KubeCon | CloudNativeCon

Europe 2020

Virtual

KEEP CLOUD NATIVE CONNECTED

KubeCon | CloudNativeCon

Europe 2020

Virtual

Kubernetes Secrets in GitOps Workflows

Alexandr Tcherniakhovski** and **Seth Vargo

About us

Seth Vargo

Advocate & Product Manager

twitter.com/sethvargo

Alexandr Tcherniakhovski

Software Engineer

linkedin.com/in/atcherniakhovski

We have a problem...

```
kubectl create secret
```

We have a problem...

Who?

When?

Why?

```
kubectl create secret
```

We have a problem...

Who?

When?

Why?

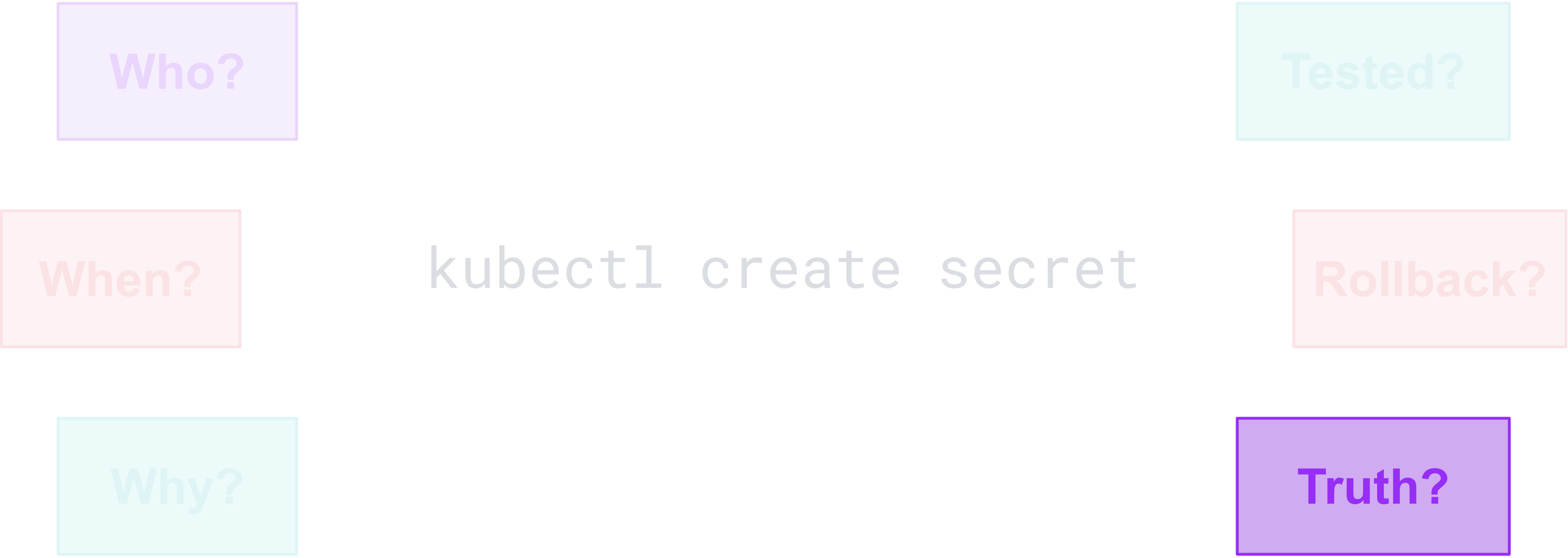
```
kubectl create secret
```

Tested?

Rollback?

Truth?

We have a problem...



Who?

When?

Why?

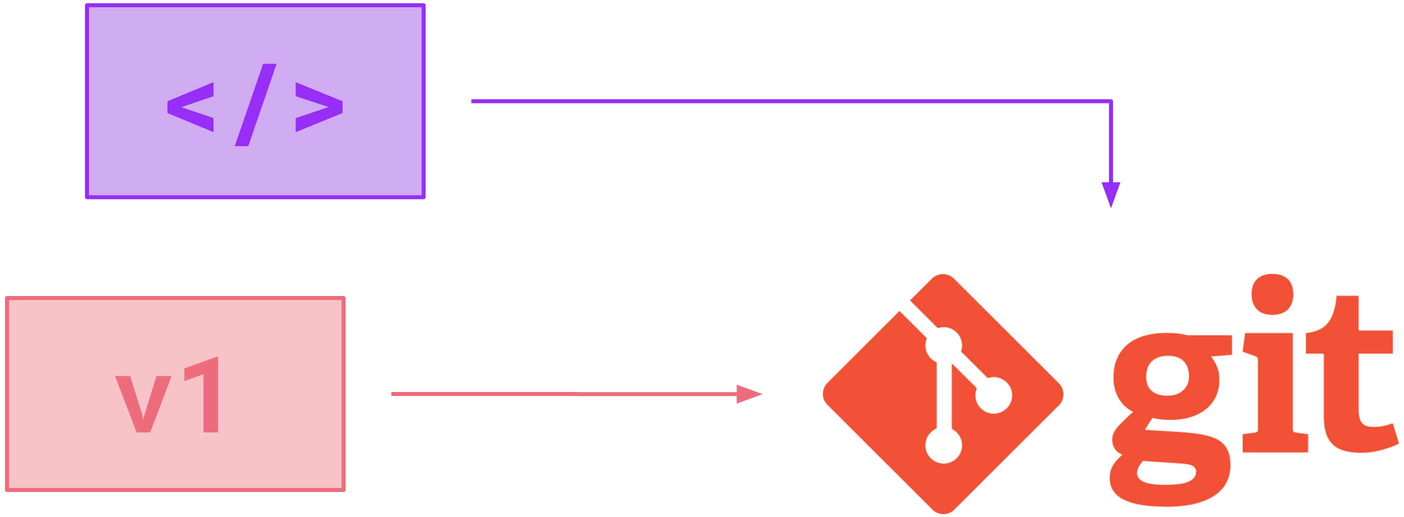
```
kubectl create secret
```

Tested?

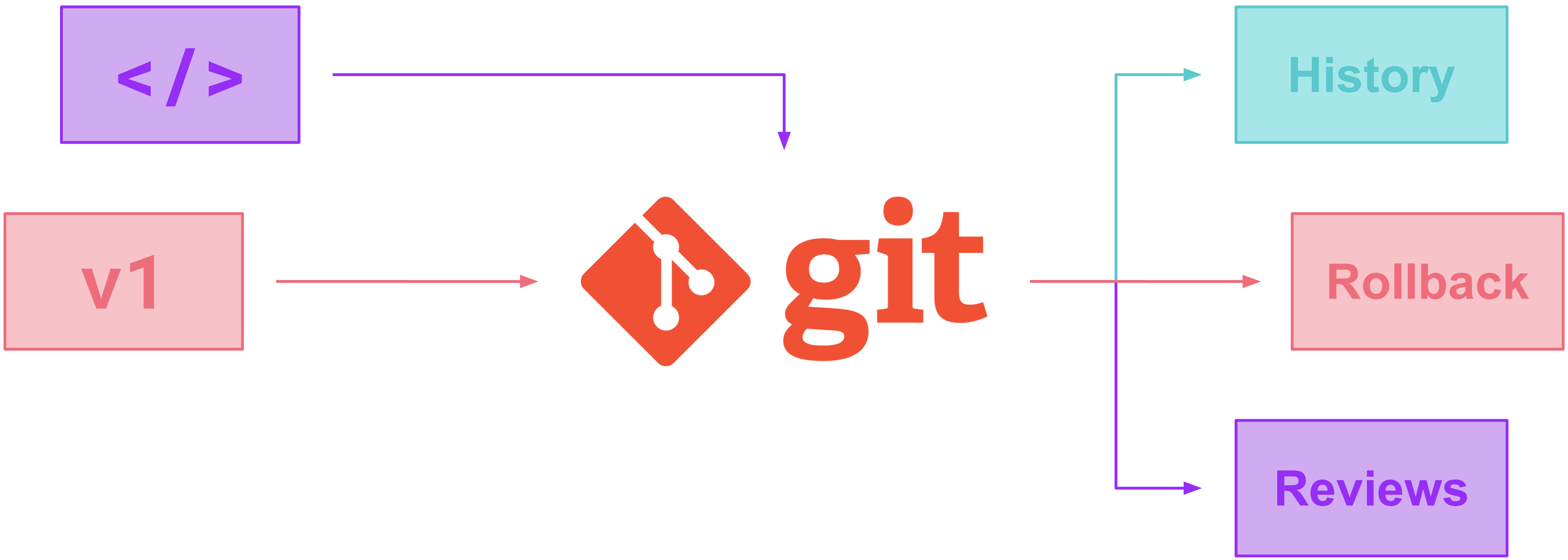
Rollback?

Truth?

What about git?



What about git?



</>

v1

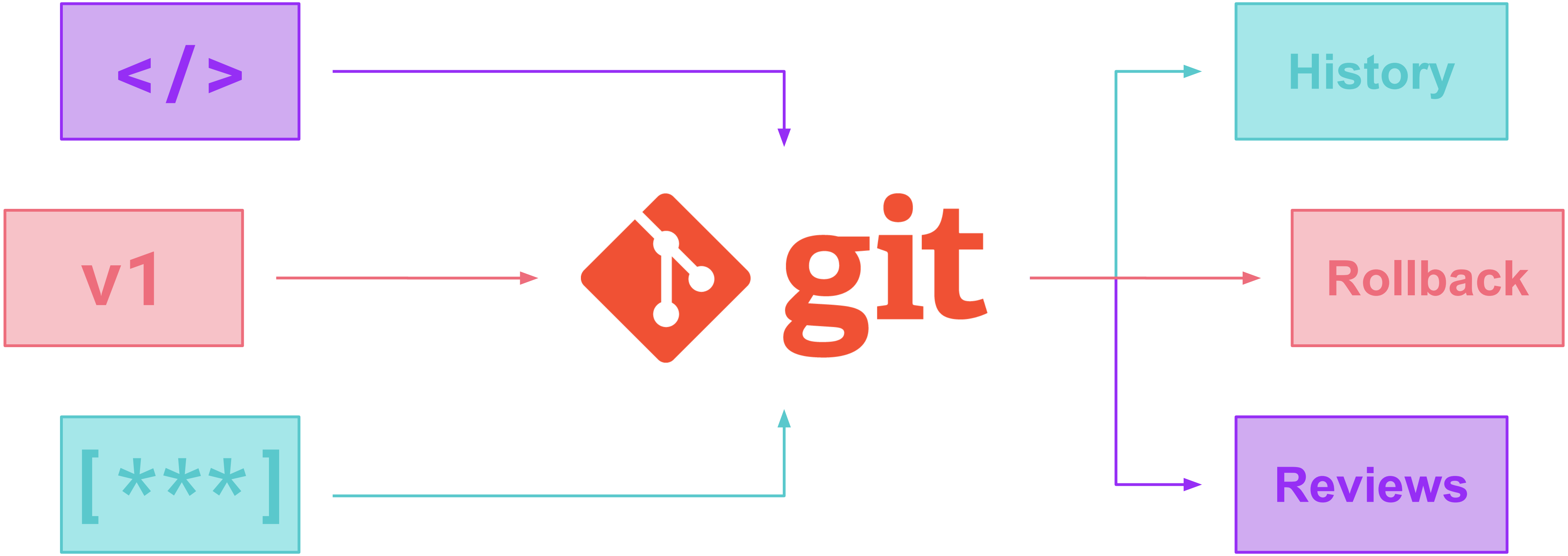
git

History

Rollback

Reviews

What about git?



Are you for real?

"I would never put plaintext secrets in my git repository - that's a privilege escalation just waiting to happen!"

Are you for real?

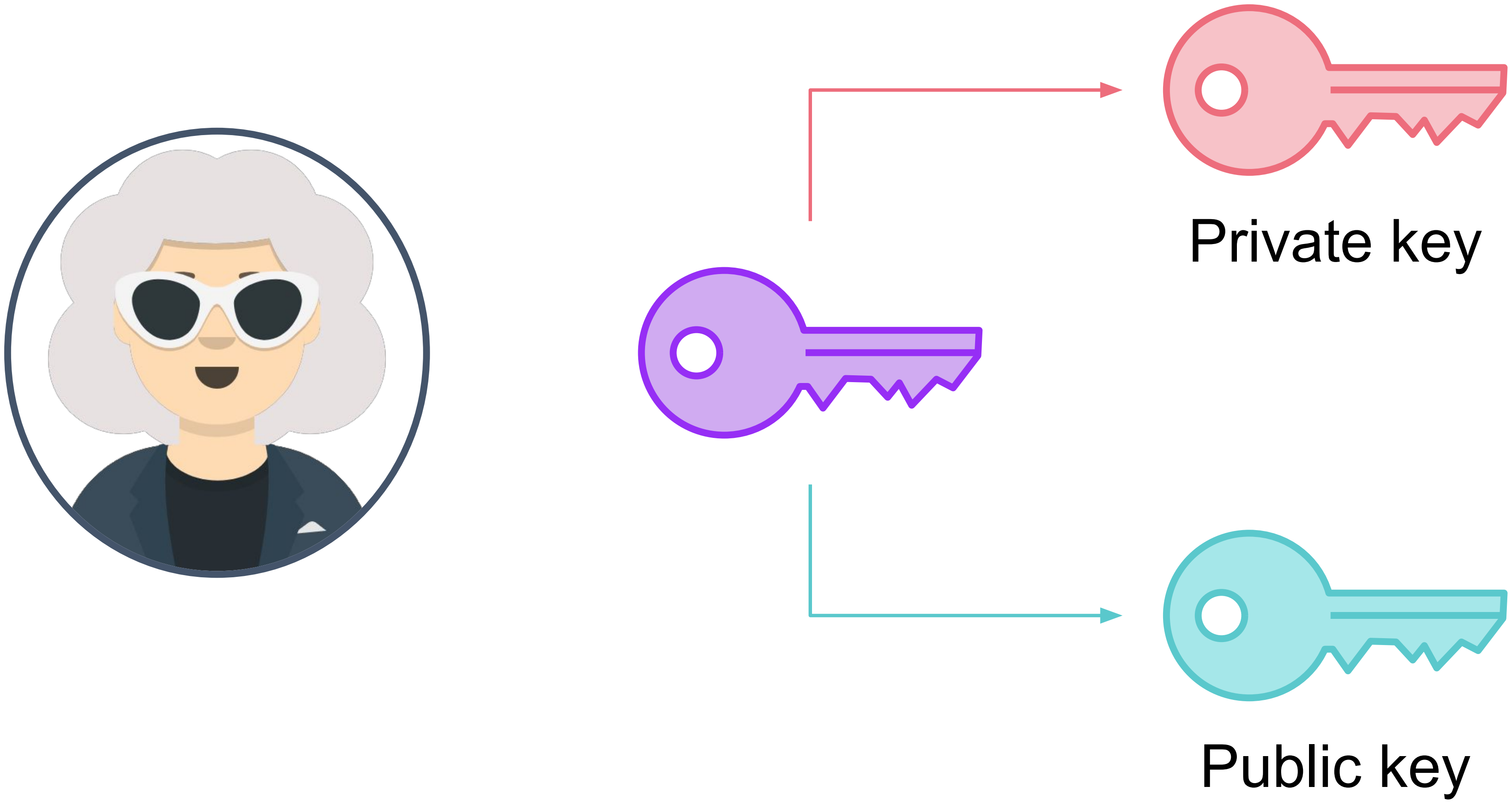
"I would never put ~~plaintext~~ secrets in my git repository - that's a privilege escalation just waiting to happen!"

I'm not using git

That's okay - this pattern works for other source control systems too.

We're just using git as an example.

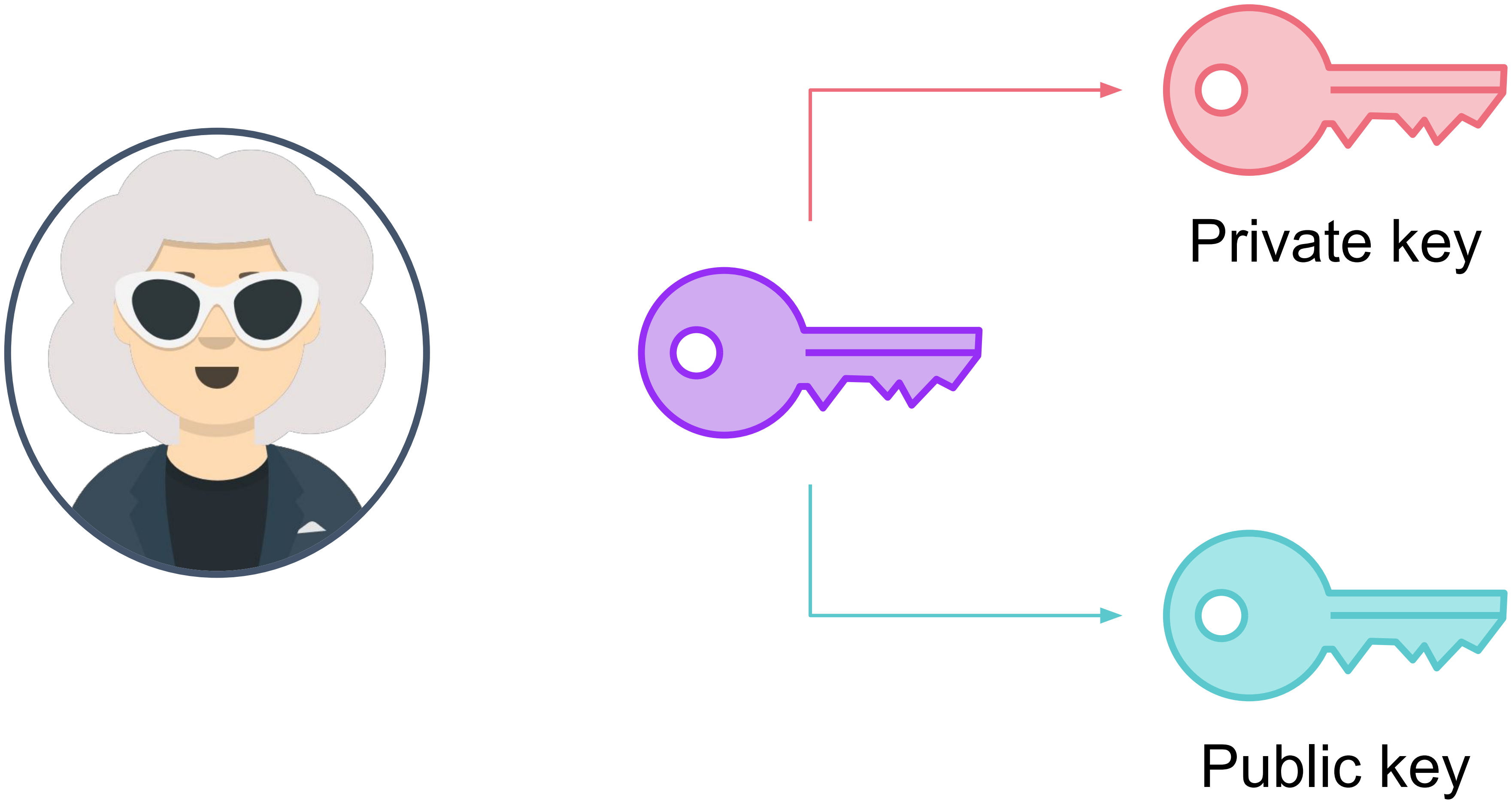
Asymmetric cryptography



Private key

Public key

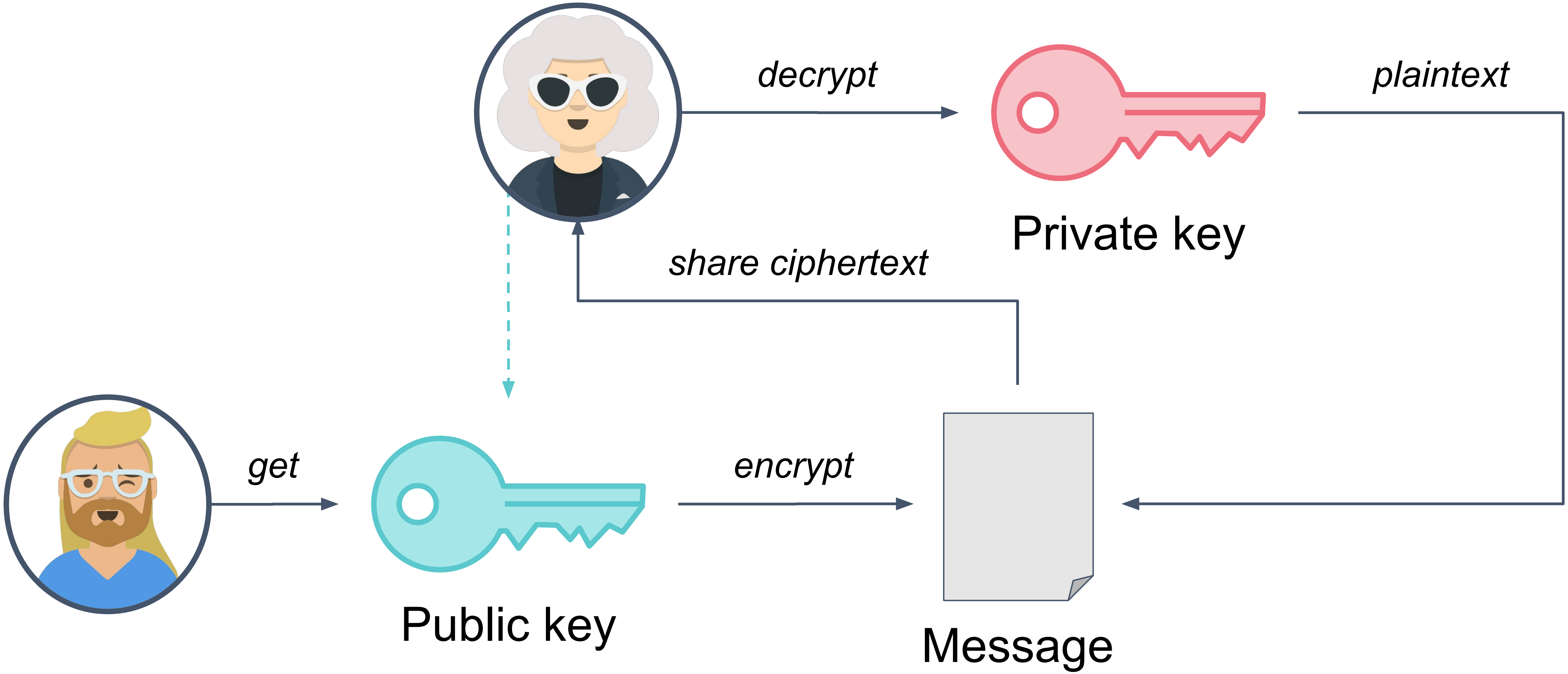
Asymmetric cryptography



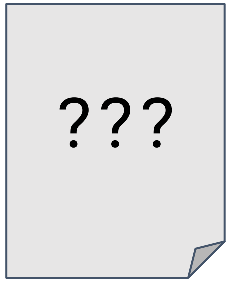
Private key

Public key

Asymmetric cryptography



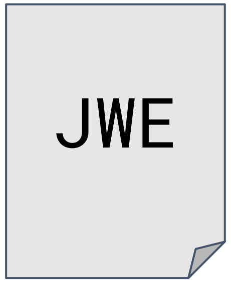
Message format



Message

JSON Web Encryption (JWE)

https://tools.ietf.org/html/rfc7516



Message

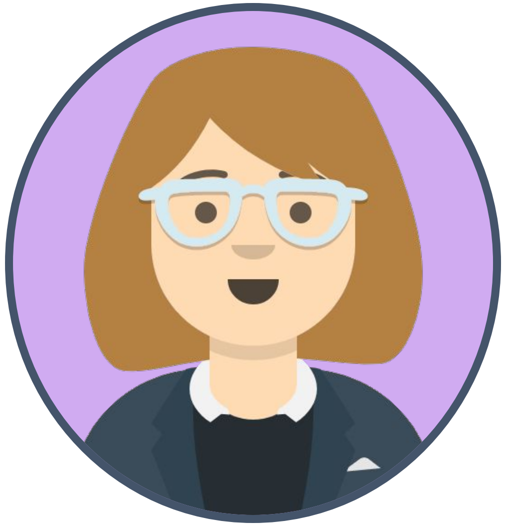
Why use an envelope?

Kubernetes secrets can be up to 1Mb, but most KMS systems are limited to 64Kb. The envelope gives us flexibility to encrypt larger payloads.

Giving the secret directly to a KMS exposes the secret to the KMS. Depending on your trust model, this might be unacceptable.

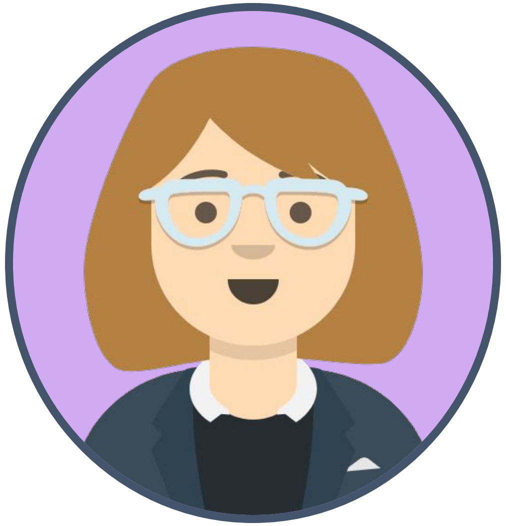
Personas

Personas



Key Admin

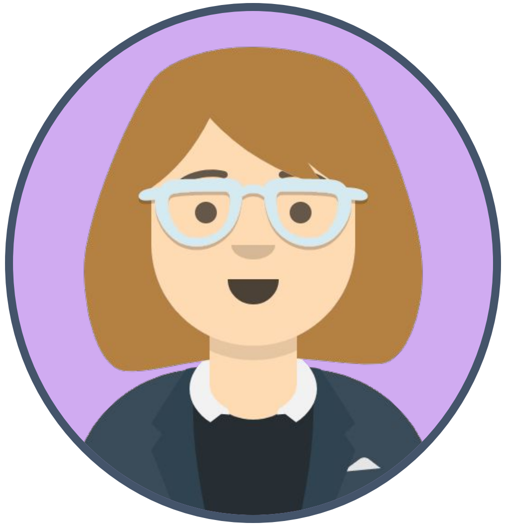
Personas



Key Admin

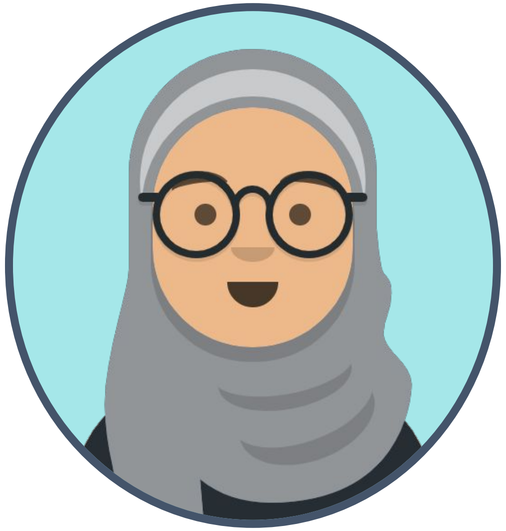
Secret Admin

Personas



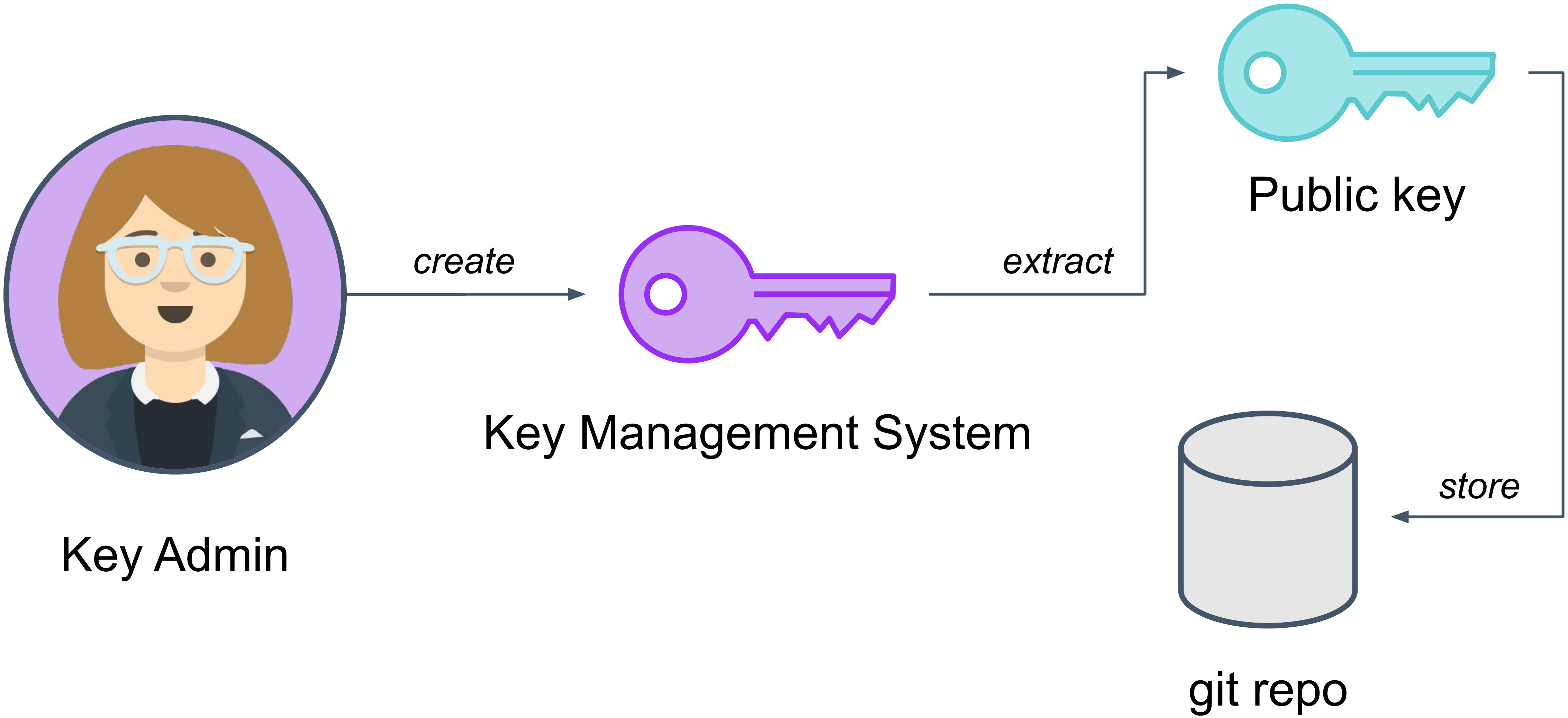
Key Admin

Secret Admin

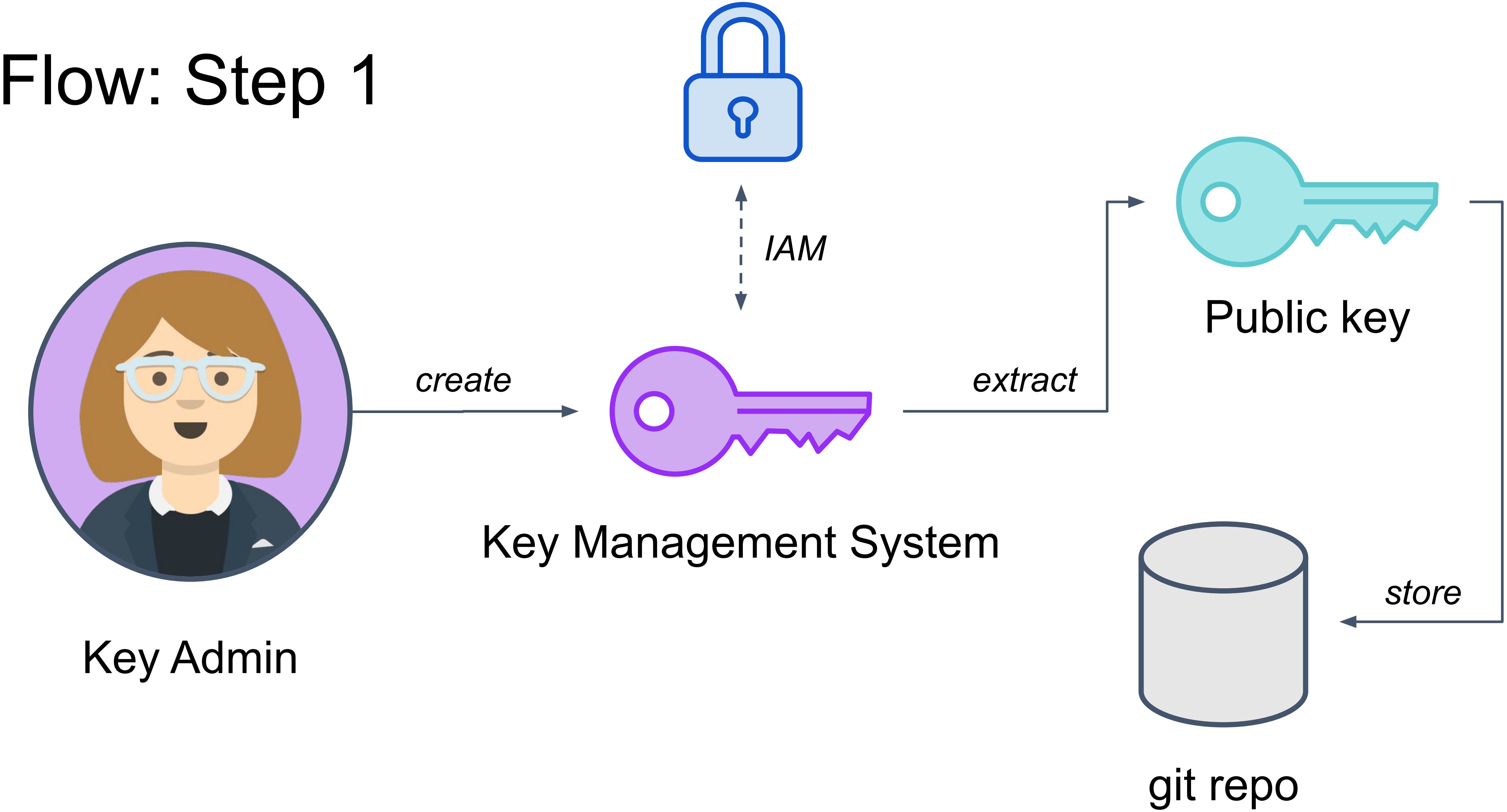


Cluster Admin

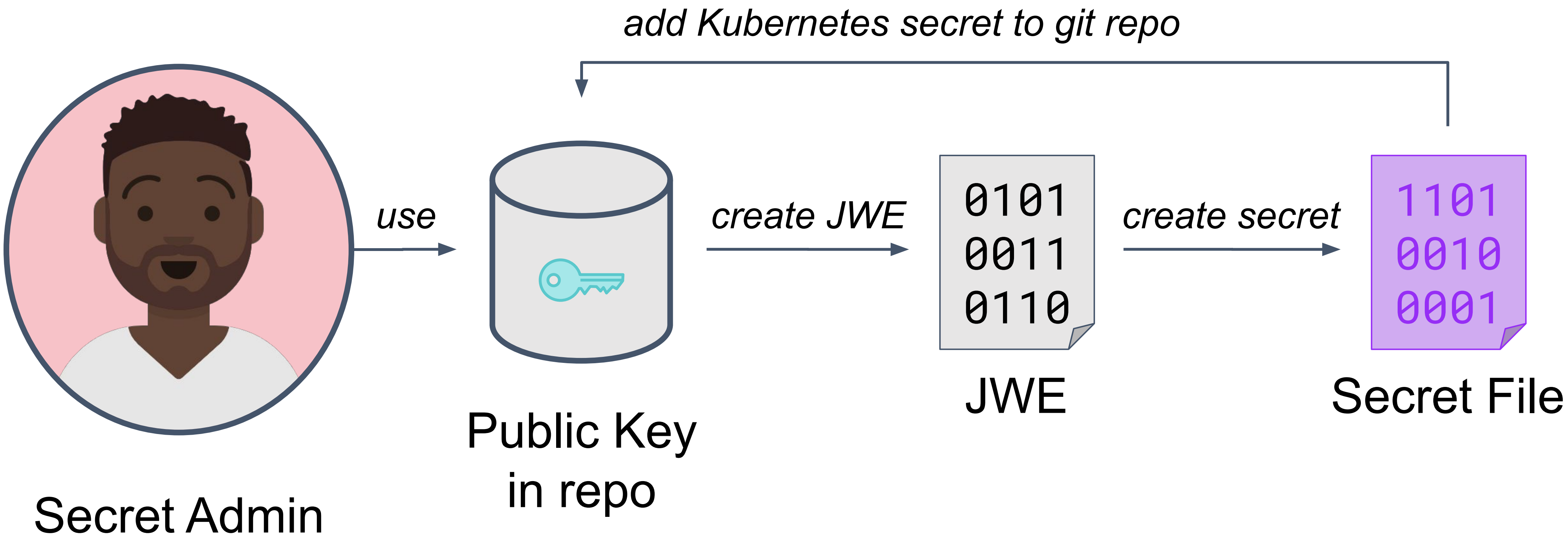
Flow: Step 1


create
extract
Public key
store
Key Admin
Key Management System
git repo

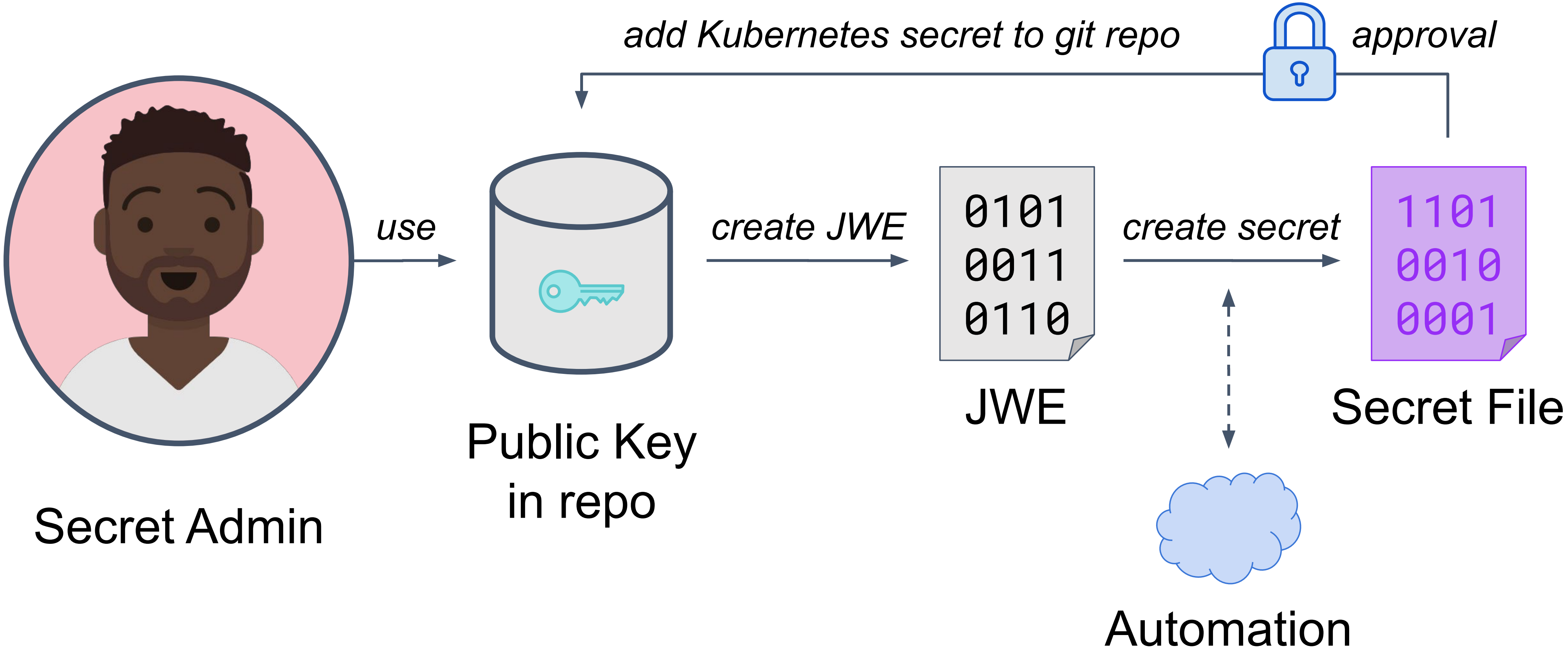
Flow: Step 1


IAM
create
extract
Public key
Key Admin
Key Management System
store
git repo

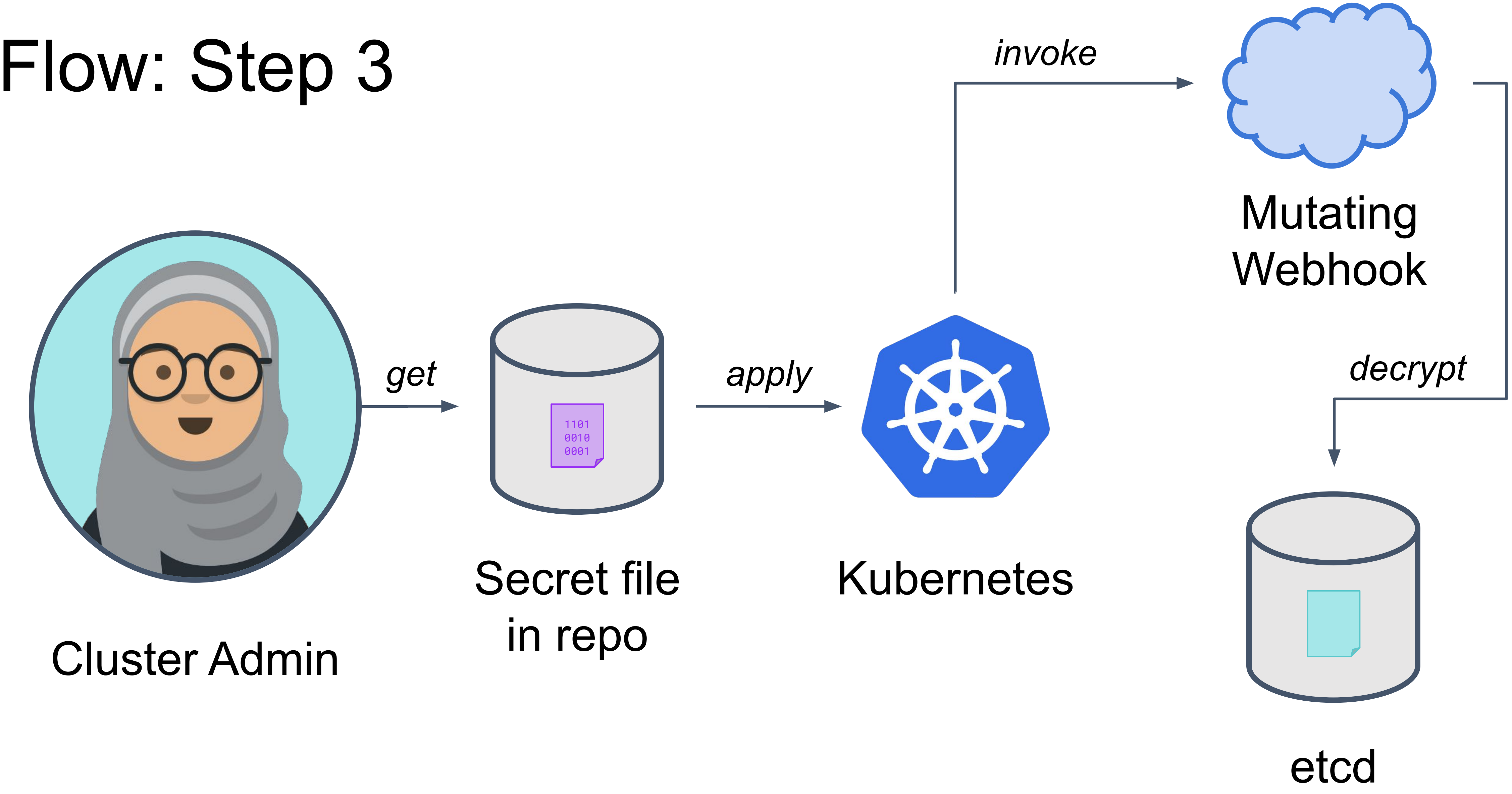
Flow: Step 2


add Kubernetes secret to git repo
use
create JWE
create secret
0101
0011
0110
1101
0010
0001
Secret Admin
Public Key
in repo
JWE
Secret File

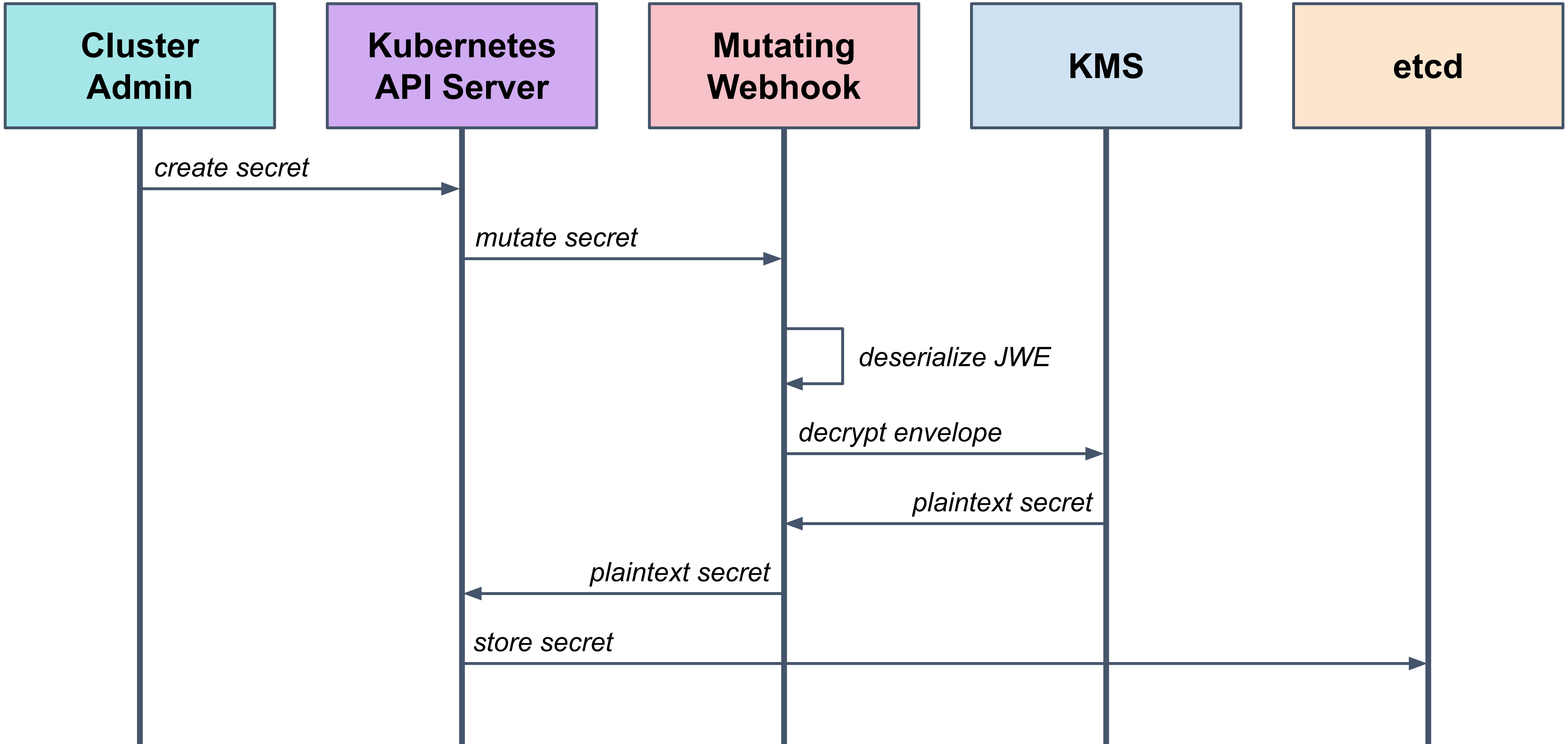
Flow: Step 2


add Kubernetes secret to git repo
approval
use
create JWE
create secret
0101
0011
0110
1101
0010
0001
Secret Admin
Public Key
in repo
JWE
Secret File
Automation

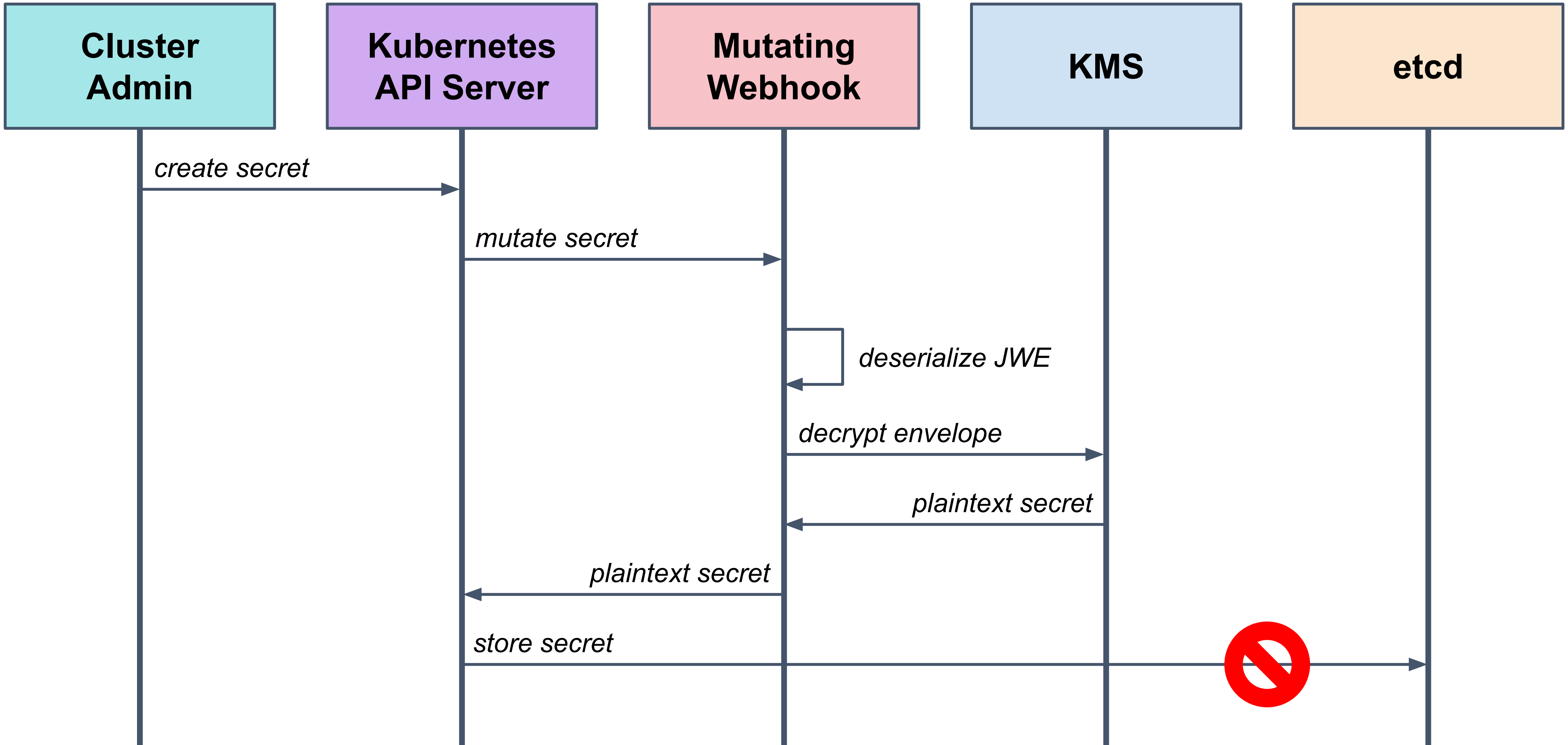
Flow: Step 3


get
1101
0010
0001
apply
invoke
Mutating
Webhook
decrypt
Cluster Admin
Secret file
in repo
Kubernetes
etcd

Flow: Step 3


Cluster Admin
Kubernetes API Server
Mutating Webhook
KMS
etcd
create secret
mutate secret
deserialize JWE
decrypt envelope
plaintext secret
plaintext secret
store secret

Flow: Step 3

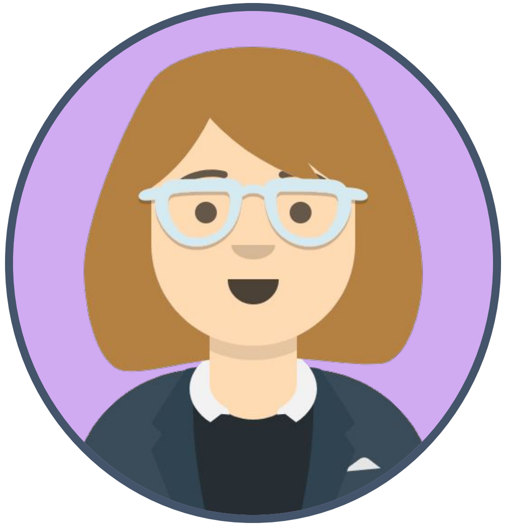

Cluster Admin
Kubernetes API Server
Mutating Webhook
KMS
etcd
create secret
mutate secret
deserialize JWE
decrypt envelope
plaintext secret
plaintext secret
store secret

Walkthrough

Environment

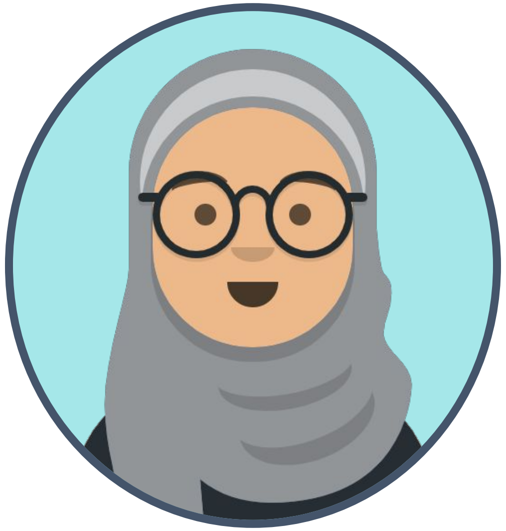
- Google Cloud KMS
- Google Compute Engine (GCE)

Personas



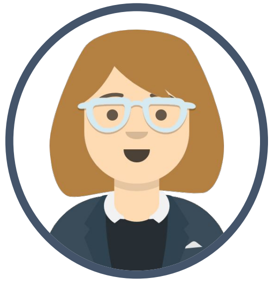
Key Admin

Secret Admin



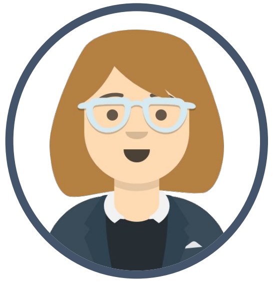
Cluster Admin

Key Admin responsibilities



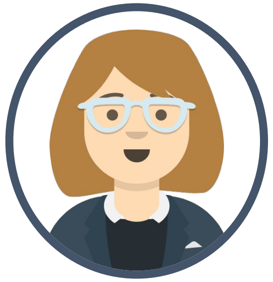
- Create an asymmetric key
- Export the public key and push it to git
- Apply access controls on the key

Create asymmetric key



```
gcloud kms keys create "my-key" \
  --location "us-east1" \
  --keyring "my-ring" \
  --purpose "asymmetric-encryption" \
  --default-algorithm "rsa-decrypt-oaep-4096-sha256"
```

Key standards



RSA

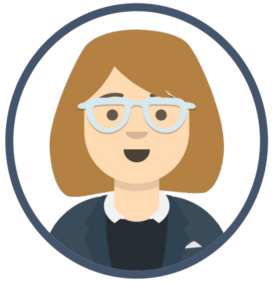
"alg" Param Value	Key Management Algorithm
RSA-OAEP	RSAES OAEP using default parameters
RSA-OAEP-256	RSA OAEP using SHA-256 and MGF1 with SHA-256

EC

"alg" Param Value	Key Management Algorithm
ECDH-ES+A128KW	ECDH-ES using Concat KDF and CEK wrapped with A128KW
ECDH-ES+A192KW	ECDH-ES using Concat KDF and CEK wrapped with A192KW
ECDH-ES+A256KW	ECDH-ES using Concat KDF and CEK wrapped with A256KW

https://tools.ietf.org/html/rfc7518

Key standards



RSA

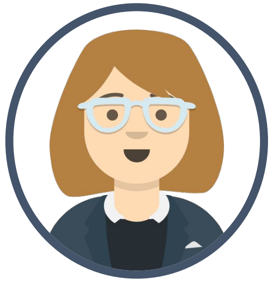
"alg" Param Value	Key Management Algorithm
RSA-OAEP	RSAES OAEP using default parameters
RSA-OAEP-256	RSA OAEP using SHA-256 and MGF1 with SHA-256

EC

"alg" Param Value	Key Management Algorithm
ECDH-ES+A128KW	ECDH-ES using Concat KDF and CEK wrapped with A128KW
ECDH-ES+A192KW	ECDH-ES using Concat KDF and CEK wrapped with A192KW
ECDH-ES+A256KW	ECDH-ES using Concat KDF and CEK wrapped with A256KW

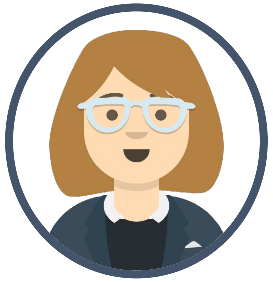
https://tools.ietf.org/html/rfc7518

Export the public key



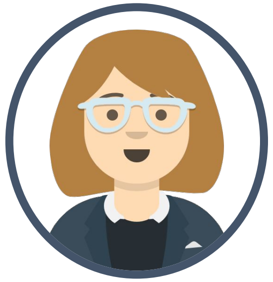
```
gcloud kms keys versions get-public-key "1" \
  --location "us-east1" \
  --keyring "my-ring" \
  --key "my-key" \
  --output-file "/tmp/key.pub"
```

Grant decrypt privileges



```
gcloud kms keys add-iam-policy-binding "my-key" \
  --location "us-east1" \
  --keyring "my-ring" \
  --member "serviceAccount:${SA_EMAIL}" \
  --role "roles/cloudkms.cryptoKeyDecrypter"
```

Key Admin responsibilities



- ✓ Create an asymmetric key
- ✓ Export the public key and push it to git
- ✓ Apply access controls on the key

Secret Admin responsibilities

- Encrypt the Credential (e.g. password)

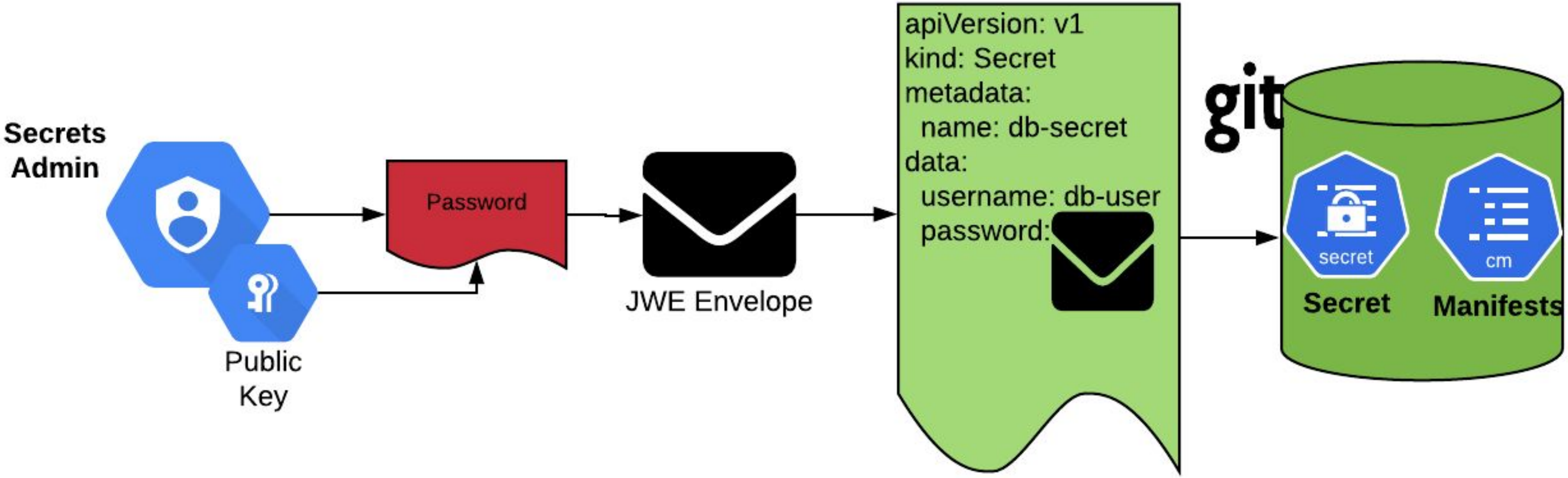
Create JWE Envelope

```
JWE=$(echo "P@ssw0rd" | jose-util encrypt --full \
  --key "/tmp/key.pub" --alg "RSA-OAEP-256" \
  --enc "A128CBC-HS256")

cat > encrypted-secret-k8s.yaml <<EOF
kind: Secret
stringData:
  password: ${JWE}
EOF
```

github.com/square/go-jose/tree/master/jose-util

Flow



```
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
data:
  username: db-user
  password:
```

Full secret

```
kind: Secret
stringData:
  password: |-
    {
      "protected": "eyJhbGci0ij...",
      "encrypted_key": "W55XJrzI_rxTnBBtMK5Al...",
      "iv": "ZKD4DLUYJVhG9T8xnSnMEQ",
      "ciphertext": "JKLXYc7C9...",
      "tag": "iZnl_VDdjMPv6..."
    }
```

Cluster Admin responsibilities

- Setup JWE Mutating Webhook

Cluster Admin

```
gcloud compute instances create "my-webook" \
  --service-account "${SA_EMAIL}" \
  --scopes "cloud-platform"
```

Config for the Mutating Webhook

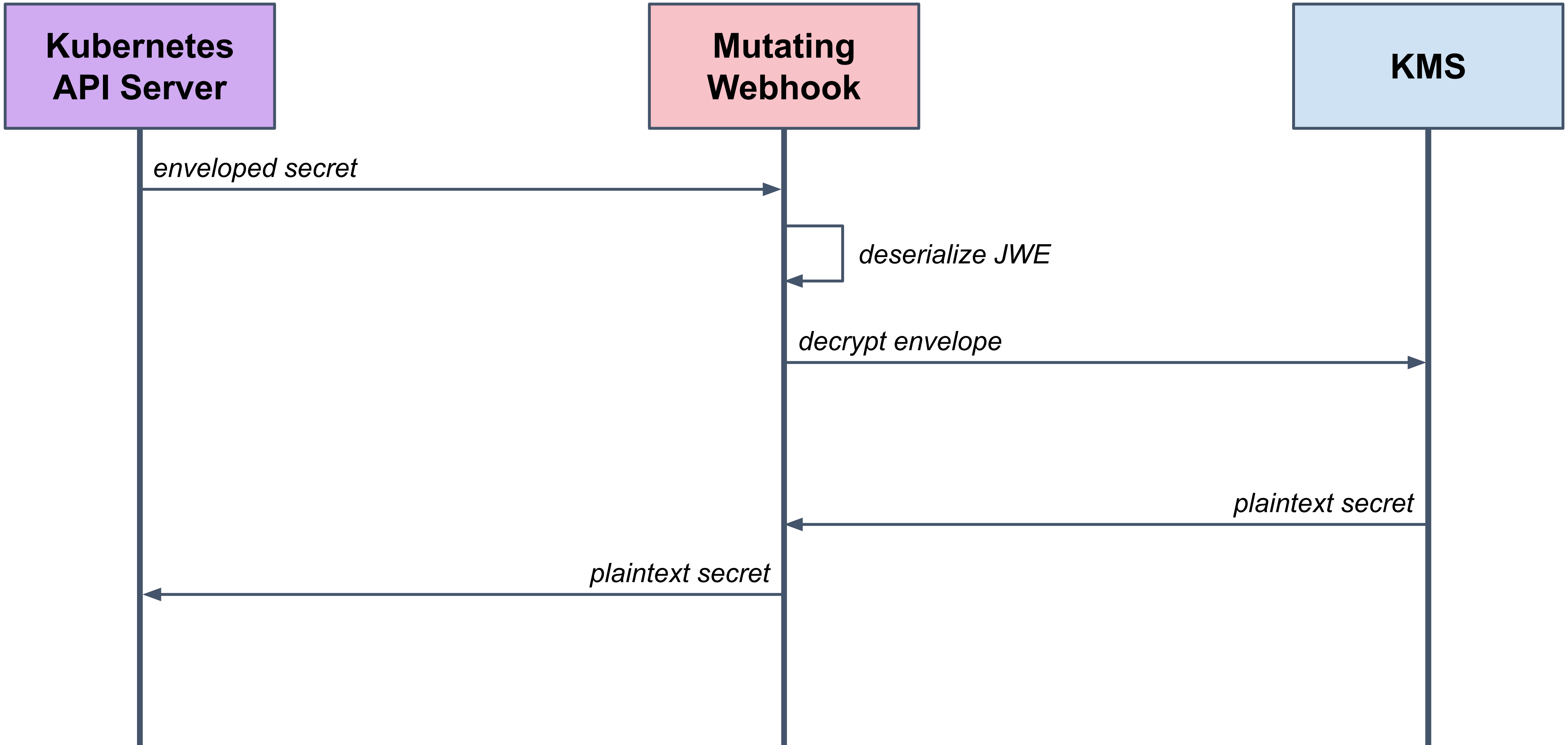
```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: MutatingWebhookConfiguration
webhooks:
- name: secrets-demo.kubecon-eu.info
  rules:
  - apiGroups: [""]
    apiVersions: ["v1"]
    operations: ["CREATE", "DELETE"]
    scope: "Namespaced"
  clientConfig:
    url: "https://jwe-webhook-farm.example.com/secrets"
    caBundle: Ls0tLs1CRUdJtiBDRVJUSUZJQ0FURS0t...
```

Deployment

```
kubectl apply -f webhook.yaml

kubectl apply -f encrypted-secret-k8s.yaml
```

Deployment


Kubernetes API Server
Mutating Webhook
KMS
enveloped secret
deserialize JWE
decrypt envelope
plaintext secret
plaintext secret

Test

```
kubectl get secret db-secret
```

```
apiVersion: v1
kind: Secret
stringData:
  password: UeBzc3cwcmQK # "P@ssw0rd" in base64
```

Thank you!

JSON Web Encryption (JWE)
tools.ietf.org/html/rfc7516

Mutating webhook source code
github.com/immutableT/k8s-secrets-and-gitops

Encrypting Kubernetes Secrets
youtube.com/watch?v=DNKcRUyz4Hw

Base64 is not encryption
youtube.com/watch?v=f4Ru6CPG1z4

KubeCon | CloudNativeCon

Europe 2020

Virtual

KEEP CLOUD NATIVE CONNECTED