

Advanced DevOps Lab Experiment 3

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

Reference: <https://www.youtube.com/watch?v=Cz7hSJNq2GU>

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:

The screenshot displays the AWS Management Console interface for launching and managing EC2 instances.

Launch an instance wizard:

- Summary:** Number of instances is set to 1.
- Name and tags:** The instance name is "Master".
- Application and OS Images (Amazon Machine Image):** A search bar is provided to find the desired AMI.
- Free tier notification:** A blue box states: "Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet."
- Buttons:** "Cancel" and "Launch instance" (orange) are visible.

Instances list:

The "Instances (3)" page shows a table of running instances. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability zone.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability zone
worker2	i-0e1f8ab5a7357054f	Running	t2.micro	Initializing	View alarms	us-east-1e
Master	i-03eb3317846815306	Running	t2.micro	2/2 checks passed	View alarms	us-east-1e
worker1	i-0e9ab0eea0c17e76a	Running	t2.micro	Initializing	View alarms	us-east-1b

[EC2](#) > [Instances](#) > [i-0e1f8ab5a7357054f](#) > [Connect to instance](#)

Connect to instance [Info](#)

Connect to your instance i-0e1f8ab5a7357054f (worker2) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console



Port 22 (SSH) is open to all IPv4 addresses

Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in [your security group](#). For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 18.206.107.24/29. [Learn more](#).

Instance ID

i-0e1f8ab5a7357054f (worker2)

Connection Type



Connect using EC2 Instance Connect

Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.



Connect using EC2 Instance Connect Endpoint

Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address

[EC2](#) > [Instances](#) > [i-03eb3317846815306](#) > [Connect to instance](#)

Connect to instance [Info](#)

Connect to your instance i-03eb3317846815306 (Master) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console



Port 22 (SSH) is open to all IPv4 addresses

Port 22 (SSH) is currently open to all IPv4 addresses, indicated by **0.0.0.0/0** in the inbound rule in [your security group](#). For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 18.206.107.24/29. [Learn more](#).

Instance ID

i-03eb3317846815306 (Master)

Connection Type



Connect using EC2 Instance Connect

Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.



Connect using EC2 Instance Connect Endpoint

Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.

Public IP address

Docker install commands :

```
sudo dnf update -y
```

```
sudo dnf install -y docker
```

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```



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

```
[ec2-user@ip-172-31-34-122 ~]$ kubeadm version
kubectrl version --client
kubelet --version
kubeadm version: &version.Info{Major:"1", Minor:"26", GitVersion:"v1.26.1", GitCommit:"8f94681cd294aa8cfd3407b8191f6c70214973a4", GitTreeState:"clean", BuildDate:"2023-01-18T15:56:50Z", GoVersion:"go1.19.5", Compiler:"gc", Platform:"linux/amd64"}
WARNING: This version information is deprecated and will be replaced with the output from kubectrl version --short. Use --output=yaml|json to get the full version.
Client Version: version.Info{Major:"1", Minor:"26", GitVersion:"v1.26.1", GitCommit:"8f94681cd294aa8cfd3407b8191f6c70214973a4", GitTreeState:"clean", BuildDate:"2023-01-18T15:58:16Z", GoVersion:"go1.19.5", Compiler:"gc", Platform:"linux/amd64"}
Kustomize Version: v4.5.7
Kubernetes v1.26.1
```

To clean:

```
sudo dnf clean all
sudo dnf makecache
sudo dnf update -y
```