KubeCon | CloudNativeCon

Europe 2018

Securing Kubernetes Clusters with Notary & TUF

Liam White
IBM

Michael Hough
IBM

Liam White

Software Engineer, IBM Cloud

IBM Cloud Container Registry
Istio Contributor

liamandrewwhite

github.com/liamawhite

Michael Hough

Software Engineer, IBM Cloud

IBM Cloud Container Registry
Notary Project Contributor

molepigeon

github.com/molepigeon

linkedin.com/in/molepigeon

Security is hard.

Containers are faster, but less secure?

How do you make sure that only trusted code runs in your production environments?

Not who, but what

How do you sign off on a release before it goes to production?

What about the bad guys?

Digitally sign it!

But how do you sign a Docker image?

Enter Notary

Implements The Update Framework (TUF)

Stores trusted data ... such as Docker image digests

Daemon

Registry

Digest for ubuntu:latest, please!

12345

Content for ubuntu@12345, please!

<stuff>

Daemon

Notary

Digest for ubuntu:latest, please!

12345, and it's signed by Alice, Bob, and Charlie

I trust Bob...

And his signature checks out!

Registry

Content for ubuntu@12345, please!

<stuff>

Why not use Docker Content Trust in your cluster?

Who else do you trust?

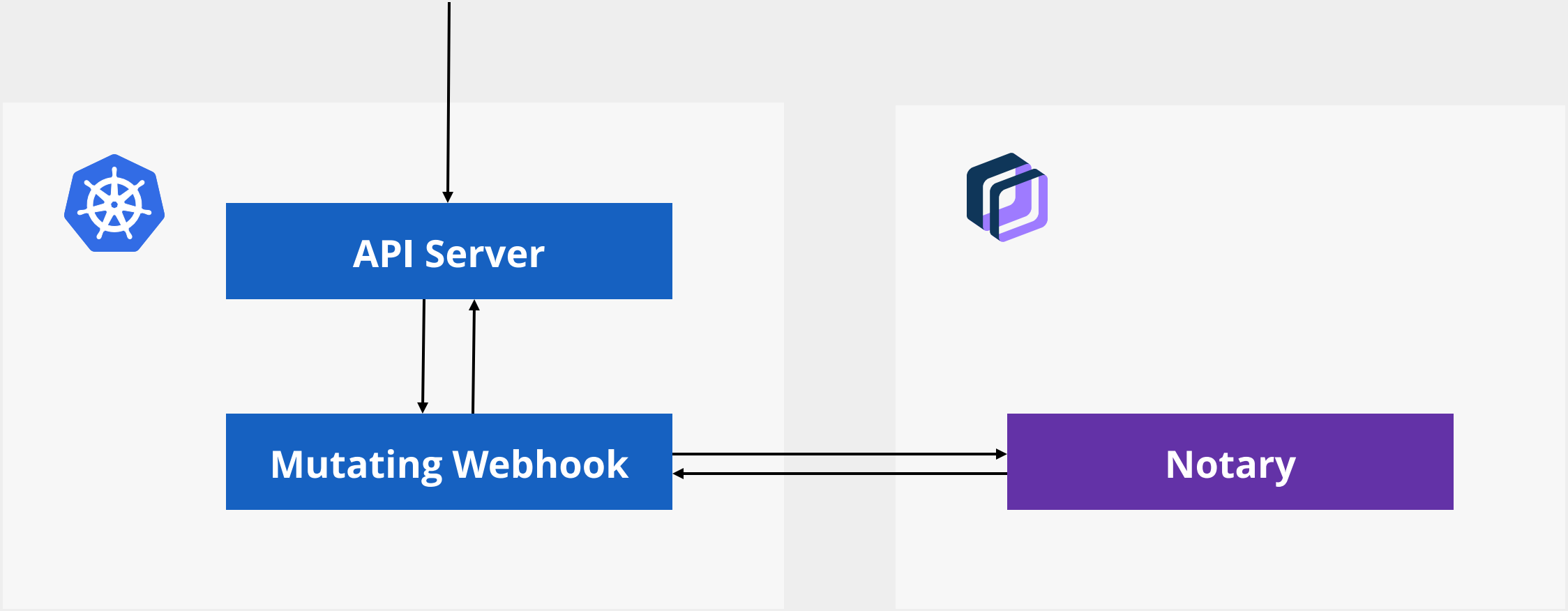
What about the kubelet images?

Kubernetes deserves powerful trust management

Admission Controllers

Validating Admission Webhook
Mutating Admission Webhook

liamwhite/kubecon@sha256:4bd87a5758f80eedb01335676a9e47347801fc...


API Server
Mutating Webhook
Notary

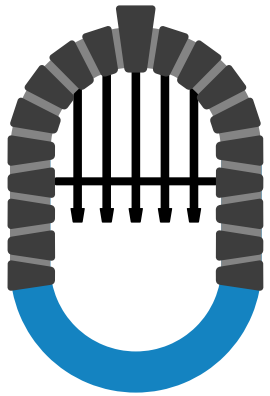
```
API Server -> Webhook (AdmissionRequest)

{
        uid: "a2e5846b-059a-4d56-a564-3b7c4fc4ccfb",

        kind: {
            group: "",
            version: "v1",
            kind: "Pod",
        },

        resource: {
            group: "",
            version: "v1",
            resource: "pods",
        },

        namespace: "default",

        operation: "CREATE",

        object: <lots-of-bytes>
}
```

API Server <- Webhook (AdmissionResponse)

```
{
        uid: "a2e5846b-059a-4d56-a564-3b7c4fc4ccfb",

        allowed: true,

        // If !allowed give a reason to inform the user
        result: {
            status: "Failure",
            message: "Untrusted Image",
            code: "401",
        }

        patchType: "JSONPatch",

        patch: <some-bytes>


}
```

API Server <- Webhook (Patch)

```
{
        operation: "replace",
        path: "/spec/containers/0/image",
        value: "liamwhite/kubecon@sha256:4bd87a5758f80eedb01335676a9e47347801fc",
}
```



PORTIERIS

github.com/ibm/portieris

Whitelist Images

Fail Closed

Namespace or Cluster Wide Policies

Extensible

```yaml
apiVersion: securityenforcement.admission.cloud.ibm.com/v1beta1
kind: ClusterImagePolicy

metadata:
  name: kubecon-cluster-image-policy

spec:
   repositories:
    - name: "docker.io/liamwhite/kubecon"
      policy:
        trust:
          enabled: true
```

```yaml
apiVersion: securityenforcement.admission.cloud.ibm.com/v1beta1
kind: ClusterImagePolicy

metadata:
  name: kubecon-cluster-image-policy-pinned

spec:
   repositories:
    - name: "docker.io/liamwhite/*"
      policy:
        trust:
          enabled: true
          signerSecrets:
          - name: <secret_name>
```

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: <secret_name>
data:
  name: c2lnbmVyMQ==
  publicKey: LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0...
```

```yaml
apiVersion: securityenforcement.admission.cloud.ibm.com/v1beta1
kind: ImagePolicy

metadata:
  name: kubecon-image-policy
  namespace: default

spec:
   repositories:
    - name: "docker.io/liamwhite/*"
      policy:
        trust:
          enabled: true
          signerSecrets:
          - name: <secret_name>
```

Demo

 liamandrewwhite

 molepigeon

github.com/ibm/portieris