

DevOps Interview Questions - Answers

- ❖ AWS
 - ❖ Linux
 - ❖ Git
 - ❖ GitHub/Jenkins/GitLab CICD
 - ❖ Monitoring
 - ❖ Architecture
 - ❖ Cost Optimization
 - ❖ Security
 - ❖ High Availability / Failover
 - ❖ Real Time Troubleshootings
 - ❖ Terraform
-
- ❖ Docker
 - ❖ What is Docker?
 - ❖ Architecture of Docker
 - ❖ Setup & Configuration
 - ❖ Docker Commands
 - ❖ Docker File
 - ❖ Docker Multi Stage Builds
 - ❖ Docker Interview Questions
 - ❖ Docker Troubleshooting Commands

❖ Kubernetes

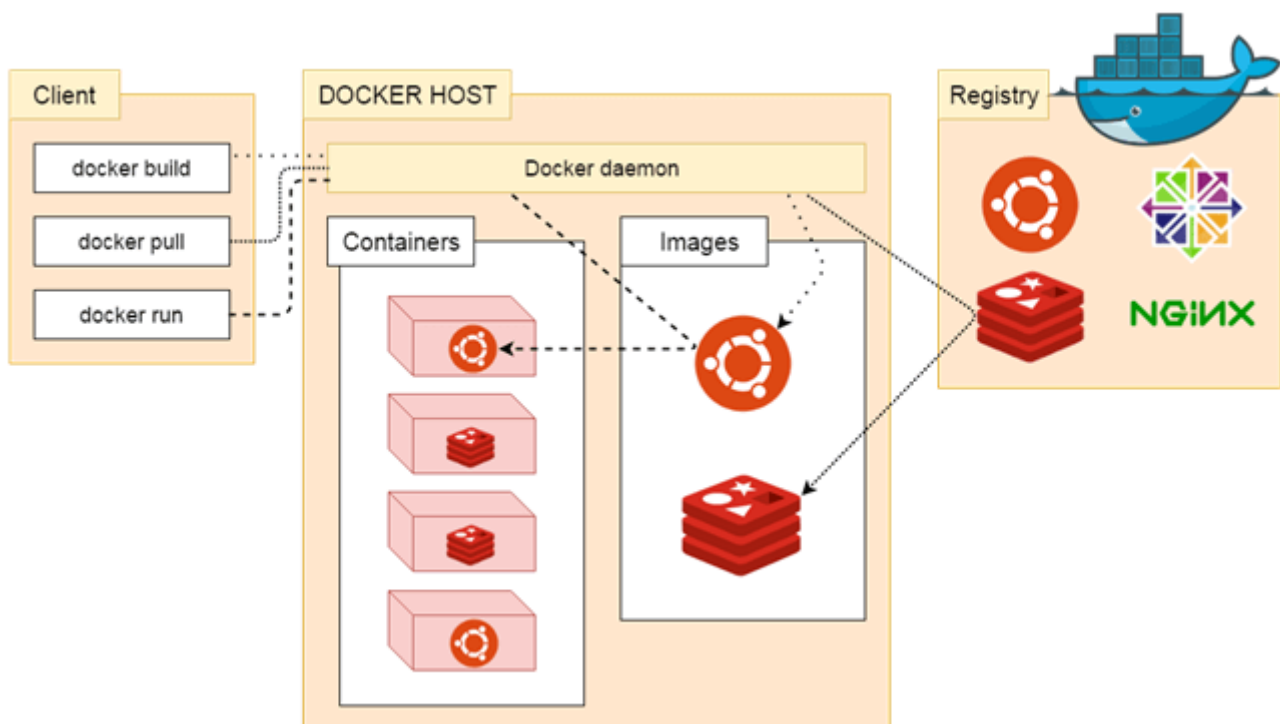
- ❖ What is Kubernetes?
- ❖ EKS Cluster Setup
- ❖ Kubernetes Architecture
- ❖ Services
 - NodePort
 - Load balancer
 - Cluster IP
 - External Name
 - Headless
 - Ingress
- ❖ Controller
 - Deployment
 - Daemon set
 - Stateful set
 - Replica set
- ❖ Liveness
- ❖ Readiness
- ❖ Namespace
- ❖ Secrets
- ❖ Real Time Troubleshootings

Docker

❖ What is Docker?

Docker is a platform for developing, shipping, and running applications using containerization technology. It allows developers to package their applications and dependencies into lightweight, portable containers that can run consistently across different environments. Docker simplifies the process of building, deploying, and scaling applications by isolating them from the underlying infrastructure, enabling faster and more reliable software delivery.

❖ Architecture of Docker



➤ Docker Daemon

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage.

➤ Docker Host

Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks, and storage.

➤ Docker Registry

Docker Registry manages and stores the Docker images.

- Public Registry – Docker Hub, Google CR & AWS ECR
- Private Registry – GitLab CR, Azure CR & JFrog

❖ Setup & Configuration

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
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 repository 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
sudo apt-get update
```

To install latest version of Docker packages

```
sudo apt-get install docker-ce docker-ce-cli containerd.io  
docker-buildx-plugin docker-compose-plugin  
docker -v
```

Verify that the Docker Engine installation is successful by running the hello-world image

```
sudo docker run hello-world
```

This command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

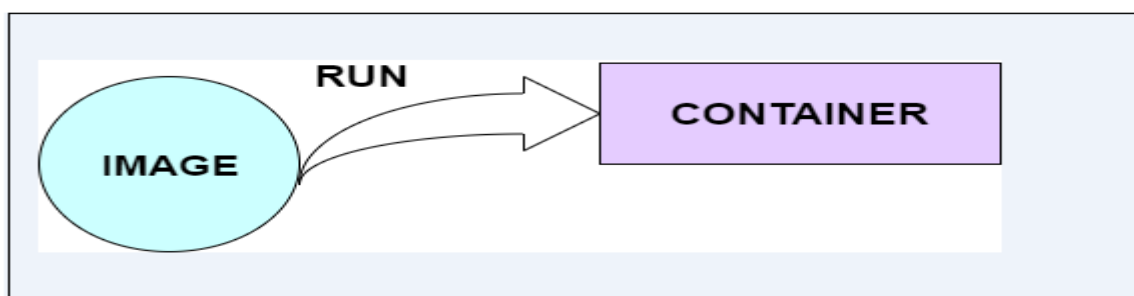
You have now successfully installed and started Docker Engine.

Configure Docker to start on boot with systemd

```
sudo systemctl enable docker.service  
sudo systemctl enable containerd.service
```

Run sample docker container

```
docker run -it --rm -d -p 8080:80 --name web nginx
```



Ref:

<https://docs.docker.com/engine/install/ubuntu/>

<https://www.docker.com/blog/how-to-use-the-official-nginx-docker-image/>

<https://docs.docker.com/get-started/>

❖ Docker File

Dockerfile is a text file that contains a list of commands (instructions), which describes how a Docker image is built based on them.



An example of Dockerfile:

```
# Use the official Nginx base image
FROM nginx:latest

# Set environment variables
ENV NGINX_VERSION=latest \
    NGINX_PORT=80 \
    NGINX_WORKDIR=/usr/share/nginx/html

# Set the working directory
WORKDIR $NGINX_WORKDIR

# Copy custom configuration file
COPY nginx.conf /etc/nginx/nginx.conf

# Copy static files to serve
COPY index.html $NGINX_WORKDIR/index.html

# Install nodejs package
RUN apt-get update && apt-get install -y \
    nodejs \
    npm \
    && npm install

# Set the local volume to the container
VOLUME /var/lib/mysql

# Expose port
EXPOSE $NGINX_PORT
```

```
# Define the default command to run when the container starts
ENTRYPOINT ["node", "app.js"]
```

```
$ docker build
```

You can name your image as well.

```
$ docker build -t my-image
```

If your Dockerfile is placed in another path,

```
$ docker build -f /path/to/a/Dockerfile .
```

To check images on the system

```
$ docker image ls
```

❖ Docker Command Difference

▪ COPY vs ADD

Both commands serve a similar purpose, to copy files into the image.

- **COPY** - let you copy files and directories from the host.
- **ADD** - does the same. Additionally it lets you use URL location and unzip files into image.

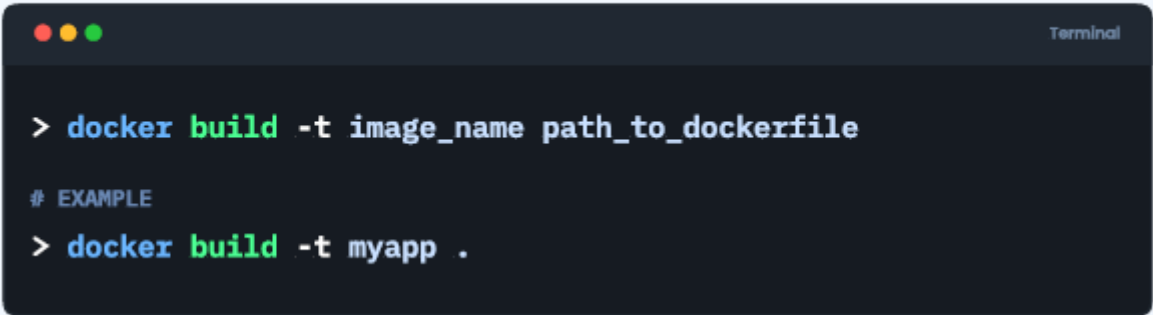
▪ ENTRYPOINT vs. CMD

- **CMD** - allows you to set a default command which will be executed only when you run a container without specifying a command. If a Docker container runs with a command, the default command will be ignored.
- **ENTRYPOINT** - allows you to configure a container that will run as an executable. ENTRYPOINT command and parameters are not ignored when Docker container runs with command line parameters.
- **Ref:** <https://www.geeksforgeeks.org/what-is-dockerfile-syntax/>
- <https://docs.docker.com/reference/dockerfile/>
- <https://www.geeksforgeeks.org/what-is-dockerfile/>

❖ Docker Commands

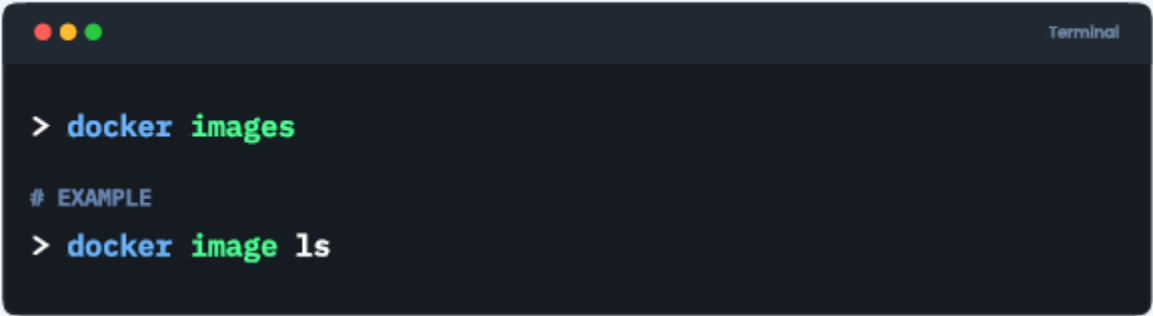
Below are some of the important docker commands

1. Build an image from a Dockerfile:

A terminal window with a dark background and a title bar with three colored dots (red, yellow, green) on the left and the word "Terminal" on the right. The terminal contains the following text:

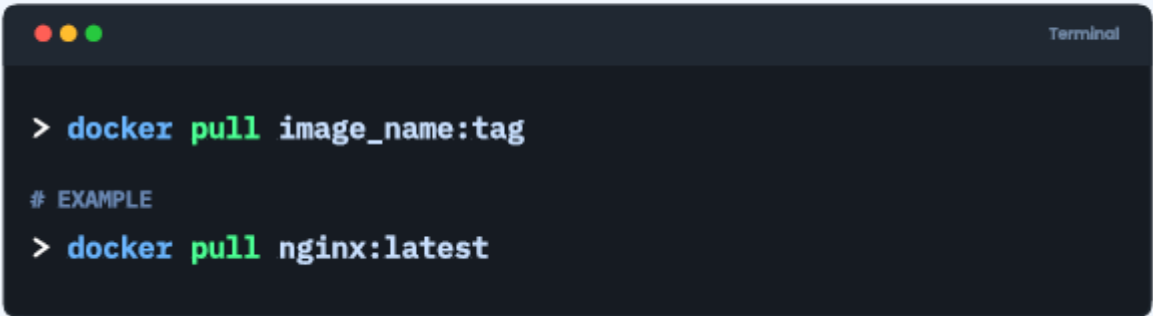
```
> docker build -t image_name path_to_dockerfile  
  
# EXAMPLE  
> docker build -t myapp .
```

2. List all local images:

A terminal window with a dark background and a title bar with three colored dots (red, yellow, green) on the left and the word "Terminal" on the right. The terminal contains the following text:

```
> docker images  
  
# EXAMPLE  
> docker image ls
```

3. Pull an image from Docker Hub:

A terminal window with a dark background and a title bar with three colored dots (red, yellow, green) on the left and the word "Terminal" on the right. The terminal contains the following text:

```
> docker pull image_name:tag  
  
# EXAMPLE  
> docker pull nginx:latest
```


4. Remove a local image:

```
> docker rmi image_name:tag

# EXAMPLE

> docker rmi myapp:latest
```

Or

```
> docker rm [image_name/image_id]

# EXAMPLE

> docker rm fd484f19954f
```

5. Tag an image:

```
> docker tag source_image:tag new_image:tag

# EXAMPLE

> docker tag myapp:latest myapp:v1
```

6. Push an image to Docker Hub:

```
> docker push image_name:tag

# EXAMPLE

> docker push myapp:v1
```

7. Inspect details of an image:

```
> docker image inspect image_name:tag

# EXAMPLE

> docker image inspect myapp:v1
```

Docker Container

1. Run a container from an image:

```
> docker run container_name image_name  
# EXAMPLE  
> docker run myapp
```

2. Run a named container from an image:

```
> docker run --name container_name image_name:tag  
# EXAMPLE  
> docker run --name my_container myapp:v1
```

3. List all running containers:

```
> docker ps
```

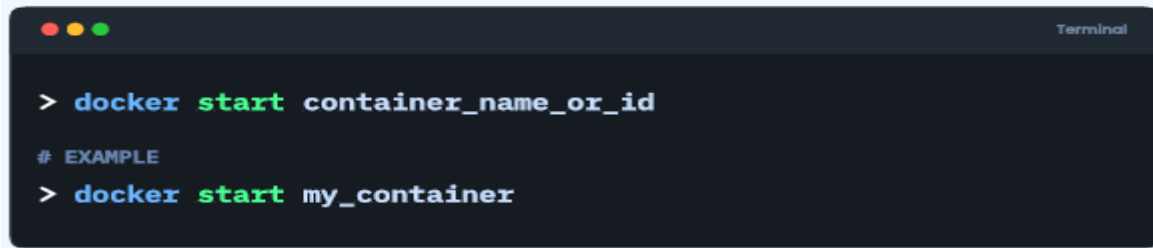
4. List all containers (including stopped ones):

```
> docker ps -a
```

5. Stop a running container:

```
> docker stop container_name_or_id  
# EXAMPLE  
> docker stop my_container
```

6. Start a stopped container:

A terminal window with a dark background and light-colored text. The title bar shows three colored circles (red, yellow, green) on the left and the word "Terminal" on the right. The terminal content shows a command prompt followed by the command to start a Docker container.

```
> docker start container_name_or_id  
  
# EXAMPLE  
> docker start my_container
```

Ref:

- <https://www.mygreatlearning.com/blog/top-essential-docker-commands/>
- <https://www.geeksforgeeks.org/docker-instruction-commands/?ref=lbp>
- <https://docs.docker.com/get-started/>

❖ Docker Compose

Docker Compose is a tool for defining and running multi-container applications

Ref: <https://www.simplilearn.com/tutorials/docker-tutorial/docker-compose>

❖ Docker Swarm

Docker Swarm is an orchestration management tool that runs on Docker applications. It helps end-users in creating and deploying a cluster of Docker nodes.

Ref:

- <https://www.simplilearn.com/tutorials/docker-tutorial/docker-swarm>
- <https://docs.docker.com/engine/swarm/swarm-tutorial/>
- <https://www.sumologic.com/glossary/docker-swarm/>

❖ Docker Multi Stage Builds

We can use multi stage builds in Docker to optimize docker image size, faster deployment etc.

Ref:

- <https://dev.to/pavanbelagatti/what-are-multi-stage-docker-builds-1mi9>
- <https://docs.docker.com/build/guide/multi-stage/>
- <https://earthly.dev/blog/docker-multistage/>
- <https://www.harness.io/blog/how-to-create-multi-stage-docker-builds-with-harness-continuous-delivery>

❖ Docker Container Status

- **Created:** The container has been created but not yet started.
- **Restarting:** The container is restarting, either due to a failure or a manual restart command.
- **Running:** The container is up and running, executing its defined tasks or processes.
- **Paused:** The container's execution has been paused, suspending all processes within the container.
- **Exited:** The container has stopped running, either because it has completed its task or due to an error. You can check the exit code to determine the reason for the exit.
- **Removing:** The container is in the process of being removed or deleted from the system.
- **Unknown:** The status of the container cannot be determined, often due to an error or communication issue with the Docker daemon.

❖ Step To Create Jenkins Container

➤ Create Jenkins Docker Container

```
# docker pull jenkins/jenkins:its
# docker run -d -p 8080:8080 -v jenkins_home:/var/jenkins_home --name
jenkins_container jenkins/jenkins:its
# docker logs jenkins_container
```

create Nginx web server container with custom index page

```
# Use the official Nginx base image
FROM nginx:latest
# Set environment variables
ENV NGINX_VERSION=latest \
    NGINX_PORT=80 \
    NGINX_WORKDIR=/usr/share/nginx/html
# Set the working directory
WORKDIR $NGINX_WORKDIR

# Copy custom configuration file
COPY nginx.conf /etc/nginx/nginx.conf
# Copy static files to serve
COPY index.html $NGINX_WORKDIR/index.html
# Expose port
EXPOSE $NGINX_PORT
```

Build Image

```
docker build -t <image_name> .
```

for example - `docker build -t demo-docker-image .`

Check docker images

- `docker image ls`

```
root@ip-192-168-4-94:~# docker image ls
REPOSITORY          TAG          IMAGE ID      CREATED        SIZE
demo-docker-image   latest       20e52af5d66d  33 seconds ago 187MB
namdevnmr/firstimage 1           8f509f3c343d  15 hours ago   187MB
namdevnmr/firstimage latest      8f509f3c343d  15 hours ago   187MB
jenkins/jenkins     lts         2371da23064a  4 days ago     462MB
nginx                latest      92b11f67642b  5 weeks ago    187MB
```

- Tag docker image for push

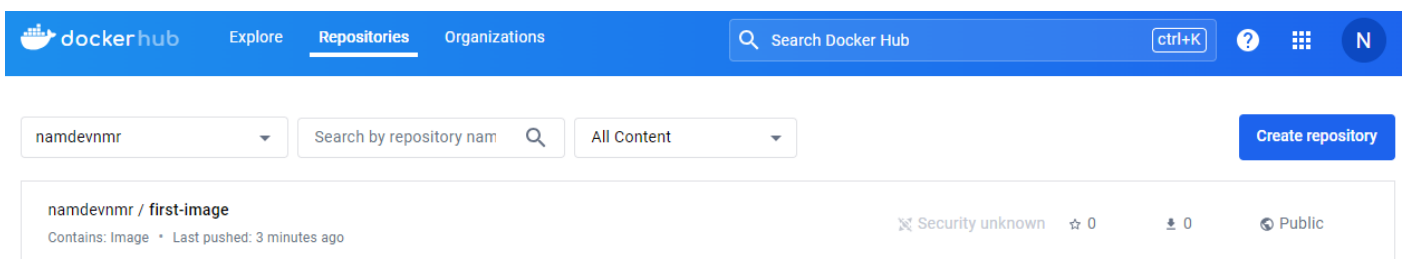
docker tag demo-docker-image namdevnmr/first-image

- Push docker image now
`docker push namdevnmr/first-image`

```
root@ip-192-168-4-94:~# docker push namdevnmr/first-image
Using default tag: latest
The push refers to repository [docker.io/namdevnmr/first-image]
b38bb6080722: Pushed
8e63057184f2: Mounted from namdevnmr/firstimage
5f70bf18a086: Mounted from namdevnmr/firstimage
fd31601f0be4: Mounted from namdevnmr/firstimage
93b4c8c4ac05: Mounted from namdevnmr/firstimage
b7df9f234b50: Mounted from namdevnmr/firstimage
ab75a0b61bd1: Mounted from namdevnmr/firstimage
c1b1bf2f95dc: Mounted from namdevnmr/firstimage
4d99aableed4: Mounted from namdevnmr/firstimage
a483da8ab3e9: Mounted from namdevnmr/firstimage
latest: digest: sha256:4d8a10108e2c71c2f1919e0968524edfe1d93206f0a8e763352d96e8ae449bc0 size: 2398
```

Login to docker hub and check image pushed or not

<https://hub.docker.com/>



The screenshot shows the Docker Hub interface. At the top, there's a navigation bar with 'dockerhub', 'Explore', 'Repositories' (selected), and 'Organizations'. A search bar is on the right. Below the navigation bar, there's a search filter section with 'namdevnmr' selected, a search input, and 'All Content' selected. A 'Create repository' button is on the right. The main content area shows the repository 'namdevnmr / first-image' with a status 'Contains: Image' and 'Last pushed: 3 minutes ago'. On the right, there are icons for 'Security unknown', '0 stars', '0 downloads', and 'Public'.

```
docker tag demo-image namdevnmr/first-image:1
```

```
docker push namdevnmr/first-image:1
```

Remove Docker Container

`docker ps -a`

`docker rm <container_id>`

docker image ls

`# docker rmi <image_name>`

❖ Docker Troubleshooting Commands

Displays the logs of a specific container, useful for debugging errors or issues.

`docker logs [container_id]`

Provides detailed information about a container, including its configuration and networking details.

`docker inspect [container_id]`

Monitors real-time events from Docker, helpful for tracking container lifecycle events or errors.

`docker events`

Displays the running processes inside a container, aiding in diagnosing performance issues or unexpected behavior.

`docker top [container_id]`

Shows resource usage statistics for a container, including CPU, memory, and network usage.

`docker stats [container_id]`

Opens an interactive shell inside a running container, allowing direct access for troubleshooting or debugging.

`docker exec -it [container_id] /bin/bash`

Lists the ports mapped to a container, aiding in verifying port bindings and networking configurations.

`docker port [container_id]`

Provides detailed information about a Docker network, including connected containers and their IP addresses.

`docker network inspect [network_id]`

Shows the file system changes made within a container since it was started, helpful for identifying modifications.

`docker diff [container_id]`

Displays information about disk usage by Docker, including images, containers, and volumes, useful for identifying storage issues.

`docker system df`

Lists recent Docker events, providing insights into system-level changes or errors.

`docker system events`

Shows detailed information about the Docker installation, including version, configuration, and supported features.

`docker info`

Displays the Docker version information, helpful for checking compatibility or troubleshooting version-related issues.

`docker version`

Lists all Docker networks, aiding in identifying network-related problems or misconfigurations.

`docker network ls`

Lists all Docker images available locally, useful for checking if required images are present.

`docker image ls`

Displays the history of an image, showing how it was built and its layers, helpful for identifying issues in the image build process.

`docker image history [image_name]`

Lists all Docker volumes, aiding in identifying volume-related problems or checking if required volumes exist.

`docker volume ls`

Removes unused data, including stopped containers, dangling images, and unused networks or volumes, helping to reclaim disk space and clean up the system.

`docker system prune`

Displays logs for a specific service defined in a `docker-compose.yml` file, useful for troubleshooting multi-container applications.

`docker-compose logs [service_name]`

Kubernetes

❖ Kubernetes

❖ What is Kubernetes?

[Kubernetes](#) is an open-source Container Management tool that automates container deployment, container scaling, descaling, and container load balancing (also called a container orchestration tool). It is written in Go language.

<https://www.geeksforgeeks.org/kubernetes-node/?ref=lbp>

<https://www.geeksforgeeks.org/kubernetes-nodeport-service/?ref=lbp>

- ❖ EKS Cluster Setup
- ❖ Kubernetes Architecture
- ❖ Services
 - NodePort
 - Load balancer
 - Cluster IP
 - External Name
 - Headless
 - Ingress
- ❖ Controller
 - Deployment
 - Daemon set
 - Stateful set
 - Replica set
- ❖ Liveness
- ❖ Readiness
- ❖ Namespace
- ❖ Secrets
- ❖ Real Time Troubleshootings

❖ EKS Cluster Setup

Pre-requisites

❖ About Author

- **Name** : Namdev Rathod
- **GitHub Projects** : <https://github.com/namdev-rathod>
- **YouTube** : <https://youtube.com/@namdev.devops>
- **Topmate** : <https://topmate.io/namdevrathod>
- **Email** : support@devopswithnamdev.com
- **WhatsApp** : +91 7249 0590 06
- **Website** : <https://devopswithnamdev.com>