Node -A node is a machine – physical or virtual – on which kubernetesis installed. A node is a worker machine and this is were containers will be launched by Kubernetes

A cluster is a set of nodes grouped together. This way even if one node fails you have your application still accessible from the other nodes. Moreover having multiple nodes helps in sharing load as well

Master -The master is another node with Kubernetes installed in it, and is configured as a Master. The master watches over the nodes in the cluster and is responsible for the actual orchestration of containers on the worker nodes.

```
adminvm@demo1:~$ minikube start --driver=docker
😄  minikube v1.37.0 on Ubuntu 24.04
```

```
adminvm@demo1:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

```
adminvm@demo1:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE     VERSION
minikube   Ready    control-plane   3m29s   v1.34.0
adminvm@demo1:~$
```

```
adminvm@demo1:~$ kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.10
deployment.apps/hello-minikube created
```

```
adminvm@demo1:~$ kubectl get deployments
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
hello-minikube   1/1     1            1           56s
```

```
adminvm@demo1:~$ kubectl expose deployment hello-minikube --type=NodePort --port=8080
service/hello-minikube exposed
```

```
adminvm@demo1:~$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
hello-minikube-858b7b9984-pgmzq      1/1     Running   0          41m
nginx                                1/1     Running   0          17s
adminvm@demo1:~$
```

```
adminvm@demo1:~$ kubectl describe pods nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Thu, 11 Sep 2025 14:10:22 +0000
Labels:           run=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.0.4
IPs:
  IP:  10.244.0.4
Containers:
  nginx:
    Container ID:   docker://df5e786e1b6cd5d78c3651b07c641df37ebb9ea167d6d5e51f0929fb20ec6986
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:d5f28ef21aabddd098f3dbc21fe5b7a7d7a184720bc07da0b6c9b9820e97f25e
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 11 Sep 2025 14:10:29 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-jk69r (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
```

```
adminvm@demo1:~$ kubectl get pods -o wide
NAME                                 READY   STATUS    RESTARTS   AGE    IP           NODE       NOMINATED NODE   READINESS GATES
hello-minikube-858b7b9984-pgmzq      1/1     Running   0          44m    10.244.0.3   minikube   <none>           <none>
nginx                                1/1     Running   0          4m4s   10.244.0.4   minikube   <none>           <none>
```

Pods with YAML

```
adminvm@demo1:~$ cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
   name: nginx
   labels:
     app: nginx
     tier: frontend
spec:
    containers:
      - name: nginx
        image: nginx
```

```
adminvm@demo1:~$ kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Thu, 11 Sep 2025 14:10:22 +0000
Labels:           app=nginx
                  run=nginx
                  tier=frontend
Annotations:      <none>
Status:           Running
IP:               10.244.0.4
IPs:
```

replicaset

```
adminvm@demo1:~$ kubectl create -f replicaset.yaml
replicaset.apps/myapp-replicaset created
```

```
adminvm@demo1:~$ kubectl get replicaset
NAME                        DESIRED   CURRENT   READY   AGE
hello-minikube-858b7b9984   1         1         1       3h
myapp-replicaset            3         3         3       83s
```

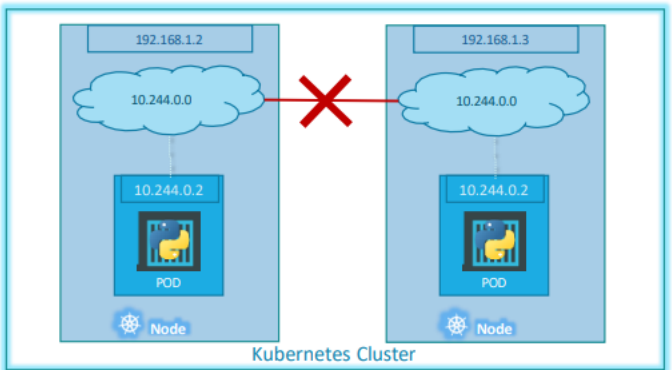
```
adminvm@demo1:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
hello-minikube-858b7b9984-pgmzq     1/1     Running   0          3h1m
myapp-replicaset-fw29w              1/1     Running   0          2m47s
myapp-replicaset-jqng2              1/1     Running   0          2m47s
myapp-replicaset-jqptm              1/1     Running   0          2m47s
nginx                               1/1     Running   0          141m
adminvm@demo1:~$ kubectl delete pod myapp-replicaset-fw29w
pod "myapp-replicaset-fw29w" deleted from default namespace
adminvm@demo1:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
hello-minikube-858b7b9984-pgmzq     1/1     Running   0          3h3m
myapp-replicaset-jqng2              1/1     Running   0          3m54s
myapp-replicaset-jqptm              1/1     Running   0          3m54s
myapp-replicaset-rjvl5              1/1     Running   0          28s
nginx                               1/1     Running   0          142m
```

Deployment

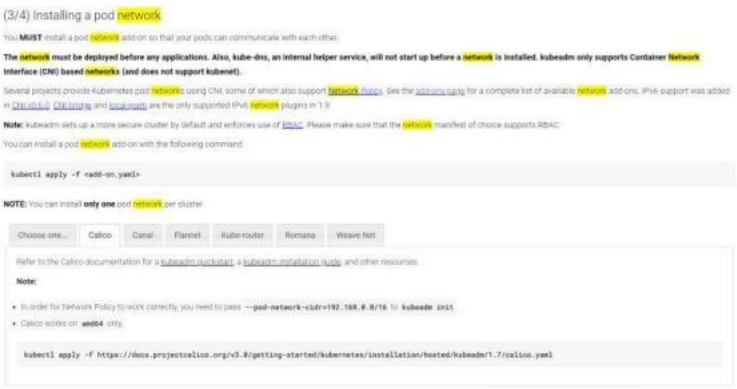
```
adminvm@demo1:~$ kubectl create -f deployment.yaml
deployment.apps/myapp-deployment created
adminvm@demo1:~$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
hello-minikube     1/1     1            1           3h9m
myapp-deployment   3/3     3            3           19s
```


Cluster Networking

- All containers/PODs can communicate to one another without NAT
- All nodes can communicate with all containers and vice-versa without NAT



Cluster Networking Setup



(3/4) Installing a pod network

You **MUST** install a pod network add-on so that your pods can communicate with each other.

The network must be deployed before any applications. Also, kube-dns, an internal helper service, will not start up before a network is installed. kubeadm only supports Container Network Interface (CNI) based networks (and does not support kubenet).

Several projects provide Kubernetes pod networks using CNI, some of which also support Network Policy. See the add-ons page for a complete list of available network add-ons. IPv6 support was added in CNI v0.6.0. CNI bridge and local-ipam are the only supported IPv6 network plugins in 1.9.

Note: kubeadm sets up a more secure cluster by default and enforces use of RBAC. Please make sure that the network manifest of choice supports RBAC.

You can install a pod network add-on with the following command:

```
kubectl apply -f <add-on.yaml>
```

NOTE: You can install **only one** pod network per cluster.

Choose one... | Calico | Canal | Flannel | Kube-router | Romana | Weave Net

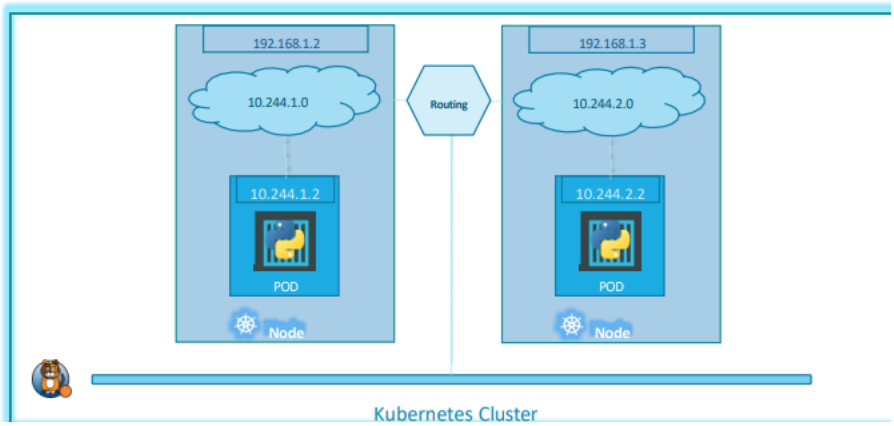
Refer to the Calico documentation for a kubeadm quickstart, a kubeadm installation guide, and other resources.

Note:

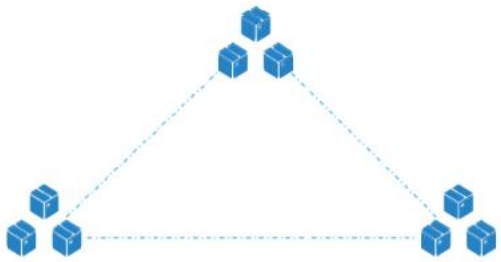
- In order for Network Policy to work correctly, you need to pass `--pod-network-cidr=192.168.0.0/16` to `kubeadm init`.
- Calico works on `amd64` only.

```
kubectl apply -f https://docs.projectcalico.org/v3.0/getting-started/kubernetes/installation/hosted/kubeadm/1.7/calico.yaml
```

Cluster Networking

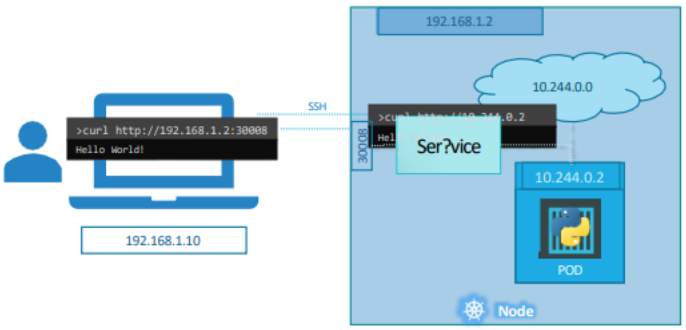


Services



Kubernetes Services enable communication between various components within and outside of the application. Kubernetes Services helps us connect applications together with other applications or users. For example, our application has groups of PODs running various sections, such as a group for serving front-end load to users, another group running back-end processes, and a third group connecting to an external data source. It is Services that enable connectivity between these groups of PODs. Services enable the front-end application to be made available to users, it helps communication between back-end and front-end PODs, and helps in establishing connectivity to an external data source. Thus services enable loose coupling between microservices in our application.

Service



Services Types

The first one is what we discussed already – NodePort were the service makes an internal POD accessible on a Port on the Node. The second is ClusterIP – and in this case the service creates a virtual IP inside the cluster to enable communication between different services such as a set of front-end servers to a set of backend servers. The third type is a LoadBalancer, were it provisions a load balancer for our service in supported cloud providers. A good example of that would be to distribute load across different web servers.

Service - NodePort

service-definition.yml

```
apiVersion: v1
kind: Service
metadata:
     name: myapp-service

spec:
     type: NodePort
     ports:
      - targetPort: 80
        port: 80
        nodePort: 30008
     selector:
```

pod-definition.yml

```
> kubectl create -f service-definition.yml
service "myapp-service" created

> kubectl get services
NAME            TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1      <none>        443/TCP        16d
myapp-service   NodePort    10.106.127.123 <none>        80:30008/TCP   5m

> curl http://192.168.1.2:30008
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
```

metadata:
  name: myapp-pod
  labels:
    app: myapp
    type: front-end

ClusterIP