Deployment on kubernetes

For deploying on kubernetes you first need to create yaml file for you application with name deployment.yaml or whatever your application name is, for example if i am having an application with name security-framework so i will name my file **security-framework.deployment.yaml**

To know more about deployment in kubernetes follow the link below:-

https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Here is how a deployment yaml file looks, in this example i have shown a nginx yaml file for deployment i named it as nginx-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

After this you need to apply this yaml file to your kubernetes cluster so that it should be ready for deployment, for doing this execute the below command:-

```
Kubectl create -f </path/to/your/yaml/file>
```

Now your deployment will be shown on your kubernetes dashboard in deployments column, to apply changes in it, suppose you have ,made few changes in your yaml file and you want those change to reflect in your deployment then you need to execute the below command:-

```
Kubectl apply -f </path/to/your/config>
```

For creating service also yo have to follow the same procedure, and name your yaml file as security-framework.service.yaml, i am showing you an nginx-service.yaml file

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

To know more about service follow the link below:-
https://kubernetes.io/docs/concepts/services-networking/service/

After this you need to apply this yaml file to your kubernetes cluster so that it should be ready for deployment, for doing this execute the below command:-

```
Kubectl create -f </path/to/your/yaml/file>
```

Now your service will be shown on your kubernetes dashboard in service column, to apply changes in it, suppose you have ,made a few changes in your yaml file and you want those changes to reflect in your deployment then you need to execute the below command:-

```
Kubectl apply -f </path/to/your/config>
```

All the yaml files for deployment and services can be found in the given repository:-

https://bitbucket.org/team99array/k8s-yaml-files/src/master/
Whenever you are creating a new deployment or service you need to execute **kubectl create** command to make you deployment available on kubernetes.

We need some of our service to externalize, in this case we will use load balancer along wih AWS certificate manager, it will provide SSL certificate for our service (in our case it is api)
We need to provide the certificate arn in our service.yaml file as shown in the repository link mentioned above, we also need to create an nginx config for this and with the help of docker we need to copy that config into our docker-image

For creating certificate for your domain follow the link below:-
https://docs.aws.amazon.com/acm/latest/userguide/gs-acm-request-public.html

The nginx config will look like this:-

```
server {
   listen 443;
       server_name domain-name;

       underscores_in_headers on;

       error_log /var/log/nginx/debug.log debug;

       location /robots.txt {return 200 "User-agent: *\nDisallow: /\n";}

       location / {

       if ($request_method = 'OPTIONS') {
                add_header 'Access-Control-Allow-Origin' '*' always;
                add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
                add_header 'Access-Control-Allow-Headers'
'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range
,A_T,P_I,T_I';
                add_header 'Access-Control-Expose-Headers'
'Content-Length,Content-Range,A_T,P_I,T_I';
                add_header 'Access-Control-Max-Age' 1728000;
                add_header 'Content-Type' 'text/plain; charset=utf-8';
                add_header 'Content-Length' 0;
                return 204;
       }
       if ($request_method = 'POST') {
                add_header 'Access-Control-Allow-Origin' '*' always;
                add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
                add_header 'Access-Control-Allow-Headers'
```

```
'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range
,A_T,P_I,T_I';
                add_header 'Access-Control-Expose-Headers'
'Content-Length,Content-Range,A_T,P_I,T_I';
        }
        if ($request_method = 'GET') {
                add_header 'Access-Control-Allow-Origin' '*' always;
                add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
                add_header 'Access-Control-Allow-Headers'
'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range
,A_T,P_I,T_I';
                add_header 'Access-Control-Expose-Headers'
'Content-Length,Content-Range,A_T,T_I,P_I';
        }

                proxy_pass_request_headers on;
                proxy_pass http://localhost:<port.no>;

        }

}

server {
 if ($host = domain-name) {
        return 301 https://$host$request_uri;
    }


    listen 80;
    server_name domain-name;
    return 404;
}
```

After this go to your route 53 and create a new record set with Type A and in the name type your domain name and choose alias column with yes and copy the load balancer host name in it and save it, you have to repeat this procedure for the all the api's.

YOU ALSO NEED TO DO SOME CHANGES INTO YOUR APPLICATION CONFIG FILE

These changes may include your environment name, access key and secret key, new url's, service names, arn, etc, according to your requirement

The modified config files are present in the repository link mentioned below along with the tag:-

https://bitbucket.org/team99array/config/src/dish/prod/

THE ONE CLICK DEPLOYMENT IS DONE THROUGH JENKINS

We have created a CI/CD pipeline in jenkins for our deployment, we will be having tag based deployment which states that the code and our application config files are tagged in our bit bucket repository. We will make the build with jenkins and we will tag our docker image with the tag number that we will provide as an argument while running the script.

Here is an example of how build script looks:-

```
#!/bin/bash

PROFILE=prod
APP_NAME=security_framework
BINDING_PORT=6062

echo $1 $2 $3

cd /home/ubuntu/config/

git fetch origin

git checkout $2

git pull origin $2

cd /home/ubuntu/enveu-saas/qa/enveu-saas-core/platform-management-module

git fetch origin

git checkout $2

git pull origin $2

../gradlew clean :main:build -x check

cd

cp "config/${PROFILE}/${APP_NAME}/application.${PROFILE}.yaml"
"${PROFILE}/dockerconfig/${APP_NAME}/"

cp "config/${PROFILE}/${APP_NAME}/log4j2-security.yml"
"${PROFILE}/dockerconfig/${APP_NAME}/"
```

```
cp "config/${PROFILE}/${APP_NAME}/default-policy.json"
"${PROFILE}/dockerconfig/${APP_NAME}/"

cp "config/${PROFILE}/${APP_NAME}/filebeat.yml"
"${PROFILE}/dockerconfig/${APP_NAME}/"

rm -r "${PROFILE}/dockerconfig/${APP_NAME}/main-app.jar"

cp
~/enveu-saas/qa/enveu-saas-core/platform-management-module/main-app/build/libs/main
-app.jar "${PROFILE}/dockerconfig/${APP_NAME}/"

cd /home/ubuntu/dockerconfig/${PROFILE}

git checkout dish

git pull origin dish

cd

cp /home/ubuntu/dockerconfig/${PROFILE}/${APP_NAME}/Dockerfile
/home/ubuntu/deployment-base/enveu-saas-devops/${APP_NAME}/

cp /home/ubuntu/dockerconfig/${PROFILE}/${APP_NAME}/entrypoint.sh
/home/ubuntu/${PROFILE}/dockerconfig/${APP_NAME}/

sudo docker build -t "${APP_NAME}_dish_${PROFILE}"
"${PROFILE}/dockerconfig/${APP_NAME}/" -f
"deployment-base/enveu-saas-devops/${APP_NAME}/Dockerfile"

sudo docker tag "${APP_NAME}_dish_${PROFILE}"
"docker-registry.enveu.com:5443/${APP_NAME}_dish_${PROFILE}"

sudo docker push "docker-registry.enveu.com:5443/${APP_NAME}_dish_${PROFILE}"

if [ $1 = "tag" ]
then
        sudo docker tag "${APP_NAME}_dish_${PROFILE}"  `echo
"docker-registry.enveu.com:5443/${APP_NAME}_dish_${PROFILE}:"$2`
        sudo docker push `echo
"docker-registry.enveu.com:5443/${APP_NAME}_dish_${PROFILE}:"$2`

fi
```

The existing build script can be found in the given repository link mentioned below:-

https://bitbucket.org/team99array/enveu-jenkins-job-script/src/qa/

For doing deployment through jenkins you need to create a pipeline on it

To know how to create a pipeline follow the link below:-
https://jenkins.io/doc/pipeline/tour/hello-world/

Always remember you have to pass the parameter on your jenkins job

For deploying on kubernetes you need to create a pipeline project
In that pipeline project you need to write the following code :-

```
node {
   def KOPS_STATE_STORE
   def NAME
   def tag
   def k8DeploymentName
   def imageName
   stage('Preparation') { // for display purposes
      echo 'Preparing build environments'
      KOPS_STATE_STORE = 's3://cluster.k8-dev.enveu.com'
      NAME = 'mycluster.k8s.local'
      tag = params.BUILD_TAG
      k8DeploymentName = 'deployment.v1.apps/security-framework-deployment'
      imageName =
'security-framework=docker-registry.enveu.com:5443/security_framework_dish_
prod:'
      echo 'Using ' + KOPS_STATE_STORE + ' as state store'
      echo 'Using ' + NAME + ' as cluster name'
      echo 'Using ' + tag + ' as build tag'
   }
   stage('Updating Deployment Image') {
      echo 'Deploying the latest application with rolling update'
      sh 'ssh -i /home/ubuntu/pem-file/enveu-saas-dish.pem
ubuntu@13.234.219.232 "kubectl set image ' + k8DeploymentName + ' ' +
imageName + tag + ' --record"'
   }
```

```
   stage('Verifying Deployment') {
       echo 'Verifying rollout status'
      sh 'ssh -i /home/ubuntu/pem-file/enveu-saas-dish.pem
ubuntu@13.234.219.232 "kubectl rollout status ' + k8DeploymentName +'"'
   }
}
```

Remember to update your `KOPS_STATE_STORE`, `k8DeploymentName`, `imageName` with your's.