Certificate Creation:

→ kuburnetes certificate creations:

- [ ] First use OpenSSL to create a private key using following command:

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

- [ ] Now we will create a certificate signing request:

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

- [ ] Now are sign the certificate using x509 command. Self signing can be done by supplying the signs key as your own private key

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

- [ ] Now to create client certificate,we do the same thing but for a user or client

① generate key using openssl:

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

② generate a csr

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

* The name "kube-admin" can be anything
* But that is the name client uses to authenticate
* This is the name that you will see in logs. So use something relevant

③ generate a signed certificate. But here we will use CA certificate and CA key

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

→ To differentiate between admin user and other users, you can specify extra parameters in certificate signing request to specify group

An example of using certificates

→ when making a request, we can now use the key, certificate and ca certificate to authenticate any request:

```
curl https://<ip>:6443/api/v1/pods \
--key admin.key --cert admin.crt \
--cacert ca.crt ↵
```

→ If there are many domain names, all the names need to be specified in the certificate signing request OR in a config

file is used for CSR.

To Add a new user Kube cluster

→ The user first creates a CSR certificate using following two steps

→ openssl genrsa -out jane.key 2048 ↵

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

→ The admin takes the CSR file and creates an object of type certificate signing request

→ this object is created like any other kube object, using a manifest file yaml

→ the request field in the yaml file should be the content of the .csr file that was created (but encoded as base 64)

→ Kube admins can use following command to see all the requests

kubectl get csr ↵

→ Admins can approve using

kubectl certificate approve <name>

→ Kube signs the certificate using CA key pair and generates a certificate for user. This can be extracted and shared w/ user

→ To view the certificate

kubectl get csr <name> -o yaml

The certificate is in the status field and is Base 64 encoded. Decode the content to extract the certificate

→ All certificate related operations is managed by KCM

→ The CA server root certificates are specified in KCM description