


Creating Users to Access the Cluster

In this lesson, we will create a user based on a fictitious character named 'John Doe' who wants to access our cluster.

WE'LL COVER THE FOLLOWING ^

- Understanding the Scenario
- Installing OpenSSL
- Creating a User
 -  A note to Windows users
 - Creating Final Certificate
 - Getting the Server Address

Understanding the Scenario

The word about Kubernetes' awesomeness is spreading in your company. People are becoming curious and would like to try it out.

Since you are the Kubernetes guru, it came as no surprise that you received a call from John Doe. He wants to “play” with Kubernetes, but he does not have time to set up his own cluster. Since he knows that you already have a cluster up and running, he'd appreciate if you would let him use yours.

Since you have no intention of giving John your certificates, you decide to let him authenticate with his user.

Installing OpenSSL

You will have to create certificates for him, so the first step you'll need to do is to verify that OpenSSL is installed on your laptop.

```
openssl version
```



It shouldn't matter which version of OpenSSL is installed. We output the `version` only to verify that the software is working.

If the output is something like `command not found: openssl`, you will have to [install the binaries](#).

Creating a User

The first thing we'll do is to create a private key for John. We'll assume that John Doe's username is `jdoue`.

```
mkdir keys
openssl genrsa \
  -out keys/jdoue.key 2048
```



We created the directory `keys` and generated a private key `jdoue.key`.

Next, we'll use the private key to generate a certificate.

```
openssl req -new \
  -key keys/jdoue.key \
  -out keys/jdoue.csr \
  -subj "/CN=jdoue/O=devs"
```



A note to Windows users

If you received an error like `Subject does not start with '/'`. Problems making Certificate Request, please replace `-subj "/CN=jdoue/O=devs"` with `-subj "//CN=jdoue\O=devs"` in the previous command and execute it again.

Creating Final Certificate

We created the certificate `jdoue.csr` with a specific subject that will help us identify John. `CN` is the username and `O` represents the organization he belongs. John is a developer, so `devs` should do.

For the final certificate, we'll need the cluster's certificate authority (CA). It will be responsible for approving the request and for generating the necessary

will be responsible for approving the request and for generating the necessary certificate John will use to access the cluster.

Since we used Minikube, the authority is already produced for us as part of the cluster creation. It should be in the `.minikube` directory inside the OS user's home folder.

Let's confirm it's there.

```
ls -l ~/.minikube/ca.*
```



i Minikube's directory might be somewhere else. If that's the case, please replace `~/.minikube` with the correct path.

The **output** is as follows.

```
/Users/vfarcic/.minikube/ca.crt  
/Users/vfarcic/.minikube/ca.key  
/Users/vfarcic/.minikube/ca.pem
```



Now we can generate the final certificate by approving the certificate sign request `jdoe.csr`.

```
openssl x509 -req \  
  -in keys/jdoe.csr \  
  -CA ~/.minikube/ca.crt \  
  -CAkey ~/.minikube/ca.key \  
  -CAcreateserial \  
  -out keys/jdoe.crt \  
  -days 365
```



Since we feel generous, we made the certificate `jdoe.crt` valid for a whole year (365 days).

To simplify the process, we'll copy the cluster's certificate authority to the `keys` directory.

```
cp ~/.minikube/ca.crt keys/ca.crt
```



Let's check what we generated.

```
ls -l keys
```



The **output** is as follows.

```
ca.crt  
jdoe.crt  
jdoe.csr  
jdoe.key
```



John does not need the `jdoe.csr` file. We used it only to generate the final certificate `jdoe.crt`. He will need all the other files though.

Getting the Server Address

Apart from the keys, John will need to know the address of the cluster. At the beginning of the chapter, we already created the `jsonpath` that retrieves the server so that part should be easy.

```
SERVER=$(kubectl config view \\\n  -o jsonpath='{.clusters[?(@.name=="minikube")].cluster.server}')  
  
echo $SERVER
```



The **output** is as follows.

```
https://192.168.99.106:8443
```



Equipped with:

- The new certificate (jdoe.crt)
- The key (jdoe.key)
- The cluster authority (ca.crt)
- The address of the server,

John can configure his `kubectl` installation.

In the next lesson, we will impersonate John Doe and access the cluster.

