Setup Kubernetes with kubeadm on AWS

Table of Content:

Open AWS Console

We will create 3 EC2 `(1 for master node)`and `(2 for worker node)`

- Specs for master node:
  - `Instance Type:` t3.medium
  - `OS:` ubuntu 22.04
  - `Storage:` 20 GB (gp2)

- Specs for worker nodes:
  - `Instance Type:` t3.medium
  - `OS:` ubuntu 22.04
  - `Storage:` 10 GB Storage, but recommend 20G

Install Docker on all Machine

Update your existing packages:

```
sudo apt update
```

Install a prerequisite package that allows apt to utilize HTTPS:

```
sudo apt-get install apt-transport-https ca-certificates curl gpg

sudo install -m 0755 -d /etc/apt/keyrings
```

Add GPG key for the official Docker repo to the Ubuntu system:

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Add the Docker repo to APT sources:

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Update the database with the Docker packages from the added repo:

```
sudo apt-get update
```

Install Docker software:

```
sudo apt install -y containerd.io docker-ce docker-ce-cli
```

Docker should now be installed, the daemon started, and the process enabled to start on boot. To verify:

```
sudo systemctl status docker
```

Make the docker enable to start automatic when reboot the machine:

```
sudo systemctl enable docker

sudo systemctl daemon-reload

sudo systemctl enable docker

sudo systemctl enable --now containerd
```

Add user to docker **Groups**:

```
sudo usermod -aG docker ${USER}
```

In AWS ec2 the user would be ubuntu

```
sudo usermod -aG docker ubuntu
```

Install CNI Plugin

```
wget
https://github.com/containernetworking/plugins/releases/download/v1.4.0/cni
-plugins-linux-amd64-v1.4.0.tgz

sudo mkdir -p /opt/cni/bin

sudo tar Cxzvf /opt/cni/bin cni-plugins-linux-amd64-v1.4.0.tgz
```

Modify containerd Configuration for systemd Support

```
sudo mkdir -p /etc/containerd

sudo containerd config default | tee /etc/containerd/config.toml

sudo vim /etc/containerd/config.toml
```

Paste the configuration in the file and save it.

```
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
version = 2

[cgroup]
  path = ""

[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0

[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
```

```
    max_recv_message_size = 16777216
    max_send_message_size = 16777216
    tcp_address = ""
    tcp_tls_cert = ""
    tcp_tls_key = ""
    uid = 0

  [metrics]
    address = ""
    grpc_histogram = false

  [plugins]

    [plugins."io.containerd.gc.v1.scheduler"]
      deletion_threshold = 0
      mutation_threshold = 100
      pause_threshold = 0.02
      schedule_delay = "0s"
      startup_delay = "100ms"

    [plugins."io.containerd.grpc.v1.cri"]
      disable_apparmor = false
      disable_cgroup = false
      disable_hugetlb_controller = true
      disable_proc_mount = false
      disable_tcp_service = true
      enable_selinux = false
      enable_tls_streaming = false
      ignore_image_defined_volumes = false
      max_concurrent_downloads = 3
      max_container_log_line_size = 16384
      netns_mounts_under_state_dir = false
      restrict_oom_score_adj = false
      sandbox_image = "k8s.gcr.io/pause:3.5"
      selinux_category_range = 1024
      stats_collect_period = 10
      stream_idle_timeout = "4h0m0s"
      stream_server_address = "127.0.0.1"
      stream_server_port = "0"
      systemd_cgroup = false
      tolerate_missing_hugetlb_controller = true
      unset_seccomp_profile = ""

      [plugins."io.containerd.grpc.v1.cri".cni]
        bin_dir = "/opt/cni/bin"
        conf_dir = "/etc/cni/net.d"
        conf_template = ""
        max_conf_num = 1

      [plugins."io.containerd.grpc.v1.cri".containerd]
        default_runtime_name = "runc"
        disable_snapshot_annotations = true
        discard_unpacked_layers = false
        no_pivot = false
```

```
      snapshotter = "overlayfs"

      [plugins."io.containerd.grpc.v1.cri".containerd.default_runtime]
        base_runtime_spec = ""
        container_annotations = []
        pod_annotations = []
        privileged_without_host_devices = false
        runtime_engine = ""
        runtime_root = ""
        runtime_type = ""


[plugins."io.containerd.grpc.v1.cri".containerd.default_runtime.options]

      [plugins."io.containerd.grpc.v1.cri".containerd.runtimes]

        [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]
          base_runtime_spec = ""
          container_annotations = []
          pod_annotations = []
          privileged_without_host_devices = false
          runtime_engine = ""
          runtime_root = ""
          runtime_type = "io.containerd.runc.v2"


[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
            BinaryName = ""
            CriuImagePath = ""
            CriuPath = ""
            CriuWorkPath = ""
            IoGid = 0
            IoUid = 0
            NoNewKeyring = false
            NoPivotRoot = false
            Root = ""
            ShimCgroup = ""
            SystemdCgroup = true


[plugins."io.containerd.grpc.v1.cri".containerd.untrusted_workload_runtime]
        base_runtime_spec = ""
        container_annotations = []
        pod_annotations = []
        privileged_without_host_devices = false
        runtime_engine = ""
        runtime_root = ""
        runtime_type = ""


[plugins."io.containerd.grpc.v1.cri".containerd.untrusted_workload_runtime.
options]

    [plugins."io.containerd.grpc.v1.cri".image_decryption]
```

```
    key_model = "node"

  [plugins."io.containerd.grpc.v1.cri".registry]
    config_path = ""

    [plugins."io.containerd.grpc.v1.cri".registry.auths]

    [plugins."io.containerd.grpc.v1.cri".registry.configs]

    [plugins."io.containerd.grpc.v1.cri".registry.headers]

    [plugins."io.containerd.grpc.v1.cri".registry.mirrors]

  [plugins."io.containerd.grpc.v1.cri".x509_key_pair_streaming]
    tls_cert_file = ""
    tls_key_file = ""

[plugins."io.containerd.internal.v1.opt"]
  path = "/opt/containerd"

[plugins."io.containerd.internal.v1.restart"]
  interval = "10s"

[plugins."io.containerd.metadata.v1.bolt"]
  content_sharing_policy = "shared"

[plugins."io.containerd.monitor.v1.cgroups"]
  no_prometheus = false

[plugins."io.containerd.runtime.v1.linux"]
  no_shim = false
  runtime = "runc"
  runtime_root = ""
  shim = "containerd-shim"
  shim_debug = false

[plugins."io.containerd.runtime.v2.task"]
  platforms = ["linux/amd64"]

[plugins."io.containerd.service.v1.diff-service"]
  default = ["walking"]

[plugins."io.containerd.snapshotter.v1.aufs"]
  root_path = ""

[plugins."io.containerd.snapshotter.v1.btrfs"]
  root_path = ""

[plugins."io.containerd.snapshotter.v1.devmapper"]
  async_remove = false
  base_image_size = ""
  pool_name = ""
  root_path = ""
```

```
    [plugins."io.containerd.snapshotter.v1.native"]
      root_path = ""

    [plugins."io.containerd.snapshotter.v1.overlayfs"]
      root_path = ""

    [plugins."io.containerd.snapshotter.v1.zfs"]
      root_path = ""

  [proxy_plugins]

  [stream_processors]

    [stream_processors."io.containerd.ocicrypt.decoder.v1.tar"]
      accepts = ["application/vnd.oci.image.layer.v1.tar+encrypted"]
      args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
      env =
["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_keyprovider
.conf"]
      path = "ctd-decoder"
      returns = "application/vnd.oci.image.layer.v1.tar"

    [stream_processors."io.containerd.ocicrypt.decoder.v1.tar.gzip"]
      accepts = ["application/vnd.oci.image.layer.v1.tar+gzip+encrypted"]
      args = ["--decryption-keys-path", "/etc/containerd/ocicrypt/keys"]
      env =
["OCICRYPT_KEYPROVIDER_CONFIG=/etc/containerd/ocicrypt/ocicrypt_keyprovider
.conf"]
      path = "ctd-decoder"
      returns = "application/vnd.oci.image.layer.v1.tar+gzip"

  [timeouts]
    "io.containerd.timeout.shim.cleanup" = "5s"
    "io.containerd.timeout.shim.load" = "5s"
    "io.containerd.timeout.shim.shutdown" = "3s"
    "io.containerd.timeout.task.state" = "2s"

  [ttrpc]
    address = ""
    gid = 0
    uid = 0
```

```
sudo systemctl restart containerd
sudo systemctl enable containerd
```

Disable swap memory (if running) on both the nodes and master:

```
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

```
sudo swapoff -a
```

Install selinux (in all Machine):

```
sudo apt install selinux-utils
```

Disable selinux (in all Machine):

```
setenforce 0
```

Enable IP forwarding temporarily:

```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

Enable IP forwarding permanently:

```
sudo sh -c "echo 'net.ipv4.ip_forward = 1' >> /etc/sysctl.conf"
```

Apply the changes:

```
sudo sysctl -p
```

Validate Containerd

```
sudo crictl info
```

Validate Containerd

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

Give a unique hostname for all machines:

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

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

```
sudo hostnamectl set-hostname node2
```

> **Restart All Machines.**

Configure Hostname

Set the hostname for all machines

```
sudo vim /etc/hosts
```

We will add:

```
<ip-master>   <hostname-master>
<ip-node1>    <hostname-node1>
<ip-node2>    <hostname-node2>
```

Check swap config, ensure swap is 0

```
free -m
```

Install Kubernetes on All Machines

Update your existing packages:

```
sudo apt-get update
```

Install packages needed to use the Kubernetes apt repository:

```
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
```

Download the public signing key for the Kubernetes package repositories.

```
sudo mkdir -p -m 755 /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo
gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

Add Kubernetes Repository:

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

Update your existing packages:

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

Install Kubeadm:

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

sudo apt-mark hold kubelet kubeadm kubectl
```

Enable the kubelet service:

```
sudo systemctl enable --now kubelet
```

Enable kernel modules

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

Update Iptables Settings

```
sudo tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

Configure persistent loading of modules

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

Reload sysctl

```
sudo sysctl --system
```

Start and enable Services

```
sudo systemctl daemon-reload

sudo systemctl restart docker

sudo systemctl enable docker

sudo systemctl enable kubelet
```

Initialize Kubernetes on the Master Node

Run the following command as sudo on the master node:

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

To start using your cluster, you need to run the following as a regular user:

```
mkdir -p $HOME/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

> In the output, you can see the `kubeadm join` command and a unique token that you will run on the worker node and all other worker nodes that you want to join onto this cluster. Next, copy-paste this command as you will use it later in the worker node.

To Create a new token as root

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

Deploy a Pod Network through the master node:

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

Joining Worker Nodes to the Kubernetes Cluster

You will use your kubeadm join command that was shown in your terminal when we initialized the master node.

The command would be similar of this:

```
sudo kubeadm join 172.31.6.233:6443 --token 9lspjd.t93etsdpwm9gyfib --
discovery-token-ca-cert-hash
sha256:37e35d7ea83599356de1fc5c80c282285cc3c749443a1dafd8e73f40
```

Reset Kubeadm

```
sudo kubeadm reset -f
```