OD - Lab 5: Kubernetes

Exercise 01

Your tasks:

1. Make a directory for holding your CA certificate, signing request and keys.
2. Follow the tutorial above:
    a. Create a root CA (private) key
    b. Create a root CA certificate
    c. Create a certificate (private) key
    d. Create a certificate signing request
    e. Issue a signed certificate from the Certificate Signing Request
    f. Verify the issued certificate from step e) has been signed by your root CA

1. Creating a directory

```
cd .../LAB-05/Exercise-1/CA-Cert
```

2.a Creating a root (private key)

```
$ openssl genrsa -des3 -out rootCA.key 2048
$ 1234
$ 1234
```

1234 is a choosen password

2.b Creating a root CA certificate

```
$ openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
$ 1234
```

Fill in the following (Up to you):

- DK
- Syddanmark
- Odense
- SDU
- SE
- SEOD
- bekut20@student.sdu.dk

2.c and d

```
openssl req -new -sha256 -nodes -out server.csr -newkey rsa:2048 -keyout berkankutuk.dk.key
```

Fill in the following

- DK
- Syddanmark
- Odense
- SDU
- SE
- SEOD
- bekut20@student.sdu.dk
- 1234

Click enter when asked for "extra attributes"

Create configuration information

Create the following file:

```
$ touch v3.ext
$ nano v3.ext
```

v3.ext

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
```

2.e Issue the certificate

```
$ openssl x509 -req -in server.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out berkankutuk.dk.crt -days 10
00 -sha256 -extfile v3.ext
$ 1234
```

See what the certificate contains

```
openssl x509 -in berkankutuk.dk.crt -text -noout
```

2.f Verify the issue

```
openssl verify -CAfile rootCA.pem berkankutuk.dk.crt
```

Exercise 2

Your tasks:

1. Tag the images in using the format mentioned above.
2. Login to Docker Hub.
3. Push the tagged image to Docker Hub.

1. Tag the image

We have to build the image before we tag it.

- So start by moving the copying the old exercises to the new folders
- Go to the folder
- And now open up the terminal and build the image

```
docker build -t exercise2 .
```

And now tag the image

```
docker tag exercise2 berkanktk/exercise2
```

2. Login to docker by typing:

```
docker login
```

And enter your username and password

3. Push the tagged image

```
docker push berkanktk/exercise2
```

And then redo all of the steps above with the second folder (/LAB2-4)

Go to the folder and open up your terminal:

```
$ docker build -t exercise4 .
$ docker tag exercise4 berkanktk/exercise4
```

```
$ docker push berkanktk/exercise4
```

Exercise 3 (This exercise is deprecated)

Exercise 4

Your tasks:

1. Install kubectl following the guide provided
2. Make sure a config is installed by using $ kubectl config view
    a. If the config is not the right one use --kubeconfig <KUBE-CONFIG>
3. Play around with the commands. What resources are present on the kubernetes cluster currently?

1. Installing kubectl

Install kubectl: https://kubernetes.io/docs/tasks/tools/install-kubectl-macos/

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl"
```

Download the checksum file

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl.sha256"
```

Validate

```
echo "$(<kubectl.sha256)  kubectl" | shasum -a 256 --check
```

Make the kubectl binary executable.

```
chmod +x ./kubectl
```

Move the kubectl binary to a file location on your system PATH.

```
sudo mv ./kubectl /usr/local/bin/kubectl
sudo chown root: /usr/local/bin/kubectl
```

Ensure version

2. Making sure the config is installed

```
kubectl config view
```

Creating the .kube and config file (First time)

Go to root folder and add a directory called .kube and put a file named config inside it with no extensions

```
$ mkdir .kube
$ cd .kube
$ touch config
```

3. Play around with the commands

The command `$ kubectl get nodes` can be used to get an overview of the nodes currently in the Kubernetes cluster.

The command below is used to deploy something. Either a deployment, service etc.

`$ kubectl apply -f <PATH-TO-YAML>`

A command very similar to the one above is used to delete or remove the .yml deployment file again.

`$ kubectl delete -f <PATH-TO-YAML>`

The command below is used to get a resource, this can be either: deployments, pods, nodes, services, events.

`$ kubectl get <RESOURCE>`

This command was mentioned above but deserves some extra attention. It is a very useful command for problem solving. If a part of the deployment is not behaving as expected or not working at all, the events might give a clue as to what happened.

`$ kubectl get events`

This command is useful for getting a further description of a resource. For example about a node, pod, deployment or service. One should first tell kubectl what the resource is (deployment, node, pod etc.) followed by the name from the get command.

`$ kubectl describe <RESOURCE> <NAME>`

This command is used to forward a local port to a pod. This is useful because one can map a local port to a port in the pod. An example usage is when using a visualizer. Using this method only the local machine has access to the visualizer. If a service was used instead everyone would have access which is not as desirable.

`$ kubectl port-forward deployment/<NAME> <HOSTPORT>:<CONTAINERPORT>`

Exercise 5

Included files:

- config
- docker-compose.yml

```
services:
    sql:
        image: muggel/lab2exercise4:latest
        command: --default-authentication-plugin=mysql_native_password

    web:
        image: muggel/lab2exercise2:latest
        depends_on:
            - sql

        ports:
            - '5050:80'%
```

- L5E5.yml

```
# L5E5.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: <NAME>
spec:
  selector:
    matchLabels:
      component: <LABEL>
  replicas: 1
  template:
    metadata:
      labels:
        component: <LABEL>
    spec:
      containers:
        - name: spekt8
          image: ahmadhamid/spekt8
          ports:
            - containerPort: 3000
```

Your tasks:

1. Create a Kubernetes Deployment using kubectl to run the above-mentioned Visualiser. Use the provided manifest (yaml-file).
2. Port-forward port 3000 from the container to your hosts port 3000, to visit the Visualiser Dashboard.
   a. Using the command described in exercise 04
3. Access the Dashboard through localhost:3000 to see Kubernetes Resources.

IMPORTANT: There is an issue with the dashboard so there won't be anything visible, but that's alright.

1. Creating a kubernetes deployment

```
nano L5E5.yml
```

```
# L5E5.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: berkan-test
spec:
  selector:
    matchLabels:
      component: berkan
  replicas: 1
  template:
    metadata:
      labels:
        component: berkan
    spec:
      containers:
        - name: spekt8
          image: ahmadhamid/spekt8
          ports:
            - containerPort: 3000
```

CMD+O (Save)→ Hit Enter (Accept override)→CMD+X (Close Nano)

Now deploy the YAML file

```
kubectl apply -f ./L5E5.yml
```

2. Port forward port 3000

```
kubectl port-forward deployment/<NAME> 3000:3000
```

3. Access the Dashboard

Open your browser and type:

```
localhost:3000
```

Exercise 06

Included files (optional):

- docker-compose.yml

```
version: '3'

services:
    sql:
        image: muggel/lab2exercise4:latest
        command: --default-authentication-plugin=mysql_native_password

    web:
        image: muggel/lab2exercise2:latest
        depends_on:
            - sql

        ports:
            - '5050:80'
```

Your tasks:

1. Make a Kubernetes Manifest that creates a Deployment containing the Docker services from Lab 02; Exercise 05.
2. Visit the spekt8 visualiser to see the running resources (might not work).
3. Visit the node03.stream.stud-srv.sdu.dk:<NODEPORT> and test that the system is functional.

1. Creating the kubernetes manifest

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: berkan-l2e5
spec:
  replicas: 3
  selector:
    matchLabels:
      app: berkan-l2e5-pod
```

```
  template:
    metadata:
      labels:
        app: berkan-l2e5-pod
    spec:
      containers:
      - name: dbcontainer
        image: docker.io/berkanktk/exercise4
        command: [“/bin/echo”]
        args: [“Message”]
        env:
        - name: ENV_NAME
          value: “Env value”
        ports:
        - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: berkan-l2e5-service
spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 8080
    nodePort: 32323
  selector:
    app: berkan-l2e5-pod
```

```
kubectl apply -f ./L5E6.yml
```

2. Visit the spekt8 visualiser
3. Visit the node03.stream.stud-srv.sdu.dk:<NODEPORT>