Kubernetes installation on Ubuntu

22.04 LTS

AI CLOUD POST

CRI-O: OCI-based
Kubernetes Runtime

Calico as CNI
plugin

Kubernetes Installation
High level steps

AI CLOUD POST

Kubernetes Cluster Setup Using Kubeadm

Following are the high-level steps involved in setting up Kubernetes cluster installation using **kubeadm** tool.

Step 1: Set **hostname** for all the nodes (master, workers)

Step 2: Disable **Swap space** on all the nodes

Step 3: Enable **iptables Bridged** Traffic on all the Nodes

Step 4: Install **Container Runtime, CRI-O** On All the Nodes

Step 5: Install **Kubeadm & Kubelet & Kubectl** on all the Nodes

Step 6: Initialize **Kubeadm init** On Master Node To Setup Control Plane

Step 7: Run **kubeadm Join** on Worker Nodes

Step 8: Install **Calico Network Plugin** for Pod Networking

Step 9: Setup Kubernetes **Metrics Server**

Step 10: Test Your Kubernetes Cluster Installation (health check)

Step 11: How to **add worker node** in to cluster.

P.S: All the steps given in this guide are referred from the official Kubernetes documentation and related official GitHub project pages.

Kubernetes Installation
High level steps

AI CLOUD POST

Kubernetes Cluster Setup Environment details

My Self has provisioned 3 AWS EC2 servers for kubernetes cluster installation,

configuration as below,(configuration taken based on the vCPU count and it should be 2 vCPU's at least otherwise the installation will not proceed further and also it will fail with error message at installation startup stage itself.

3 node cluster configuration

1 master node:

configuration:

vCPU: 2

Mem: 4 GiB

2 worker nodes:

configuration:

vCPU: 2

Mem: 4 GiB

Kubernetes Installation
High level steps

AI CLOUD POST

Step 1: Set hostname for all the nodes (master, workers)

Login to **master node** and set hostname via *hostnamectl* command,

```
$ sudo hostnamectl set-hostname “controller.demo.com"          // on master node
$ exec bash                                                    // run to effect the hostname changes
```

On the worker nodes,

```
$ sudo hostnamectl set-hostname "worker1.demo.com"             // 1st worker node
$ exec bash                                                    // run to effect the hostname changes
```

Add the IP and hostname lines in /etc/hosts file on each node, below are my server entries

```
172.31.60.2      controller.demo.com
172.31.54.218    worker1.demo.com
```

Kubernetes Installation

AI CLOUD POST

Step 2: Disable Swap space on all the nodes

For kubeadm to work properly, you need to disable swap on all the nodes using the following command.

You can do it with the following command. The sudo swapoff -a command temporarily disables swap on your system.

```
$ sudo swapoff –a
```

Then, the sudo sed -i '/ swap / s/^/#/' /etc/fstab command modifies a configuration file to keep the swap remains off even after a system reboot.

```
sudo sed -i '/ swap / s/^/#/' /etc/fstab
```

Make sure to run the above commands on all the cluster nodes.

Kubernetes Installation

Step 3: Enable iptables Bridged Traffic on all the Nodes

Execute the following commands on all the nodes for Enable IPv4 packet forwarding.
Here we are tweaking some kernel parameters and setting them using sysctl.

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# sysctl params required by setup, params persist across reboots
     cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
     net.bridge.bridge-nf-call-iptables  = 1
     net.bridge.bridge-nf-call-ip6tables = 1
     net.ipv4.ip_forward                 = 1
     EOF

# Apply sysctl params without reboot
     sudo sysctl --system
```

The **br_netfilter module** is required to enable transparent masquerading and to facilitate Virtual Extensible LAN (VxLAN) traffic for communication between Kubernetes pods across the cluster nodes.

In **Overlay networking**, only the Kubernetes cluster nodes are assigned IPs from subnets. Pods receive IPs from a private CIDR provided at the time of cluster creation. Each node is assigned a /24 address space carved out from the same CIDR.

Kubernetes Installation

AI CLOUD POST

Step 4: Install Container Runtime On All The Nodes

Note: We are using cri-o instead if containerd because, in Kubernetes certification exams, cri-o is used as the container runtime.

The basic requirement for a Kubernetes cluster is a container runtime. You can have any one of the following container runtimes.

1) CRI-O
2) containerd
3) Docker Engine (using cri-dockerd)

But, we will be using CRI-O instead of Docker for this setup.

Execute the following commands **on all the nodes** to install required dependencies and the latest version of CRIO.

Kubernetes Installation

For deb based distributions, you can run the following commands as a root user:

Install dependencies for adding the repositories

```
sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates
```

Add the CRI-O repository

```
curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key |
    gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" |
    tee /etc/apt/sources.list.d/cri-o.list
```

```
apt-get install -y cri-o
```

```
sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service
```

Install crictl - crictl, a CLI utility to interact with the containers created by the container runtime.

```
VERSION="v1.28.0"
wget https://github.com/kubernetes-sigs/cri-tools/releases/download/$VERSION/crictl-$VERSION-linux-amd64.tar.gz
sudo tar zxvf crictl-$VERSION-linux-amd64.tar.gz -C /usr/local/bin
rm -f crictl-$VERSION-linux-amd64.tar.gz
```

When you use container runtimes other than Docker, you can use the **crictl utility** to debug containers on the nodes.

Kubernetes Installation

Step 5: Install Kubeadm & Kubelet & Kubectl on all Nodes

Download the GPG key for the Kubernetes APT repository **on all the nodes.**

```
KUBERNETES_VERSION=1.28
sudo mkdir -p /etc/apt/keyrings
```

Add the Kubernetes repository

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key |
    gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /" |
    tee /etc/apt/sources.list.d/kubernetes.list
```

Update apt repo

```
sudo apt-get update -y
```

You can use the following commands on master node to find the latest versions.

```
apt-cache madison kubeadm | tac
```

Install the first version in 1.28 so that we can practice cluster upgrade task later.

Install the packages

```
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-get install -y kubelet=1.28.10-1.1 kubectl=1.28.10-1.1 kubeadm=1.28.10-1.1
```

Note: If you are preparing for Kubernetes certification, install the specific version of kubernetes by changing KUBERNETES_VERSION variable. For example, the current Kubernetes version for CKA, CKAD and CKS exams is Kubernetes version 1.29

Kubernetes Installation

Install the first version in 1.29 so that we can practice cluster upgrade task later.

Specify the version as shown below. Here I am using **1.29.0-1.1**

```
sudo apt-get install -y kubelet=1.28 kubectl=1.28 kubeadm=1.28
```

Or, to **install the latest version** from the repo use the following command without specifying any version.

```
sudo apt-get install -y kubelet kubeadm kubectl
```

Add hold to the packages to prevent upgrades

```
sudo apt-mark hold kubelet kubeadm kubectl
```

Kubernetes Installation

AI CLOUD POST

Step 6: Initialize Kubeadm On Master Node To Setup Control Plane

Set the following environment variables. Replace 172.31.60.2 with the IP of your master node.

```
IPADDR="172.31.60.2"
NODENAME=$(hostname -s)
POD_CIDR="192.168.0.0/16"
```

```
sudo kubeadm init --apiserver-advertise-address=$IPADDR  --apiserver-cert-extra-sans=$IPADDR
--pod-network-cidr=$POD_CIDR --node-name $NODENAME --ignore-preflight-errors Swap
```

`--ignore-preflight-errors Swap` is actually not required as we disabled the swap initially.

On a successful kubeadm initialization, you should get an output with kubeconfig file location and the **join command with the token**. Copy that and save it to the file.

we will need it for **joining the worker node to the master.**

Follow the instructions given in the command output to configure master node properly.

Kubernetes Installation
Health check

Now, verify the kubeconfig by executing the following kubectl command to list all the pods in the kube-system namespace.

```
kubectl get po -n kube-system
```

You verify all the cluster component health statuses using the following command.

```
kubectl get --raw='/readyz?verbose'
```

You can get the cluster info using the following command.

```
kubectl cluster-info
```

By default, apps won't get scheduled on the master node. If you want to use the master node for scheduling apps, taint the master node.

```
kubectl taint nodes --all node-role.kubernetes.io/control-plane-
```

Kubernetes Installation
Worker node setup

AI CLOUD POST

Step 7: Join Worker Nodes To Kubernetes Master Node

We have set up **cri-o, kubelet, and kubeadm** utilities on the worker nodes as well. Along with master node preparation.

Now, let's join the worker node to the master node using the Kubeadm join command with which you have got in the output while setting up the master node.

On successful execution, you will see the output saying, "This node has joined the cluster".

Now execute the **kubectl command from the master node** to check if the node is added to the master.

```
kubectl get nodes -o wide
```

Kubernetes Installation

AI CLOUD POST

How Does Kubeadm Work?

When you initialize a Kubernetes cluster using Kubeadm, it does the following.

1) When you initialize kubeadm, first it runs all the **preflight checks** to validate the system state and it downloads all the required cluster container images from the **registry.k8s.io** container registry.

2) It then generates required TLS certificates and stores them in the **/etc/kubernetes/pki** folder.

3) Next, it generates all the kubeconfig files for the cluster components in the **/etc/kubernetes** folder.

4) Then it starts the kubelet service generates the static pod manifests for all the cluster components and saves it in the **/etc/kubernetes/manifests** folder.

5) Next, it starts all the **control plane components** from the **static pod manifests**.

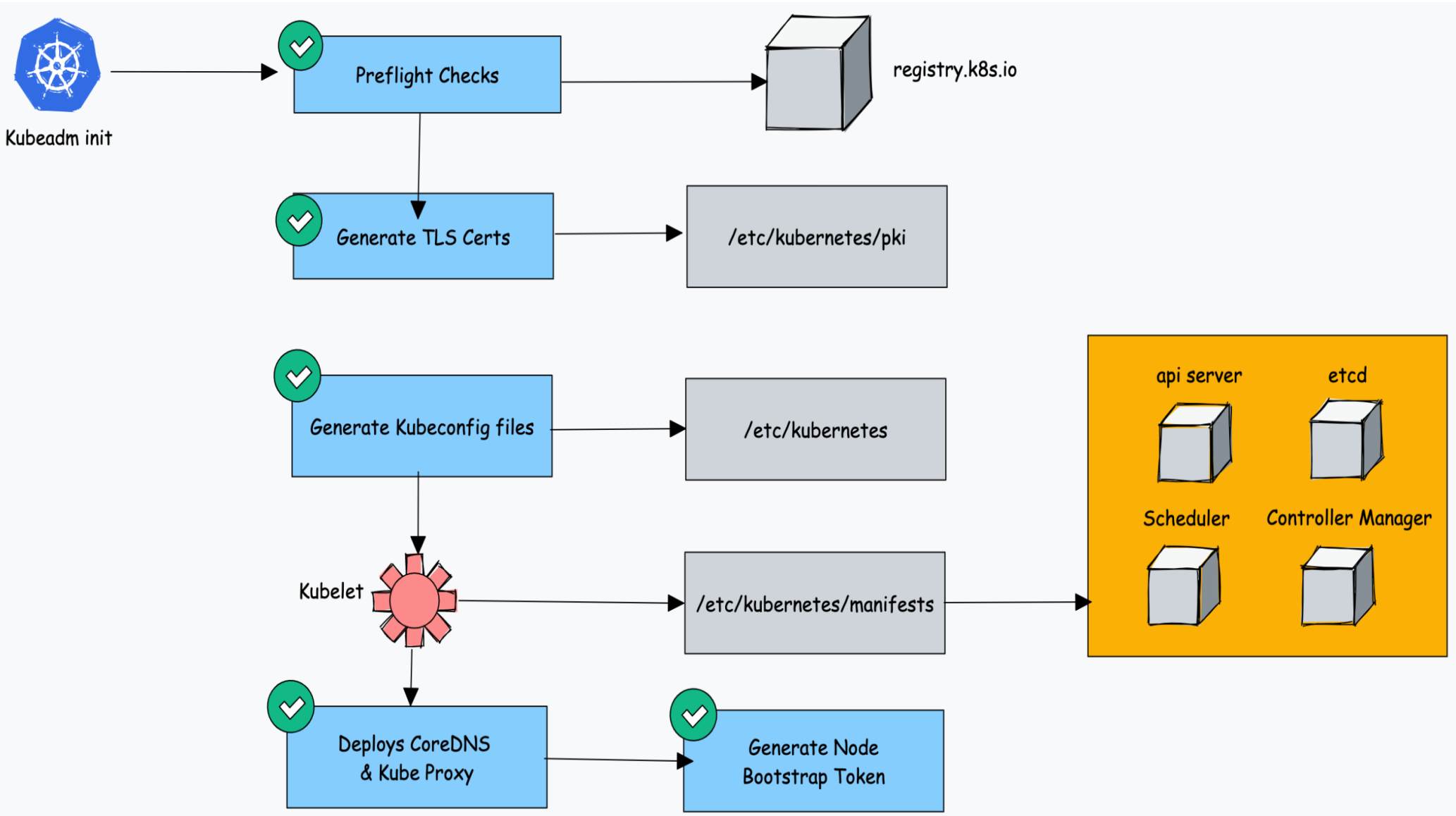
6) Then it installs **core DNS** and **Kubeproxy components**

7) Finally, it generates the **node bootstrap token**.

8) Worker nodes use this token to **join** the control plane.

Kubernetes Installation

How Does Kubeadm Work?



Kubernetes Installation
Pod network setup

AI CLOUD POST

Step 8: Install Calico Network Plugin for Pod Networking

Kubeadm does not configure any network plugin.
You need to install a network plugin of your choice for kubernetes pod networking and enable network policy.

I am using the Calico network plugin for this setup.

Execute the following commands to install the Calico network plugin operator on the cluster.

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

After a couple of minutes, if you check the pods in kube-system namespace, you will see calico pods and running CoreDNS pods.

```
kubectl get po -n kube-system
```

Kubernetes Installation
Metric server setup

AI CLOUD POST

Step 9: Setup Kubernetes Metrics Server

Kubeadm doesn't install metrics server component during its initialization. We have to install it separately.

To verify this, if you run the top command, you will see the Metrics API not available error.

```
root@controller:~# kubectl top nodes
 error: Metrics API not available
```

To install the metrics server, execute the following metric server manifest file. It deploys metrics server version v0.6.2

```
kubectl apply -f https://raw.githubusercontent.com/aicloudpost/k8s-metric-server/main/metric-server.yaml
```

This manifest is taken from the official metrics server repo. I have added the `--kubelet-insecure-tls` flag to the container to make it work in the local setup and hosted it separately. Or else, you will get the following error.
because it doesn't contain any IP SANs" node=""

After couple of minutes, you should be able to view the node metrics as shown below.

```
root@controlplane:~# kubectl top nodes

NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
controlplane   142m         7%     1317Mi          34%
node01         36m          1%     915Mi           23%
```

Kubernetes Installation
Testing final setup

AI CLOUD POST

Step 10: Test Your Kubernetes Cluster Installation

To test Kubernetes installation, let's try to deploy nginx based application and try to access it.

```
$ kubectl create deployment nginx-app --image=nginx --replicas=2
```

Check the status of nginx-app deployment

```
$ kubectl get deployment nginx-app
```

Expose the deployment as NodePort,

```
$ kubectl expose deployment nginx-app --type=NodePort --port=80
service/nginx-app exposed
```

Run following commands to view service status

```
$ kubectl get svc nginx-app
$ kubectl describe svc nginx-app
```

AI CLOUD POST