Module 3: Deploy App To Kubernetes Cluster

Demo Document

Deploy a containerized app image in the locally setup Kubernetes cluster

Step 1:

Before putting the application on Kubernetes cluster, you need to perform the following steps:

a. Deploy a simple Docker container for webserver
b. Put a custom file for a static web-page
c. Create custom container image
d. Push it to Docker hub
e. Using the same image, deploy it on your Kubernetes cluster

a. Deploy a simple docker container for webserver

Note: You can choose either apache or nginx
We will use the latest release of the nginx and download from the default docker registry using the following command:

```
$sudo docker run -d -P --name webserver nginx
```

```
edureka@kmaster:~/webserver$ sudo docker run -d -P --name webserver nginx
9a3a7b9effcfeb238789dbb4355d0e4a9b9cba2383375d3209a2659f719e6675
```

Let's see the default page of nginx. Get the port on which it is exposed using this command:

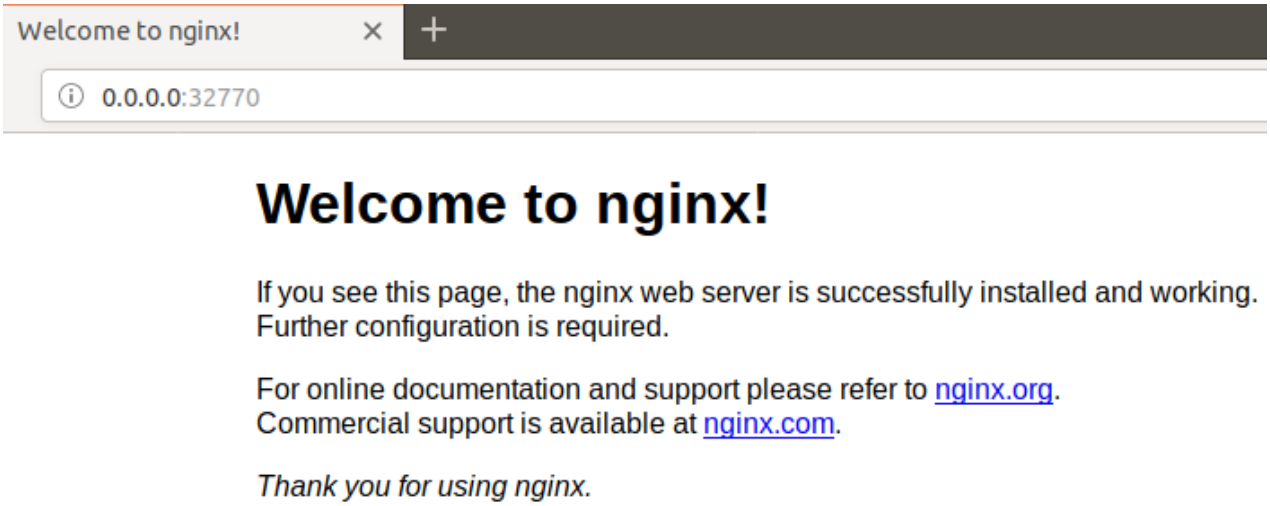
```
$sudo docker port webserver
```

```
edureka@kmaster:~/webserver$ sudo docker port webserver
80/tcp -> 0.0.0.0:32770
```

Note: webserver is the name of the container which we provided

Open the browser and use the routable IP to your host on which container is running. In our case, it is on host 'kmaster':

http://kmaster:32770



b. Put a custom file for a static webpage

- Create a directory webserver using these commands

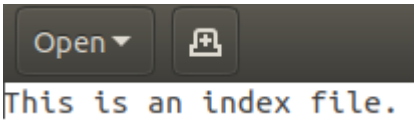
```
$mkdir webserver
cd webserver
```

- Create a file 'index.html' with some content to it using the command:

```
$gedit index.html
```

- View the contents of the file:

```
$cat index.html
```



- Under this, create a file 'Dockerfile' using the command:

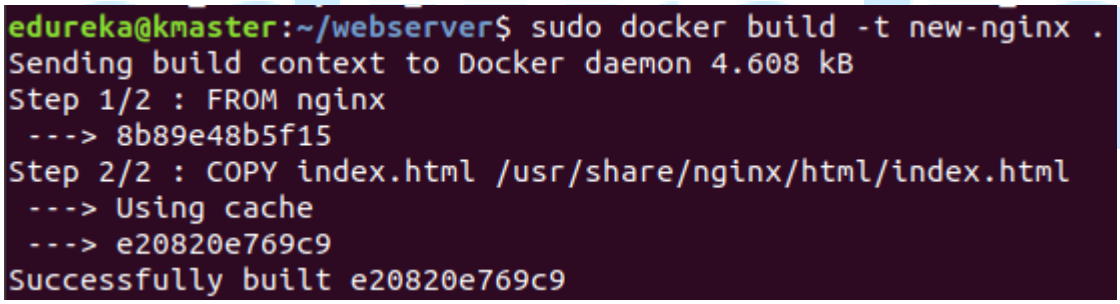
```
$gedit Dockerfile
```

Enter the following inside it:

```
FROM nginx
COPY index.html /usr/share/nginx/html/index.html
```

c. Create custom container image

```
$sudo docker build -t new-nginx .
```



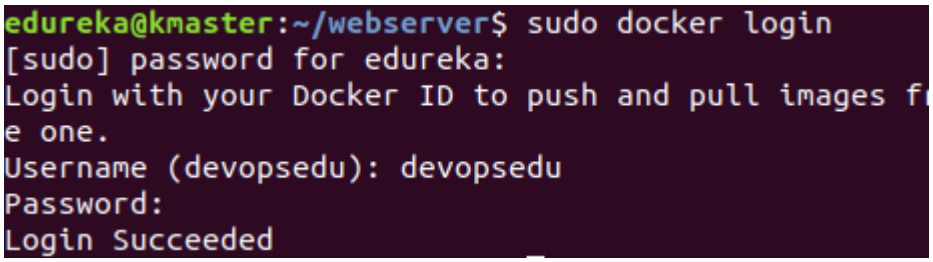
- Verify the image:

```
$sudo docker images
```

d. Push the image to the docker hub

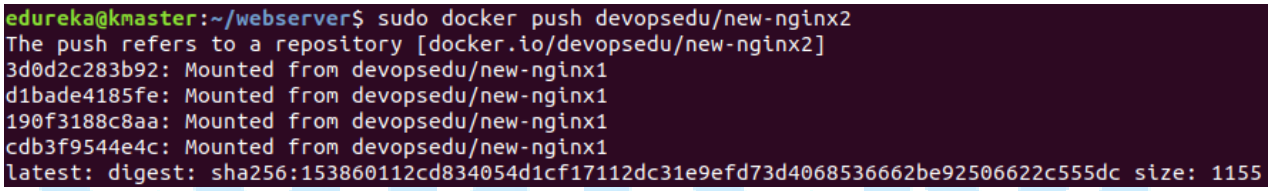
- First login to it:

```
$sudo docker login
```



- Then push the image:

```
$sudo docker push devopsedu/newnginx1
```



- Now, let's create a .yaml file to create Kubernetes deployment with 2 replicaset:

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
 name: new-nginx-deployment
spec:
 selector:
  matchLabels:
   app: new-nginx1
 replicas: 2
 template:
  metadata:
   labels:
    app: new-nginx1
  spec:
   containers:
   - name: new-nginx
    image: devopsedu/new-nginx1
    ports:
    - containerPort: 80
```


e. Create the Kubernetes deployment

```
$kubectl create -f new-nginx1.yaml
$kubectl get deployments
```

```
edureka@kmaster:~/webserver$ kubectl create -f new-nginx1.yaml
deployment.apps/new-nginx-deployment created
```

```
edureka@kmaster:~/webserver$ kubectl get deployments
NAME                   DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
new-nginx-deployment   2         2         2            2           2m
```

- Expose the service to external network and note the port to which it is exposed. Here it is 31134:

```
$kubectl expose deployment new-nginx-deployment --
type=NodePort --port=80

$kubectl get services
```

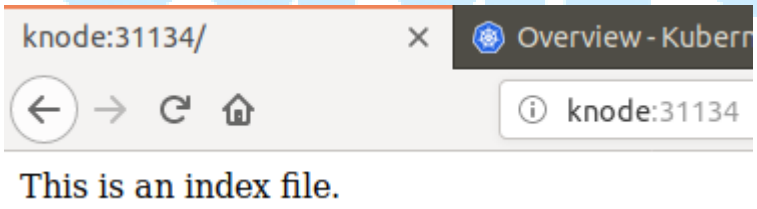
```
edureka@kmaster:~/webserver$ kubectl expose deployment new-nginx-deployment --type=NodePort --port=80
service/new-nginx-deployment exposed
edureka@kmaster:~/webserver$ kubectl get service
NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes            ClusterIP   10.96.0.1        <none>        443/TCP        23m
new-nginx-deployment  NodePort    10.104.156.169   <none>        80:31134/TCP   27s
```

- Note the node on which it is running:

```
$kubectl get pods -o wide
```

```
edureka@kmaster:~/webserver$ kubectl get pods --all-namespaces -o wide
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE     IP               NODE
default       new-nginx-deployment-b64c59d5d-fhxbm  1/1     Running   0          4m      192.168.178.66   knode
default       new-nginx-deployment-b64c59d5d-t65jt  1/1     Running   0          4m      192.168.178.65   knode
```

- Now, use knode:31134 to browse through



Step 2:

List all local deployments:

```
$kubectl get deployments
```

```
edureka@kmaster:~/webserver$ kubectl get deployments
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
new-nginx-deployment  2         2         2            2           2m
```

Step 3:

Create a kubectl proxy for forwarding communication to cluster-wide private network:

```
$kubectl proxy
```

```
edureka@kmaster:~/webserver$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

Note: The proxy can be terminated by pressing control-C and won't show any output while its running.
The proxy enables direct access to the API from these terminals.
All those APIs hosted through the proxy endpoint, can be seen through:
http://localhost:8001

Step 4:

Curl to verify that the app is running:

```
$curl localhost:31134
```

```
edureka@kmaster:~/webserver$ curl localhost:31134
This is an index file.
```

Step 5:

List all existing pods:

```
$ kubectl get pods --all-namespaces
```

```
edureka@kmaster:~/webserver$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                    READY     STATUS    RESTARTS   AGE
default       new-nginx-deployment-b64c59d5d-fhxbm    1/1       Running   0          4m
default       new-nginx-deployment-b64c59d5d-t65jt    1/1       Running   0          4m
kube-system   calico-etcd-4j7r2                       1/1       Running   0          18m
kube-system   calico-kube-controllers-cd589c58b-q2g97 1/1       Running   0          18m
kube-system   calico-node-hcc6z                       2/2       Running   1          13m
kube-system   calico-node-rbcgj                       2/2       Running   0          18m
kube-system   coredns-78fcdf6894-4zfhg                1/1       Running   0          25m
kube-system   coredns-78fcdf6894-5schb                1/1       Running   0          25m
kube-system   etcd-kmaster                            1/1       Running   0          17m
kube-system   kube-apiserver-kmaster                  1/1       Running   0          17m
kube-system   kube-controller-manager-kmaster         1/1       Running   0          17m
kube-system   kube-proxy-kc86j                        1/1       Running   0          25m
kube-system   kube-proxy-scg5k                        1/1       Running   0          13m
kube-system   kube-scheduler-kmaster                  1/1       Running   0          17m
kube-system   kubernetes-dashboard-6948bdb78-884nk    1/1       Running   0          17m
```

Note: You can also fetch all the details by API
$ curl localhost:8001/api/v1/namespaces/default/pods?

Step 6:

Get description of a specific pod:

```
$ kubectl describe pod <pod-name>
```

```
edureka@kmaster:~/webserver$ kubectl describe pod new-nginx-deployment-b64c59d5d-fhxbm
Name:               new-nginx-deployment-b64c59d5d-fhxbm
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               knode/10.0.2.15
Start Time:         Thu, 19 Jul 2018 21:04:40 +0530
Labels:             app=new-nginx1
                    pod-template-hash=620715818
Annotations:        <none>
Status:             Running
IP:                 192.168.178.66
Controlled By:      ReplicaSet/new-nginx-deployment-b64c59d5d
Containers:
  new-nginx:
    Container ID:   docker://16143c4a35a26f4c0c09a96d2b88bd6b5708c10ddcbe9672ca2538fa9e91460b
    Image:          devopsedu/new-nginx1
    Image ID:       docker-pullable://devopsedu/new-nginx1@sha256:153860112cd834054d1cf17112dc31e9efd73d4068536662be92506622c555dc
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 19 Jul 2018 21:04:48 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-sw5r4 (ro)
Conditions:
```

Step 7:

View logs of the container:

- To view the logs of the container, get the respective container id.

  ```
  $docker ps -a
  ```

  It will show all the containers which are running through the docker engine.

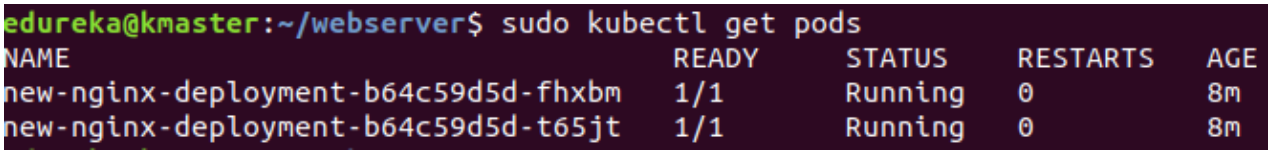
```
edureka@kmaster:~/webserver$ sudo docker ps -a
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS                     NAMES
9a3a7b9effcf        nginx                  "nginx -g 'daemon ..."   12 minutes ago      Up 12 minutes       0.0.0.0:32770->80/tcp     webserver
cae3bc42d403        0c60bcf89900           "/dashboard --inse..."   20 minutes ago      Up 20 minutes                                 k8s_kubern
etes-dashboard_kubernetes-dashboard-6948bdb78-884nk_kube-system_69a04779-8b67-11e8-896b-0800270c87d2_0
807b8461cf15        k8s.gcr.io/pause:3.1   "/pause"                 20 minutes ago      Up 20 minutes                                 k8s_POD_ku
bernetes-dashboard-6948bdb78-884nk_kube-system_69a04779-8b67-11e8-896b-0800270c87d2_0
3d273a7412e9        b3b94275d97c           "/coredns -conf /e..."   20 minutes ago      Up 20 minutes                                 k8s_coredn
s_coredns-78fcdf6894-4zfhg_kube-system_5a97935d-8b66-11e8-896b-0800270c87d2_0
8b7aabe1d7e9        b3b94275d97c           "/coredns -conf /e..."   20 minutes ago      Up 20 minutes                                 k8s_coredn
s_coredns-78fcdf6894-5schb_kube-system_5a939ce2-8b66-11e8-896b-0800270c87d2_0
ccb03c15335e        d9298bd6eae2           "/usr/bin/kube-con..."   20 minutes ago      Up 20 minutes                                 k8s_calico
-kube-controllers_calico-kube-controllers-cd589c58b-q2g97_kube-system_511b8473-8b67-11e8-896b-0800270c87d2_0
cfd7cb831ebb        k8s.gcr.io/pause:3.1   "/pause"                 20 minutes ago      Up 20 minutes                                 k8s_POD_co
redns-78fcdf6894-4zfhg_kube-system_5a97935d-8b66-11e8-896b-0800270c87d2_0
234bf64133c6        k8s.gcr.io/pause:3.1   "/pause"                 20 minutes ago      Up 20 minutes                                 k8s_POD_ca
```

- Once you get the container id, use the following command to see the logs

```
$docker logs <container ID>
```

- First get the pods name

```
$ kubectl get pods
```



- Once you have the pod name, then use the command

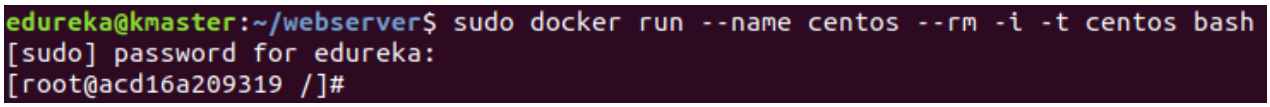
```
$kubectl logs <podname>
```

Step 8:

Execute command directly on container:

- First let's start a new container

```
$sudo docker run --name centos --rm -i -t centos bash
```



Once you have the shell, you are free to run the command like a normal linux CLI. Leave this terminal as it is and open another session

- Verify that docker is still running

```
$sudo docker ps -a
```

```
edureka@kmaster:~/webserver$ sudo docker ps -a
[sudo] password for edureka:
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
acd16a209319        centos              "bash"              About a minute ago   Up About a minute                       centos
```

- Now run the command directly on centos container which you have created

```
$sudo docker exec -it centos sh -c "touch edureka"
```

```
edureka@kmaster:~/webserver$ sudo docker exec -it centos sh -c "touch edureka"
```

- Now go back to the first terminal and see if the file is created

```
[root@acd16a209319 /]# ls
bin  dev  edureka  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```