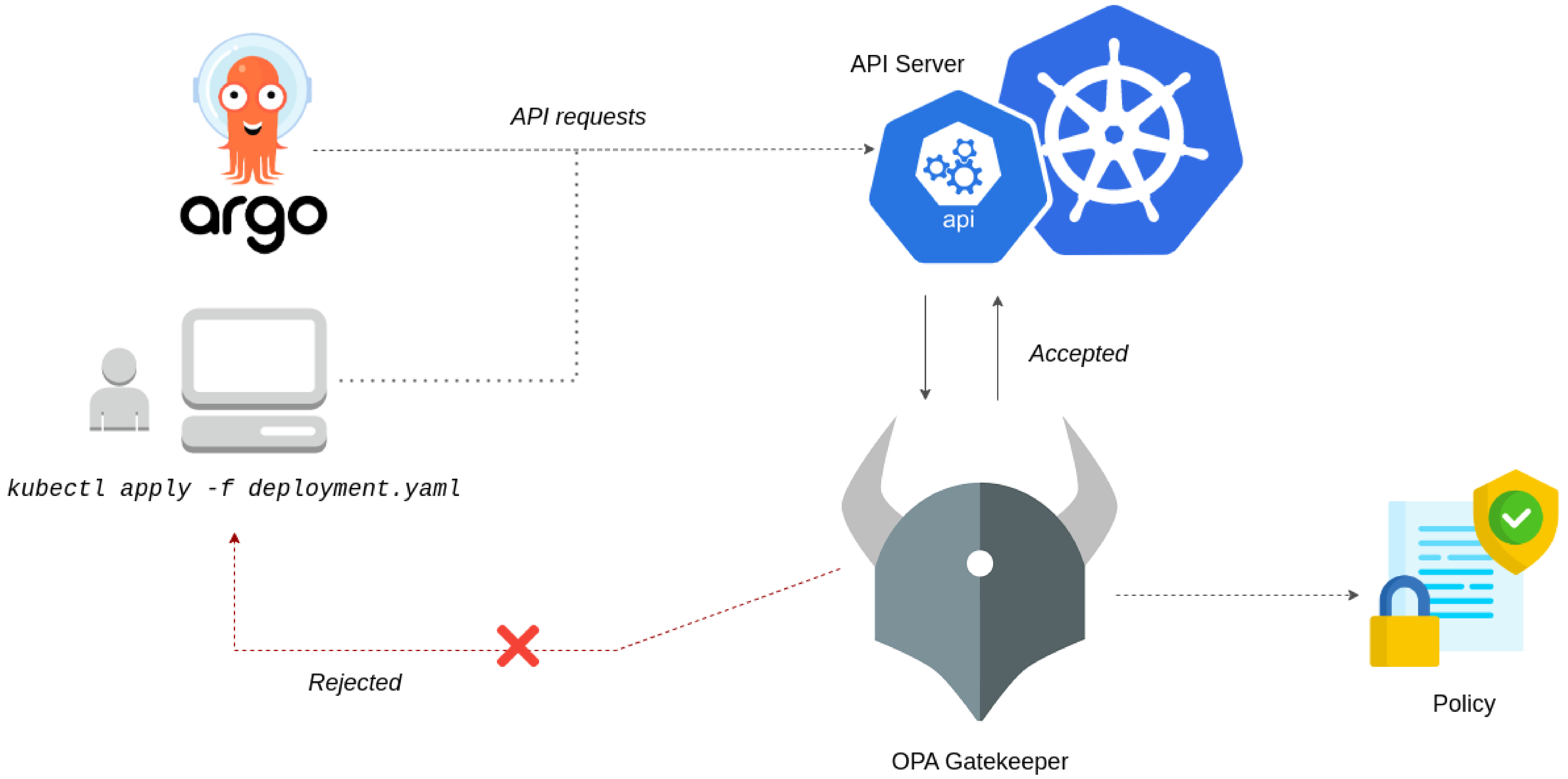

Container
mountPath: /usr/app/secret
Volume
Cloud Storage Volume (attached to host)

How do we prevent violations that risk secret exposure?

Prevent violations with admission controllers



Prevent violations with admission controllers



Let's see this in action…