Advance DevOps Lab

Experiment:3

**Aim:** To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

# Theory:

Container-based microservices architectures have profoundly changed the way development and operations teams test and deploy modern software. Containers help companies modernize by making it easier to scale and deploy applications, but containers have also introduced new challenges and more complexity by creating an entirely new infrastructure ecosystem.

Large and small software companies alike are now deploying thousands of container instances daily, and that’s a complexity of scale they have to manage. So how do they do it?

Enter the age of Kubernetes.

Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. In fact, Kubernetes has established itself as the defacto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), backed by key players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

Kubernetes makes it easy to deploy and operate applications in a microservice architecture. It does so by creating an abstraction layer on top of a group of hosts so that development teams can deploy their applications and let Kubernetes manage the following activities:

* Controlling resource consumption by application or team
* Evenly spreading application load across a hosting infrastructure
* Automatically load balancing requests across the different instances of an application
* Monitoring resource consumption and resource limits to automatically stop applications from consuming too many resources and restarting the applications again
* Moving an application instance from one host to another if there is a shortage of resources in a host, or if the host dies
* Automatically leveraging additional resources made available when a new host is added to the cluster

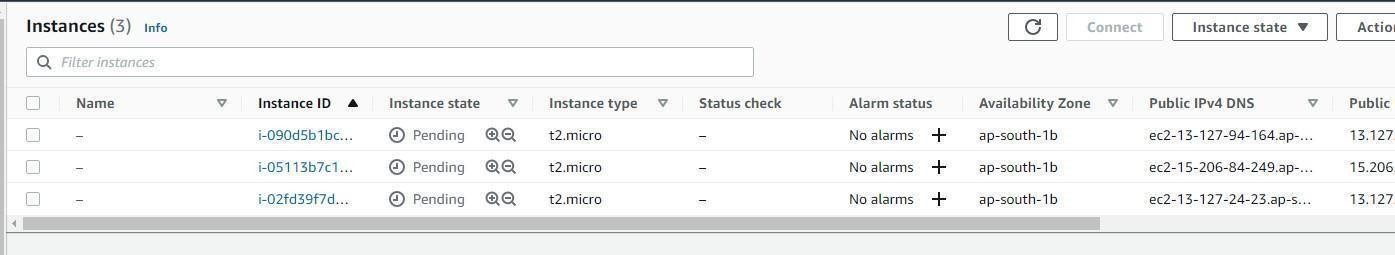


* Easily performing canary deployments and rollbacks

# Steps:

1. Create 3 EC2 Ubuntu Instances on AWS.

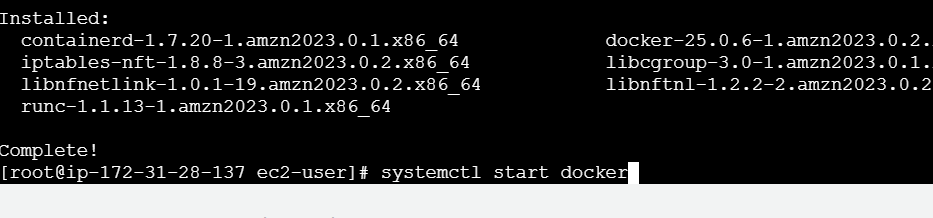
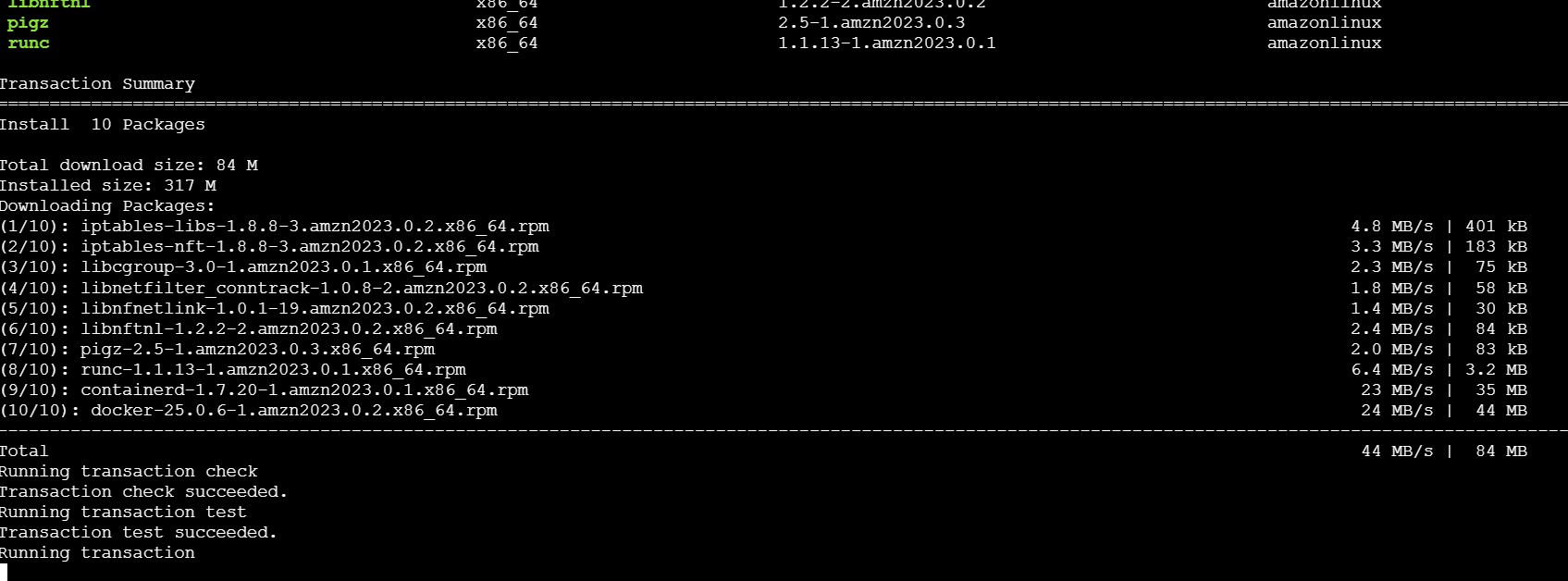
(Name 1 as Master, the other 2 as worker-1 and worker-2)



1. From now on, until mentioned, perform these steps on all 3 machines.

Install Docker

Yum install docker -y



Install Kubernetes on all 3 machines

*# Set SELinux in permissive mode (effectively disabling it)*

sudo setenforce 0

sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

*# This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo*

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo [kubernetes]

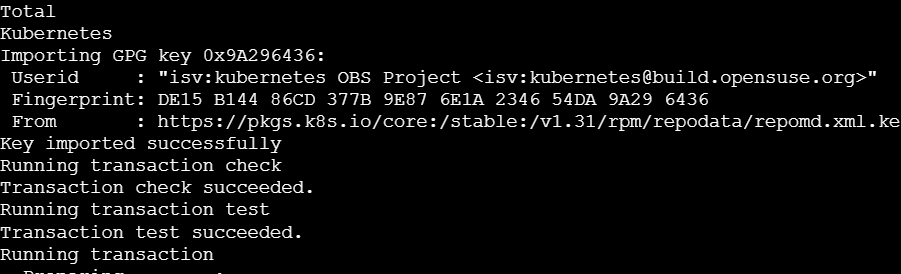
name=Kubernetes baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/ enabled=1

gpgcheck=1 gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml.key exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni

EOF

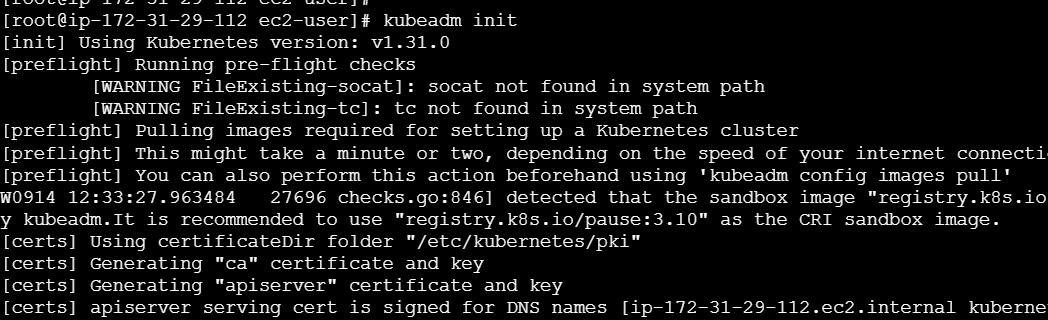
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes (Optional) Enable the kubelet service before running kubeadm:

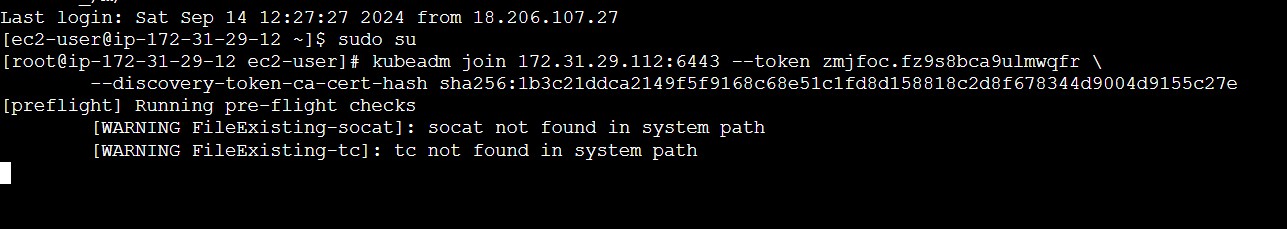
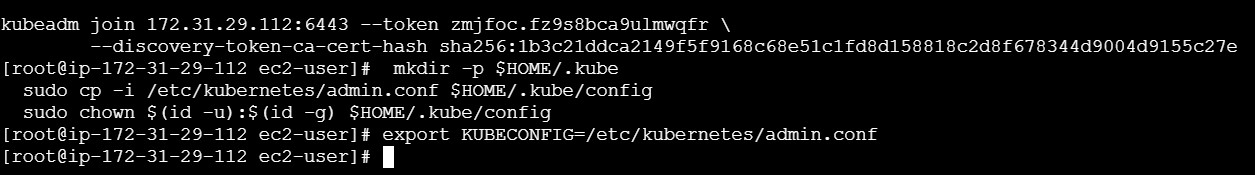
sudo systemctl enable --now kubelet



1. Perform this **ONLY on the Master machine**

Initialize the Kubecluster





That’s it, we now have a Kubernetes cluster running across 3 AWS EC2 Instances. This cluster can be used to further deploy applications and their loads being distributed across these machines.

# Conclusion:

**Kubernetes simplifies containerized application management by automating deployment, scaling, and resource allocation. It efficiently handles load balancing, resource optimization, and fault tolerance, making it the standard for scalable and reliable microservice architectures.**