Table of Contents

Software Hardware List

Chapter number	Software required (With version)	Free/Proprietary	If proprietary, can code testing be performed using a trial version	If proprietary, then cost of the software	Download links to the software	Hardware specifications	OS required
1	Samba 4.x Server Software	Proprietary	Yes	$600	http://www.sampleURl.com	Common Unix Printing System with 4GB RAM and Graphic Card supporting DirectX 11	Windows
2	Latest Available version on Site.	Free	-	-	https://serverless.com/ http://apex.run/ https://github.com/aws/chalice https://kubernetes.io/docs/setup/release/		
4	Latest Available version on Site.	Free	-	-	https://jenkins.io/download/ https://www.terraform.io/downloads.html https://git-scm.com/downloads		

5	Latest Available version on Site.	Free	-	-	https://kubernetes.io/docs/setup/release/ https://docs.docker.com/install/		
7	Latest Available version on Site.	Free	-	-	https://ballerina.io/downloads/ https://www.rust-lang.org/en-US/install.html		
9							
10	Prometheus.* 2.1.3 or any latest version	Free	-	-	https://github.com/istio/istio/releases/download/1.1.0.snapshot.1/istio-1.1.0.snapshot.1-linux.tar.gz		Linux

Kubernetes installation steps on CentOS 7 or RHEL 7:

In this article we will install latest version of Kubernetes 1.7 on CentOS 7 / RHEL 7 with kubeadm utility. In my setup I am taking three CentOS 7 servers with minimal installation. One server will acts master node and rest two servers will be minion or worker nodes.

- Worker 1: 192.168.1.40
- Worker 2: 192.168.1.50
- Master Node: 192.168.1.30

Step 1: Disable SELinux & setup firewall rules

Login to your kubernetes master node and set the hostname and disable selinux using following commands:

```
~]# hostnamectl set-hostname 'k8s-master'
~]# exec bash
~]# setenforce 0
~]# sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/sysconfig/selinux
```

Set the following firewall rules:

```
[root@k8s-master ~]# firewall-cmd --permanent --add-port=6443/tcp
[root@k8s-master ~]# firewall-cmd --permanent --add-port=2379-2380/tcp
[root@k8s-master ~]# firewall-cmd --permanent --add-port=10250/tcp
[root@k8s-master ~]# firewall-cmd --permanent --add-port=10251/tcp
[root@k8s-master ~]# firewall-cmd --permanent --add-port=10252/tcp
[root@k8s-master ~]# firewall-cmd --permanent --add-port=10255/tcp
[root@k8s-master ~]# firewall-cmd --reload
[root@k8s-master ~]# modprobe br_netfilter
[root@k8s-master ~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-
iptables
```

Note: In case you don't have your own dns server then update `/etc/hosts` file on master and worker nodes:

```
192.168.1.30 k8s-master
192.168.1.40 worker-node1
192.168.1.50 worker-node2
```

```
[root@k8s-master ~]# 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
>         https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
> EOF [root@k8s-master ~]#
```

Step 3: Install Kubeadm and Docker

Once the package repositories are configured, run the beneath command to install `kubeadm` and `docker` packages.

```
[root@k8s-master ~]# yum install kubeadm docker -y
```

Start and enable kubectl and docker service

```
[root@k8s-master ~]# systemctl restart docker && systemctl enable docker
[root@k8s-master ~]# systemctl  restart kubelet && systemctl enable kubelet
```

Step 4: Initialize Kubernetes Master with 'kubeadm init'

Run the beneath command to initialize and setup Kubernetes master.

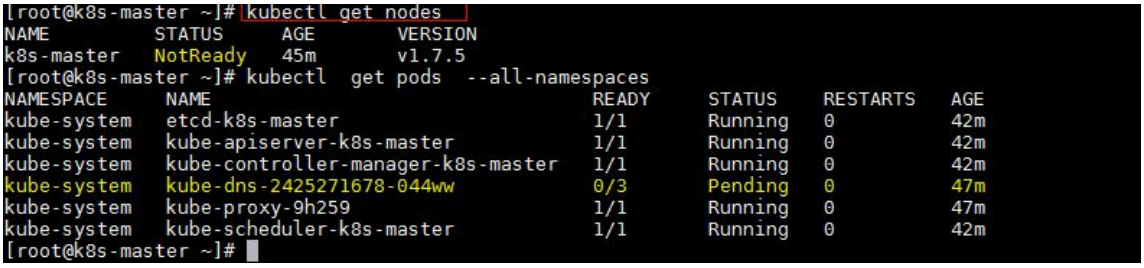
```
[root@k8s-master ~]# kubeadm init
```

As we can see in the output that Kubernetes master has been initialized successfully. Execute the beneath commands to use the cluster as root user.

```
[root@k8s-master ~]# mkdir -p $HOME/.kube
[root@k8s-master ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@k8s-master ~]# chown $(id -u):$(id -g) $HOME/.kube/config
```

Step 5: Deploy pod network to the cluster

Try to run below commands to get status of cluster and pods:



To make the cluster status ready and kube-dns status running, deploy the pod network so that containers of different host communicated each other. POD network is the overlay network between the worker nodes.

Run the beneath command to deploy network:

```
[root@k8s-master ~]# export kubever=$(kubectl version | base64 | tr -d
'\n')
[root@k8s-master ~]# kubectl apply -f
"https://cloud.weave.works/k8s/net?k8s-version=$kubever"
serviceaccount "weave-net" created
clusterrole "weave-net" created
clusterrolebinding "weave-net" created
daemonset "weave-net" created
[root@k8s-master ~]#
```

Now run the following commands to verify the status:

```
[root@k8s-master ~]# kubectl get nodes
NAME         STATUS    AGE       VERSION
k8s-master   Ready     1h        v1.7.5
[root@k8s-master ~]# kubectl  get pods  --all-namespaces
NAMESPACE     NAME                                    READY     STATUS
RESTARTS   AGE
kube-system   etcd-k8s-master                         1/1       Running   0
57m
kube-system   kube-apiserver-k8s-master               1/1       Running   0
57m
kube-system   kube-controller-manager-k8s-master      1/1       Running   0
57m
kube-system   kube-dns-2425271678-044ww               3/3       Running   0
1h
kube-system   kube-proxy-9h259                        1/1       Running   0
1h
kube-system   kube-scheduler-k8s-master               1/1       Running   0
57m
kube-system   weave-net-hdjzd                         2/2       Running   0
7m
[root@k8s-master ~]#
```

Now let's add worker nodes to the Kubernetes master nodes.

Perform the following steps on each worker node

Step 1: Disable SELinux & configure firewall rules on both the nodes

Before disabling SELinux set the hostname on the both nodes as 'worker-node1' and 'worker-node2' respectively

```
~]# setenforce 0
~]# sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/sysconfig/selinux
~]# firewall-cmd --permanent --add-port=10250/tcp
~]# firewall-cmd --permanent --add-port=10255/tcp
~]# firewall-cmd --permanent --add-port=30000-32767/tcp
~]# firewall-cmd --permanent --add-port=6783/tcp
~]# firewall-cmd  --reload
~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

Step 2: Configure Kubernetes Repositories on both worker nodes

```
~]# 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
>         https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
> EOF
```

Step 3: Install kubeadm and docker package on both nodes

```
[root@worker-node1 ~]# yum  install kubeadm docker -y
[root@worker-node2 ~]# yum  install kubeadm docker -y
```

Start and enable docker service

```
[root@worker-node1 ~]# systemctl restart docker && systemctl enable docker
[root@worker-node2 ~]# systemctl restart docker && systemctl enable docker
```

Step 4: Now Join worker nodes to master node

To join worker nodes to Master node, a token is required. Whenever kubernetes master initialized , then in the output we get command and token. Copy that command and run on both nodes.

```
[root@worker-node1 ~]# kubeadm join --token a3bd48.1bc42347c3b35851
192.168.1.30:6443


[root@worker-node2 ~]# kubeadm join --token a3bd48.1bc42347c3b35851
192.168.1.30:6443
```

Now verify Nodes status from master node using `kubectl` command

```
[root@k8s-master ~]# kubectl get nodes
NAME           STATUS    AGE       VERSION
k8s-master     Ready     2h        v1.7.5
worker-node1   Ready     20m       v1.7.5
worker-node2   Ready     18m       v1.7.5
[root@k8s-master ~]#
```

As we can see master and worker nodes are in ready status. This concludes that Kubernetes 1.7 has been installed successfully and also we have successfully joined two worker nodes. Now we can create pods and services.

Ballerina installation steps on Ubuntu

STEP 1: Login into any Linux (Ubuntu or Debina System) and make sure you execute steps using root user.

STEP 2: Download Ballerina Deb package from their website

```
wget
https://product-dist.ballerina.io/downloads/0.983.0/ballerina-linux-ins
taller-x64-0.983.0.deb
```

STEP 3: Install deb package

```
dpkg -i ballerina-linux-installer-x64-0.983.0.deb
```

This installs the Ballerina distribution to `/usr/lib/ballerina`

It completes installation of Ballerina on Linux system.

Serverless Installation Steps

Serverless is a Node.js (`https://nodejs.org/en/`) CLI tool so the first thing you need to do is to install Node.js on your machine and make sure you execute it as root user.

STEP 1: Go to the official Node.js website (`https://nodejs.org/en/`), download and follow the installation instructions (`https://nodejs.org/en/download/`) to install Node.js on your local machine.

Note: Serverless runs on Node v4 or higher:

```
wget https://nodejs.org/dist/v10.13.0/node-v10.13.0-linux-x64.tar.xz
```

STEP 2: Untar and Unzip it on same Linux prompt:

```
tar xf archive.tar.xz
```

STEP 3: Install Serverless:

```
npm install -g serverless
```

It completes serverless installation

Jenkins Installation

To install Jenkins (`https://linuxize.com/post/how-to-install-jenkins-on-centos-7/#installing-jenkins`) on your CentOS system, follow the below mentioned steps and make sure you execute as "root" user:

STEP 1: Jenkins is a Java application, so the first step is to install Java. Run the following command to install the OpenJDK 8 package.

The current version of Jenkins does not support Java 10 (and Java 11) yet. If you have multiple versions of Java installed on your machine make sure Java 8 is the default Java version(`https://linuxize.com/post/how-to-install-jenkins-on-centos-7/#installing-jenkins`).

STEP 2: The next step is to enable the Jenkins repository. To do that, import the GPG key using the following curl command:

```
curl --silent --location
http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo | sudo tee
/etc/yum.repos.d/jenkins.repo
```

STEP 3: And add the repository to your system with:

```
sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
```

STEP 4: Once the repository is enabled, install the latest stable version of Jenkins by typing:

```
sudo yum install jenkins
```

STEP 5: After the installation process is completed, start the Jenkins service with:

```
sudo systemctl start jenkins
```

STEP 6: To check whether it started successfully run:

```
systemctl status jenkins
```

It completes Jenkins installation on Linux system.

Reference the following links:

Serverless: `https://serverless.com/framework/docs/providers/aws/guide/installation/`

Jenkins : `https://serverless.com/framework/docs/providers/aws/guide/installation/`

K8s: `https://www.linuxtechi.com/install-kubernetes-1-7-centos7-rhel7/`

Ballerina: `https://ballerina.io/learn/getting-started/`

Index