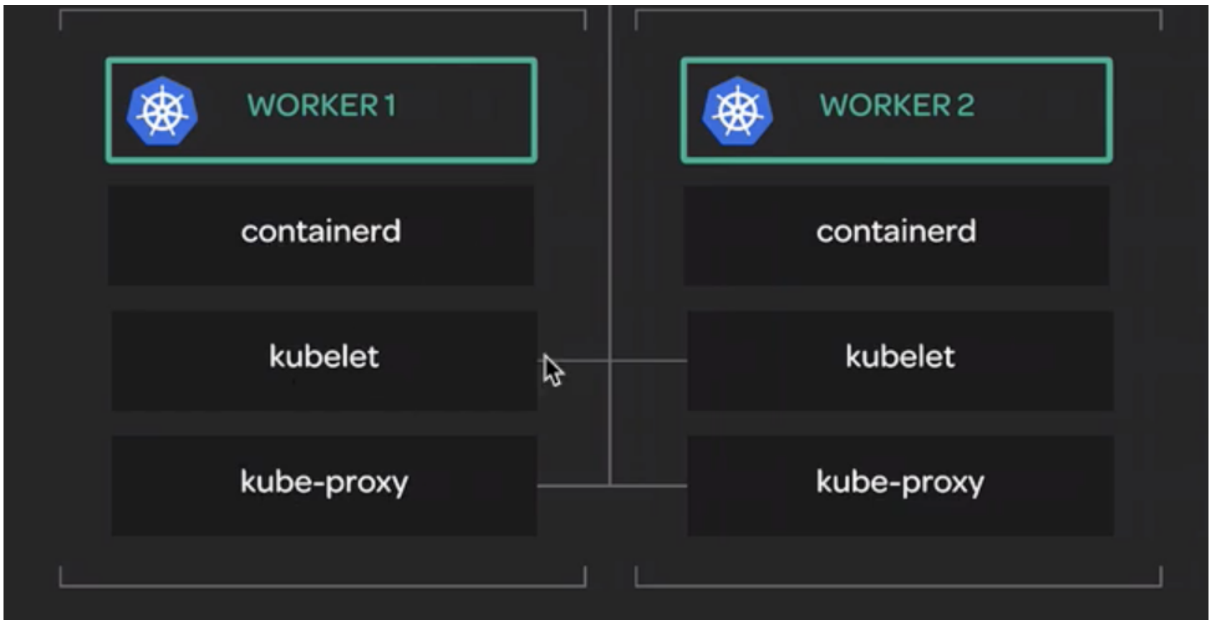
Cloud Provider:AWS
Server:Ubuntu 18.04 x6
Server List: 2 Controller(Master), 2(Worker), 1 Nginx(Reverse Proxy), 1 Ubuntu 18.04(Local)

Worker Node Architecture Overview



Installing Worker Node Binaries

// We are now ready to begin the process of setting up our worker nodes. The first step is to download and install the binary file which we will later use to configure our worker nodes services. In this lesson, we will be downloading and installing the binaries for containerd, kubectl, kubelet, and kube-proxy, as well as other software that they depend on. After completing this lesson, you should have these binaries downloaded and all of the files moved into the correct locations in preparation for configuring the worker node services.

// You can install the worker binaries like so. Run these commands on both worker nodes:

```
$ sudo apt-get -y install socat conntrack ipset

$ wget -q --show-progress --https-only --timestamping \

https://github.com/kubernetes-incubator/cri-tools/releases/download/v1.0.0-beta.0/crictl-v1.0.0-beta.0-linux-amd64.tar.gz \
  https://storage.googleapis.com/kubernetes-the-hard-way/runsc \
  https://github.com/opencontainers/runc/releases/download/v1.0.0-rc5/runc.amd64 \

https://github.com/containernetworking/plugins/releases/download/v0.6.0/cni-plugins-amd64-v0.6.0.tgz \
https://github.com/containerd/containerd/releases/download/v1.1.0/containerd-1.1.0.linux-am
d64.tar.gz \
  https://storage.googleapis.com/kubernetes-release/release/v1.10.2/bin/linux/amd64/kubectl
\

https://storage.googleapis.com/kubernetes-release/release/v1.10.2/bin/linux/amd64/kube-pr
oxy \
  https://storage.googleapis.com/kubernetes-release/release/v1.10.2/bin/linux/amd64/kubelet

$ sudo mkdir -p \
  /etc/cni/net.d \
  /opt/cni/bin \
  /var/lib/kubelet \
  /var/lib/kube-proxy \
  /var/lib/kubernetes \
  /var/run/kubernetes

$ chmod +x kubectl kube-proxy kubelet runc.amd64 runsc
$ sudo mv runc.amd64 runc
$ sudo mv kubectl kube-proxy kubelet runc runsc /usr/local/bin/
$ sudo tar -xvf crictl-v1.0.0-beta.0-linux-amd64.tar.gz -C /usr/local/bin/
$ sudo tar -xvf cni-plugins-amd64-v0.6.0.tgz -C /opt/cni/bin/
$ sudo tar -xvf containerd-1.1.0.linux-amd64.tar.gz -C /
```

Configuring Containerd

// Containerd is the container runtime used to run containers managed by Kubernetes in this course. In this lesson, we will configure a systemd service for containerd on both of our worker node servers. This containerd service will be used to run containerd as a component of each worker node. After completing this lesson, you should have a containerd configured to run as a systemd service on both workers.

// You can configure the containerd service like so. Run these commands on both worker nodes:

```
$ sudo mkdir -p /etc/containerd/
```

// Create the containerd config.toml:

```
$ cat << EOF | sudo tee /etc/containerd/config.toml
[plugins]
  [plugins.cri.containerd]
    snapshotter = "overlayfs"
    [plugins.cri.containerd.default_runtime]
      runtime_type = "io.containerd.runtime.v1.linux"
      runtime_engine = "/usr/local/bin/runc"
      runtime_root = ""
    [plugins.cri.containerd.untrusted_workload_runtime]
      runtime_type = "io.containerd.runtime.v1.linux"
      runtime_engine = "/usr/local/bin/runsc"
      runtime_root = "/run/containerd/runsc"
EOF
```

// Create the containerd unit file:

```
$ cat << EOF | sudo tee /etc/systemd/system/containerd.service
[Unit]
Description=containerd container runtime
Documentation=https://containerd.io
After=network.target

[Service]
ExecStartPre=/sbin/modprobe overlay
ExecStart=/bin/containerd
Restart=always
RestartSec=5
Delegate=yes
KillMode=process
OOMScoreAdjust=-999
LimitNOFILE=1048576
LimitNPROC=infinity
LimitCORE=infinity

[Install]
WantedBy=multi-user.target
EOF
```

Configuring Kubelet

// Kubelet is the Kubernetes agent which runs on each worker node. Acting as a middleman between the Kubernetes control plane and the underlying container runtime, it coordinates the running of containers on the worker node. In this lesson, we will configure our systemd service for kubelet. After completing this lesson, you should have a systemd service configured and ready to run on each worker node.

// You can configure the kubelet service like so. Run these commands on both worker nodes.

// Set a HOSTNAME environment variable that will be used to generate your config files. Make sure you set the HOSTNAME appropriately for each worker node:

```
$ HOSTNAME=$(echo ‘public-ipv4 dns addr’)
```

```
$ sudo mv ${HOSTNAME}-key.pem ${HOSTNAME}.pem /var/lib/kubelet/
$ sudo mv ${HOSTNAME}.kubeconfig /var/lib/kubelet/kubeconfig

$ sudo mv ca.pem /var/lib/kubernetes/
```

// Create the kubelet config file:

```
$ cat << EOF | sudo tee /var/lib/kubelet/kubelet-config.yaml
kind: KubeletConfiguration
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
  webhook:
    enabled: true
  x509:
    clientCAFile: "/var/lib/kubernetes/ca.pem"
authorization:
  mode: Webhook
clusterDomain: "cluster.local"
clusterDNS:
  - "10.32.0.10"
runtimeRequestTimeout: "15m"
tlsCertFile: "/var/lib/kubelet/${HOSTNAME}.pem"
tlsPrivateKeyFile: "/var/lib/kubelet/${HOSTNAME}-key.pem"
EOF
```

// Create the kubelet unit file:

```
$ cat << EOF | sudo tee /etc/systemd/system/kubelet.service
[Unit]
Description=Kubernetes Kubelet
Documentation=https://github.com/kubernetes/kubernetes
After=containerd.service
Requires=containerd.service

[Service]
ExecStart=/usr/local/bin/kubelet \\
  --config=/var/lib/kubelet/kubelet-config.yaml \\
  --container-runtime=remote \\
  --container-runtime-endpoint=unix:///var/run/containerd/containerd.sock \\
  --image-pull-progress-deadline=2m \\
  --kubeconfig=/var/lib/kubelet/kubeconfig \\
  --network-plugin=cni \\
  --register-node=true \\
  --v=2 \\
  --hostname-override=${HOSTNAME} \\
  --allow-privileged=true
Restart=on-failure
```

```
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF
```

Configuring Kube-Proxy

// Kube-proxy is an important component of each Kubernetes worker node. It is responsible for providing network routing to support Kubernetes networking components. In this lesson, we will configure our kube-proxy systemd service. Since this is the last of the three worker node services that we need to configure, we will also go ahead and start all of our worker node services once we're done. Finally, we will complete some steps to verify that our cluster is set up properly and functioning as expected so far. After completing this lesson, you should have two Kubernetes worker nodes up and running, and they should be able to successfully register themselves with the cluster.

// You can configure the kube-proxy service like so. Run these commands on both worker nodes:

```
$ sudo mv kube-proxy.kubeconfig /var/lib/kube-proxy/kubeconfig
```

// Create the kube-proxy config file:

```
$ cat << EOF | sudo tee /var/lib/kube-proxy/kube-proxy-config.yaml
kind: KubeProxyConfiguration
apiVersion: kubeproxy.config.k8s.io/v1alpha1
clientConnection:
  kubeconfig: "/var/lib/kube-proxy/kubeconfig"
mode: "iptables"
clusterCIDR: "10.200.0.0/16"
EOF
```

// Create the kube-proxy unit file:

```
$ cat << EOF | sudo tee /etc/systemd/system/kube-proxy.service
[Unit]
Description=Kubernetes Kube Proxy
Documentation=https://github.com/kubernetes/kubernetes

[Service]
ExecStart=/usr/local/bin/kube-proxy \\
  --config=/var/lib/kube-proxy/kube-proxy-config.yaml
Restart=on-failure
RestartSec=5

[Install]
```

```
WantedBy=multi-user.target
EOF
```

// Now you are ready to start up the worker node services! Run these:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable containerd kubelet kube-proxy
$ sudo systemctl start containerd kubelet kube-proxy
```

// Check the status of each service to make sure they are all active (running) on both worker nodes:

```
$ sudo systemctl status containerd kubelet kube-proxy
```

// Finally, verify that both workers have registered themselves with the cluster. Log in to one of your control nodes and run this:

```
$ kubectl get nodes
```

// You should see the hostnames for both worker nodes listed. Note that it is expected for them to be in the NotReady state at this point.