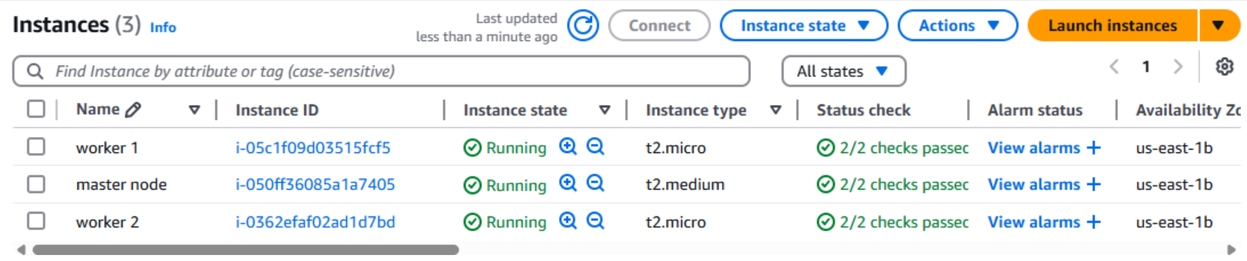
Creating Kubernetes Cluster

Workflow

- ☑ Master – node, worker -1 and worker - 2
  - Install container runtime – Containerd
  - Disable swap
  - Install kubelet, kubectl and kubeadm.
  - Install runc and CNI plugins.
  - Enable IPv4 forwarding on all nodes.
- ☑ Master – node
  - Initialize the cluster. (kubeadm init)
- ☑ Worker – nodes.
  - Join each worker node using kubeadm init
- ☑ Master – node
  - Install pod-network add on.
- ☑ Master – node
  - Deploy workload.


Instances (3) Info
Last updated less than a minute ago
Connect
Instance state ▼
Actions ▼
Launch instances ▼
Find Instance by attribute or tag (case-sensitive)
All states ▼
1

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zc
□	worker 1	i-05c1f09d03515fcf5	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-1b
□	master node	i-050ff36085a1a7405	✓ Running	t2.medium	✓ 2/2 checks passec	View alarms +	us-east-1b
□	worker 2	i-0362efaf02ad1d7bd	✓ Running	t2.micro	✓ 2/2 checks passec	View alarms +	us-east-1b

Install Containerd – container runtime

👉 Containerd documentation.

Please note that you have to install runc and CNI plugins from this documentation as well.

Also enable containerd as a systemd service.

```
curl -o /usr/lib/systemd/system/containerd.service https://raw.githubusercontent.com/containerd/containerd/main/containerd.service
```

- ✓ This command download the content of the provided link to the specified path that it systemd. -o flag indicates the path to be downloaded.

After installing containerd, CNI plugins and runc, check whether containerd is running as a sytemd service.

Install kubernetes

Then install kubeadm using this documentation : 👉 Install kubeadm

Then you have to enable IPv4 forwarding in all nodes.

```
sudo sysctl -w net.ipv4.ip_forward=1
```

Verify the changes

```
cat /proc/sys/net/ipv4/ip_forward
```

Make the changes persistent

```
sudo nano /etc/sysctl.conf
```

Add or modify the following line:

net.ipv4.ip_forward=1

Apply the changes :

```
sudo sysctl -p
```

Before proceeding to the next step, we have to decide which pod network add-on to be installed.

Initiate Cluster

In this particular example, we will be using Calico pod network add-on and Calico uses 192.168.0.0/16 CIDR block. Therefore, in master node, initiate cluster using;

```
sudo kubeadm init --pod-network-cidr=192.168.0.0/16
```

Then it will display the commands to proceed with and how to join worker nodes. To install a pod network add-on, you can use the following command to install Calico pod network add-on.

```
kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.29.3/manifests/calico.yaml
```

After that, you can join any number of worker nodes to the cluster.

You can regenerate the command on the master node:

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

Deploy Workload

Creating a Deployment

👉 Application Deployment

- Write a Deployment Manifest

  Example: nginx-deployment.yaml

```yaml
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:1.23.2
        ports:
        - containerPort: 80
```

- Apply the Deployment

```
kubectl apply -f nginx-deployment.yaml
```

- Check the Deployment:

```
kubectl get deployments
```

- Expose the Deployment
  To make the Nginx service accessible, create a **Service**. For simplicity, we'll use a NodePort type.
  Example: `nginx-service.yaml`

```
apiVersion: v1

kind: Service

metadata:

  name: nginx-service

spec:

  selector:

    app: nginx

  ports:

    - protocol: TCP

      port: 80

      targetPort: 80

  type: NodePort
```

- Apply the Service:

```
kubectl apply -f nginx-service.yaml
```

NOTE: ⚠

Indentation is very important in yaml files.

- Access the Application

```
kubectl get services
```

You'll see a port like 30000-32767. Access the application using:

http://<NodeIP>:<NodePort>

NOTE: ⚠

You have to open the <NodePort> Port in security groups or any firewalls used.

Monitor and Manage

- **View Pods**: kubectl get pods
- **Inspect Pod Logs**: kubectl logs <pod-name>
- **Scale Deployment**: Increase or decrease replicas:

```
kubectl scale deployment nginx-deployment --replicas=3
```