Deploy App On Pod & Basic Networking

[Edition 3]

[Last Update 210322]

For any issues/help contact : ***support@k21academy.com***

Contents

1 INTRODUCTION

A **Kubernetes cluster** is a set of node machines for running containerized applications. If you're running **Kubernetes**, you're running a **cluster**. At a minimum, a **cluster** contains a control plane and one or more compute machines, or nodes.

This guide Covers:

- Deploying a simple application as pod

2 DOCUMENTATION

2.1 Kubernetes Documentation

1. Autoscaling in Kubernetes
   https://kubernetes.io/blog/2016/07/autoscaling-in-kubernetes/
2. Using kubectl to Create a Deployment
   https://kubernetes.io/docs/tutorials/kubernetes-basics/deploy-app/deploy-intro/
3. Cluster Networking
   https://kubernetes.io/docs/concepts/cluster-administration/networking/

2.2 Linux Commands and VIM Commands

Note: If you are new to Linux and wanted to learn basic linux for Kubernetes, then drop us a mail at support@k21academy.com and get Bonus Linux for Beginners course free.

1. Basic Linux Commands
   https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners
   https://www.hostinger.in/tutorials/linux-commands
2. Basic VIM Commands
   https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started
3. Popular VIM Commands
   https://www.keycdn.com/blog/vim-commands

3 PRE-REQUISITE

Ensure that you have completed following three activity guides (or you have an Ubuntu Server)

- Create account (Trial or Paid) on Azure Cloud.

  Note: *Follow Activity Guide* ***AG_Bootstrap_Kubernetes_Cluster_Using_Kubeadm_Guide_ed**** *from portal*

4 RUNNING NGINX SERVER AS POD IN CLUSTER

Note: In below Sections we are going to use YAML files no need write complete yaml file because in CKA exam you can official documentation use Below GIT url to clone repo and use yaml files

```
$ git clone https://github.com/k21academyuk/Kubernetes
$ cd Kubernetes
```

```
Cloning into 'Kubernetes'...
remote: Enumerating objects: 179, done.
remote: Counting objects: 100% (179/179), done.
remote: Compressing objects: 100% (125/125), done.
remote: Total 179 (delta 61), reused 167 (delta 50), pack-reused 0
Receiving objects: 100% (179/179), 17.08 MiB | 39.22 MiB/s, done.
Resolving deltas: 100% (61/61), done.
root@master:/home/ubuntu# cd Kubernetes/
root@master:/home/ubuntu/Kubernetes# ls
Dockerfile                        nfs-pvc.yaml
README.md                         nfspv-pod.yaml
__pycache__                       nginx-deployment.yaml
app.py                            nginx-hpa.yaml
apple.yaml                        nginx-svc.yaml
banana.yaml                       nodeaffinity-deployment.yaml
counter-pod.yaml                  nodeaffinity1-deployment.yaml
daemonset.yaml                    nodeanti-affinity-deployment.yaml
demo-pod.yaml                     nodeanti-affinity1-deployment.yaml
docker-compose.yaml               oke-admin-service-account.yaml
elasticsearch-rbac.yaml           pod-dynamicpv-oci.yaml
elasticsearch-stfullset-oci.yaml  pod-dynamicpv.yaml
elasticsearch-stfullset.yaml      podaffinity-deployment.yaml
elasticsearch-svc.yaml            podaffinity1-deployment.yaml
elasticsearch.yaml                podanti-affinity-deployment.yaml
example-ingress.yaml              podanti-affinity1-deployment.yaml
filebeat-agent.yaml               pvc-oci.yaml
fluentd.yaml                      pvc.yaml
foo-allow-to-hello.yaml           quota-pod.yaml
guestbook-frontend-svc.yaml       quota-pod1.yaml
guestbook-frontend.yaml           quota.yaml
hello-allow-from-foo.yaml         readiness-pod.yaml
ingress-app1.yaml                 readiness-svc.yaml
ingress-app2.yaml                 redis-master-svc.yaml
ingress-route.yaml                redis-master.yaml
kibana-elk.yaml                   redis-slave-svc.yaml
kibana.yaml                       redis-slave.yaml
label-deployment.yaml             requirements.txt
liveness-pod.yaml                 role-dev.yaml
logstash-configmap.yaml           rolebind.yaml
logstash-deployment.yaml          script.sh
logstash-svc.yaml                 security-cxt-nonroot.yaml
metrics-server.yaml               security-cxt-priv.yaml
multi-container.yaml              security-cxt-readonly.yaml
multi-pod-configmap.yaml          security-cxt-rmcap.yaml
multi-pod-nginx.yaml              security-cxt-time.yaml
multi-prod-consumer.yaml          security-cxt.yaml
namespace.yaml                    tt-pod.yaml
network-policy.yaml               tt-pod1.yaml
nfs-pv.yaml
root@master:/home/ubuntu/Kubernetes#
```

1. Create the demo-pod.yaml file to define the nginx server

```
$ vim demo-pod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: demo-pod
  labels:
    name: demo-pod
spec:
  containers:
  - name: demo-container
    image: nginx
~
~
~
~
```

2. Create nginx pod by applying the demo-pod.yaml file

```
$ kubectl apply -f demo-pod.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl apply -f demo-pod.yaml
pod/demo-pod created
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

3. Verifying the status of the newly created nginx pod

```
$ kubectl get pods
$ kubectl get pods -w
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pods
NAME       READY   STATUS              RESTARTS   AGE
demo-pod   0/1     ContainerCreating   0          17s
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pods -w
NAME       READY   STATUS    RESTARTS   AGE
demo-pod   1/1     Running   0          23s
```

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

```
^Croot@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE     IP          NODE      NOMINATED NODE   READINESS GATES
demo-pod   1/1     Running   0          2m16s   10.46.0.1   worker1   <none>           <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

5 EXPOSING NGINX SERVER WITH IN CLUSTER USING EXPOSE COMMAND

1. Exposing the pod on cluster level with ClusterIP type of service

```
$ kubectl expose pod demo-pod --port 80 --target-port 80 --type ClusterIP
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl expose pod demo-pod --port 80 --target-port 80 --type ClusterIP
service/demo-pod exposed
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2. Listing the demo-pod service with kubectl get command. Get the ClusterIP address of the service

```
$ kubectl get svc
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get svc
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
demo-pod      ClusterIP   10.108.93.193   <none>        80/TCP    6s
kubernetes    ClusterIP   10.96.0.1       <none>        443/TCP   3h21m
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

3. Describing the demo-pod service to see the port and endpoint details

```
$ kubectl describe svc demo-pod
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe svc demo-pod
Name:              demo-pod
Namespace:         default
Labels:            name=demo-pod
Annotations:       <none>
Selector:          name=demo-pod
Type:              ClusterIP
IP:                10.108.93.193
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         10.46.0.1:80
Session Affinity:  None
Events:            <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

4. Check the reachability of the service and pod from master node with curl command

```
$ curl <Service Cluster-IP>
$ curl 10.108.93.193  (Use your own Cluster IP)
```

6 EXPOSING NGINX SERVER OUTSIDE CLUSTER USING EXPOSE COMMAND

1. Exposing the pod outside cluster level with **NodePort** type of service. Edit the existing service to NodePort type

```
$ kubectl edit svc demo-pod
```

```
spec:
  clusterIP: 10.108.93.193
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    name: demo-pod
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

Before Exposing the pod –

```
root@master: ~/Kubernetes
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2020-09-15T18
  labels:
    name: demo-pod
  name: demo-pod
  namespace: default
  resourceVersion: "4467"
  selfLink: /api/v1/namespaces/defa
  uid: 721c05fa-6e13-49fc-a5a5-c0c6
spec:
  clusterIP: 10.103.114.66
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    name: demo-pod
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
```

After Exposing the pod –

```
root@master: ~/Kubernetes
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2020-09-15T1
  labels:
    name: demo-pod
  name: demo-pod
  namespace: default
  resourceVersion: "4467"
  selfLink: /api/v1/namespaces/def
  uid: 721c05fa-6e13-49fc-a5a5-c0c
spec:
  clusterIP: 10.103.114.66
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    name: demo-pod
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl edit svc demo-pod
service/demo-pod edited
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

$ kubectl get svc

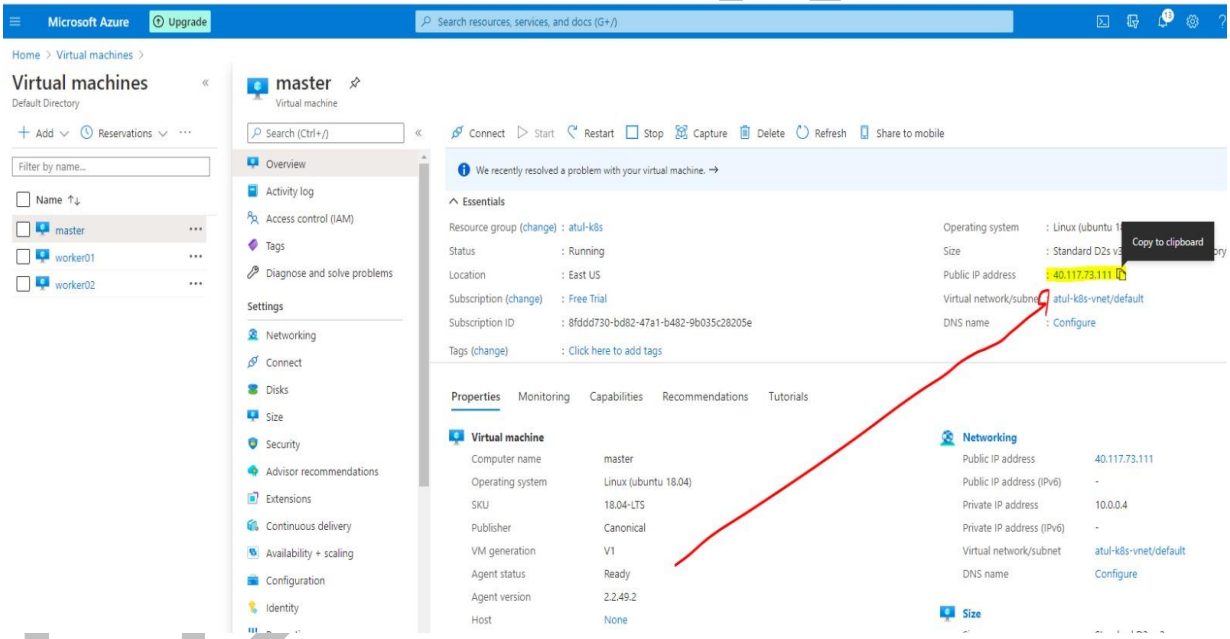
```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
demo-pod     NodePort    10.108.93.193   <none>        80:32561/TCP   18m
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        3h39m
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2. Describe the service to get details on the service type and port

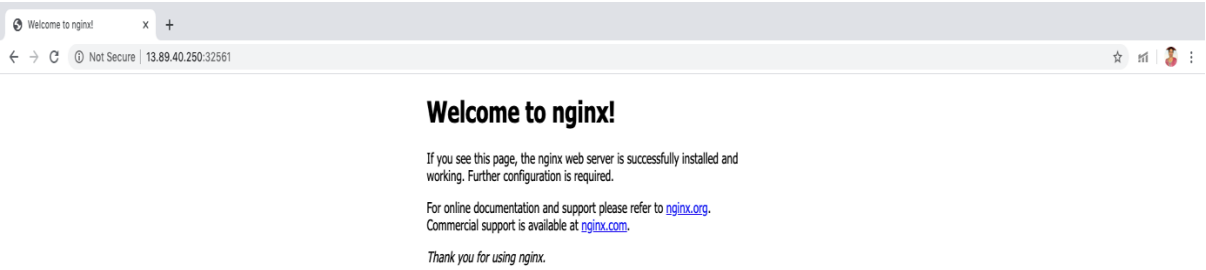
$ kubectl describe svc demo-pod

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe svc demo-pod
Name:                     demo-pod
Namespace:                default
Labels:                   name=demo-pod
Annotations:              <none>
Selector:                 name=demo-pod
Type:                     NodePort
IP:                       10.108.93.193
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  32561/TCP
Endpoints:                10.46.0.1:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

3. Check for application reachability from outside the cluster. Open web browser on the laptop and access the nginx application through **any node's publicIP address** and port number allocated by Kubernetes, as in this example its **32561**



(Check your port number From NodePort as given in above screen is for example 32561 and to access nginx use your azure any worker node IP address)


Welcome to nginx!
Not Secure | 13.89.40.250:32561
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.

7 DELETING RESOURCES CREATED IN THIS LAB EXERCISE

1. Deleting the pod using kubectl delete command

```
$ kubectl delete -f demo-pod.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl delete -f demo-pod.yaml
pod "demo-pod" deleted
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2. Deleting the service demo-pod

```
$ kubectl delete svc demo-pod
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl delete svc demo-pod
service "demo-pod" deleted
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

8 SUMMARY

In this guide we Covered:

- Deploying a simple application as pod