

Managing Certificates and kubeconfig Files



Anthony E. Nocentino

ENTERPRISE ARCHITECT @ CENTINO SYSTEMS

@nocentino www.centinosystems.com

Course Overview



Kubernetes Security Fundamentals

Managing Certificates and kubeconfig Files

Managing Role Based Access Controls

Summary

Certificates and PKI in Kubernetes

Creating and managing certificates

**Configuring and Managing kubeconfig files
for accessing clusters**

Certificates and PKI in Kubernetes



TLS Encryption

User and System Component authentication

kubeadm-based clusters

Creates a self-signed Certificate Authority

Generates certificates for System Components

kubernetes-admin user

Can use an external/trusted CA

<https://bit.ly/39KLm9j>

Certificates and PKI in Kubernetes



Single self-signed Certificate Authority

`/etc/kubernetes/pki/`

`ca.key` - this is the private key

`ca.crt` - CA Certificate

Certificates and PKI in Kubernetes - ca.crt



Distribute to clients to trust your self-signed CA

Copied to Nodes in the cluster during build

kubeconfig files

ServiceAccount

kubeconfig Files and Certificate Based Authentication

Cluster Location

Client authentication

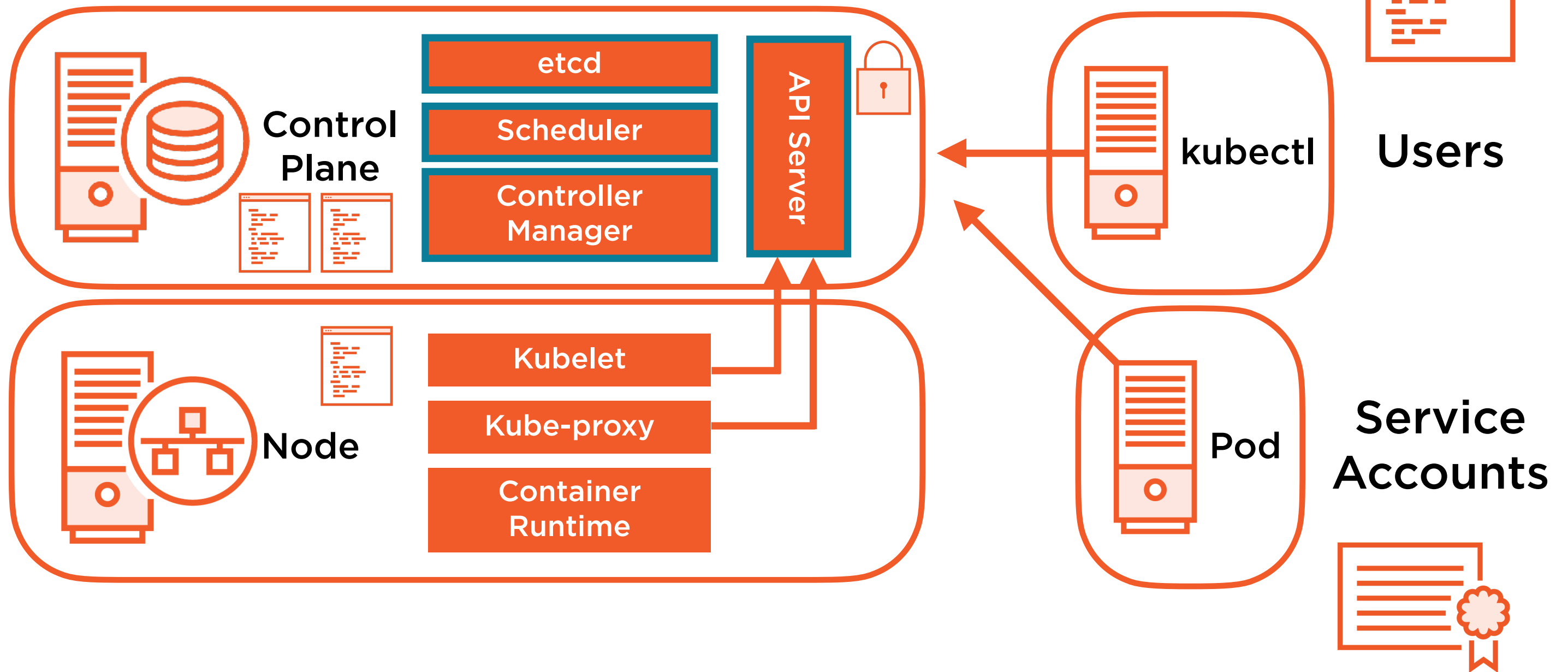
Users or system
components

CA Certificate
ca.crt

Client Certificate

Client Private Key

Certificates in Kubernetes Clusters

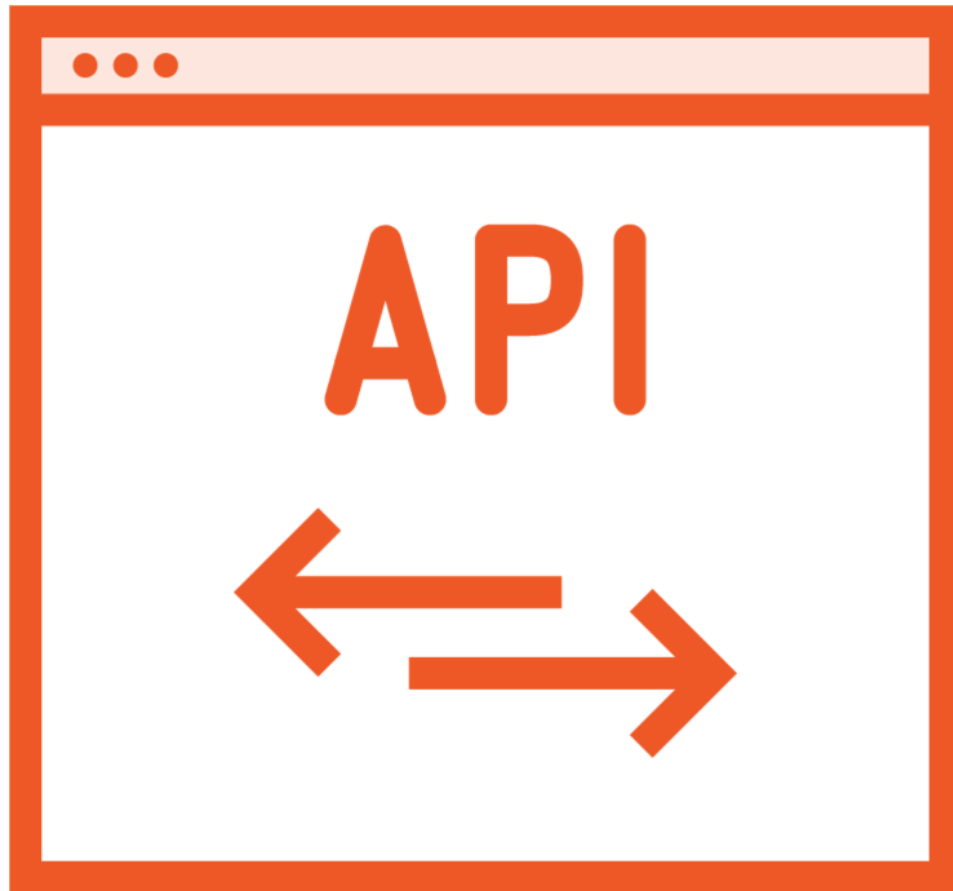


Demo

Investigating the PKI setup on the Control Plane Node

- **Certificate authority**
- **Control Plane Pod kubeconfig configuration**

Creating Certificates with the Certificate API



Submit and sign Certificate Signing Requests (CSR) via the API Server

Encryption and authentication in the cluster

Programmatic interface

Creating a New Certificate

Create a Private Key
with openssl or cfssl

Create a Certificate
Signing Request with
openssl or cfssl

Create and submit
CertificateSigning
Request Object

Approve the
CertificateSigning
Request

Retrieve the
Certificate

This process is used to create new certificates for new users

Creating a Certificate Signing Request in openssl

#Create a private key

```
openssl genrsa -out demouser.key 2048
```

#Generate a CSR

#CN (Common Name) is your username, O (Organization) is the Group

```
openssl req -new -key demouser.key -out demouser.csr -subj "/CN=demouser"
```

#The CertificateSigningRequest needs to be base64 encoded

#And also have the header and trailer pulled out.

```
cat demouser.csr | base64 | tr -d "\n" > demouser.base64.csr
```

Creating a CertificateSigningRequest Object

```
cat <<EOF | kubectl apply -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: demouser
spec:
  groups:
  - system:authenticated
  request: $(cat demouser.base64.csr)
  usages:
  - client auth
EOF
```

Approving a CSR and Retrieving a Certificate

#Approve the CSR

```
kubectl certificate approve demouser
```

#Retrieve the certificate from the CSR object, it's base64 encoded

```
kubectl get certificatesigningrequests demouser \
  -o jsonpath='{ .status.certificate }' | base64 --decode > demouser.crt
```

Demo

Creating a new certificate for a new user

kubeconfig File Overview

Cluster access

**Context is a cluster's
location and
credentials**

**Multiple
configuration
contexts**

**Multiple kubeconfig
files**

Users

System components

kubeconfig File Components

| Users | Clusters | Contexts |
|----------------------------|--|---|
| Credentials | Network location of the API Server | Access parameters to the cluster |
| Username | Certificate Authority's Certificate (ca.crt) | Comprised of a Cluster and User in this kubeconfig file |
| Certificate/Token/Password | | Set the namespace |

kubeconfig Files - admin.conf

apiVersion: v1

clusters:

- cluster:

certificate-authority-data: DATA+OMITTED

server: https://172.16.94.10:6443

name: kubernetes

users:

- name: kubernetes-admin

user:

client-certificate-data: REDACTED

client-key-data: REDACTED

contexts:

- context:

cluster: kubernetes

user: kubernetes-admin

name: kubernetes-admin@kubernetes

current-context: kubernetes-admin@kubernetes

Creating kubeconfig Files Manually

#Define a cluster

```
kubectl config set-cluster kubernetes-demo \  
  --server=https://172.16.94.10:6443 \  
  --certificate-authority=/etc/kubernetes/pki/ca.crt \  
  --embed-certs=true \  
  --kubeconfig=demouser.conf
```

#Define a credential

```
kubectl config set-credentials demouser \  
  --client-key=demouser.key \  
  --client-certificate=demouser.crt \  
  --embed-certs=true \  
  --kubeconfig=demouser.conf
```

#Define the context

```
kubectl config set-context demouser@kubernetes-demo \  
  --cluster=kubernetes-demo \  
  --user=demouser \  
  --kubeconfig=demouser.conf
```

#Set the context

```
kubectl config use-context \  
  demouser@kubernetes-demo --kubeconfig=demouser.conf
```

Demo

Working with kubeconfig files and contexts

Creating a kubeconfig file for a new user

Using a kubeconfig file for a new user

Review

Certificates in Kubernetes

Creating and managing certificates

Managing kubeconfig Files for accessing clusters

Up Next:

Managing Role Based Access Controls
