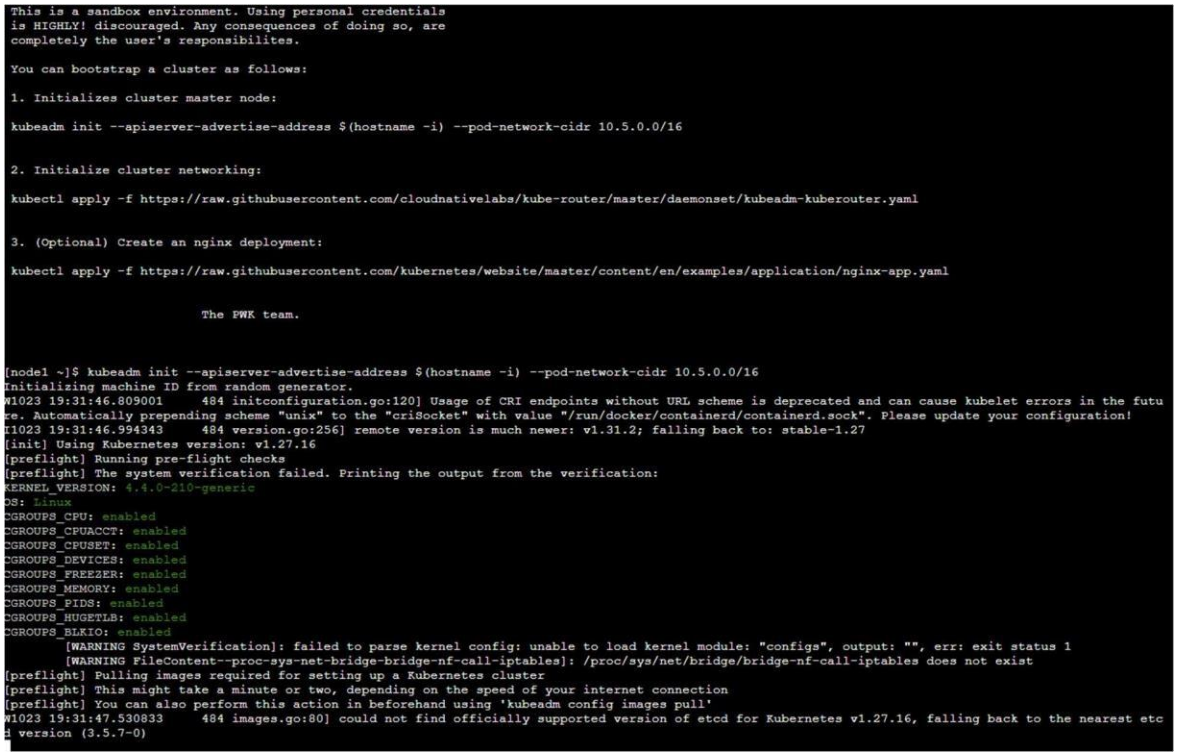
SVKM'S NMIMS (Deemed-to-be University)
MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT AND ENGINEERING
NAVI MUMBAI CAMPUS

Roll Number: A053	Name: Nishant Baruah
Class: B. Tech CE (Sem VII)	Batch: B2
Date of Experiment: 22-10-24	Date of Submission: 05-11-24
Subject: Cloud Computing	Practical No: 10

AIM:

Lab Manual: Working with Kubernetes to run a highly available web server

Task:



```
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to
 4m0s
[apiclient] All control plane components are healthy after 6.004001 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node node1 as control-plane by adding the labels: [node-role.kubernetes.io/control-plane node.kubernetes.io/exclude-fro
m-external-load-balancers]
[mark-control-plane] Marking the node node1 as control-plane by adding the taints [node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: emj87o.cwrix5r5y6jubzhz
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.0.23:6443 --token emj87o.cwrix5r5y6jubzhz \
        --discovery-token-ca-cert-hash sha256:57a04cdb6fa4265e9a98c89ea380578b1fca096633ecdef0163b399bbe2057c0
Waiting for api server to startup
Warning: resource daemonsets/kube-proxy is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. k
ubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will
 be patched automatically.
daemonset.apps/kube-proxy configured
No resources found
```

```
[node1 ~]$ kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml
configmap/kube-router-cfg created
daemonset.apps/kube-router created
serviceaccount/kube-router created
clusterrole.rbac.authorization.k8s.io/kube-router created
clusterrolebinding.rbac.authorization.k8s.io/kube-router created
[node1 ~]$
```

```
[node2 ~]$ kubeadm join 192.168.0.23:6443 --token 81rstv.jno0j886vt6sx2vu --discovery-token-ca-cert-hash sha256:57a04cdb6fa4265e9a98c89ea380578b1fca0966
33ecdef0163b399bbe2057c0
Initializing machine ID from random generator.
W1023 19:35:47.911619     865 initconfiguration.go:120] Usage of CRI endpoints without URL scheme is deprecated and can cause kubelet errors in the futu
re. Automatically prepending scheme "unix" to the "criSocket" with value "/run/docker/containerd/containerd.sock". Please update your configuration!
[preflight] Running pre-flight checks
[preflight] The system verification failed. Printing the output from the verification:
KERNEL_VERSION: 4.4.0-210-generic
OS: Linux
CGROUPS_CPU: enabled
CGROUPS_CPUACCT: enabled
CGROUPS_CPUSET: enabled
CGROUPS_DEVICES: enabled
CGROUPS_FREEZER: enabled
CGROUPS_MEMORY: enabled
CGROUPS_PIDS: enabled
CGROUPS_HUGETLB: enabled
CGROUPS_BLKIO: enabled
        [WARNING SystemVerification]: failed to parse kernel config: unable to load kernel module: "configs", output: "", err: exit status 1
        [WARNING FileContent--proc-sys-net-bridge-bridge-nf-call-iptables]: /proc/sys/net/bridge/bridge-nf-call-iptables does not exist
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.
```

```
[node1 ~]$ kubectl get nodes
NAME     STATUS   ROLES           AGE     VERSION
node1    Ready    control-plane   3m55s   v1.27.2
node2    Ready    <none>          18s     v1.27.2
```

```
[node1 project]$ vi deployment.yaml
[node1 project]$ kubectl apply -f deployment.yaml
deployment.apps/nginx-deployment created
[node1 project]$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-584dbbbc9f-5r88j   1/1     Running   0          13s
nginx-deployment-584dbbbc9f-69xww   1/1     Running   0          13s
nginx-deployment-584dbbbc9f-g79rz   1/1     Running   0          13s
[node1 project]$
```

Conclusion

Learned how to successfully set up a Nginx web server running with 3 replicas using Kubernetes cluster.