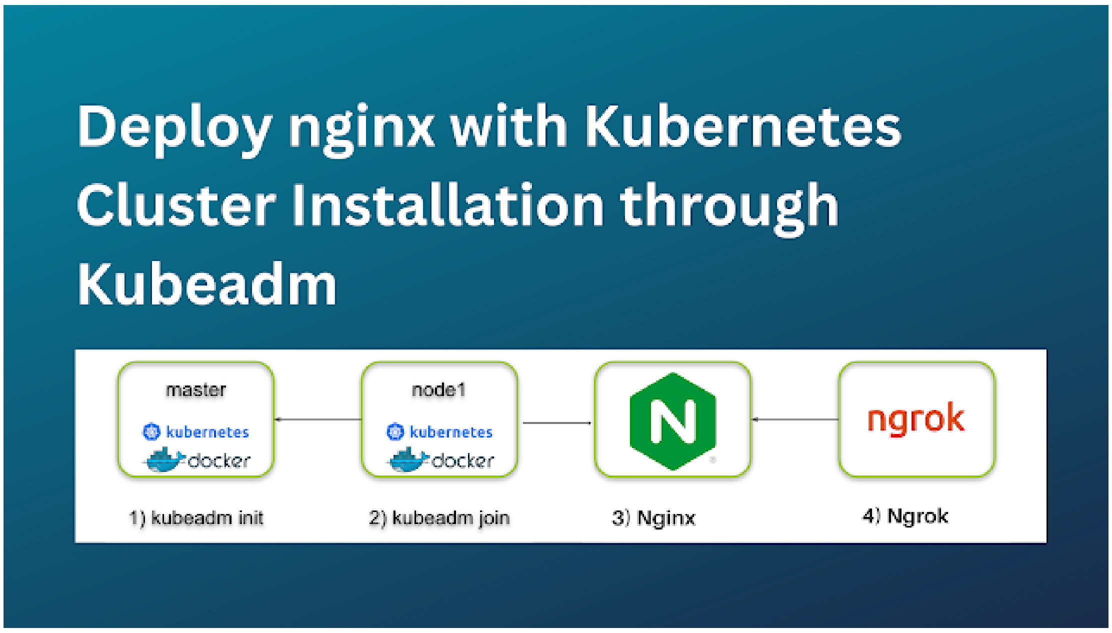
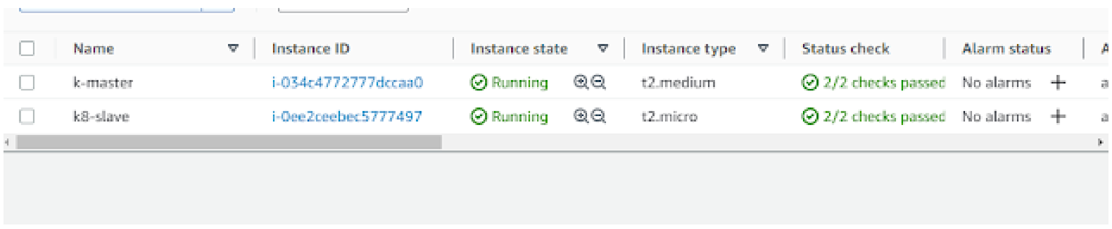
Deploy nginx with Kubernetes Cluster Installation through Kubeadm

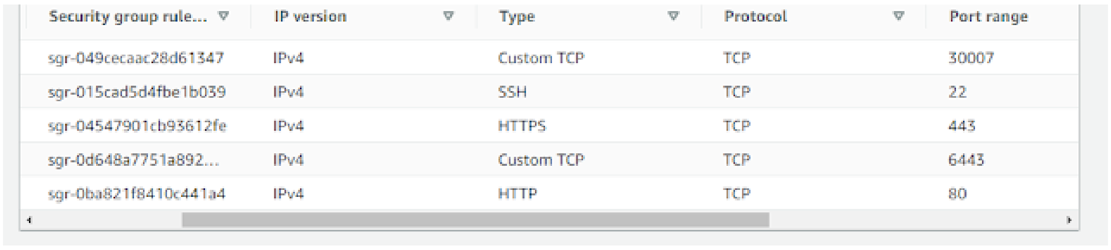
May 13, 2023



First we need to create 2 EC2 instances i.e t2 medium for master node and t2 micro for worker node. And allow ports for their connection.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status
k-master	i-034c4772777dccaa0	Running	t2.medium	2/2 checks passed	No alarms
k8-slave	i-0ee2ceebec5777497	Running	t2.micro	2/2 checks passed	No alarms



Security group rule...	IP version	Type	Protocol	Port range
sgr-049cecaac28d61347	IPv4	Custom TCP	TCP	30007
sgr-015cad5d4fbe1b039	IPv4	SSH	TCP	22
sgr-04547901cb93612fe	IPv4	HTTPS	TCP	443
sgr-0d648a7751a892...	IPv4	Custom TCP	TCP	6443
sgr-0ba821f8410c441a4	IPv4	HTTP	TCP	80

Run all below command on both master and worker nodes

```
sudo apt update -y
```

```
sudo apt install docker.io -y
```

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

```
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

```
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
sudo apt update -y sudo apt install kubeadm=1.20.0-00 kubectl=1.20.0-00 kubelet=1.20.0-00 -y
```

Run these commands on the master node.

```
sudo su
```

```
kubeadm init
```

```
mkdir -p $HOME/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
kubectl apply -f
https://github.com/weaveworks/weave/releases/download/v2.8.1/weave
-daemonset-k8s.yaml

sudo apt-get update

sudo apt-get -y install containerd

kubeadm token create --print-join-command
```

Run these commands on the worker node.

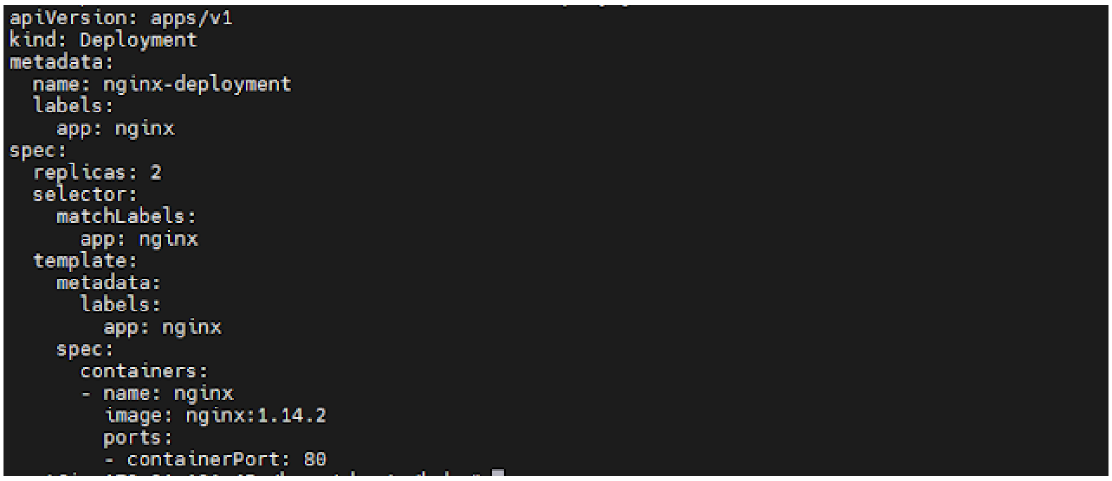
```
sudo su

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f
https://github.com/weaveworks/weave/releases/download/v2.8.1/weave
-daemonset-k8s.yaml

sudo apt-get update

sudo apt-get -y install containerd

kubeadm reset pre-flight checks
```

-----> Paste the Join command on worker node with --v=5

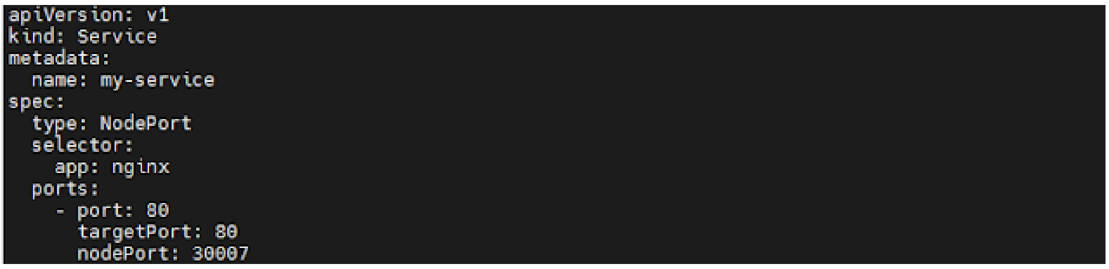
Note: Containerd is a container runtime that provides a set of high-level APIs to manage the lifecycle of container images and containers. Kubernetes is a container orchestration platform that automates the deployment, scaling, and management of containerized applications.

Also Read: Provision and Manage AWS EC2 Instances and S3 Bucket Using Terraform IaC

Now it's time for Deploying Nginx on a Kubernetes cluster that requires the creation of two important files: the deployment.yaml and service.yaml. These files are crucial as they provide instructions to Kubernetes on how to manage the deployment of Nginx and how to make it accessible to other components of the cluster. The deployment.yaml file defines the desired state of the Nginx deployment, including the number of replicas and the container image to be used. On the other hand, the service.yaml file specifies how the Nginx deployment should be exposed to the rest of the cluster by defining the network endpoints and the ports to be used. Together, these files enable seamless deployment and management of Nginx on a Kubernetes cluster, providing a reliable and scalable solution for web serving.



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```



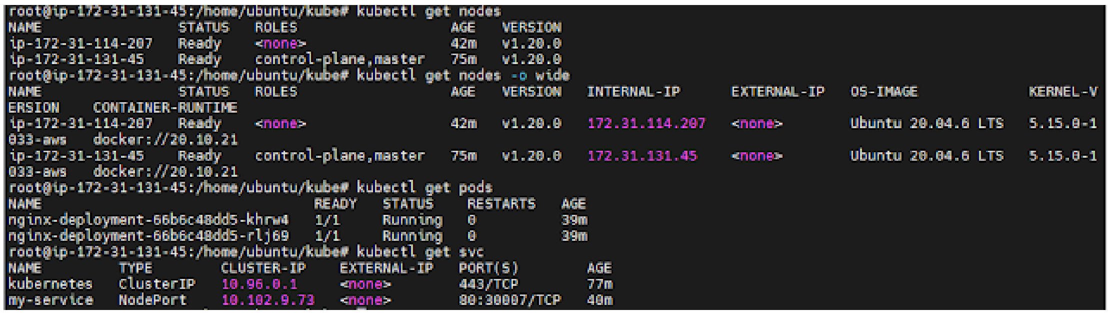
```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30007
```

kubectl apply -f service.yaml

kubectl get svc

kubectl get pods

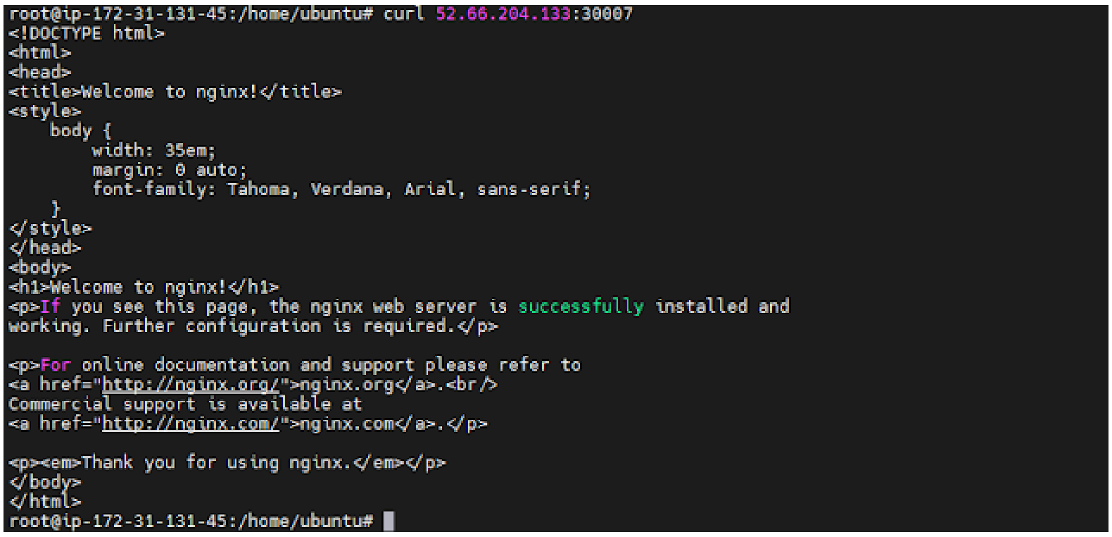
kubectl cluster-info



```
root@ip-172-31-131-45:/home/ubuntu/kube# kubectl get nodes
NAME                STATUS   ROLES                  AGE   VERSION
ip-172-31-114-207   Ready    <none>                 42m   v1.20.0
ip-172-31-131-45    Ready    control-plane,master   75m   v1.20.0
root@ip-172-31-131-45:/home/ubuntu/kube# kubectl get nodes -o wide
NAME                STATUS   ROLES                  AGE   VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE             KERNEL-V
ERSION    CONTAINER-RUNTIME
ip-172-31-114-207   Ready    <none>                 42m   v1.20.0   172.31.114.207   <none>        Ubuntu 20.04.6 LTS   5.15.0-1
033-aws   docker://20.10.21
ip-172-31-131-45    Ready    control-plane,master   75m   v1.20.0   172.31.131.45    <none>        Ubuntu 20.04.6 LTS   5.15.0-1
033-aws   docker://20.10.21
root@ip-172-31-131-45:/home/ubuntu/kube# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-66b6c48dd5-khrw4   1/1     Running   0          39m
nginx-deployment-66b6c48dd5-rlj69   1/1     Running   0          39m
root@ip-172-31-131-45:/home/ubuntu/kube# kubectl get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP        77m
my-service   NodePort    10.102.9.73   <none>        80:30007/TCP   40m
```

Now check deployment locally

curl 52.66.204.133:30007



```
root@ip-172-31-131-45:/home/ubuntu# curl 52.66.204.133:30007
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@ip-172-31-131-45:/home/ubuntu#
```

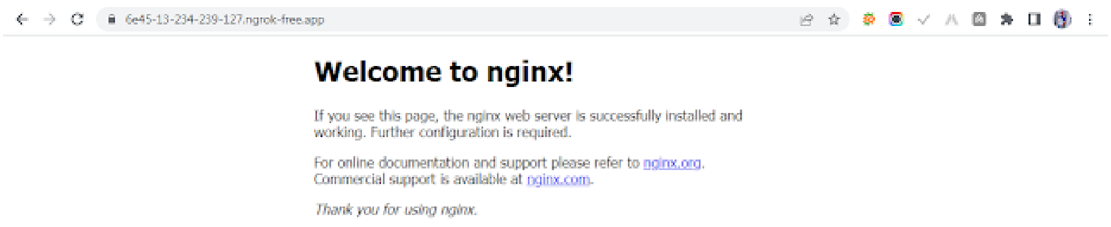
After verifying Nginx's functionality locally, we can now assess its performance globally using ngrok. Ngrok provides a secure way to expose a web server running locally to the internet, making it possible to test and debug web applications from anywhere. By using ngrok,

Commands to install ngrok (Master Nodes)

sudo snap install ngrok

ngrok config add-authtoken 2MOTCXpmZcoDRE7gGp2QFVj5ZAr_X3qCwSeDgS1osnEXXrDU

ngrok http 52.66.204.133:30007


6e45-13-234-239-127.ngrok-free.app

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.

In conclusion,

we have connected the master node and worker node using Kubeadm and deployed Nginx on our Kubernetes cluster. We have also verified the deployment's functionality both locally and globally using ngrok. This achievement showcases the power and flexibility of Kubernetes as a platform for container orchestration and highlights the importance of efficient and scalable web serving in modern software development.

KUBERNETES