Prerequisites

- Multiple Linux servers running CentOS 7 (1 Master Node, Multiple Worker Nodes)
- A user account on every system with **sudo** or root privileges
- The **yum** package manager, included by default
- Command-line/terminal window

Steps for Installing Kubernetes on CentOS 7

To use Kubernetes, you need to install a **containerization engine**. Currently, the most popular container solution is **Docker**. Docker needs to be installed on CentOS, **both** on the Master Node and the Worker Nodes.

Step 1: Configure Kubernetes Repository

Kubernetes packages are not available from official CentOS 7 repositories. This step needs to be performed on the Master Node, and each Worker Node you plan on utilizing for your container setup. Enter the following command to retrieve the Kubernetes repositories.

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes
-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg h
ttps://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

Note: If using the sudo command, append it not only to the **cat** command but to the restricted file as well.

Step 2: Install *kubelet*, *kubeadm*, and *kubectl*

These 3 basic packages are required to be able to use Kubernetes. Install the following package(s) on each node:

```
sudo yum install -y kubelet kubeadm kubectl
```

```
systemctl enable kubelet
```

```
systemctl start kubelet
```

You have now successfully installed Kubernetes, including its tools and basic packages.

```
Installed:
  kubeadm.x86_64 0:1.16.2-0   kubectl.x86_64 0:1.16.2-0   kubelet.x86_64 0:1.16.2-0

Dependency Installed:
  conntrack-tools.x86_64 0:1.4.4-5.el7_7.2
  cri-tools.x86_64 0:1.13.0-0
  kubernetes-cni.x86_64 0:0.7.5-0
  libnetfilter_cthelper.x86_64 0:1.0.0-10.el7_7.1
  libnetfilter_cttimeout.x86_64 0:1.0.0-6.el7_7.1
  libnetfilter_queue.x86_64 0:1.0.2-2.el7_2
  socat.x86_64 0:1.7.3.2-2.el7

Complete!
```

Before deploying a cluster, make sure to set hostnames, configure the firewall, and kernel settings.

Step 3: Set Hostname on Nodes

To give a unique hostname to each of your nodes, use this command:

```
sudo hostnamectl set-hostname master-node
```

or

```
sudo hostnamectl set-hostname worker-node1
```

In this example, the master node is now named master-node, while a worker node is named worker-node1.

Make a host entry or DNS record to resolve the hostname for all nodes:

```
sudo vi /etc/hosts
```

With the entry:

```
192.168.1.10 master.phoenixnap.com master-node
192.168.1.20 node1. phoenixnap.com node1 worker-node
```

Step 4: Configure Firewall

The nodes, containers, and pods need to be able to communicate across the cluster to perform their functions. Firewalld is enabled in CentOS by default on the front-end. Add the following ports by entering the listed commands.

On the Master Node enter:

```
sudo firewall-cmd --permanent --add-port=6443/tcp
sudo firewall-cmd --permanent --add-port=2379-2380/tcp
sudo firewall-cmd --permanent --add-port=10250/tcp
sudo firewall-cmd --permanent --add-port=10251/tcp
sudo firewall-cmd --permanent --add-port=10252/tcp
sudo firewall-cmd --permanent --add-port=10255/tcp
sudo firewall-cmd --reload
```

Each time a port is added the system confirms with a ‘success’ message.

```
[phoenixnap@localhost ~]$ sudo firewall-cmd --permanent --add-port=6443/tcp
success
[phoenixnap@localhost ~]$ sudo firewall-cmd --permanent --add-port=2379-2380/tcp
success
[phoenixnap@localhost ~]$ sudo firewall-cmd --permanent --add-port=10250/tcp
success
[phoenixnap@localhost ~]$ sudo firewall-cmd --permanent --add-port=10251/tcp
success
[phoenixnap@localhost ~]$ sudo firewall-cmd --permanent --add-port=10252/tcp
success
[phoenixnap@localhost ~]$ sudo firewall-cmd --permanent --add-port=10255/tcp
success
```

Enter the following commands on each worker node:

```
sudo firewall-cmd --permanent --add-port=10251/tcp
sudo firewall-cmd --permanent --add-port=10255/tcp
firewall-cmd --reload
```

Step 5: Update Iptables Settings

Set the **`net.bridge.bridge-nf-call-iptables`** to '1' in your sysctl config file. This ensures that packets are properly processed by IP tables during filtering and port forwarding.

```
cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sysctl --system
```

Step 6: Disable SELinux

The containers need to access the host filesystem. SELinux needs to be set to permissive mode, which effectively disables its security functions.

Use following commands to disable SELinux:

```
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/s
elinux/config
```

Step 7: Disable SWAP

Lastly, we need to disable SWAP to enable the kubelet to work properly:

```
sudo sed -i '/swap/d' /etc/fstab
sudo swapoff -a
```

How to Deploy a Kubernetes Cluster

Step 1: Create Cluster with kubeadm

Initialize a cluster by executing the following command:

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

The process might take several minutes to complete based on network speed. Once this command finishes, it displays a kubeadm join message. Make a note of the entry and use it to join worker nodes to the cluster at a later stage.

Note: This tutorial uses the **flannel** virtual network add-on. The **10.244.0.0/16** network value reflects the configuration of the *kube-flannel.yml* file. If you plan to use a different third-party provider, change the **`--pod-network-cidr`** value to match your provider's requirements.

Step 2: Manage Cluster as Regular User

To start using the cluster you need to run it as a regular user by typing:

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

Step 3: Set Up Pod Network

A Pod Network allows nodes within the cluster to communicate. There are several available Kubernetes networking options. Use the following command to install the **flannel** pod network add-on:

```
sudo kubectl apply -f https://raw.githubusercontent.com/coreos
/flannel/master/Documentation/kube-flannel.yml
```

If you decide to use flannel, edit your firewall rules to allow traffic for the flannel default port **8285**.

Step 4: Check Status of Cluster

Check the status of the nodes by entering the following command on the master server:

```
sudo kubectl get nodes
```

Once a pod network has been installed, you can confirm that it is working by checking that the CoreDNS pod is running by typing:

```
sudo kubectl get pods --all-namespaces
```

Step 5: Join Worker Node to Cluster

As indicated in **Step 1**, you can use the **`kubeadm join`** command on each worker node to connect it to the cluster.

```
kubeadm join --discovery-token cfgrty.1234567890jyrfgd --disco
very-token-ca-cert-hash sha256:1234..cdef 1.2.3.4:6443
```

Replace the codes with the ones from your master server. Repeat this action for each worker node on your cluster.

Conclusion

You have installed Kubernetes on CentOS successfully and can now manage clusters across multiple servers. If you have bare metal server, you may want to look into our guide on how to install Kubernetes on such servers.

This Kubernetes tutorial provides a good starting point for exploring the many options this versatile platform has to offer. Use Kubernetes to scale your operations more efficiently and spend less time on micromanaging containers.

For beginners who still have no experience of deploying multiple containers, **Minikube** is a great way to start. Minikube is a system for running a single node cluster locally and is excellent for learning the basics, before moving on to Kubernetes.