Step-by-Step Guide to Configuring a Kubernetes v1.31 Cluster on AWS EC2

By
Mahendran Selvakumar
https://devopstronaut.com/

https://www.linkedin.com/in/mahendran-selvakumar-36444a77/

What is Kubernetes?

Kubernetes, often abbreviated as **K8s**, is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. Originally developed by Google, Kubernetes is now maintained by the Cloud Native Computing Foundation (CNCF).

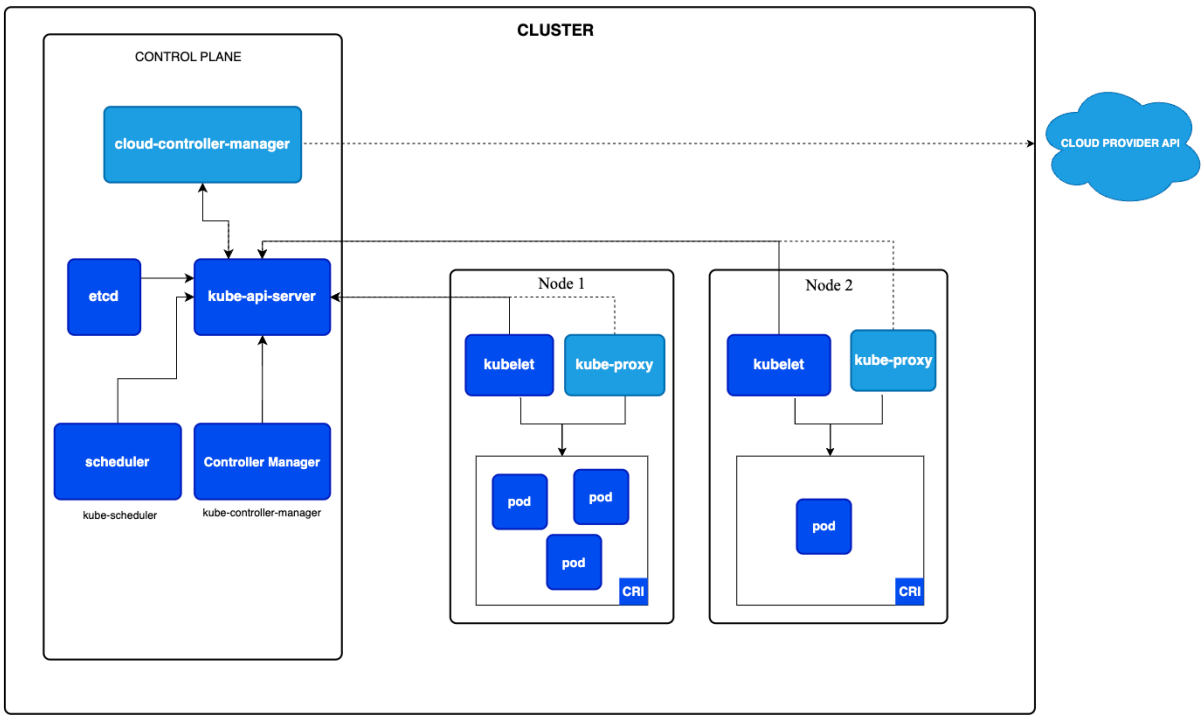
Key Features of Kubernetes:

1. **Automated Deployment and Scaling:**
   - Kubernetes automates the deployment of applications in containers and can dynamically scale the number of containers based on resource usage or demand.
2. **Self-Healing:**
   - It automatically monitors the health of containers and replaces or reschedules them if they fail, ensuring high availability and reliability.
3. **Load Balancing and Service Discovery:**
   - Kubernetes provides built-in load balancing to distribute traffic among containers, and it automatically discovers services to help applications communicate with each other.
4. **Storage Orchestration:**
   - It allows you to automatically mount storage systems such as local storage, public cloud providers, or network storage solutions.
5. **Declarative Configuration:**
   - Users can define the desired state of their applications using YAML or JSON configuration files, and Kubernetes will work to maintain that state.
6. **Extensibility:**
   - Kubernetes supports various extensions and plugins, enabling users to customize the platform according to their needs, including network policies, storage configurations, and application runtimes.
7. **Multi-Cloud and Hybrid Deployments:**
   - Kubernetes can run on various environments, including on-premises servers, public clouds, or hybrid setups, making it versatile for different deployment strategies.

Benefits of Using Kubernetes:

- **Portability:** Kubernetes allows you to run applications consistently across different environments, whether on-premises or in the cloud.
- **Resource Efficiency:** By effectively managing containerized workloads, Kubernetes optimizes resource usage, resulting in lower costs.
- **Enhanced Development Speed:** Developers can quickly deploy and scale applications, enabling faster innovation and iteration.
- **Robust Ecosystem:** Kubernetes has a large and active community that provides numerous tools, libraries, and integrations, enhancing its functionality and usability.

Kubernetes Cluster Architecture:



1. Control Plane

The **Control Plane** serves as the control plane of the Kubernetes cluster, responsible for managing the cluster's overall state and operations. Its key components include:

- **API Server:**
  - Acts as the front-end for the Kubernetes control plane, exposing the Kubernetes API and handling REST commands from clients (e.g., kubectl).
- **etcd:**
  - A distributed key-value store that holds the cluster's configuration data and state information, ensuring data consistency and reliability.
- **Scheduler:**
  - Responsible for assigning pods to worker nodes based on available resources and scheduling policies.
- **Controller Manager:**
  - Monitors the cluster and manages the different controllers that regulate the state of the cluster, ensuring that the current state matches the desired state.
- **Cloud Controller Manager (optional):**
  - Integrates with cloud provider APIs to manage resources specific to cloud environments, enabling features such as dynamic provisioning of cloud resources.

2. Worker Node

Worker Nodes are the machines where application workloads run. They host the necessary services to execute and manage pods. The key components include:

- **kubelet:**
  - An agent that runs on each worker node, ensuring that containers are running as specified by the API server and reporting the node's status back to the control plane.
- **Container Runtime:**
  - The software responsible for running containers on the worker node. Examples include Docker, containerd, and CRI-O.
- **kube-proxy:**
  - Manages network routing and load balancing for services and pods, ensuring that traffic is directed to the appropriate containers based on service definitions.
- **Pods:**
  - The smallest deployable units in Kubernetes, which can contain one or more containers, along with shared storage and network resources.

Minimum Requirements for Kubernetes Installation

To successfully install and run Kubernetes, you need to meet the following minimum requirements:

1. Hardware Requirements

- **Master Node:**
  - **CPU:** 2 or more cores
  - **Memory:** 2 GB RAM (recommended: 4 GB or more)
  - **Disk:** 20 GB of available disk space (SSD preferred)
- **Worker Nodes:**
  - **CPU:** 1 or more cores
  - **Memory:** 1 GB RAM (recommended: 2 GB or more)
  - **Disk:** 20 GB of available disk space (SSD preferred)

2. Operating System Requirements

- Supported Linux distributions include:
  - Ubuntu (18.04, 20.04, 22.04 LTS)
  - CentOS (7 and 8)
  - Debian (9 and 10)
  - RHEL (7 and 8)
  - Fedora (latest stable version)
- **Note:** Kubernetes may not work on older operating systems or other non-Linux operating systems.

3. Network Requirements

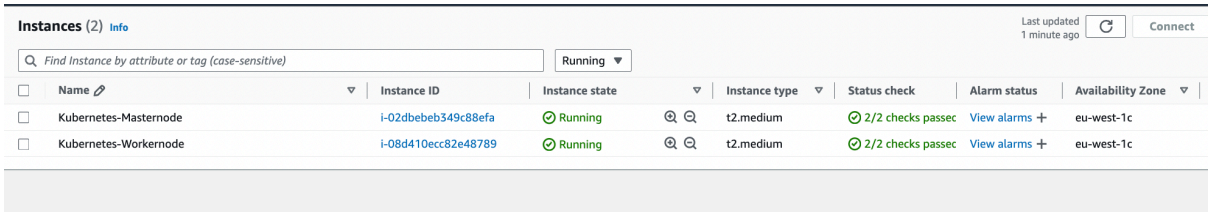
- A reliable network connection between all nodes (master and worker nodes).
- The ability to reach the Kubernetes API server from all nodes.
- Ensure that the required ports are open for communication (e.g., 6443 for API server, 10250 for Kubelet, etc.).

4. Software Requirements

- **Container Runtime:** Docker, containerd, or CRI-O (Docker is commonly used).
- **Kubeadm, Kubelet, and Kubectl:** Must be installed on all nodes.
- **Swap Memory:** Must be disabled. Kubernetes requires that swap memory is turned off.

Step1: Create EC2 instances for Master and Worker Nodes

Create one Master and one Worker node EC2 instance of type t2.medium, using Ubuntu 22.04 LTS as the operating system



Modify the hostname using the command **hostnamectl set-hostname <hostname of the Instance>**

```
ubuntu@ip-172-31-38-92:~$ sudo su -
root@ip-172-31-38-92:~# hostnamectl set-hostname workernode2
root@ip-172-31-38-92:~#
```

I have modified the hostname for all instances

Step2: Install Docker engine on Master and Worker Nodes

Install docker using apt package: Add docker's official GPG key
Update Ubuntu package lists

```
root@masternode:~# sudo apt-get update
Hit:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2066 kB]
Get:12 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [358 kB]
Get:13 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [17.9 kB]
Get:14 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1130 kB]
Get:15 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [264 kB]
Get:16 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [26.3 kB]
Get:17 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [43.3 kB]
Get:18 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [10.8 kB]
Get:19 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [444 B]
Get:20 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [67.7 kB]
Get:21 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [11.1 kB]
Get:22 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [388 B]
Get:23 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:24 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [28.8 kB]
Get:25 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.5 kB]
Get:26 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [672 B]
Get:27 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:28 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1848 kB]
Get:29 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [299 kB]
Get:30 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [13.3 kB]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [909 kB]
Get:32 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [179 kB]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [19.4 kB]
Get:34 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [37.2 kB]
Get:35 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [7588 B]
Get:36 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [228 B]
Fetched 28.1 MB in 4s (7626 kB/s)
Reading package lists... Done
root@masternode:~#
```

Install the **ca-certificates** and **curl** packages

```
root@masternode:~# sudo apt-get install ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203~22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.18).
curl set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
root@masternode:~#
```

Create a directory with the necessary permissions to hold Docker's keyring, download Docker's GPG key for package verification, and save it in the newly created directory

```
root@masternode:~# sudo install -m 0755 -d /etc/apt/keyrings
root@masternode:~# sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
root@masternode:~#
```

Add the Docker repository to the apt sources

```
root@masternode:~# 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
root@masternode:~#
```

- **dpkg** --print-architecture ensures you get the correct architecture (like amd64).
- **signed-by=/etc/apt/keyrings/docker.asc** ensures the repository is verified with Docker's GPG key.
- **$(. /etc/os-release && echo "$VERSION_CODENAME")** pulls your Ubuntu version

Update the package again using the command: **sudo apt-get update**

```
root@masternode:~# sudo apt-get update
Hit:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:6 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [40.7 kB]
Fetched 89.5 kB in 1s (167 kB/s)
Reading package lists... Done
root@masternode:~#
```

Install the latest version of Docker

```
root@masternode:~# sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 5 not upgraded.
Need to get 123 MB of archives.
After this operation, 442 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libltdl7 amd64 2.4.6-15build2 [39.6 kB]
Get:3 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libslirp0 amd64 4.6.1-1build1 [61.5 kB]
Get:4 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 slirp4netns amd64 1.0.1-2 [28.2 kB]
Get:5 https://download.docker.com/linux/ubuntu jammy/stable amd64 containerd.io amd64 1.7.22-1 [29.5 MB]
Get:6 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-buildx-plugin amd64 0.17.1-1~ubuntu.22.04~jammy [30.3 MB]
Get:7 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-cli amd64 5:27.3.1-1~ubuntu.22.04~jammy [15.0 MB]
Get:8 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce amd64 5:27.3.1-1~ubuntu.22.04~jammy [25.6 MB]
Get:9 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-rootless-extras amd64 5:27.3.1-1~ubuntu.22.04~jammy [9589 kB]
Get:10 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-compose-plugin amd64 2.29.7-1~ubuntu.22.04~jammy [12.7 MB]
Fetched 123 MB in 2s (77.8 MB/s)
```

This command installs the complete set of tools needed to run Docker, manage containers, and build Docker images, ensuring a fully functional Docker environment on your system.

Verify the Docker version using the command: **docker –version**

```
root@masternode:~# docker --version
Docker version 27.3.1, build ce12230
root@masternode:~#
```

Repeat the same steps on the worker nodes as well

Step3: Install Kubeadm, Kubelet and Kubectl on Master and Worker nodes

Swap is a portion of the hard drive used as virtual memory when the system's RAM is full. Running the swapoff command disables this virtual memory, forcing the system to rely solely on physical RAM. Kubernetes has known performance and stability issues when swap is enabled

Run **sudo swapoff -a** to disable all active swap partitions or swap files on your system

```
ubuntu@masternode:~$ sudo su -
root@masternode:~# sudo swapoff -a
root@masternode:~#
```

Update the APT package list using command **sudo apt-get update**

```
root@masternode:~# sudo apt-get update
Hit:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu jammy InRelease
Reading package lists... Done
root@masternode:~#
```

Install the packages needed to use the Kubernetes APT repository using this command **sudo apt-get install -y apt-transport-https ca-certificates curl gpg**

```
root@masternode:~# sudo apt-get install -y apt-transport-https ca-certificates curl gpg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203~22.04.1).
curl is already the newest version (7.81.0-1ubuntu1.18).
gpg is already the newest version (2.2.27-3ubuntu2.1).
gpg set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 1510 B of archives.
After this operation, 170 kB of additional disk space will be used.
Get:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-transport-https all 2.4.13 [1510 B]
Fetched 1510 B in 0s (83.7 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 66050 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.4.13_all.deb ...
Unpacking apt-transport-https (2.4.13) ...
Setting up apt-transport-https (2.4.13) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@masternode:~#
```

Download the public signing key for the Kubernetes package repositories. Add the Kubernetes package repository key to your system, which allows you to securely install Kubernetes packages via APT using command **curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg**

```
root@masternode:~# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
root@masternode:~#
root@masternode:~#
root@masternode:~#
```

Add the Kubernetes stable package repository for version v1.31 to your system's APT sources list to ensure you can install and manage Kubernetes packages using command **echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list**

```
root@masternode:~# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
root@masternode:~#
```

Update the apt package again

```
root@masternode:~# sudo apt-get update
Hit:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:3 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:5 https://download.docker.com/linux/ubuntu jammy InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  InRelease [1186 B]
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  Packages [4865 B]
Fetched 263 kB in 1s (476 kB/s)
Reading package lists... Done
root@masternode:~#
```

Installs the core Kubernetes components on your system using command **sudo apt-get install -y kubelet kubeadm kubectl**

```
root@masternode:~# sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 5 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 conntrack amd64 1:1.4.6-2build2 [33.5 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb  kubelet 1.31.1-1.1 [15.2 MB]
Fetched 87.4 MB in 1s (78.4 MB/s)
Selecting previously unselected package conntrack.
(Reading database ... 66054 files and directories currently installed.)
Preparing to unpack .../0-conntrack_1%3a1.4.6-2build2_amd64.deb ...
Unpacking conntrack (1:1.4.6-2build2) ...
Selecting previously unselected package cri-tools.
Preparing to unpack .../1-cri-tools_1.31.1-1.1_amd64.deb ...
Unpacking cri-tools (1.31.1-1.1) ...
```

Run this command **sudo apt-mark hold kubelet kubeadm kubectl** to prevent these specific Kubernetes packages from being automatically upgraded during regular system updates

```
root@masternode:~# sudo apt-mark hold kubelet kubeadm kubectl
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@masternode:~#
```

Verify the Kubeadm version using **kubeadm version** command

```
root@masternode:~# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"31", GitVersion:"v1.31.1", GitCommit:"948afe5ca072329a73c8e79ed5938717a5cb3d21", GitTreeState:"clean", BuildDate:"2024-09-11T21:26:49Z", GoVersion:"go1.22.6", Compiler:"gc", Platform:"linux/amd64"}
root@masternode:~#
```

Install the same core components on the worker nodes as well

Step4: Initializing Your Kubernetes Cluster

Run the **sudo kubeadm init** command on the master node to initialize the Kubernetes cluster

```
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.21.49:6443 --token 3drto9.uijs9evnnf9w09xi \
        --discovery-token-ca-cert-hash sha256:71f83328389361279f788c418d7607ce34f28bef8f92000e5239d5d44ab9aa1c
```

Run this below command to apply the Flannel network configuration

```
root@masternode:~# kubectl apply -f https://github.com/coreos/flannel/raw/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
root@masternode:~#
```

The Kubernetes control plane has now been initialized successfully

Step5: Setting Up kubectl Configuration for Non-Root/Root Users:

create the .kube directory, copy the Kubernetes admin configuration file, and change the ownership of the configuration file to the current user. Run this commad to configure Kubectl if you are not running non-root user

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

If you are using the root user, run the following command to set the KUBECONFIG environment variable (**export KUBECONFIG=/etc/kubernetes/admin.conf**)

```
root@masternode:~# export KUBECONFIG=/etc/kubernetes/admin.conf
root@masternode:~#
```

Use the **kubectl get ns** command to verify the namespaces

```
root@masternode:~# export KUBECONFIG=/etc/kubernetes/admin.conf
root@masternode:~# kubectl get ns
NAME              STATUS   AGE
default           Active   83s
kube-node-lease   Active   83s
kube-public       Active   83s
kube-system       Active   83s
```

Step6: Join the worker node to the cluster

Run kubeadm join command which copied from master node **(kubeadm join 172.31.21.49:6443 --token jz0i5y.6sbr302uua0x153s --discovery-token-ca-cert-hash sha256:915a957807e2e56e3a66bfca1f2229645e69f4c70d9adeff495e3a7c5cc11ec5) and we**

```
root@workernode1:~# kubeadm join 172.31.21.49:6443 --token jz0i5y.6sbr302uua0x153s \
        --discovery-token-ca-cert-hash sha256:915a957807e2e56e3a66bfca1f2229645e69f4c70d9adeff495e3a7c5cc11ec5
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.503405309s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Run the **kubectl get nodes** command from the master node to verify the status of the nodes

```
root@masternode:~# kubectl get nodes
NAME          STATUS     ROLES           AGE    VERSION
masternode    Ready      control-plane   117s   v1.31.2
workernode1   NotReady   <none>          8s     v1.31.2
```

Conclusion:

In this guide, we covered the steps to create a fully functional Kubernetes cluster on AWS EC2 instances. You learned how to set up EC2 instances, install Docker, and configure Kubernetes components. With this foundation, you can now manage and deploy containerized applications effectively in a scalable cloud environment.

Keep Learning, Keep Kuberneting!!

Feel free to reach out to me, if you have any other queries or suggestions Stay connected on LinkedIn: Mahendran Selvakumar

Stay connected on Medium: https://devopstronaut.com/