

# AUTHENTICATION



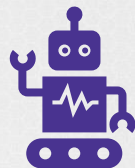
Admins



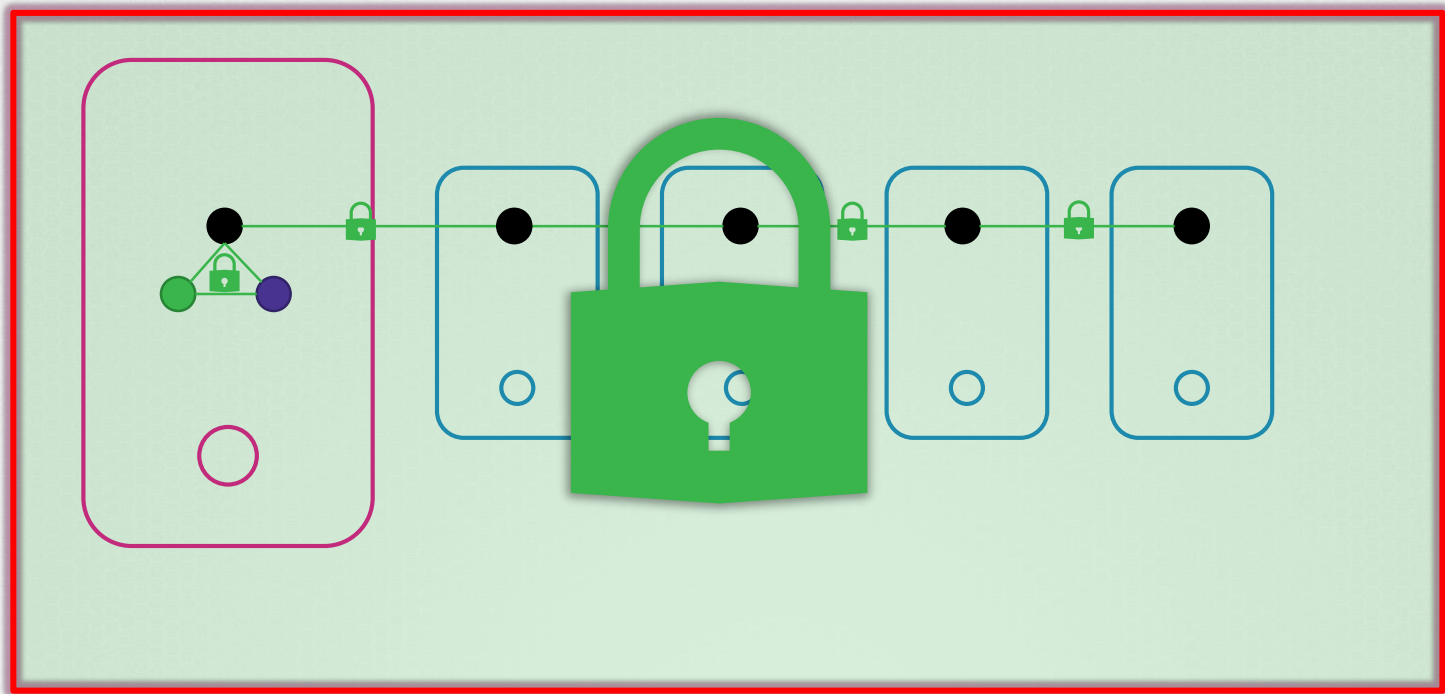
Developers



End Users



Bots



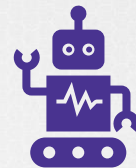
# |Accounts



Admins



Developers



Bots

User

Service Accounts

# Accounts



Admins



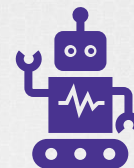
Developers

User

```
▶ kubectl create user user1  
user1 Created
```



```
▶ kubectl list users  
  
Username  
user1  
user2
```



Bots

Service Accounts

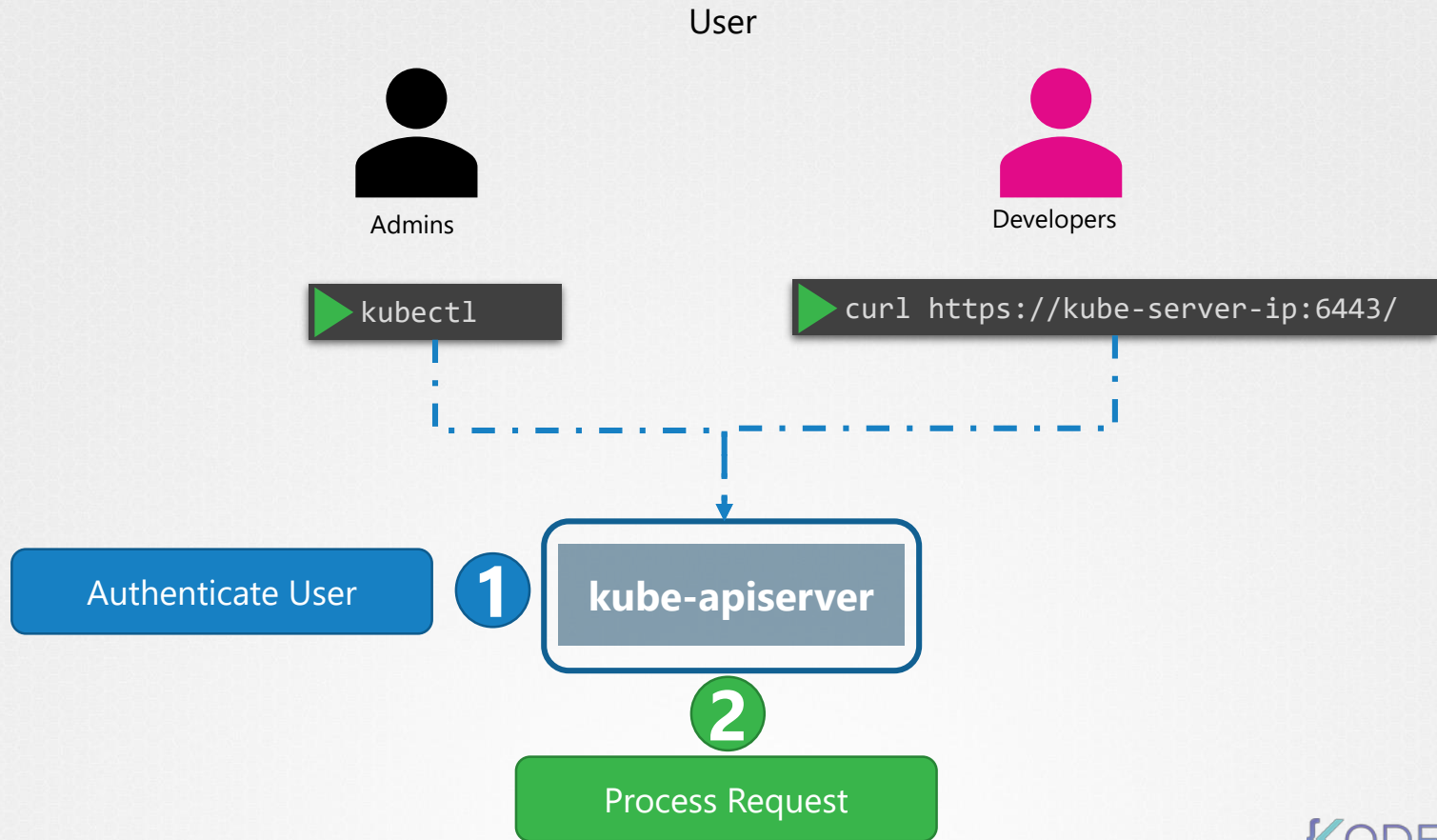
```
▶ kubectl create serviceaccount sa1  
Service Account sa1 Created
```



```
▶ kubectl list serviceaccount  
  
ServiceAccount  
sa1
```



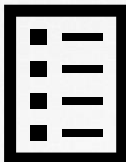
# Accounts



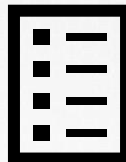
# | Auth Mechanisms

kube-apiserver

Static Password File



Static Token File





# Auth Mechanisms - Basic

kube-apiserver

~~--basic-auth-file=user-details.csv~~

user-details.csv

```
password123,user1,u0001
password123,user2,u0002
password123,user3,u0003
password123,user4,u0004
password123,user5,u0005
```

kube-apiserver.service

```
ExecStart=/usr/local/bin/kube-apiserver \\  
  --advertise-address=${INTERNAL_IP} \\  
  --allow-privileged=true \\  
  --apiserver-count=3 \\  
  --authorization-mode=Node,RBAC \\  
  --bind-address=0.0.0.0 \\  
  --enable-swagger-ui=true \\  
  --etcd-servers=https://127.0.0.1:2379 \\  
  --event-ttl=1h \\  
  --runtime-config=api/all \\  
  --service-cluster-ip-range=10.32.0.0/24 \\  
  --service-node-port-range=30000-32767 \\  
  --v=2
```

Note: Showing fewer options for simplicity

# Kube-api Server Configuration

## kube-apiserver.service

```
ExecStart=/usr/local/bin/kube-apiserver \\  
  --advertise-address=${INTERNAL_IP} \\  
  --allow-privileged=true \\  
  --apiserver-count=3 \\  
  --authorization-mode=Node,RBAC \\  
  --bind-address=0.0.0.0 \\  
  --enable-swagger-ui=true \\  
  --etcd-servers=https://127.0.0.1:2379 \\  
  --event-ttl=1h \\  
  --runtime-config=api/all \\  
  --service-cluster-ip-range=10.32.0.0/24 \\  
  --service-node-port-range=30000-32767 \\  
  --v=2  
--basic-auth-file=/etc/kubernetes/user-details.csv
```

## /etc/kubernetes/manifests/kube-apiserver.yaml

```
apiVersion: v1  
kind: Pod  
metadata:  
  creationTimestamp: null  
  name: kube-apiserver  
  namespace: kube-system  
spec:  
  containers:  
  - command:  
    - kube-apiserver  
    - --authorization-mode=Node,RBAC  
    - --advertise-address=172.17.0.107  
    - --allow-privileged=true  
    - --enable-admission-plugins=NodeRestriction  
    - --enable-bootstrap-token-auth=true  
  
  image: k8s.gcr.io/kube-apiserver-amd64:v1.11.3  
  name: kube-apiserver
```

Note: Showing fewer options for simplicity



# Authenticate User

```
▶ curl -v -k https://master-node-ip:6443/api/v1/pods -u "user1:password123"
```

```
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/pods",
    "resourceVersion": "3594"
  },
  "items": [
    {
      "metadata": {
        "name": "nginx-64f497f8fd-krkg6",
        "generateName": "nginx-64f497f8fd-",
        "namespace": "default",
        "selfLink": "/api/v1/namespaces/default/pods/nginx-64f497f8fd-krkg6",
        "uid": "77dd7dfb-2914-11e9-b468-0242ac11006b",
        "resourceVersion": "3569",
        "creationTimestamp": "2019-02-05T07:05:49Z",
        "labels": {
          "pod-template-hash": "2090539498",
          "run": "nginx"
        }
      }
    }
  ]
}
```

# Auth Mechanisms - Basic

## Static Password File

user-details.csv

```
password123,user1,u0001,group1  
password123,user2,u0002,group1  
password123,user3,u0003,group2  
password123,user4,u0004,group2  
password123,user5,u0005,group2
```

## Static Token File

user-token-details.csv

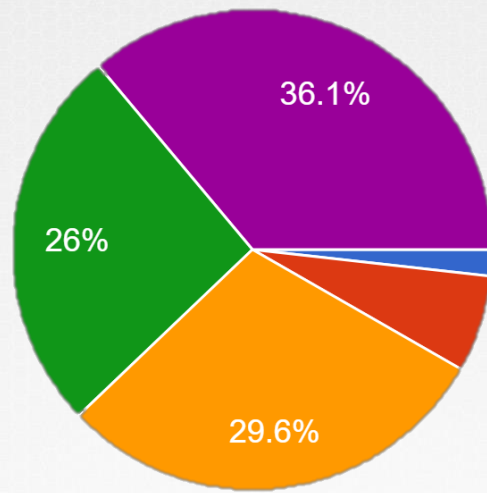
```
KpjCVbI7rCFAHYpKByTIzRb7gu1cUc4B,user10,u0010,group1  
rJjncHmvtXHc6MlWQddhtvNyyhgTdxSC,user11,u0011,group1  
mjpOFIEiFOkL9toikaRNtt59ePtczZSq,user12,u0012,group2  
PG41IXhs7QjqwWkmBkvgGT9glOyUqZij,user13,u0013,group2
```

```
--token-auth-file=user-details.csv
```

```
curl -v -k https://master-node-ip:6443/api/v1/pods --header "Authorization: Bearer KpjCVbI7rCFAHYpKbZbRb7gu1cUc4B"
```

# I Note

- This is not a recommended authentication mechanism
- Consider volume mount while providing the auth file in a kubeadm setup
- Setup Role Based Authorization for the new users



No Clue

Not very  
Comfortable

# | Goals!

- ☐ What are TLS Certificates?
- ☐ How does Kubernetes use Certificates?
- ☐ How to generate them?
- ☐ How to configure them?
- ☐ How to view them?
- ☐ How to troubleshoot issues related to Certificates



Certificate (Public Key)

-----  
\*.cert \*.pem

server.crt  
server.pem  
client.crt  
client.pem

Private Key

-----  
\*.key \*-key.pem

server.key  
server-key.pem  
client.key  
client-key.pem



Public Key (Lock)



Private Key

**User:** John  
**Password:** Pass123

**XCVB:** DKSJD  
**LKJSDFK:** XZKJSDLF

**User:** John  
**Password:** Pass123

**XCVB:** DKSJD  
**LKJSDFK:** XZKJSDLF

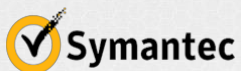


# TLS CERTIFICATES

What Certificates?



# CERTIFICATE AUTHORITY (CA)



Root Certificates



Client Certificates

Certificate (Public Key)

-----  
\*.crt \*.pem

server.crt  
server.pem  
client.crt  
client.pem

Private Key

-----  
\*.key \*-key.pem

server.key  
server-key.pem  
client.key  
client-key.pem

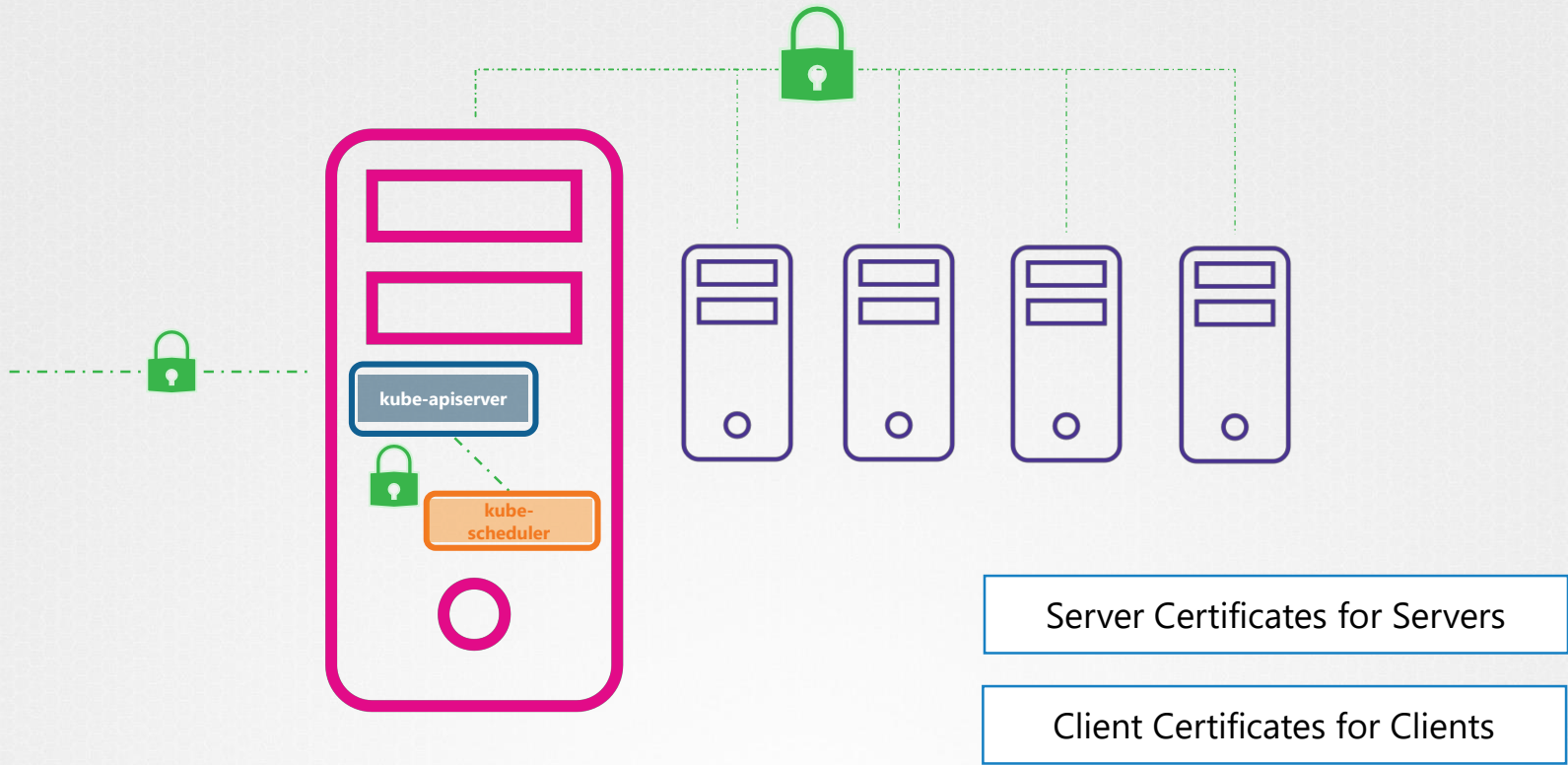


Certificate  
(Public Key)

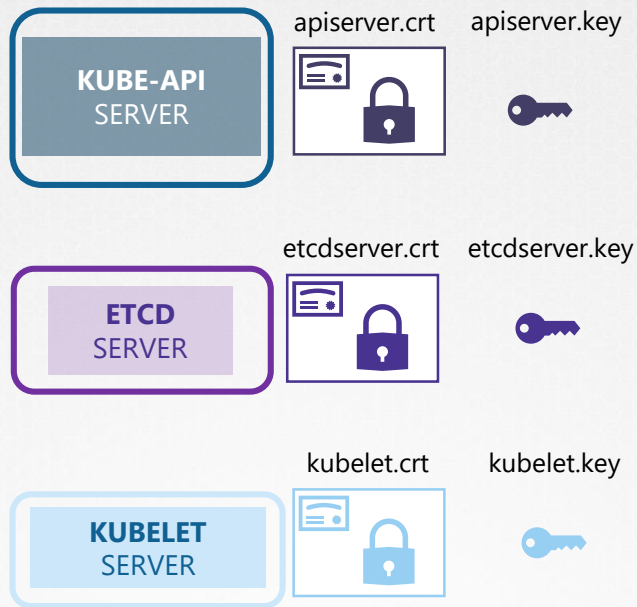


Private Key

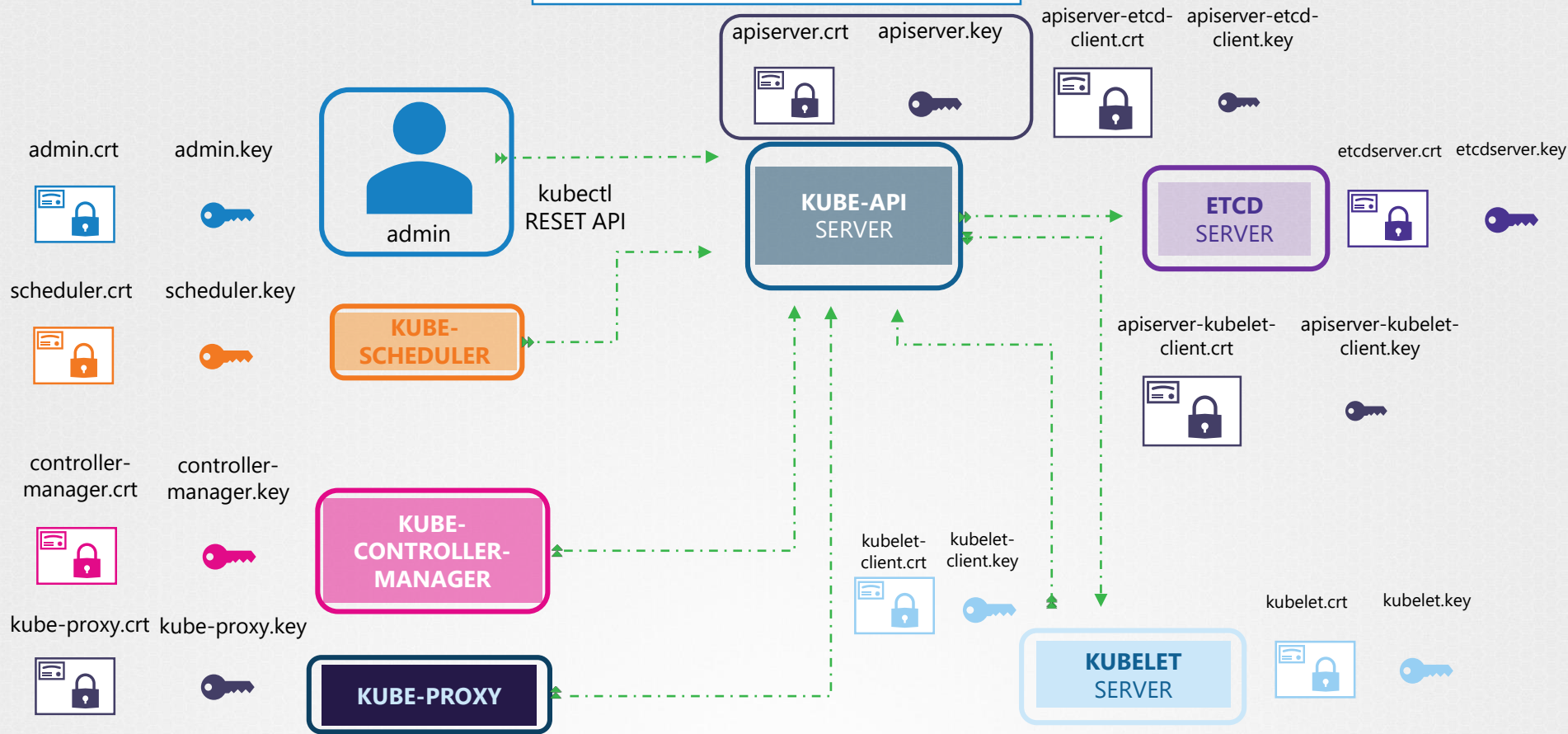
Server Certificates



## Server Certificates for Servers



## Client Certificates for Clients





# CERTIFICATE AUTHORITY (CA)

## Client Certificates for Clients

admin.crt admin.key



admin

scheduler.crt scheduler.key



KUBE-SCHEDULER

controller-manager.crt controller-manager.key



KUBE-CONTROLLER-MANAGER

kube-proxy.crt kube-proxy.key



KUBE-PROXY

apiserver-kubelet-client.crt apiserver-kubelet-client.key



KUBE-API SERVER

apiserver-etcd-client.crt apiserver-etcd-client.key



KUBE-API SERVER

kubelet-client.crt kubelet-client.key



KUBELET SERVER

## Server Certificates for Servers

etcdserver.crt etcdserver.key



ETCD SERVER

apiserver.crt apiserver.key



KUBE-API SERVER

kubelet.crt kubelet.key



KUBELET SERVER





CERTIFICATE AUTHORITY (CA)



CERTIFICATE AUTHORITY (CA)

### Client Certificates for Clients

admin.crt admin.key



admin

scheduler.crt scheduler.key



KUBE-SCHEDULER

controller-manager.crt controller-manager.key



KUBE-CONTROLLER-MANAGER

kube-proxy.crt kube-proxy.key



KUBE-PROXY

### Server Certificates for Servers

apiserver-kubelet-client.crt apiserver-kubelet-client.key



KUBE-API SERVER

apiserver.crt apiserver.key



KUBE-API SERVER

kubelet-client.crt kubelet-client.key



KUBELET SERVER

kubelet.crt kubelet.key



KUBELET SERVER

### Server Certificates for Servers

etcdserver.crt etcdserver.key



ETCD SERVER

apiserver-etcd-client.crt apiserver-etcd-client.key



KUBE-API SERVER



# CERTIFICATE AUTHORITY (CA)

## Client Certificates for Clients

admin.crt

admin.key



admin

scheduler.crt

scheduler.key



KUBE-SCHEDULER

controller-  
manager.crt

controller-  
manager.key



KUBE-CONTROLLER-MANAGER

kube-proxy.crt

kube-proxy.key



KUBE-PROXY

apiserver-kubelet-  
client.crt

apiserver-kubelet-  
client.key



KUBE-API SERVER

apiserver-etcd-  
client.crt

apiserver-etcd-  
client.key



KUBE-API SERVER

kubelet-  
client.crt

kubelet-  
client.key



KUBELET SERVER

## Server Certificates for Servers

etcdserver.crt

etcdserver.key



ETCD SERVER

apiserver.crt

apiserver.key



KUBE-API SERVER

kubelet.crt

kubelet.key



KUBELET SERVER



{K}ODE{K}LOUD

# Course Objectives

Core Concepts

Scheduling

Logging Monitoring

Application Lifecycle Management

Cluster Maintenance

Security

☐ Kubernetes Security Primitives

☐ Secure Persistent Key Value Store

☐ Authentication

☐ Authorization

☐ Security Contexts

☐ TLS Certificates for Cluster Components

☐ Images Securely

☐ Network Policies

Storage

Networking

Installation, Configuration & Validation

Troubleshooting

# TLS CERTIFICATES

Generate Certificates

EASYRSA

OPENSSL

CFSSL



OPENSSL



# CERTIFICATE AUTHORITY (CA)

## Client Certificates for Clients

admin.crt admin.key



admin

scheduler.crt scheduler.key



KUBE-SCHEDULER

controller-manager.crt controller-manager.key



KUBE-CONTROLLER-MANAGER

kube-proxy.crt kube-proxy.key



KUBE-PROXY

apiserver-kubelet-client.crt apiserver-kubelet-client.key



apiserver-etcd-client.crt apiserver-etcd-client.key



KUBE-API SERVER

kubelet-client.crt kubelet-client.key



KUBELET SERVER

## Server Certificates for Servers

etcdserver.crt etcdserver.key



ETCD SERVER

apiserver.crt apiserver.key



KUBE-API SERVER

kubelet.crt kubelet.key



KUBELET SERVER



# CERTIFICATE AUTHORITY (CA)

Generate Keys



ca.key

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

ca.key



```
openssl req -new -key ca.key -subj "/CN=KUBERNETES-CA" -out ca.csr
```

```
openssl x509 -req -in ca.csr -signkey ca.key -out ca.crt
```



ca.key



ca.crt



# ADMIN USER

admin.key



Generate Keys

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

Certificate  
Signing  
Request

admin.csr



```
openssl req -new -key admin.key -subj \
"/CN=kube-admin/OU=system:masters" -out admin.csr
```

```
-in admin.csr -CA ca.crt -CAkey ca.key -out admin.crt
```







ca.key



ca.crt

# KUBE SCHEDULER

Generate Keys

scheduler.key



Certificate  
Signing  
Request

scheduler.csr



Sign  
Certificates

scheduler.crt





ca.key



ca.crt

# KUBE CONTROLLER MANGER

Generate Keys

controller-manager.key

Certificate  
Signing  
Request



controller-manager.csr



Sign  
Certificates

controller-manager.crt







ca.key



ca.crt

# KUBE PROXY

Generate Keys

kube-proxy.key



Certificate  
Signing  
Request

kube-proxy.csr



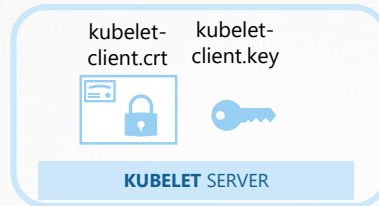
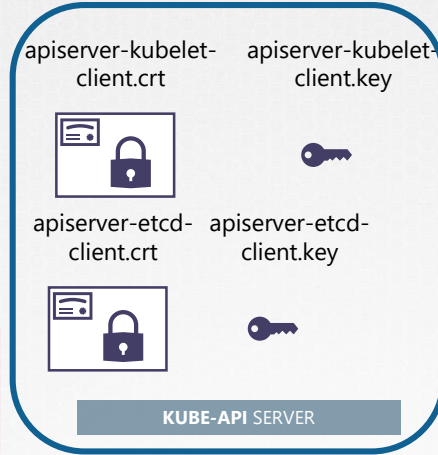
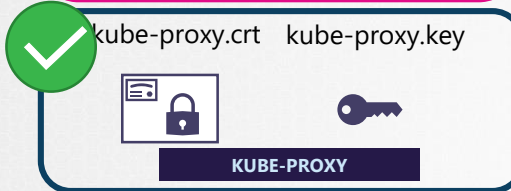
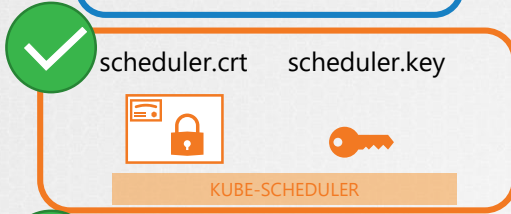
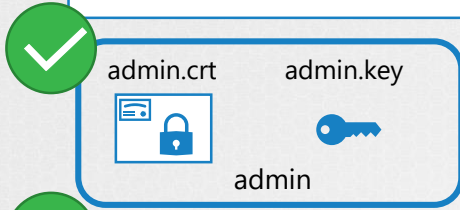
Sign  
Certificates

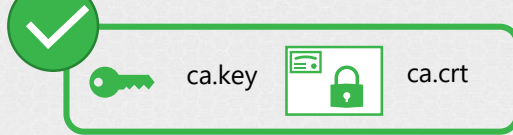
kube-proxy.crt



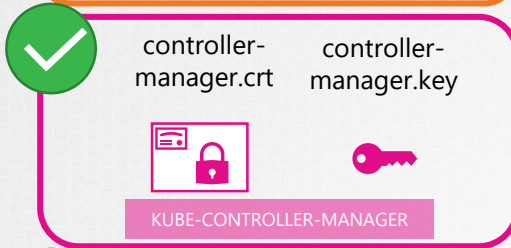
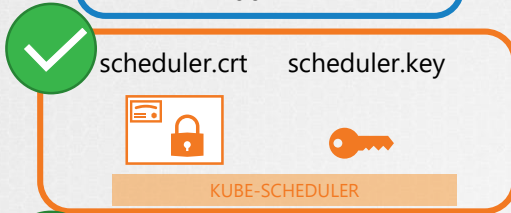
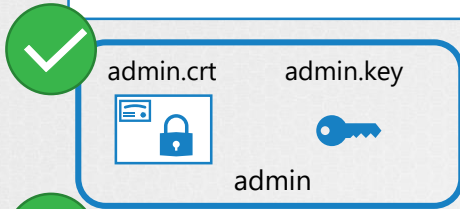


## Client Certificates for Clients





## Client Certificates for Clients

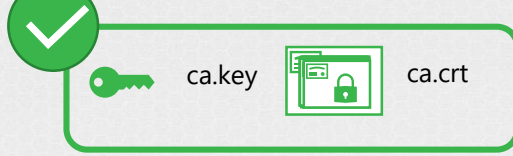


```
curl https://kube-apiserver:6443/api/v1/pods \
--key admin.key --cert admin.crt
--cacert ca.crt
```

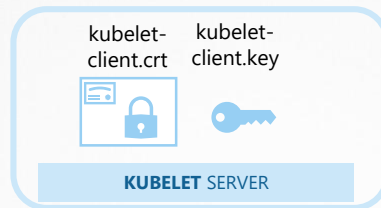
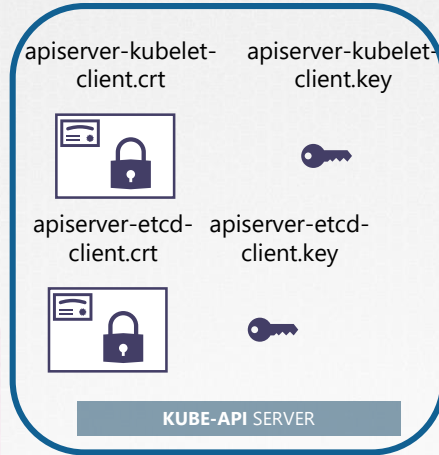
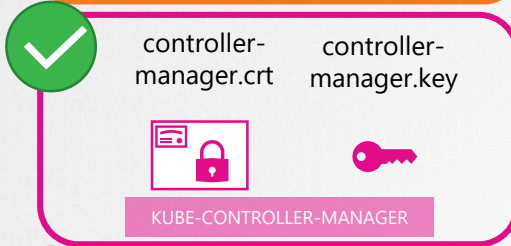
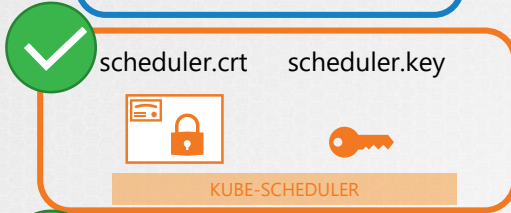
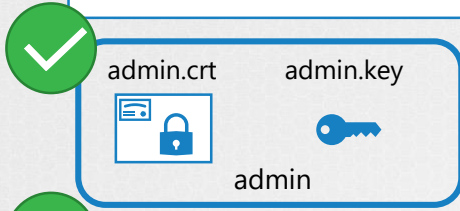
```
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/pods",
  },
  "items": []
}
```

### kube-config.yaml

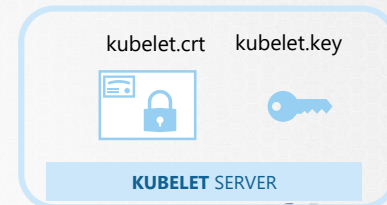
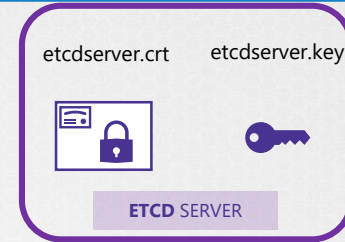
```
apiVersion: v1
clusters:
- cluster:
    certificate-authority: ca.crt
    server: https://kube-apiserver:6443
    name: kubernetes
kind: Config
users:
- name: kubernetes-admin
  user:
    client-certificate: admin.crt
    client-key: admin.key
```



## Client Certificates for Clients

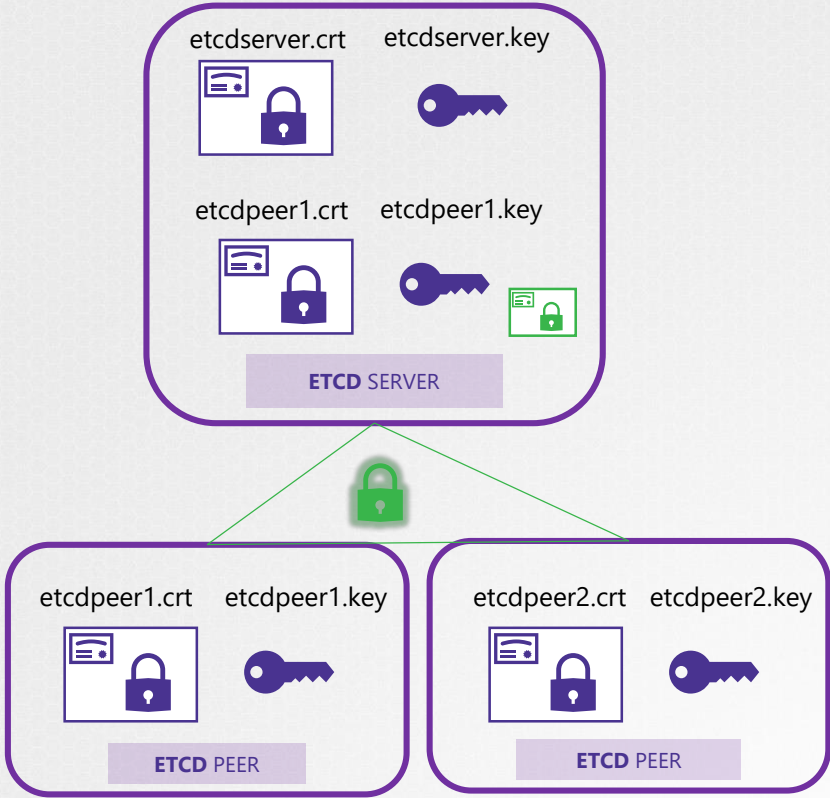


## Server Certificates for Servers

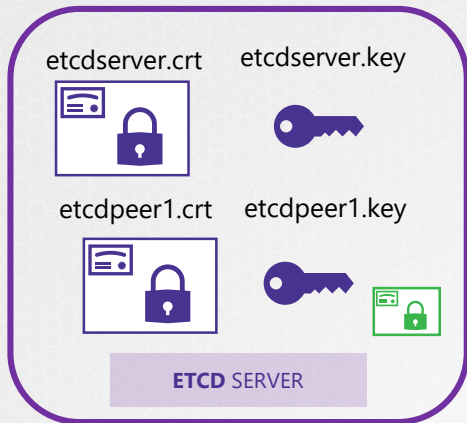




## ETCD SERVERS



## ETCD SERVERS



```
cat etcd.yaml
```

```
- etcd
  - --advertise-client-urls=https://127.0.0.1:2379
  - --key-file=/path-to-certs/etcdserver.key
  - --cert-file=/path-to-certs/etcdserver.crt
  - --client-cert-auth=true
  - --data-dir=/var/lib/etcd
  - --initial-advertise-peer-urls=https://127.0.0.1:2380
  - --initial-cluster=master=https://127.0.0.1:2380
  - --listen-client-urls=https://127.0.0.1:2379
  - --listen-peer-urls=https://127.0.0.1:2380
  - --name=master
  - --peer-cert-file=/path-to-certs/etcdpeer1.crt
  - --peer-client-cert-auth=true
  - --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
  - --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
  - --snapshot-count=10000
  - --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
```



## KUBE API SERVER

apiserver.crt apiserver.key



KUBE-API SERVER

### CERTIFICATE

kubernetes

kubernetes.default

kubernetes.default.svc

kubernetes.default.svc.cluster.local

10.96.0.1

172.17.0.87

*This Certificate Proudly Presented to*

**KUBE-API SERVER**

MIIDvDCCAgSgAwIBAgIUfZ1+94HNB+Nf4jjZ56cVQg5d3pAwDQYJKoZIhvcNAQELBQAwZDELMAkGA1UEBhMCVmxvDzANBgNVBAgTBk9yZWdvbjERMhA8GA1UEBmIMU9yZGxhbmQwETAPBgNVBAoTCFNF5bnFudG9jMQswCQYDVQQLExJ0TERMA8GA1UEAwIU31tYW50ZWwHhcnMTkwMjA4MDIxMzAwHhcnMTQwMjA3MDIxMzAwWjBkMQswCQYDVQGEwJVUzEPMA0GA1UECBMT3JlZ29uMREwDwYDVQQHEWhQb3J0bGFuZDERMA8G

NEW YORK  
NY, US



Issued by:



apiserver.crt apiserver.key



KUBE-API SERVER

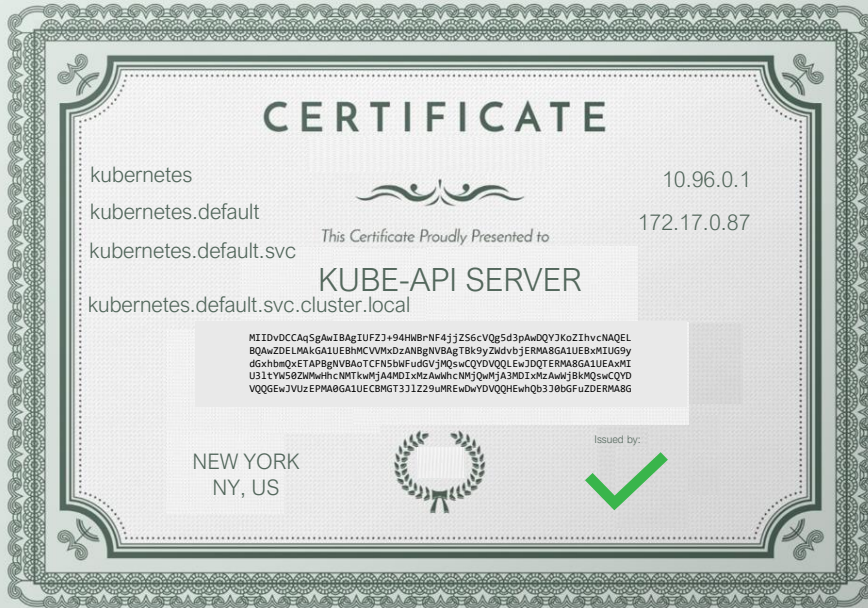
## KUBE API SERVER

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

apiserver.key

```
openssl req -new -key apiserrver.key -subj \"/CN=kube-apiserver" -out apiserver.csr
```

apiserver.csr



apiserver.crt apiserver.key



KUBE-API SERVER

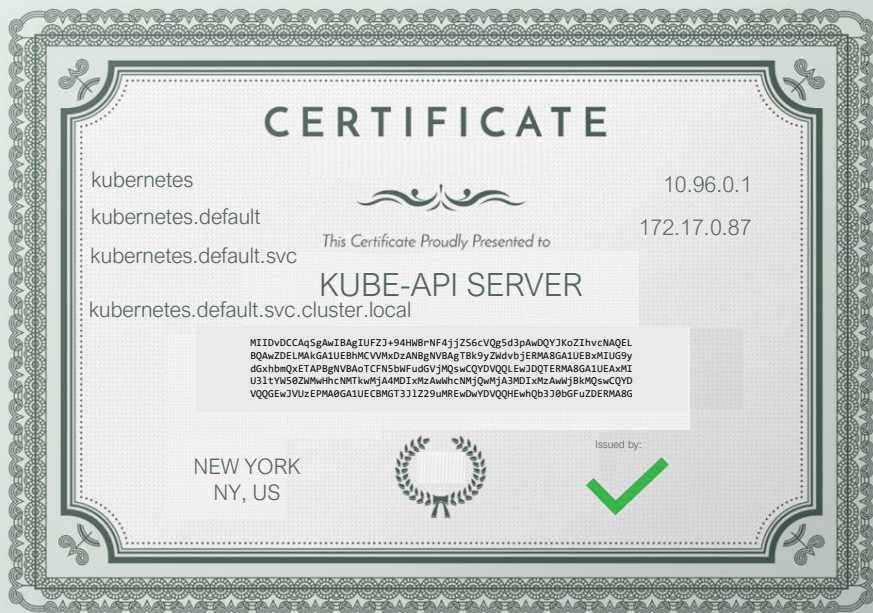
## KUBE API SERVER

```
openssl req -new -key apiserver.key -subj \
"/CN=kube-apiserver" -out apiserver.csr -config openssl.cnf
apiserver.csr
```

openssl.cnf

```
[req]
req_extensions = v3_req
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation,
subjectAltName = @alt_names
[alt_names]
DNS.1 = kubernetes
DNS.2 = kubernetes.default
DNS.3 = kubernetes.default.svc
DNS.4 = kubernetes.default.svc.cluster.local
IP.1 = 10.96.0.1
IP.2 = 172.17.0.87
```

```
openssl x509 -req -in apiserver.csr \
-CA ca.crt -CAkey ca.key -out apiserver.crt
apiserver.crt
```





## KUBE API SERVER

apiserver.crt apiserver.key



KUBE-API SERVER

apiserver-kubelet-  
client.crt apiserver-kubelet-  
client.key



apiserver-etcd-  
client.crt apiserver-etcd-  
client.key



KUBE-API SERVER

```
ExecStart=/usr/local/bin/kube-apiserver \\  
  --advertise-address=${INTERNAL_IP} \\  
  --allow-privileged=true \\  
  --apiserver-count=3 \\  
  --authorization-mode=Node,RBAC \\  
  --bind-address=0.0.0.0 \\  
  --enable-swagger-ui=true \\  
  --etcd-cafile=/var/lib/kubernetes/ca.pem \\  
  --etcd-certfile=/var/lib/kubernetes/apiserver-etcd-client.crt \\  
  --etcd-keyfile=/var/lib/kubernetes/apiserver-etcd-client.key \\  
  --etcd-servers=https://127.0.0.1:2379 \\  
  --event-ttl=1h \\  
  --kubelet-certificate-authority=/var/lib/kubernetes/ca.pem \\  
  --kubelet-client-certificate=/var/lib/kubernetes/apiserver-etcd-client.crt \\  
  --kubelet-client-key=/var/lib/kubernetes/apiserver-etcd-client.key \\  
  --kubelet-https=true \\  
  --runtime-config=api/all \\  
  --service-account-key-file=/var/lib/kubernetes/service-account.pem \\  
  --service-cluster-ip-range=10.32.0.0/24 \\  
  --service-node-port-range=30000-32767 \\  
  --client-ca-file=/var/lib/kubernetes/ca.pem \\  
  --tls-cert-file=/var/lib/kubernetes/apiserver.crt \\  
  --tls-private-key-file=/var/lib/kubernetes/apiserver.key \\  
  --v=2
```

## KUBECTL NODES (SERVER CERT)

kubelet.crt   kubelet.key



KUBELET SERVER



node01



node02



node03

kubelet-config.yaml (node01)

```
kind: KubeletConfiguration
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  x509:
    clientCAFile: "/var/lib/kubernetes/ca.pem"
authorization:
  mode: Webhook
clusterDomain: "cluster.local"
clusterDNS:
  - "10.32.0.10"
podCIDR: "${POD_CIDR}"
resolvConf: "/run/systemd/resolve/resolv.conf"
runtimeRequestTimeout: "15m"
tlsCertFile: "/var/lib/kubelet/kubelet-node01.crt"
tlsPrivateKeyFile: "/var/lib/kubelet/kubelet-
node01.key"
```

## KUBECTL NODES (CLIENT CERT)

Kubelet-client.crt Kubelet-client.key



KUBELET SERVER



node01



node02



node03





{K}ODE{K}LOUD

# Course Objectives

Core Concepts

Scheduling

Logging Monitoring

Application Lifecycle Management

Cluster Maintenance

Security

☐ Kubernetes Security Primitives

☐ Secure Persistent Key Value Store

☐ Authentication

☐ Authorization

☐ Security Contexts

☐ TLS Certificates for Cluster Components

☐ Images Securely

☐ Network Policies

Storage

Networking

Installation, Configuration & Validation

Troubleshooting

# TLS CERTIFICATES

View Certificate Details

"The Hard Way"

kubeadm

## "The Hard Way"

```
▶ cat /etc/systemd/system/kube-apiserver.service
```

```
[Service]
ExecStart=/usr/local/bin/kube-apiserver \
  --advertise-address=172.17.0.32 \
  --allow-privileged=true \
  --apiserver-count=3 \
  --authorization-mode=Node,RBAC \
  --bind-address=0.0.0.0 \
  --client-ca-file=/var/lib/kubernetes/ca.pem \
  --enable-swagger-ui=true \
  --etcd-cafile=/var/lib/kubernetes/ca.pem \
  --etcd-certfile=/var/lib/kubernetes/kubernetes.pem \
  --etcd-keyfile=/var/lib/kubernetes/kubernetes-key.pem \
  --event-ttl=1h \
  --kubelet-certificate-authority=/var/lib/kubernetes/ca.pem \
  --kubelet-client-key=/var/lib/kubernetes/kubernetes-key.pem \
  --kubelet-https=true \
  --service-node-port-range=30000-32767 \
  --tls-cert-file=/var/lib/kubernetes/kubernetes.pem \
  --tls-private-key-file=/var/lib/kubernetes/kubernetes-key.pem \
  --v=2
```

## kubeadm

```
▶ cat /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
spec:
  containers:
  - command:
    - kube-apiserver
    - --authorization-mode=Node,RBAC
    - --advertise-address=172.17.0.32
    - --allow-privileged=true
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --disable-admission-plugins=PersistentVolumeLabel
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,HostIP
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
    - --requestheader-allowed-names=front-proxy-client
```

# kubeadm

Component	Type	Certificate Path	CN Name	ALT Names	Organization	Issuer	Expiration
kube-apiserver	Server						
kube-apiserver	Server						
kube-apiserver	Server						
kube-apiserver	Client (Kubelet)						
kube-apiserver	Client (Kubelet)						
kube-apiserver	Client (Etcd)						
kube-apiserver	Client (Etcd)						
kube-apiserver	Client (Etcd)						



```
▶ cat /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
spec:
  containers:
  - command:
    - kube-apiserver
    - --authorization-mode=Node,RBAC
    - --advertise-address=172.17.0.32
    - --allow-privileged=true
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --disable-admission-plugins=PersistentVolumeLabel
    - --enable-admission-plugins=NodeRestriction
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
    - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
    - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
    - --etcd-servers=https://127.0.0.1:2379
    - --insecure-port=0
    - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
    - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
    - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
    - --secure-port=6443
    - --service-account-key-file=/etc/kubernetes/pki/sa.pub
    - --service-cluster-ip-range=10.96.0.0/12
    - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
    - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
```

/etc/kubernetes/pki/apiserver.crt

```
▶ openssl x509 -in /etc/kubernetes/pki/apiserver.crt -text -noout
```

Certificate:

Data:

Version: 3 (0x2)

Serial Number: 3147495682089747350 (0x2bae26a58f090396)

Signature Algorithm: sha256WithRSAEncryption

Issuer: CN=kubernetes

Validity

Not Before: Feb 11 05:39:19 2019 GMT

Not After : Feb 11 05:39:20 2020 GMT

Subject: CN=kube-apiserver

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

Public-Key: (2048 bit)

Modulus:

00:d9:69:38:80:68:3b:b7:2e:9e:25:00:e8:fd:01:

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Key Usage: critical

Digital Signature, Key Encipherment

X509v3 Extended Key Usage:

TLS Web Server Authentication

X509v3 Subject Alternative Name:

DNS:master, DNS:kubernetes, DNS:kubernetes.default,  
DNS:kubernetes.default.svc, DNS:kubernetes.default.svc.cluster.local, IP  
Address:10.96.0.1, IP Address:172.17.0.27

# kubeadm

Certificate Path	CN Name	ALT Names	Organization	Issuer	Expiration
/etc/kubernetes/pki/apiserver.crt /etc/kubernetes/pki/apiserver.key	kube-apiserver	DNS:master DNS:kubernetes DNS:kubernetes.default DNS:kubernetes.default.svc IP Address:10.96.0.1 IP Address:172.17.0.27		kubernetes	Feb 11 05:39:20 2020
/etc/kubernetes/pki/ca.crt	kubernetes			kubernetes	Feb 8 05:39:19 2029
/etc/kubernetes/pki/apiserver-kubelet-client.crt /etc/kubernetes/pki/apiserver-kubelet-client.key	kube-apiserver-kubelet-client		system:masters	kubernetes	Feb 11 05:39:20 2020
/etc/kubernetes/pki/apiserver-etcd-client.crt /etc/kubernetes/pki/apiserver-etcd-client.key	kube-apiserver-etcd-client		system:masters	self	Feb 11 05:39:22 2020
/etc/kubernetes/pki/etcd/ca.crt	kubernetes			kubernetes	Feb 8 05:39:21 2017

Default CN	Parent CA	O (in Subject)	kind	hosts (SAN)
kube-etcd	etcd-ca		server, client [1][etcdbug]	localhost , 127.0.0.1
kube-etcd-peer	etcd-ca		server, client	<hostname> , <Host_IP> , localhost , 127.0.0.1
kube-etcd-healthcheck-client	etcd-ca		client	
kube-apiserver-etcd-client	etcd-ca	system:masters	client	
kube-apiserver	kubernetes-ca		server	<hostname> , <Host_IP> , <advertise_IP> , [1]
kube-apiserver-kubelet-client	kubernetes-ca	system:masters	client	
front-proxy-client	kubernetes-front-proxy-ca		client	

Default CN	recommend key path	recommended cert path	command	key argument	cert argument
etcd-ca		etcd/ca.crt	kube-apiserver		-etcd-cafile
etcd-client	apiserver-etcd-client.key	apiserver-etcd-client.crt	kube-apiserver	-etcd-keyfile	-etcd-certfile
kubernetes-ca		ca.crt	kube-apiserver		-client-ca-file
kube-apiserver	apiserver.key	apiserver.crt	kube-apiserver	-tls-private-key-file	-tls-cert-file
apiserver-kubelet-client		apiserver-kubelet-client.crt	kube-apiserver		-kubelet-client-certificate
front-proxy-ca		front-proxy-ca.crt	kube-apiserver		-requestheader-client-ca-file
front-proxy-client	front-proxy-client.key	front-proxy-client.crt	kube-apiserver	-proxy-client-key-file	-proxy-client-cert-file
etcd-ca		etcd/ca.crt	etcd		-trusted-ca-file, -peer-trusted-ca-file
kube-etcd	etcd/server.key	etcd/server.crt	etcd	-key-file	-cert-file
kube-etcd-peer	etcd/peer.key	etcd/peer.crt	etcd	-peer-key-file	-peer-cert-file
etcd-ca		etcd/ca.crt	etcdctl[2]		-cacert
kube-etcd-healthcheck-client	etcd/healthcheck-client.key	etcd/healthcheck-client.crt	etcdctl[2]	-key	-cert

# Inspect Service Logs

```
▶ journalctl -u etcd.service -l
```

```
2019-02-13 02:53:28.144631 I | etcdmain: etcd Version: 3.2.18
2019-02-13 02:53:28.144680 I | etcdmain: Git SHA: eddf599c6
2019-02-13 02:53:28.144684 I | etcdmain: Go Version: go1.8.7
2019-02-13 02:53:28.144688 I | etcdmain: Go OS/Arch: linux/amd64
2019-02-13 02:53:28.144692 I | etcdmain: setting maximum number of CPUs to 4, total number of available CPUs is 4
2019-02-13 02:53:28.144734 N | etcdmain: the server is already initialized as member before, starting as etcd
member...
2019-02-13 02:53:28.146625 I | etcdserver: name = master
2019-02-13 02:53:28.146637 I | etcdserver: data dir = /var/lib/etcd
2019-02-13 02:53:28.146642 I | etcdserver: member dir = /var/lib/etcd/member
2019-02-13 02:53:28.146645 I | etcdserver: heartbeat = 100ms
2019-02-13 02:53:28.146648 I | etcdserver: election = 1000ms
2019-02-13 02:53:28.146651 I | etcdserver: snapshot count = 10000
2019-02-13 02:53:28.146677 I | etcdserver: advertise client URLs = 2019-02-13 02:53:28.185353 I | etcdserver/api:
enabled capabilities for version 3.2
2019-02-13 02:53:28.185588 I | embed: ClientTLS: cert = /etc/kubernetes/pki/etcd/server.crt, key =
/etc/kubernetes/pki/etcd/server.key, ca = , trusted-ca = /etc/kubernetes/pki/etcd/old-ca.crt, client-cert-auth =
true
2019-02-13 02:53:30.080017 I | embed: ready to serve client requests
2019-02-13 02:53:30.080130 I | etcdserver: published {Name:master ClientURLs:[https://127.0.0.1:2379]} to cluster
c9be114fc2da2776
2019-02-13 02:53:30.080281 I | embed: serving client requests on 127.0.0.1:2379
WARNING: 2019/02/13 02:53:30 Failed to dial 127.0.0.1:2379: connection error: desc = "transport: authentication
handshake failed: remote error: tls: bad certificate"; please retry.
```



# View Logs

```
▶ kubectl logs etcd-master
```

```
2019-02-13 02:53:28.144631 I | etcdmain: etcd Version: 3.2.18
2019-02-13 02:53:28.144680 I | etcdmain: Git SHA: eddf599c6
2019-02-13 02:53:28.144684 I | etcdmain: Go Version: go1.8.7
2019-02-13 02:53:28.144688 I | etcdmain: Go OS/Arch: linux/amd64
2019-02-13 02:53:28.144692 I | etcdmain: setting maximum number of CPUs to 4, total number of available CPUs is 4
2019-02-13 02:53:28.144734 N | etcdmain: the server is already initialized as member before, starting as etcd
member...
2019-02-13 02:53:28.146625 I | etcdserver: name = master
2019-02-13 02:53:28.146637 I | etcdserver: data dir = /var/lib/etcd
2019-02-13 02:53:28.146642 I | etcdserver: member dir = /var/lib/etcd/member
2019-02-13 02:53:28.146645 I | etcdserver: heartbeat = 100ms
2019-02-13 02:53:28.146648 I | etcdserver: election = 1000ms
2019-02-13 02:53:28.146651 I | etcdserver: snapshot count = 10000
2019-02-13 02:53:28.146677 I | etcdserver: advertise client URLs = 2019-02-13 02:53:28.185353 I | etcdserver/api:
enabled capabilities for version 3.2
2019-02-13 02:53:28.185588 I | embed: ClientTLS: cert = /etc/kubernetes/pki/etcd/server.crt, key =
/etc/kubernetes/pki/etcd/server.key, ca = , trusted-ca = /etc/kubernetes/pki/etcd/old-ca.crt, client-cert-auth =
true
2019-02-13 02:53:30.080017 I | embed: ready to serve client requests
2019-02-13 02:53:30.080130 I | etcdserver: published {Name:master ClientURLs:[https://127.0.0.1:2379]} to cluster
c9be114fc2da2776
2019-02-13 02:53:30.080281 I | embed: serving client requests on 127.0.0.1:2379
WARNING: 2019/02/13 02:53:30 Failed to dial 127.0.0.1:2379: connection error: desc = "transport: authentication
handshake failed: remote error: tls: bad certificate"; please retry.
```

# View Logs

▶ docker ps -a

CONTAINER ID	STATUS	NAMES
23482a09f25b	Up 12 minutes	k8s_kube-apiserver_kube-apiserver-master_kube-system_8758a3d10776bb527e043f
b9bf77348c96	Up 18 minutes	k8s_etcd_etcd-master_kube-system_2cc1c8a24b68ab9b46bca47e153e74c6_0
87fc69913973	Up 18 minutes	k8s_POD_etcd-master_kube-system_2cc1c8a24b68ab9b46bca47e153e74c6_0
fda322157b86	Exited (255) 18 minutes ago	k8s_kube-apiserver_kube-apiserver-master_kube-system_8758a3d10776bb527e043f
0794bdfd57d8	Up 40 minutes	k8s_kube-scheduler_kube-scheduler-master_kube-system_009228e74aef4d7babd796
00f3f95d2102	Up 40 minutes	k8s_kube-controller-manager_kube-controller-manager-master_kube-system_ac1d4c5ae0f
b8e6a0e173dd	Up About an hour	k8s_weave_weave-net-8dzw_kube-system_22cd7993-2f2d-11e9-a2a6-0242ac110021_0
18e47bad320e	Up About an hour	k8s_weave-npc_weave-net-8dzw_kube-system_22cd7993-2f2d-11e9-a2a6-0242ac110021_0
4d087daf0380	Exited (1) About an hour ago	k8s_weave_weave-net-8dzw_kube-system_22cd7993-2f2d-11e9-a2a6-0242ac110021_0
e923140101a3	Up About an hour	k8s_kube-proxy_kube-proxy-cdmlz_kube-system_22cd267f-2f2d-11e9-a2a6-0242ac110021_0
e0db7e63d18e	Up About an hour	k8s_POD_weave-net-8dzw_kube-system_22cd7993-2f2d-11e9-a2a6-0242ac110021_0
74c257366f65	Up About an hour	k8s_POD_kube-proxy-cdmlz_kube-system_22cd267f-2f2d-11e9-a2a6-0242ac110021_0
8f514eac9d04	Exited (255) 40 minutes ago	k8s_kube-controller-manager_kube-controller-manager-master_kube-system_ac1d4c5ae0f
b39c5c594913	Exited (1) 40 minutes ago	k8s_kube-scheduler_kube-scheduler-master_kube-system_009228e74aef4d7babd796
3aefcb20ed30	Up 2 hours	k8s_POD_kube-apiserver-master_kube-system_8758a3d10776bb527e043fccfc835986_0
576c8a273b50	Up 2 hours	k8s_POD_kube-controller-manager-master_kube-system_ac1d4c5ae0f553b664a6c2_0
4b3c5f34efde	Up 2 hours	k8s_POD_kube-scheduler-master_kube-system_009228e74aef4d7babd7968782118d5e_0

# View Logs

▶ docker logs 87fc

```
2019-02-13 02:53:28.144631 I | etcdmain: etcd Version: 3.2.18
2019-02-13 02:53:28.144680 I | etcdmain: Git SHA: eddf599c6
2019-02-13 02:53:28.144684 I | etcdmain: Go Version: go1.8.7
2019-02-13 02:53:28.144688 I | etcdmain: Go OS/Arch: linux/amd64
2019-02-13 02:53:28.144692 I | etcdmain: setting maximum number of CPUs to 4, total number of available CPUs is 4
2019-02-13 02:53:28.144734 N | etcdmain: the server is already initialized as member before, starting as etcd member...
2019-02-13 02:53:28.146625 I | etcdserver: name = master
2019-02-13 02:53:28.146637 I | etcdserver: data dir = /var/lib/etcd
2019-02-13 02:53:28.146642 I | etcdserver: member dir = /var/lib/etcd/member
2019-02-13 02:53:28.146645 I | etcdserver: heartbeat = 100ms
2019-02-13 02:53:28.146648 I | etcdserver: election = 1000ms
2019-02-13 02:53:28.146651 I | etcdserver: snapshot count = 10000
2019-02-13 02:53:28.146677 I | etcdserver: advertise client URLs = 2019-02-13 02:53:28.185353 I | etcdserver/api: enabled capabilities for version 3.2
2019-02-13 02:53:28.185588 I | embed: ClientTLS: cert = /etc/kubernetes/pki/etcd/server.crt, key = /etc/kubernetes/pki/etcd/server.key, ca = , trusted-ca = /etc/kubernetes/pki/etcd/old-ca.crt, client-cert-auth = true
2019-02-13 02:53:30.080017 I | embed: ready to serve client requests
2019-02-13 02:53:30.080130 I | etcdserver: published {Name:master ClientURLs:[https://127.0.0.1:2379]} to cluster c9be114fc2da2776
2019-02-13 02:53:30.080281 I | embed: serving client requests on 127.0.0.1:2379
WARNING: 2019/02/13 02:53:30 Failed to dial 127.0.0.1:2379: connection error: desc = "transport: authentication handshake failed: remote error: tls: bad certificate"; please retry.
```

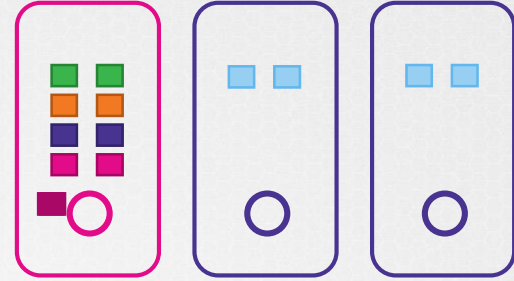


{K}ODE{K}LOUD

# TLS CERTIFICATES

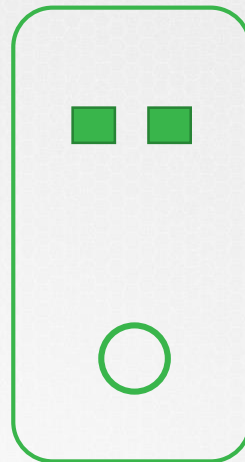
Certificate Workflow & API







## CERTIFICATE AUTHORITY (CA)

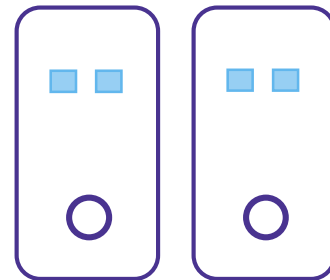
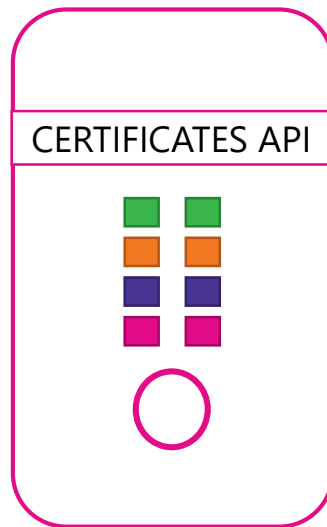
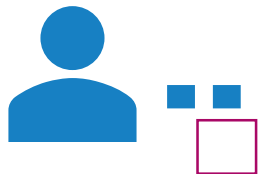


1. Create CertificateSigningRequest Object

2. Review Requests

3. Approve Requests

4. Share Certs to Users





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

```
jane.key
```

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

```
jane.csr
```

```
-----BEGIN CERTIFICATE REQUEST-----
MIICWDCCAUAQAQAwEzERMA8GA1UEAwwIbmV3LXVzZXIwggEiMA0GCSqGSIb3DQEB
AQUAA4IBDwAwggEKAoIBAQQD00WJW+DXsAJSInrjpNo5vRIBp1nZg+6xc9+UVwkKi0
LfC27t+1eEnON5Muq99NevmMEOnrDUO/thyVqP2w2XNIDRXjYyF40Fbmd+5zWyCK
9w0BAQsFAA0CAQEAS9iS6C1uxTuf5BBYSU7QFQUZa1NxAdYsaORRQNWHzHqGi4
hOK4a2zyNy14400ijyaD6tUW8DSxkr8BLK8Kg3srREtJq15rLZy9LRVrsJghD4gY
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF531esNSNMLQ2++RMnjDQJ7juPEic8/dhk
Wr2EUM6UawzykrdHIImwTv2m1MY0R+DntV1Yie+0H9/YE1t+FSGjh5L5YUvI1Dqiy
413E/y3qL71WfAcuH30sVpUUnQISMdQs0qWCsbE56CC5DhPGZIpUbnKUpAwka+8E
vwQ07jG+hpknxmuFAeXxgUwodALaJ7ju/TD1cw==
-----END CERTIFICATE REQUEST-----
```



jane.csr

```
-----BEGIN CERTIFICATE REQUEST-----
MIICWDCCAUAQAwEzERMA8GA1UEAwIbmV3LXVzZXIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQD00WJW+DXsAJSIrjpNo5vRIBplnzg+6xc9+UVwkKi0
LfC27t+1eEnON5Muq99NevmMEOnrDU0/thyVqP2w2XNIDRXjYyF40FbmD+5zWyCK
9w0BAQsFAA0CAQEAS9iS6C1uxTuf5BBYSU7QFQHUza1NxAdYsaORRQNwHZwHqGi4
hOK4a2zyNyI4400iJyaD6tUW8DSxkr8BLK8Kg3srRetJq15rLZy9LRVrsJghD4gY
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF531esNSNMLQ2++RMnjDQJ7juPEic8/dhk
Wr2EUM6UawzykrdHImwTv2mIMY0R+DNTV1Yie+0H9/YE1t+FSGjh5L5YUvI1Dqiy
413E/y3qL71WfAcuH30sVpUUNQISMdQs0qWCsbE56CC5DhPGZIpUbnKUpAwka+8E
vwQ07jG+hpknxmuFAeXgUwodALaJ7ju/TDIcw==
-----END CERTIFICATE REQUEST-----
```

jane-csr.yaml

```
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: jane
spec:
  groups:
    - system:authenticated
  usages:
    - digital signature
    - key encipherment
    - server auth
  request:
```

cat jane.csr | base64

```
LS0tLS1CRUdJTiBDRXJ0USUzZXIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQD00WJW+DXsAJSIrjpNo5vRIBplnzg+6xc9+UVwkKi0
LfC27t+1eEnON5Muq99NevmMEOnrDU0/thyVqP2w2XNIDRXjYyF40FbmD+5zWyCK
9w0BAQsFAA0CAQEAS9iS6C1uxTuf5BBYSU7QFQHUza1NxAdYsaORRQNwHZwHqGi4
hOK4a2zyNyI4400iJyaD6tUW8DSxkr8BLK8Kg3srRetJq15rLZy9LRVrsJghD4gY
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF531esNSNMLQ2++RMnjDQJ7juPEic8/dhk
Wr2EUM6UawzykrdHImwTv2mIMY0R+DNTV1Yie+0H9/YE1t+FSGjh5L5YUvI1Dqiy
413E/y3qL71WfAcuH30sVpUUNQISMdQs0qWCsbE56CC5DhPGZIpUbnKUpAwka+8E
vwQ07jG+hpknxmuFAeXgUwodALaJ7ju/TDIcw==
-----END CERTIFICATE REQUEST-----
```



```
▶ kubectl get csr
```

NAME	AGE	REQUESTOR	CONDITION
jane	10m	admin@example.com	Pending

```
▶ kubectl certificate approve jane
```

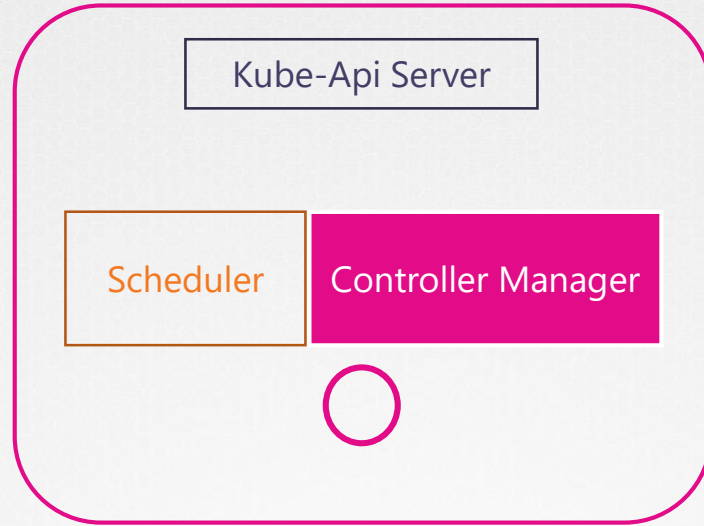
```
jane approved!
```

```
kubectl get csr jane -o yaml
```

```
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  creationTimestamp: 2019-02-13T16:36:43Z
  name: new-user
spec:
  groups:
  - system:masters
  - system:authenticated
usages:
  - digital signature
  - key encipherment
  - server auth
  username: kubernetes-admin
status:
  certificate:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSURDakNDQWZLZ0F3SUJBZ0lVRmwy
Q2wxYX0xawW15M3JNVisreFRYQUowU3dnd0RRWUpLb1pJaHZjTkFRRUwKQ1FBd0ZURVRN
QkVHQTFVRUF4TUthM1ZpWlhKdVpYUmxjekFlRncweE9UQXlNVE14TmPNeU1EQmFGd1dn
Y0ZFell2ajNuSXh3eFdDS1NIRm5sU041c0t5Z0VxUkwzTFM5V29GelhHZDdWcm1EZ2F0
MVRMRFBXTVhjN09FVnVjSwc1Yk4weEVHTkVwRU5tdU1BNlZWeHvjS1h6aG9ldDY0MEd1
MGU0YXFKWVlKwVMbjBvRTRFCY3dod2xic0I1ND0KLS0tLS1FTkQgQ0VSVElGSUNBVEUt
LS0tLQo=
  conditions:
  - lastUpdateTime: 2019-02-13T16:37:21Z
    message: This CSR was approved by kubectl certificate approve.
    reason: KubectlApprove
    type: Approved
```

```
echo "LS0...Qo=" | base64 --decode
```

```
-----BEGIN CERTIFICATE -----
MIICWDCCAUAQAQAwEzERMA8GA1UEAwwIbmV3LXVzZXIwgg
AQUAA4IBDwAwggEKAoIBAQD00WJW+DXsAJSIrjpNo5vRIB
LfC27t+1eEnON5Muq99NevmME0nrDU0/thyVqP2w2XNIDR
y3BiHHB93MJ70q13UTvZ8TELqyaDknRl/jv/SxgXkok0AB
IF5nxAttMVkDPQ7NbeZRG43b+QWlVGR/z6DW0fJnbfez0t
EcCXAwwqChjBLkz2BHPR4J89D6Xb8k39pu6jpyngV6uP0tI
j2qEL+hZEWkkFz80lNNtyT5LxMqENDCnIgwC4GZiRGbrAg
9w0BAQsFAAOCAQEAS9iS6C1uxTuf5BBYSU7QFQHUza1NxA
hOK4a2zyNyi4400ijyaD6tUW8DSxkr8BLK8Kg3srREtJq1
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF53lesNSNMLQ2++R
Wr2EUM6UawzykrdHImwTv2m1MY0R+DntV1Yie+0H9/YElT
4l3E/y3qL71WfAcuH30sVpUUnQISMdQs0qWCsbE56CC5Dh
vwQ07jG+hpknxmuFAeXxgUwodALaJ7ju/TDIcw==
-----END CERTIFICATE -----
```



## Controller Manager

CSR-APPROVING

CSR-SIGNING

```
cat /etc/kubernetes/manifests/kube-controller-manager.yaml
```

```
spec:
```

```
  containers:
```

```
  - command:
```

```
    - kube-controller-manager
```

```
    - --address=127.0.0.1
```

```
    - --cluster-signing-cert-file=/etc/kubernetes/pki/ca.crt
```

```
    - --cluster-signing-key-file=/etc/kubernetes/pki/ca.key
```

```
    - --controllers=*,bootstrapsigner,tokencleaner
```

```
    - --kubeconfig=/etc/kubernetes/controller-manager.conf
```

```
    - --leader-elect=true
```

```
    - --root-ca-file=/etc/kubernetes/pki/ca.crt
```

```
    - --service-account-private-key-file=/etc/kubernetes/pki/sa.key
```

```
    - --use-service-account-credentials=true
```



## Client Certificates for Clients



kube-scheduler

kube-controller-  
manager

- CA CERT for Cluster Signing
- CA KEY for Cluster Signing
- CA CERT
- KEY for SERVICE ACCOUNT

kubelet

kube-proxy

## Server Certificates for Servers

kube-apiserver

Kube-api.service

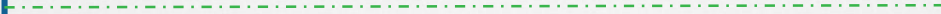
- CA CERT for ETCD
- CERT for ETCD
- KEY for ETCD
- CA CERT for KUBELET
- CERT for KUBELET CLIENT
- KEY for KUBELET CLIENT
- CERT for Service Account
- CERT for TLS
- KEY for TLS

kubelet

Kubelet-config.yaml

- tlsCertFile
- tlsPrivateKeyFile

ETCD  
CLUSTER



# Provision Certificate Authority



ca-key.pem



ca.pem

kube-apiserver



kubernetes-key.pem



kubernetes.pem



admin user



admin-key.pem



admin.pem

kube-controller-manager



kcm-key.pem



kcm.pem

kube-scheduler



Kube-scheduler-key.pem



Kube-scheduler.pem

Kube-proxy



kube-proxy-key.pem



kube-proxy.pem

kubelet



worker01-key.pem



worker01.pem



worker02-key.pem



worker02.pem

Default CN	Parent CA	O (in Subject)	kind	hosts (SAN)
kube-etcd	etcd-ca		server, client [1][etcdbug]	localhost , 127.0.0.1
kube-etcd-peer	etcd-ca		server, client	<hostname> , <Host_IP> , localhost , 127.0.0.1
kube-etcd-healthcheck-client	etcd-ca		client	
kube-apiserver-etcd-client	etcd-ca	system:masters	client	
kube-apiserver	kubernetes-ca		server	<hostname> , <Host_IP> , <advertise_IP> , [1]
kube-apiserver-kubelet-client	kubernetes-ca	system:masters	client	
front-proxy-client	kubernetes-front-proxy-ca		client	

Default CN	recommend key path	recommended cert path	command	key argument	cert argument
etcd-ca		etcd/ca.crt	kube-apiserver		-etcd-cafile
etcd-client	apiserver-etcd-client.key	apiserver-etcd-client.crt	kube-apiserver	-etcd-keyfile	-etcd-certfile
kubernetes-ca		ca.crt	kube-apiserver		-client-ca-file
kube-apiserver	apiserver.key	apiserver.crt	kube-apiserver	-tls-private-key-file	-tls-cert-file
apiserver-kubelet-client		apiserver-kubelet-client.crt	kube-apiserver		-kubelet-client-certificate
front-proxy-ca		front-proxy-ca.crt	kube-apiserver		-requestheader-client-ca-file
front-proxy-client	front-proxy-client.key	front-proxy-client.crt	kube-apiserver	-proxy-client-key-file	-proxy-client-cert-file
etcd-ca		etcd/ca.crt	etcd		-trusted-ca-file, -peer-trusted-ca-file
kube-etcd	etcd/server.key	etcd/server.crt	etcd	-key-file	-cert-file
kube-etcd-peer	etcd/peer.key	etcd/peer.crt	etcd	-peer-key-file	-peer-cert-file
etcd-ca		etcd/ca.crt	etcdctl[2]		-cacert
kube-etcd-healthcheck-client	etcd/healthcheck-client.key	etcd/healthcheck-client.crt	etcdctl[2]	-key	-cert

# Configure certificates for user accounts

You must manually configure these administrator account and service accounts:

filename	credential name	Default CN	O (in Subject)
admin.conf	default-admin	kubernetes-admin	system:masters
kubelet.conf	default-auth	system:node: <b>&lt;nodeName&gt;</b> (see note)	system:nodes
controller-manager.conf	default-controller-manager	system:kube-controller-manager	
scheduler.conf	default-manager	system:kube-scheduler	



Component	Certificate	Purpose
API server	Cluster CA	Authenticate clients, TLS
API server	Etcd CA	Etcd server authentication
API server	Etcd client cert	Etcd client authentication
API server	Serving certificate	Serving API over HTTPS
API server	Kubelet client cert	Authenticating against Kubelet
Controller Manager	Client certificate	Authenticating against API server
Controller Manager	Cluster CA	Embedding in service account secrets
Scheduler	Client certificate	Authenticating against API server
Kubelet	Serving certificate	Serving API over HTTPS
Kubelet	Client certificate	Authenticating against API server
Kubelet	Cluster CA	Authenticating clients
Kube Proxy	Client certificate	Authenticating against API server

```
- kube-apiserver
- --authorization-mode=Node,RBAC
- --advertise-address=172.17.0.18
- --allow-privileged=true
- --client-ca-file=/etc/kubernetes/pki/ca.crt
- --disable-admission-plugins=PersistentVolumeLabel
- --enable-admission-plugins=NodeRestriction
- --enable-bootstrap-token-auth=true
- --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
- --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
- --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
- --etcd-servers=https://127.0.0.1:2379
- --insecure-port=0
- --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
- --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
- --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
- --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
- --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
- --requestheader-allowed-names=front-proxy-client
- --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
- --requestheader-extra-headers-prefix=X-Remote-Extra-
- --requestheader-group-headers=X-Remote-Group
- --requestheader-username-headers=X-Remote-User
- --secure-port=6443
- --service-account-key-file=/etc/kubernetes/pki/sa.pub
- --service-cluster-ip-range=10.96.0.0/12
- --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
- --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
```

```
openssl genrsa -out old-ca.key 2048
openssl req -new -key old-ca.key -subj "/CN=old-ca" -out old-ca.csr
openssl x509 -req -in old-ca.csr -signkey old-ca.key -out old-ca.crt -days 365

openssl x509 -req -in ca.csr -signkey ca.key -out server.crt -days 365

openssl req -new -key apiserver-kubelet-client.key -out apiserver-kubelet-client.csr -subj "/CN=kube-apiserver-kubelet-client/O=system:masters"

openssl req -new -key apiserver-kubelet-client.key -out apiserver-kubelet-client.csr -subj "/CN=kube-apiserver-kubelet-client/O=system:masters"
openssl x509 -req -in apiserver-kubelet-client.csr -CA /root/new-ca/old-ca.crt -CAkey /root/new-ca/old-ca.key -CAcreateserial -out apiserver-kubelet-client-new.crt -days 365

openssl req -new -key apiserver-etcd-client.key -out apiserver-etcd-client.csr -subj "/CN=kube-apiserver-etcd-client/O=system:masters"
openssl x509 -req -in apiserver-etcd-client.csr -CA /root/new-ca/old-ca.crt -CAkey /root/new-ca/old-ca.key -CAcreateserial -out apiserver-etcd-client-new.crt -days 365

openssl req -new -key apiserver-etcd-client.key -out apiserver-etcd-client.csr -subj "/CN=kube-apiserver-etcd-client/O=system:masters"
openssl x509 -req -in apiserver-etcd-client.csr -CA /root/new-ca/old-ca.crt -CAkey /root/new-ca/old-ca.key -CAcreateserial -out apiserver-etcd-client-new.crt -days 365

openssl req -new -key /etc/kubernetes/pki/apiserver-etcd-client.key -out apiserver-etcd-client.csr -subj "/CN=kube-apiserver-etcd-client/O=system:masters"

openssl x509 -req -in apiserver-etcd-client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out apiserver-etcd-client.crt -days -10

openssl x509 -req -in apiserver-etcd-client.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out apiserver-etcd-client.crt -startdate 190101010101Z
20170101000000Z
200801010000Z

"openssl", "req", "-new", "-key", "/etc/kubernetes/pki/apiserver-etcd-client.key", "-out", "/etc/kubernetes/pki/apiserver-etcd-client.csr", "-subj", "/CN=kube-apiserver-etcd-client/O=system:masters"

"openssl", "x509", "-req", "-in", "/etc/kubernetes/pki/apiserver-etcd-client.csr", "-CA", "/etc/kubernetes/pki/etcd/ca.crt", "-CAkey", "/etc/kubernetes/pki/etcd/ca.key", "-CAcreateserial", "-out",
"/etc/kubernetes/pki/apiserver-etcd-client.crt"

openssl x509 -req -in /etc/kubernetes/pki/apiserver-etcd-client.csr -CA /etc/kubernetes/pki/etcd/ca.crt -CAkey /etc/kubernetes/pki/etcd/ca.key -CAcreateserial -out
/etc/kubernetes/pki/apiserver-etcd-client.crt -days 100

openssl x509 -req -in apiserver.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out apiserver.crt
```



{K}ODE{K}LOUD

# Security

# KUBECONFIG



```
▶ curl https://my-kube-playground:6443/api/v1/pods \
  --key admin.key
  --cert admin.crt
  --cacert ca.crt
```

```
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/pods",
  },
  "items": []
}
```

```
▶ kubectl get pods
  --server my-kube-playground:6443
  --client-key admin.key
  --client-certificate admin.crt
  --certificate-authority ca.crt
```

No resources found.



\$HOME/.kube/config

### KubeConfig File

```
--server my-kube-playground:6443  
--client-key admin.key  
--client-certificate admin.crt  
--certificate-authority ca.crt
```

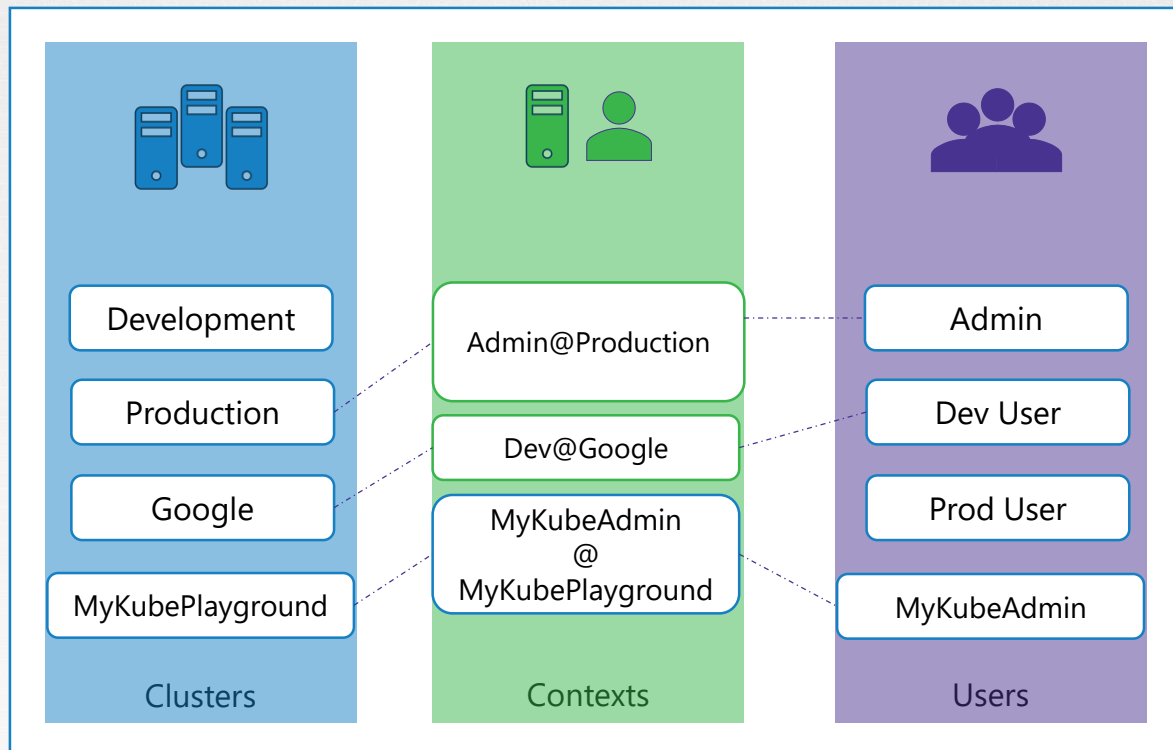
```
▶ kubectl get pods  
   --kubeconfig config
```

No resources found.

# KubeConfig File

\$HOME/.kube/config

```
--server my-kube-playground:6443  
--client-key admin.key  
--client-certificate admin.crt  
--certificate-authority ca.crt
```



# KubeConfig File

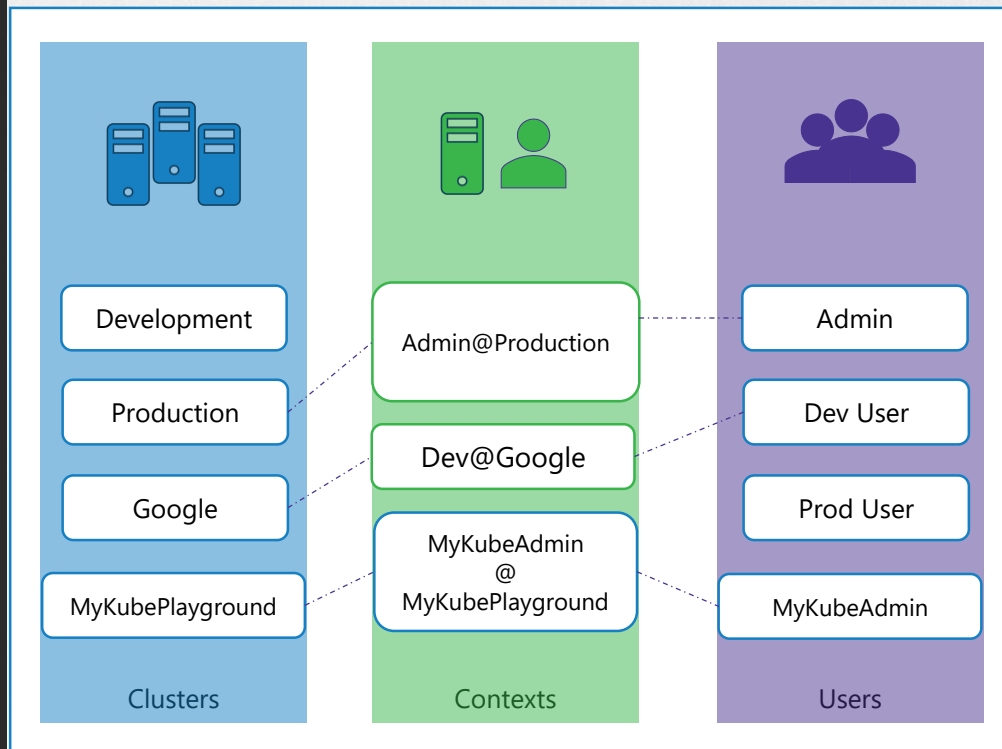
```
apiVersion: v1
kind: Config

clusters:
- name: my-kube-playground
  cluster:
    certificate-authority: ca.crt
    server: https://my-kube-playground:6443

contexts:
- name: my-kube-admin@my-kube-playground
  context:
    cluster:
    user:

users:
- name: my-kube-admin
  user:
    client-certificate: admin.crt
    client-key: admin.key
```

\$HOME/.kube/config



# KubeConfig File

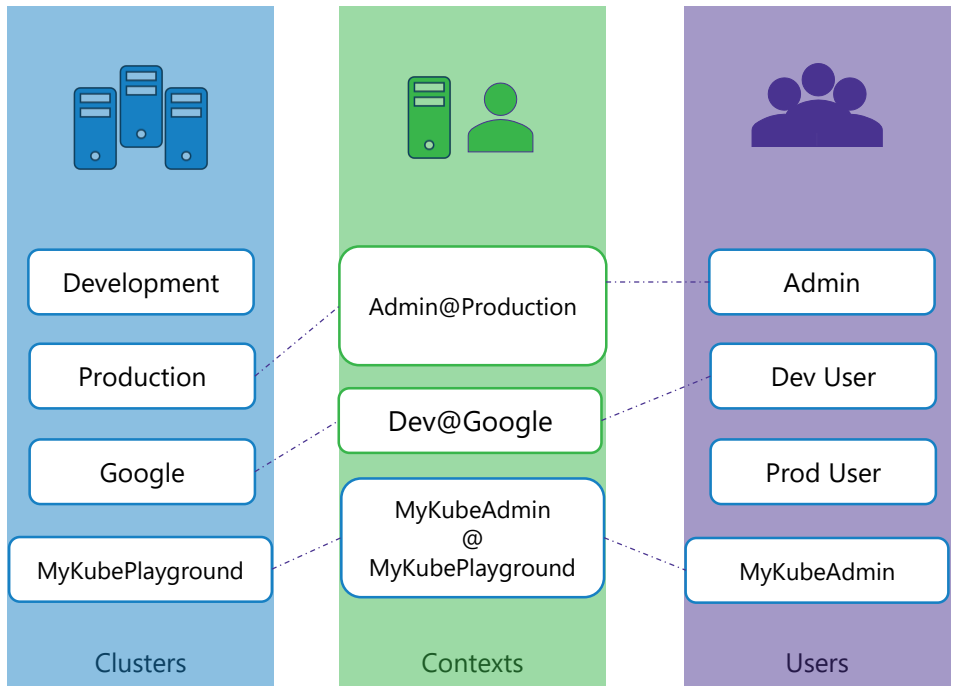
```
apiVersion: v1
kind: Config
current-context: dev-user@google

clusters:
- name: my-kube-playground (values hidden...)
- name: development
- name: production
- name: google

contexts:
- name: my-kube-admin@my-kube-playground
- name: dev-user@google
- name: prod-user@production

users:
- name: my-kube-admin
- name: admin
- name: dev-user
- name: prod-user
```

\$HOME/.kube/config



# Kubectl config

```
▶ kubectl config view
```

```
apiVersion: v1

kind: Config
current-context: kubernetes-admin@kubernetes

clusters:
- cluster:
    certificate-authority-data: REDACTED
    server: https://172.17.0.5:6443
    name: kubernetes

contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
    name: kubernetes-admin@kubernetes

users:
- name: kubernetes-admin
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
```

```
▶ kubectl config view --kubeconfig=my-custom-config
```

```
apiVersion: v1

kind: Config
current-context: my-kube-admin@my-kube-playground

clusters:
- name: my-kube-playground
- name: development
- name: production

contexts:
- name: my-kube-admin@my-kube-playground
- Name: prod-user@production

users:
- name: my-kube-admin
- name: prod-user
```

# Kubectl config

▶ kubectl config view

```
apiVersion: v1

kind: Config
current-context: my-kube-admin@my-kube-playground

clusters:
- name: my-kube-playground
- name: development
- name: production

contexts:
- name: my-kube-admin@my-kube-playground
- Name: prod-user@production

users:
- name: my-kube-admin
- name: prod-user
```

▶ kubectl config use-context prod-user@production

```
apiVersion: v1

kind: Config
current-context: prod-user@production

clusters:
- name: my-kube-playground
- name: development
- name: production

contexts:
- name: my-kube-admin@my-kube-playground
- Name: prod-user@production

users:
- name: my-kube-admin
- name: prod-user
```



# Kubectl config

```
▶ kubectl config -h
```

## Available Commands:

current-context	Displays the current-context
delete-cluster	Delete the specified cluster from the kubeconfig
delete-context	Delete the specified context from the kubeconfig
get-clusters	Display clusters defined in the kubeconfig
get-contexts	Describe one or many contexts
rename-context	Renames a context from the kubeconfig file.
set	Sets an individual value in a kubeconfig file
set-cluster	Sets a cluster entry in kubeconfig
set-context	Sets a context entry in kubeconfig
set-credentials	Sets a user entry in kubeconfig
unset	Unsets an individual value in a kubeconfig file
use-context	Sets the current-context in a kubeconfig file
view	Display merged kubeconfig settings or a specified kubeconfig file

# Namespaces

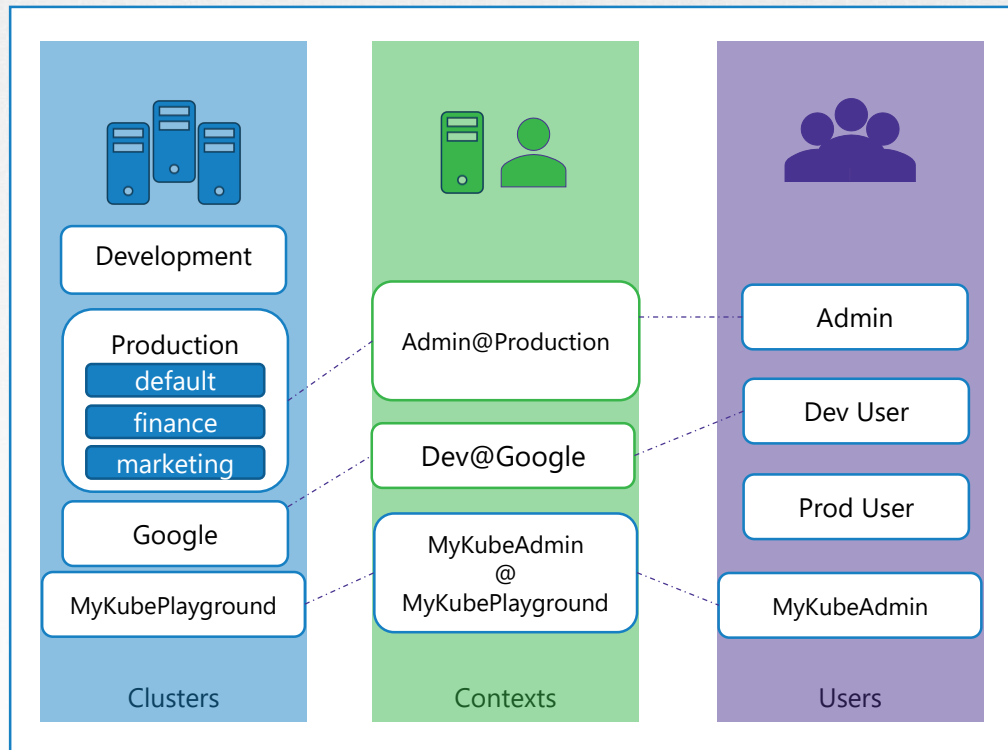
```
apiVersion: v1
kind: Config

clusters:
- name: production
  cluster:
    certificate-authority: ca.crt
    server: https://172.17.0.51:6443

contexts:
- name: admin@production
  context:
    cluster: production
    user: admin
    namespace: finance

users:
- name: admin
  user:
    client-certificate: admin.crt
    client-key: admin.key
```

\$HOME/.kube/config



# | Certificates in KubeConfig

```
apiVersion: v1
kind: Config

clusters:
- name: production
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt
    server: https://172.17.0.51:6443

contexts:
- name: admin@production
  context:
    cluster: production
    user: admin
    namespace: finance

users:
- name: admin
  user:
    client-certificate: /etc/kubernetes/pki/users/admin.crt
    client-key: /etc/kubernetes/pki/users/admin.key
```

# Certificates in KubeConfig

```
apiVersion: v1
kind: Config

clusters:
- name: production
  cluster:
    certificate-authority: /etc/kubernetes/pki/ca.crt

    certificate-authority-data:
```

-----BEGIN CERTIFICATE -----

```
MIICWDCCAUAQAQAwEzERMA8GA1UEAwwIbmV3LXVzZXIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQD00WJW+DXsAJSIrjpNo5vRIBpInzg+6
LFC27t+1eEnON5Muq99NevmMEOnrDU0/thyVqP2w2XNIDRXjYyF40
y3BiHhB93MJ70q13UTvZ8TELqyaDknR1/jv/SxgXkok0ABUTpWMx4
IF5nxAttMVkDPQ7NbeZRG43b+QW1VGR/z6DW0fJnbfez0taAydGLT
EcCXAqwChjBLkz2BHPR4J89D6Xb8k39pu6jpyngV6uP0tIb0zpqNv
j2qEL+hZEwkkFz801NNtyT5LxMqENDCnIgwC4GZiRGbrAgMBAAGGA
9w0BAQsFAAOCAQEAS9iS6C1uxTuf5BBYSU7QFQHUza1NxAdYsaORR
hOK4a2zyNy14400ijyaD6tUW8DSxkr8BLK8Kg3srREtJq15rLzy9L
P9NL+aDRSxROVSqBaB2nWeYpM5cJ5TF531esNSNMLQ2++RMnjDQJ7
Wr2EUM6UawzykrdHImwTv2m1MY0R+DNTV1Yie+0H9/YE1t+FSGjh5
4l3E/y3qL71WfAcuH30sVpUUNQISMdQs0qWCsbE56CC5DhPGZIpU
vwQ07jG+hpknxmuFAeXxgUwodALaJ7ju/TDIcw==
-----END CERTIFICATE -----
```

cat ca.crt | base64

```
LS0tLS1CRUdJTiBDRVJ1SUZ1Q0FURSBSRVFVRVNN
tLS0KTU1JQ1dE00NBVUFDOVFBRD0V6RVJNOThHOT
F3d0libVYzTFhwe1pYSXdnZ0VpTUFWR0NTcUdTS
FFFQgnBUVVB0TRJ0krR30XdZ0VlOW9J0kFRRE8w
K0RYc0FKU01yanB0bzV2UklCcGxuemcrNnhjOST
rS2kwCkxmQzI3dCsxZUVuT041TXVxOT10ZXZtTU
U01yanB0bzV2UklCcGxuemcrNnhjOSTV
VndrS2kwCkxmQzI3dCsxZUVuT041TXVx
```



{K}ODE{K}LOUD





# Course Objectives

Core Concepts

Scheduling

Logging Monitoring

Application Lifecycle Management

Cluster Maintenance

Security

○ Kubernetes Security Primitives

○ Secure Persistent Key Value Store

○ Authentication

○ Authorization

○ Security Contexts

○ TLS Certificates for Cluster Components

○ Images Securely

○ Network Policies

Storage

Networking

Installation, Configuration & Validation

Troubleshooting

# API Groups

## Pre-Requisite

```
▶ curl https://kube-master:6443/version
```

```
{
  "major": "1",
  "minor": "13",
  "gitVersion": "v1.13.0",
  "gitCommit": "ddf47ac13c1a9483ea035a79cd7c1005ff21a6d",
  "gitTreeState": "clean",
  "buildDate": "2018-12-03T20:56:12Z",
  "goVersion": "go1.11.2",
  "compiler": "gc",
  "platform": "linux/amd64"
}
```

```
▶ curl https://kube-master:6443/api/v1/pods
```

```
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/pods",
    "resourceVersion": "153068"
  },
  "items": [
    {
      "metadata": {
        "name": "nginx-5c7588df-ghsbd",
        "generateName": "nginx-5c7588df-",
        "namespace": "default",
        "creationTimestamp": "2019-03-20T10:57:48Z",
        "labels": {
          "app": "nginx",
          "pod-template-hash": "5c7588df"
        },
        "ownerReferences": [
          {
            "apiVersion": "apps/v1",
            "kind": "ReplicaSet",
            "name": "nginx-5c7588df",
            "uid": "398ce179-4af9-11e9-beb6-020d3114c7a7",
            "controller": true,
            "blockOwnerDeletion": true
          }
        ]
      },
      "spec": {
        "volumes": [
          {
            "name": "default-token-8p888",
            "secretName": "default-token-8p888"
          }
        ],
        "containers": [
          {
            "name": "nginx",
            "image": "nginx:1.15",
            "ports": [
              {
                "containerPort": 80
              }
            ]
          }
        ]
      },
      "status": {
        "phase": "Running",
        "ip": "10.0.1.15",
        "podIP": "10.0.1.15"
      }
    }
  ]
}
```

/metrics

/healthz

/version

/api

/apis

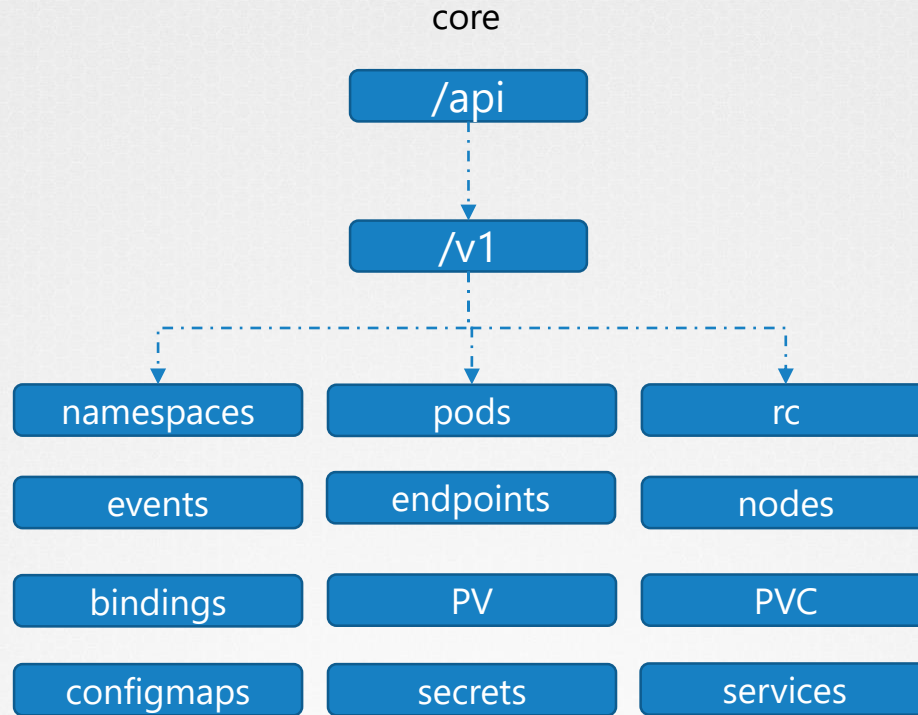
/logs

core

/api

named

/apis





named

/apis

API Groups

/apps

/extensions

/networking.k8s.io

/storage.k8s.io

/authentication.k8s.io

/certificates.k8s.io

/v1

/v1

/v1

/deployments

/replicasets

/statefulsets

list

get

create

delete

update

watch

/networkpolicies

/certificatesigningrequests

Resources

Verbs

## Overview

### WORKLOADS APIS

Container v1 core

CronJob v1beta1 batch

DaemonSet v1 apps

Deployment v1 apps

Job v1 batch

### Pod v1 core

Write Operations

Read Operations

Status Operations

Proxy Operations

Misc Operations

ReplicaSet v1 apps

ReplicationController v1 core

StatefulSet v1 apps

# Pod v1 core

kubectl example

curl example

Group	Version
core	v1

### ⚠ Warning:

It is recommended that users create Pods only through a Controller, and not directly. See Controllers: [Deploy](#)

### ℹ Appears In:

- PodList [core/v1]

Field	Description
<code>apiVersion</code> <i>string</i>	APIVersion defines the versioned schema of this representation of an object. Servers should c <a href="https://git.k8s.io/community/contributors/devel/api-conventions.md#resources">https://git.k8s.io/community/contributors/devel/api-conventions.md#resources</a>

```
▶ curl http://localhost:6443 -k
```

```
{  
  "paths": [  
    "/api",  
    "/api/v1",  
    "/apis",  
    "/apis/",  
    "/healthz",  
    "/logs",  
    "/metrics",  
    "/openapi/v2",  
    "/swagger-2.0.0.json",  
  ],  
}
```

```
▶ curl http://localhost:6443/apis -k | grep "name"
```

```
  "name": "extensions",  
  "name": "apps",  
  "name": "events.k8s.io",  
  "name": "authentication.k8s.io",  
  "name": "authorization.k8s.io",  
  "name": "autoscaling",  
  "name": "batch",  
  "name": "certificates.k8s.io",  
  "name": "networking.k8s.io",  
  "name": "policy",  
  "name": "rbac.authorization.k8s.io",  
  "name": "storage.k8s.io",  
  "name": "admissionregistration.k8s.io",  
  "name": "apiextensions.k8s.io",  
  "name": "scheduling.k8s.io",
```



Kube ApiServer



```
► curl http://localhost:6443 -k
```

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {

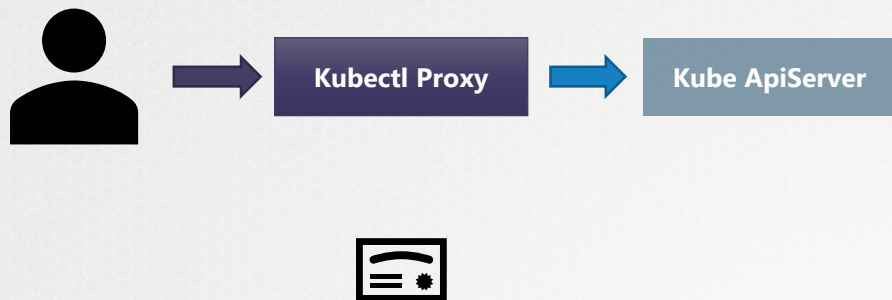
  },
  "code": 403
}
```

```
► curl http://localhost:6443 -k
```

```
--key admin.key
--cert admin.crt
--cacert ca.crt
```

```
{
  "paths": [
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/healthz",
    "/logs",
    "/metrics"
```

# kubectl proxy



```
▶ kubectl proxy
```

```
Starting to serve on 127.0.0.1:8001
```

```
▶ curl http://localhost:8001 -k
```

```
{
  "paths": [
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/healthz",
    "/logs",
    "/metrics",
    "/openapi/v2",
    "/swagger-2.0.0.json",
```

Kube proxy



Kubectl proxy



# Key Takeaways

named

/apis

## API Groups

/apps   /extensions   /networking.k8s.io   /storage.k8s.io   /authentication.k8s.io   /certificates.k8s.io

/v1

/v1

/v1

/deployments

/replicasets

/statefulsets

/networkpolicies

/certificatesigningrequests

list

get

create

delete

update

watch

Resources

Verbs



{K}ODE{K}LOUD

# Course Objectives

Core Concepts

Scheduling

Logging Monitoring

Application Lifecycle Management

Cluster Maintenance

Security

○ Kubernetes Security Primitives

○ Secure Persistent Key Value Store

○ Authentication

○ Authorization

○ Security Contexts

○ TLS Certificates for Cluster Components

○ Images Securely

○ Network Policies

Storage

Networking

Installation, Configuration & Validation

Troubleshooting

# AUTHORIZATION

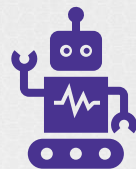
# Why Authorization?



Admins



Developers



Bots

```
▶ kubectl get pods
```

NAME	STATUS	ROLES	AGE	VERSION
worker-1	Ready	<none>	5d21h	v1.13.0
worker-2	Ready	<none>	5d21h	v1.13.0

```
▶ kubectl get pods
```

NAME	STATUS	ROLES	AGE	VERSION
worker-1	Ready	<none>	5d21h	v1.13.0
worker-2	Ready	<none>	5d21h	v1.13.0

```
▶ kubectl get pods
```

Error from server (**Forbidden**): nodes "worker-1" is **forbidden**: User "Bot-1" **delete resource** "nodes"

```
▶ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
worker-1	Ready	<none>	5d21h	v1.13.0
worker-2	Ready	<none>	5d21h	v1.13.0

```
▶ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
worker-1	Ready	<none>	5d21h	v1.13.0
worker-2	Ready	<none>	5d21h	v1.13.0

```
▶ kubectl get nodes
```

Error from server (**Forbidden**): nodes "worker-1" is **forbidden**: User "Bot-1" **delete resource** "nodes"

```
▶ kubectl delete node worker-2
```

Node worker-2 Deleted!

```
▶ kubectl delete node worker-2
```

Error from server (**Forbidden**): nodes "worker-1" is **forbidden**: User "developer" **cannot delete resource** "nodes"

```
▶ kubectl delete node worker
```

Error from server (**Forbidden**): nodes "worker-1" is **forbidden**: User "Bot-1" **delete resource** "nodes"

# | Authorization Mechanisms

Node

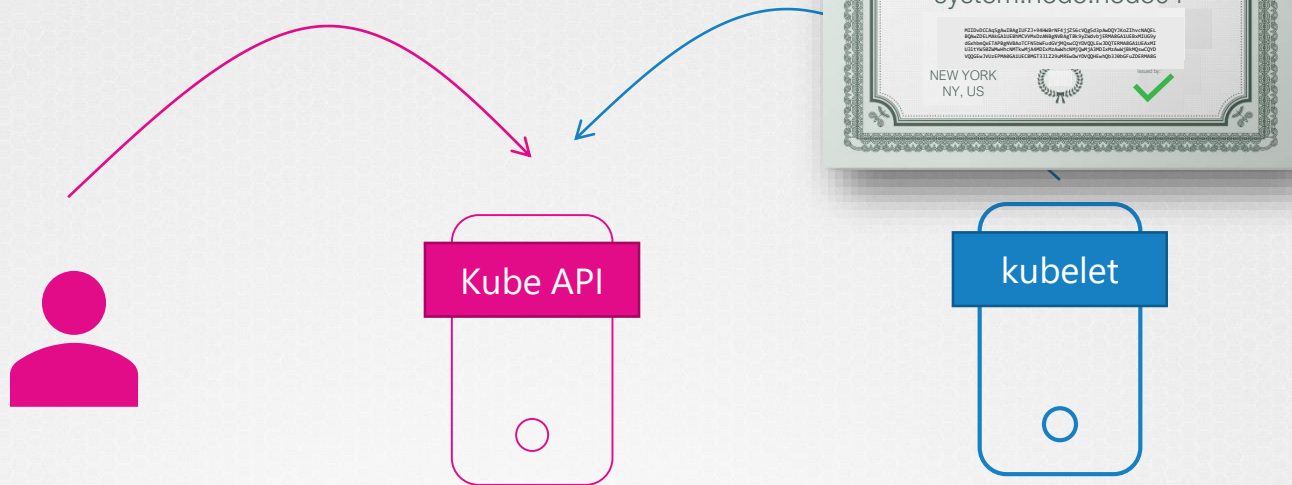
ABAC

RBAC

Webhook



# Node Authorizer



- Read
  - Services
  - Endpoints
  - Nodes
  - Pods
- Write
  - Node status
  - Pod status
  - events

# ABAC



dev-user



- ✓ Can view PODs
- ✓ Can create PODs
- ✓ Can Delete PODs

```
{"kind": "Policy", "spec": {"user": "dev-user", "namespace": "*", "resource": "pods", "apiGroup": "*"}}
```

# ABAC



dev-user



- ✓ Can view PODs
- ✓ Can create PODs
- ✓ Can Delete PODs



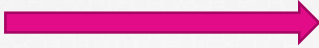
dev-user-2



- ✓ Can view PODs
- ✓ Can create PODs
- ✓ Can Delete PODs



dev-users



- ✓ Can view PODs
- ✓ Can create PODs
- ✓ Can Delete PODs



security-1



- ✓ Can view CSR
- ✓ Can approve CSR

```
{ "kind": "Policy", "spec": { "user": "dev-user", "namespace": "*", "resource": "pods", "apiGroup": "*" } }  
{ "kind": "Policy", "spec": { "user": "dev-user-2", "namespace": "*", "resource": "pods", "apiGroup": "*" } }  
{ "kind": "Policy", "spec": { "group": "dev-users", "namespace": "*", "resource": "pods", "apiGroup": "*" } }  
{ "kind": "Policy", "spec": { "user": "security-1", "namespace": "*", "resource": "csr", "apiGroup": "*" } }
```

# RBAC



dev-user



dev-user-2



dev-users



security-1



- ✓ Can view PODs
- ✓ Can create PODs
- ✓ Can Delete PODs
- ✓ Can Create ConfigMaps

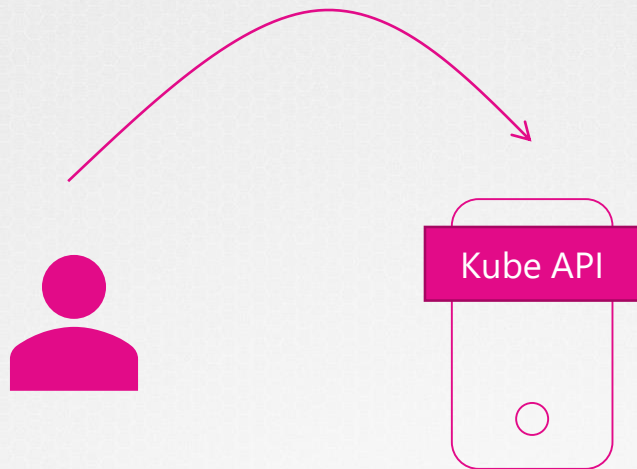
Developer



- ✓ Can view CSR
- ✓ Can approve CSR

Security

# Webhook



User **dev-user**  
requested read  
access to **Pods**.  
Should I allow?



Open Policy Agent

I checked. Yes!

**{CODE}{CLOUD}**



# | Authorization Mode

NODE

ABAC

RBAC

WEBHOOK

AlwaysAllow

AlwaysDeny



# Authorization Mode

AlwaysAllow

NODE

ABAC

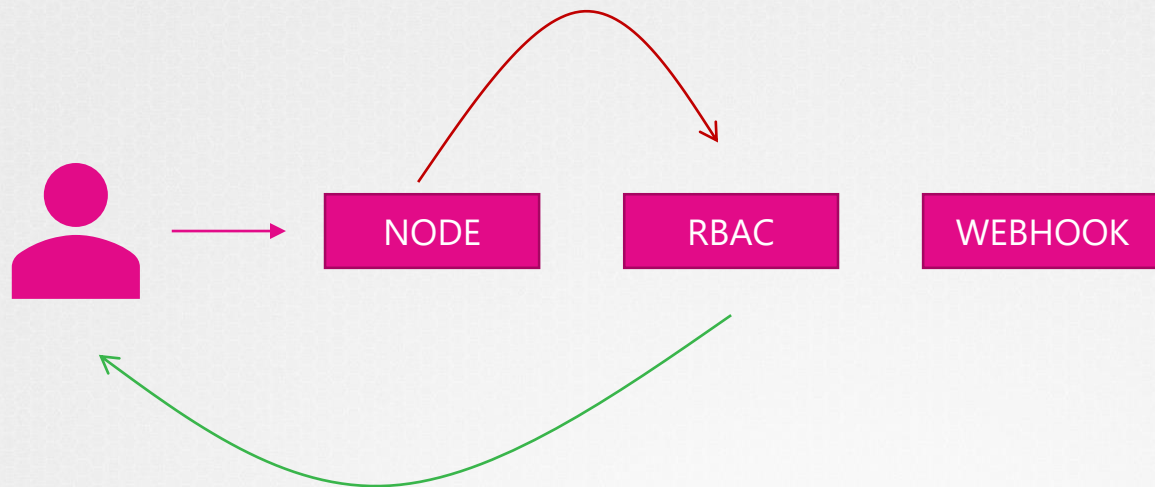
RBAC

WEBHOOK

AlwaysDeny

```
ExecStart=/usr/local/bin/kube-apiserver \\  
  --advertise-address=${INTERNAL_IP} \\  
  --allow-privileged=true \\  
  --apiserver-count=3 \\  
  --authorization-mode=Node,RBAC,Webhook \\  
  --bind-address=0.0.0.0 \\  
  --enable-swagger-ui=true \\  
  --etcd-cafile=/var/lib/kubernetes/ca.pem \\  
  --etcd-certfile=/var/lib/kubernetes/apiserver-etcd-client.crt \\  
  --etcd-keyfile=/var/lib/kubernetes/apiserver-etcd-client.key \\  
  --etcd-servers=https://127.0.0.1:2379 \\  
  --event-ttl=1h \\  
  --kubelet-certificate-authority=/var/lib/kubernetes/ca.pem \\  
  --kubelet-client-certificate=/var/lib/kubernetes/apiserver-etcd-client.crt \\  
  --kubelet-client-key=/var/lib/kubernetes/apiserver-etcd-client.key \\  
  --service-node-port-range=30000-32767 \\  
  --client-ca-file=/var/lib/kubernetes/ca.pem \\  
  --tls-cert-file=/var/lib/kubernetes/apiserver.crt \\  
  --tls-private-key-file=/var/lib/kubernetes/apiserver.key \\  
  --v=2
```

# Authorization Mode



```
ExecStart=/usr/local/bin/kube-apiserver \\  
  --advertise-address=${INTERNAL_IP} \\  
  --allow-privileged=true \\  
  --apiserver-count=3 \\  
  --authorization-mode=Node,RBAC,Webhook \\  
  --bind-address=0.0.0.0 \\  

```



{K}ODE{K}LOUD

# Course Objectives

Core Concepts

Scheduling

Logging Monitoring

Application Lifecycle Management

Cluster Maintenance

Security

○ Kubernetes Security Primitives

○ Secure Persistent Key Value Store

○ Authentication

○ Authorization

○ Security Contexts

○ TLS Certificates for Cluster Components

○ Images Securely

○ Network Policies

Storage

Networking

Installation, Configuration & Validation

Troubleshooting

# RBAC



# RBAC



- ✓ Can view PODs
- ✓ Can create PODs
- ✓ Can Delete PODs
- ✓ Can Create ConfigMaps

Developer

developer-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["list", "get", "create", "update", "delete"]
- apiGroups: [""]
  resources: ["ConfigMap"]
  verbs: ["create"]
```



```
kubectl create -f developer-role.yaml
```



# RBAC



dev-user



- ✓ Can view PODs
- ✓ Can create PODs
- ✓ Can Delete PODs
- ✓ Can Create ConfigMaps

Developer

Namespace: default

```
▶ kubectl create -f devuser-developer-binding.yaml
```

developer-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["list", "get", "create", "update", "delete"]
- apiGroups: [""]
  resources: ["ConfigMap"]
  verbs: ["create"]
```

devuser-developer-binding.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: devuser-developer-binding
subjects:
- kind: User
  name: dev-user
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: developer
  apiGroup: rbac.authorization.k8s.io
```

# View RBAC

```
▶ kubectl get roles
```

NAME	AGE
developer	4s

```
▶ kubectl get rolebindings
```

NAME	AGE
devuser-developer-binding	24s

```
▶ kubectl describe role developer
```

```
Name:          developer
Labels:        <none>
Annotations:   <none>
PolicyRule:
  Resources      Non-Resource URLs  Resource Names  Verbs
  -----
  ConfigMap      []                  []              [create]
  pods           []                  []              [get watch list create delete]
```

# View RBAC

```
▶ kubectl describe rolebinding devuser-developer-binding
```

```
Name:          devuser-developer-binding
```

```
Labels:        <none>
```

```
Annotations:   <none>
```

```
Role:
```

```
  Kind:  Role
```

```
  Name:  developer
```

```
Subjects:
```

```
  Kind  Name      Namespace
```

```
  ----  ----      -
```

```
User   dev-user
```

# Check Access

```
▶ kubectl auth can-i create deployments  
yes
```

```
▶ kubectl auth can-i delete nodes  
no
```

```
▶ kubectl auth can-i create deployments --as dev-user  
no
```

```
▶ kubectl auth can-i create pods --as dev-user  
yes
```

```
▶ kubectl auth can-i create pods --as dev-user --namespace test  
no
```

# Resource Names



blue



green



orange



purple



pink

developer-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "create", "update"]
  resourceNames: ["blue", "orange"]
```



{K}ODE{K}LOUD



# Course Objectives

Core Concepts

Scheduling

Logging Monitoring

Application Lifecycle Management

Cluster Maintenance

Security

○ Kubernetes Security Primitives

○ Secure Persistent Key Value Store

○ Authentication

○ Authorization

○ Security Contexts

○ TLS Certificates for Cluster Components

○ Images Securely

○ Network Policies

Storage

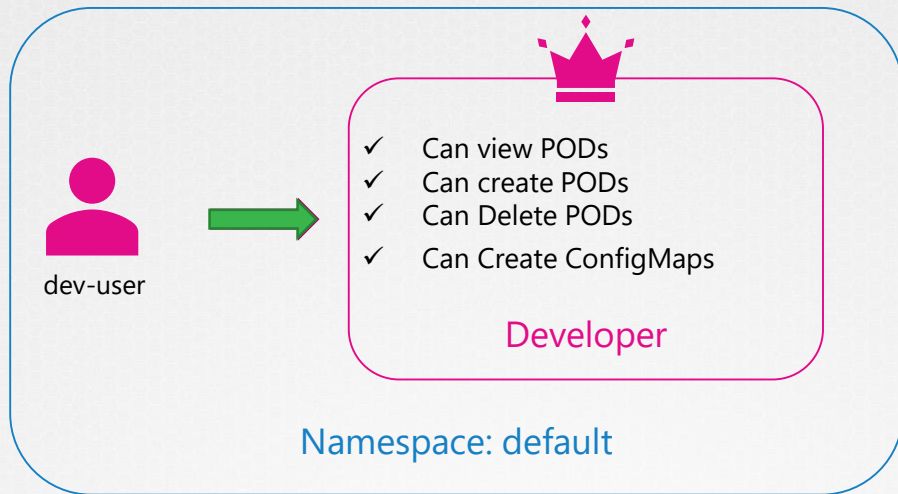
Networking

Installation, Configuration & Validation

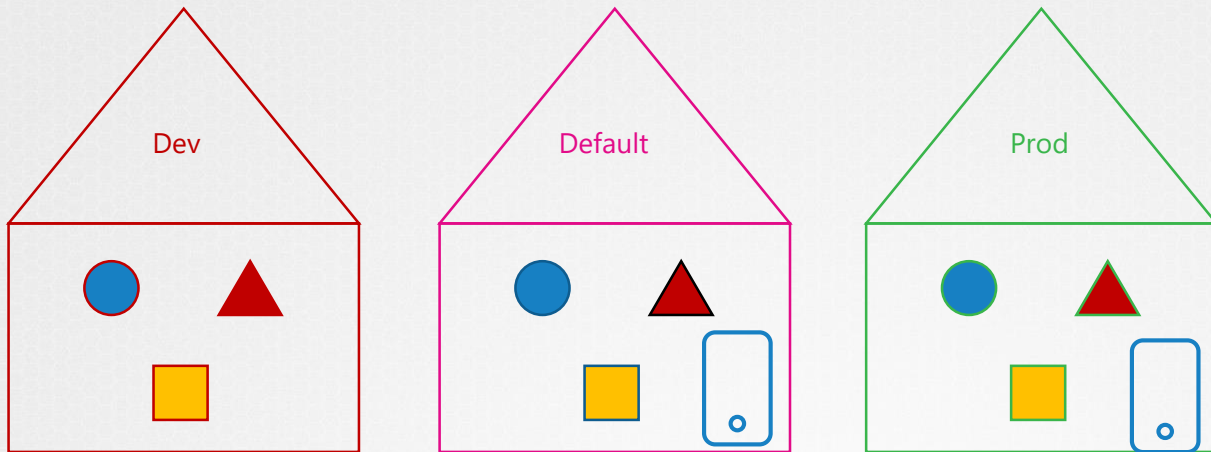
Troubleshooting

# Cluster Roles

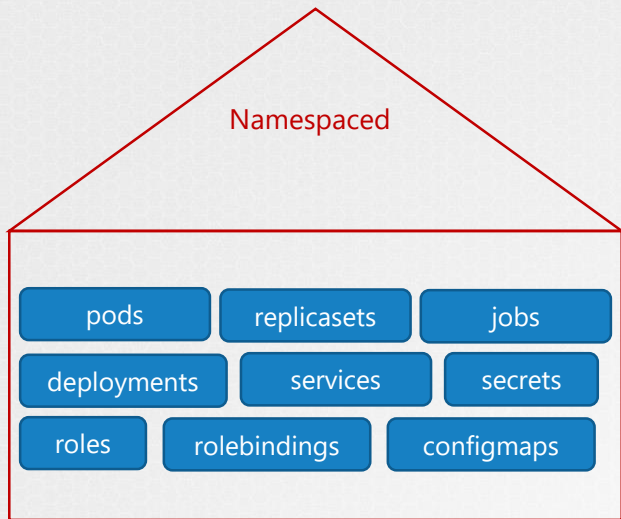
# | Roles



# Namespace

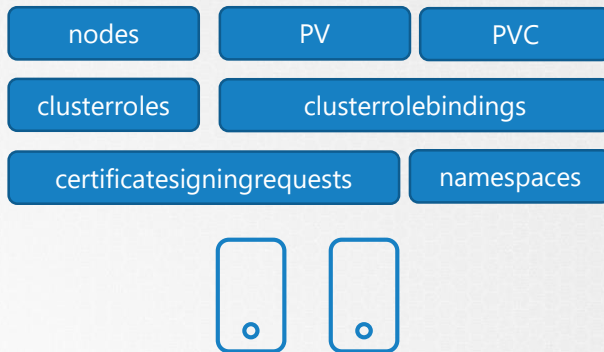


# Namespace



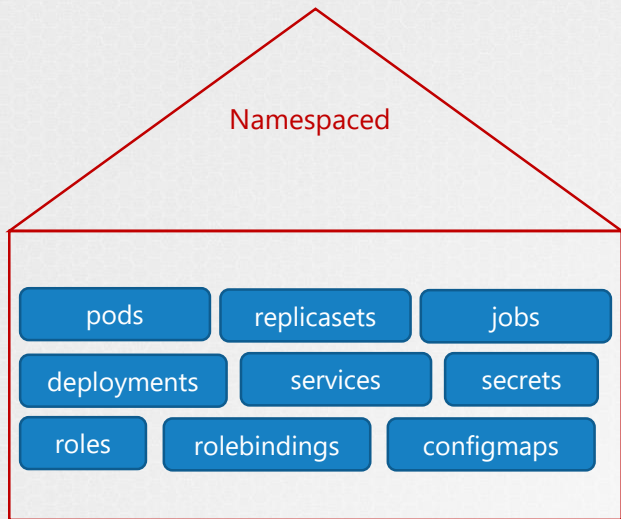
```
▶ kubectl api-resources --namespaced=true
```

Cluster Scoped

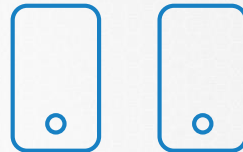
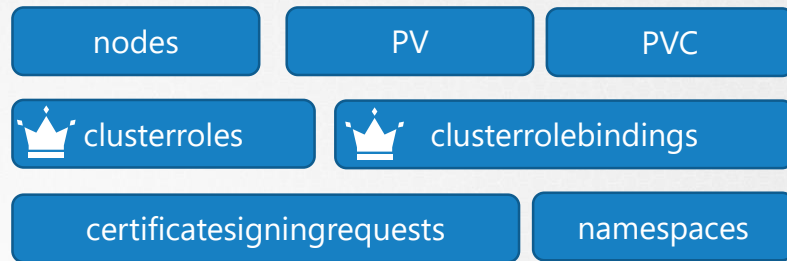


```
▶ kubectl api-resources --namespaced=false
```

# Namespace



## Cluster Scoped





# clusterroles



- ✓ Can view Nodes
- ✓ Can create Nodes
- ✓ Can delete Nodes

Cluster Admin



- ✓ Can view PVs
- ✓ Can create PVs
- ✓ Can delete PVCs

Storage Admin

cluster-admin-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-administrator
rules:
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["list", "get", "create", "delete"]
```



```
kubectl create -f cluster-admin-role.yaml
```


# clusterrolebinding

  
cluster-admin



- ✓ Can view Nodes
- ✓ Can create Nodes
- ✓ Can delete Nodes

  
Cluster Admin

 `kubectl create -f cluster-admin-role-binding.yaml`

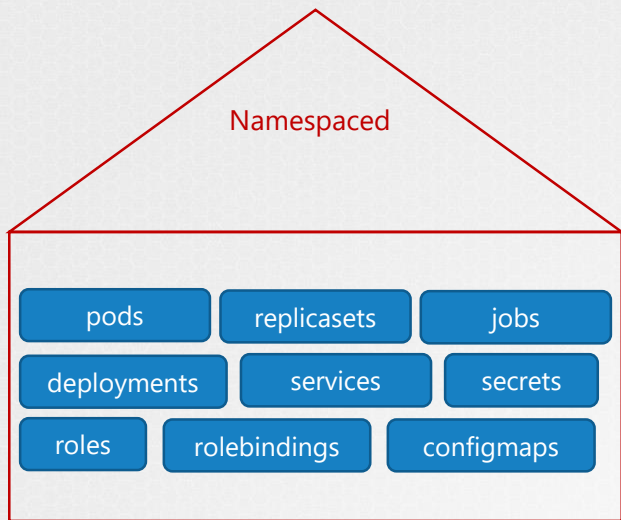
cluster-admin-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-administrator
rules:
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["list", "get", "create", "delete"]
```

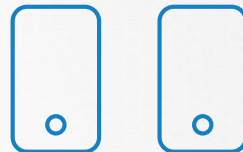
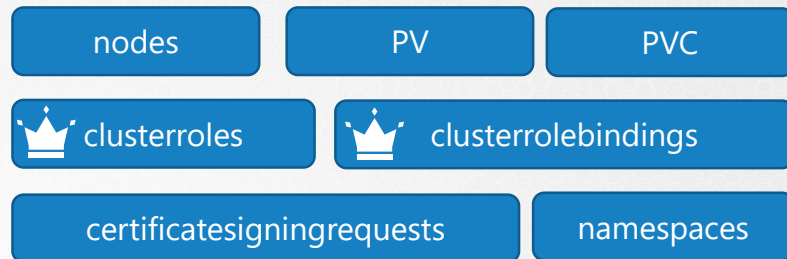
cluster-admin-role-binding.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cluster-admin-role-binding
subjects:
- kind: User
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: cluster-administrator
  apiGroup: rbac.authorization.k8s.io
```

# Cluster Roles



## Cluster Scoped





{K}ODE{K}LOUD

# Course Objectives

Core Concepts

Scheduling

Logging Monitoring

Application Lifecycle Management

Cluster Maintenance

Security

☐ Kubernetes Security Primitives

☐ Secure Persistent Key Value Store

☐ Authentication

☐ Authorization

☐ Security Contexts

☐ TLS Certificates for Cluster Components

☐ Images Securely

☐ Network Policies

Storage

Networking

Installation, Configuration & Validation

Troubleshooting

# Image Security



# Image

nginx-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx
    image: nginx
```

# Image

**image:** `docker.io/nginx/nginx`



Registry

User/  
Account

Image/  
Repository

`gcr.io/ kubernetes-e2e-test-images/dnsutils`

# Private Repository

```
▶ docker login private-registry.io
```

Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to <https://hub.docker.com> to create one.

**Username:** registry-user

**Password:**

WARNING! Your password will be stored unencrypted in /home/vagrant/.docker/config.json.

Login Succeeded

```
▶ docker run private-registry.io/apps/internal-app
```

# Private Repository

```
▶ docker login private-registry.io
```

```
▶ docker run private-registry.io/apps/internal-app
```

```
▶ kubectl create secret docker-registry regcred \
  --docker-server= private-registry.io \
  --docker-username= registry-user \
  --docker-password= registry-password \
  --docker-email= registry-user@org.com
```

nginx-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx
    image:
      imagePullSecrets:
      - name: regcred
```