DOCKER

Docker, the containerisation platform, has become increasingly popular in recent years due to its ability to simplify application deployment and management.

One of the key areas where Docker has made an impact is in DevOps, which is the practice of combining software development and IT operations to increase efficiency and productivity.

With Docker, DevOps teams can package an application and its dependencies into a single container that can be deployed across multiple environments, including development, testing, and production. This not only reduces the risk of compatibility issues but also streamlines the deployment process, allowing for faster release cycles and more frequent updates.

Simple way to understand - **Docker, the containerisation platform**

Containerisation is lightweight and efficient, as it allows multiple applications to run on the same operating system, without the need for a complete virtual machine. This can be especially useful for running microservices.

Docker

Docker is an open-source containerization platform which helps you create containers that allows developers to easily package, deploy, and run their applications in any environment.

It provides a way to isolate applications from the underlying system, making it easier to ensure that they run consistently across different environments.

Docker is especially useful for **microservices architecture**, where applications are broken down into smaller, independent components that can be developed and deployed separately.

With Docker, developers can build, ship, and run distributed applications easily and quickly, without worrying about the underlying infrastructure.

Docker Architecture

Docker architecture refers to the structure of components, both hardware and software, that make up Docker.

At its core, Docker architecture consists of

- **Docker Daemon / Docker Service**
- **Docker Client**

● **Docker Registry**

● The Daemon/Service is responsible for managing Docker objects such as images, containers, and networks. All the heavy lifting is done by daemon.

● The client sends commands to the daemon and receives output.

● The registry stores Docker images and allows them to be shared among users and systems.

Note: When using Docker, it is important to have a basic understanding of its architecture in order to effectively deploy and manage Docker containers. By understanding the various components and their roles, you can better troubleshoot issues and optimize your Docker environment for your specific use case.

Docker Registry

Docker Registry is a piece of software that stores and distributes Docker images. It enables developers to share and collaborate on container images with ease. Docker Registry can be used to host both public and private images. Following are a few registries docker hub, aws registry.

Docker Hub

Docker Hub is one of the most popular ways to distribute Docker images and is widely used in the software development industry. With its ability to store, manage, and distribute container images, Docker Hub has become an essential tool for developers and organizations looking to streamline their container workflows.

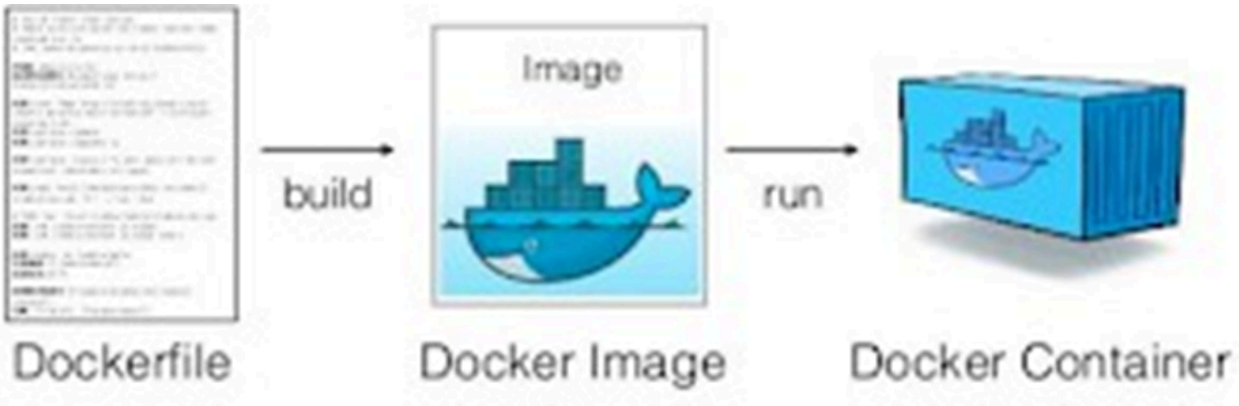
By using Docker Hub, developers can save time and effort by taking advantage of pre-built images and by collaborating with other developers to improve the quality and functionality of their applications.

https://hub.docker.com/

Docker image

A Docker image is a lightweight, standalone, and executable package that includes everything needed to run the desired software application.

Docker Images are built from a Dockerfile, which is a script that contains all the commands to assemble the image.

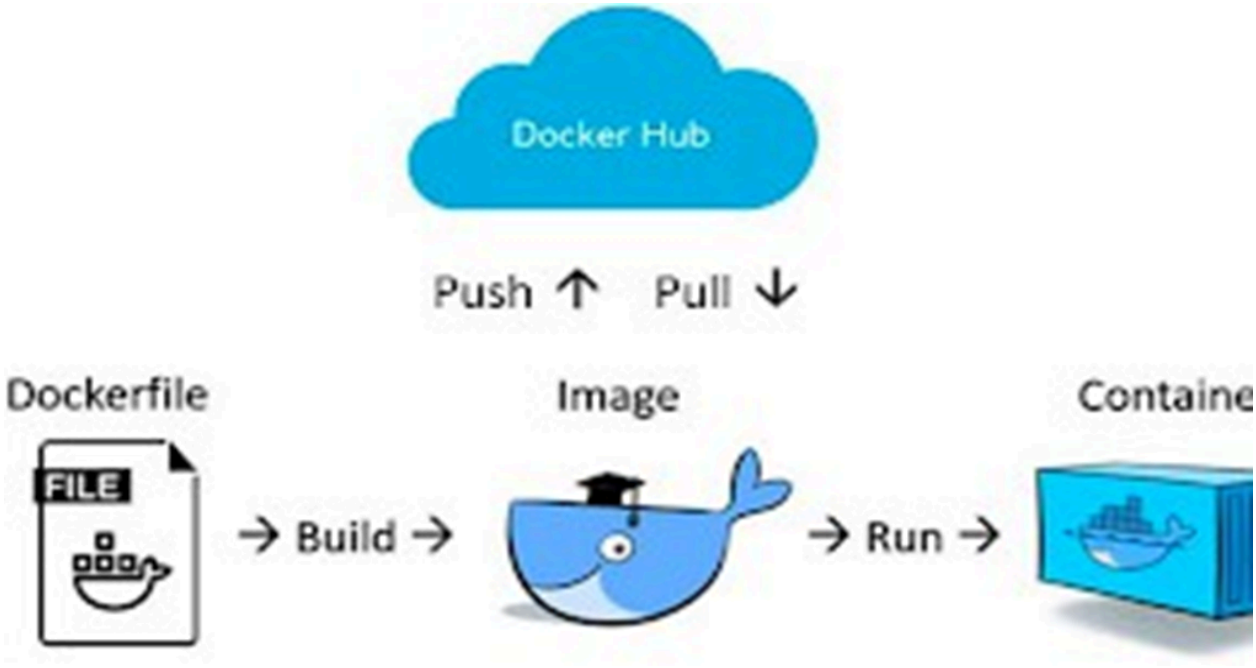


Docker images can be stored in a registry, such as Docker Hub, and can be pulled from there to run on any machine with Docker installed.

Additionally, Docker images can be customized and layered on top of each other to create more complex applications with multiple components. Overall, Docker images provide an efficient and portable way to package and distribute software applications.

Docker Container

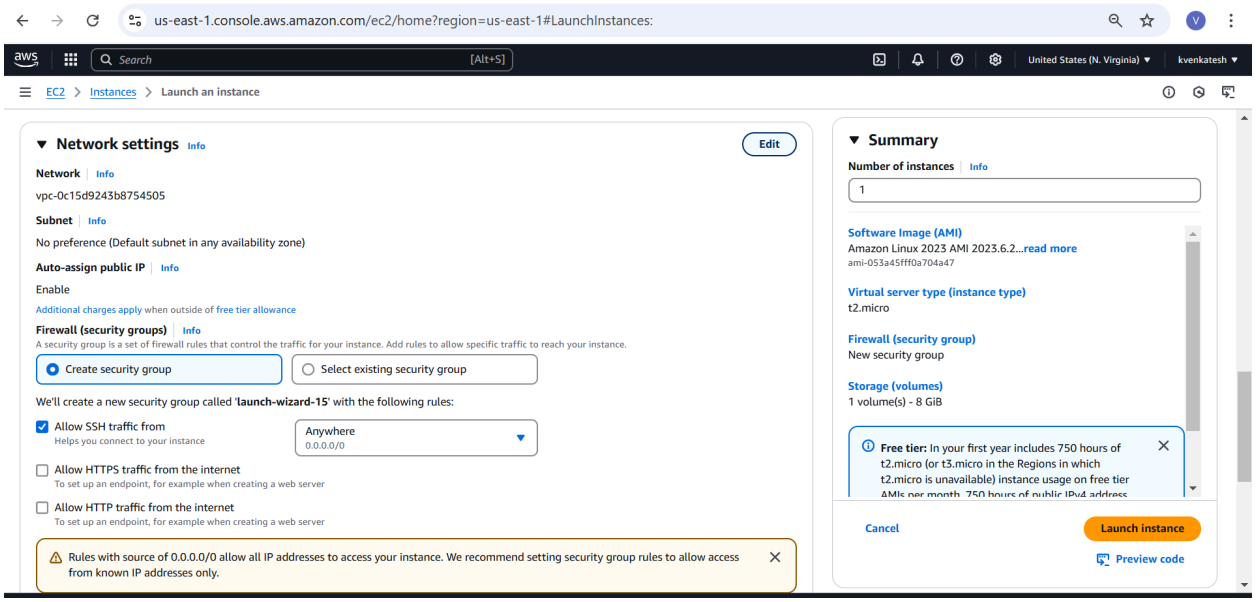
A container in Docker is an isolated environment that allows you to run an application or service with all the dependencies it needs, without interfering with other applications or services running on the same system. To create a container we need Docker Image.



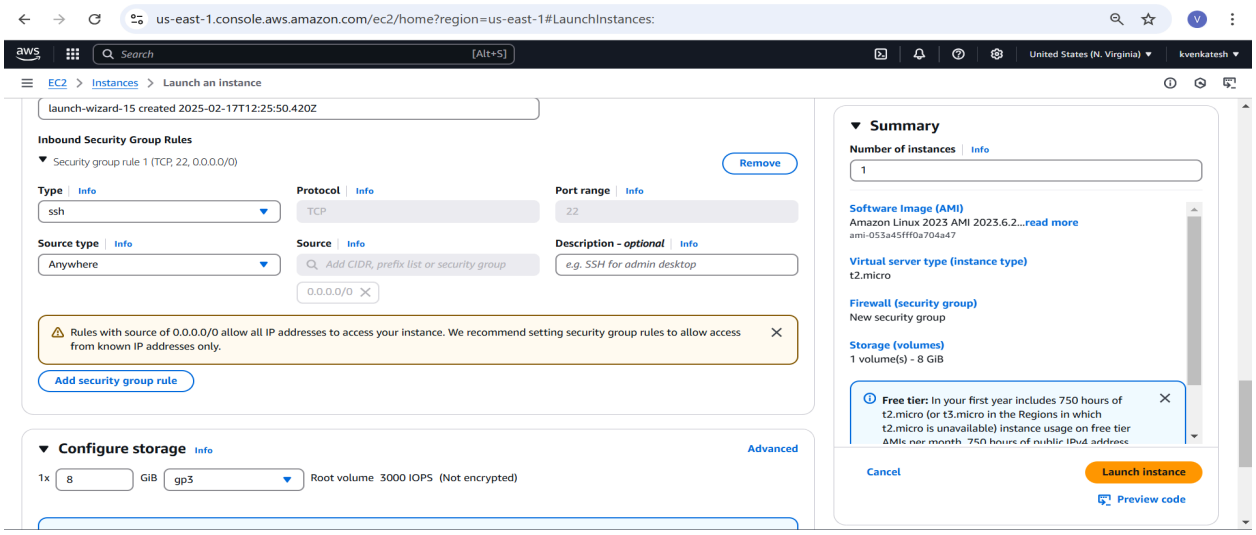
Setup Docker

- Setup t2.large instance with 20 GB Volume
- Add Following Rules in Security Groups 32768-61000, and 8080-9090, 80, 22
- We can install docker on any operating system whether it is Linux, Windows or MAC.

Go To Network settings click on **Edit**



Now click **add security groups rule**

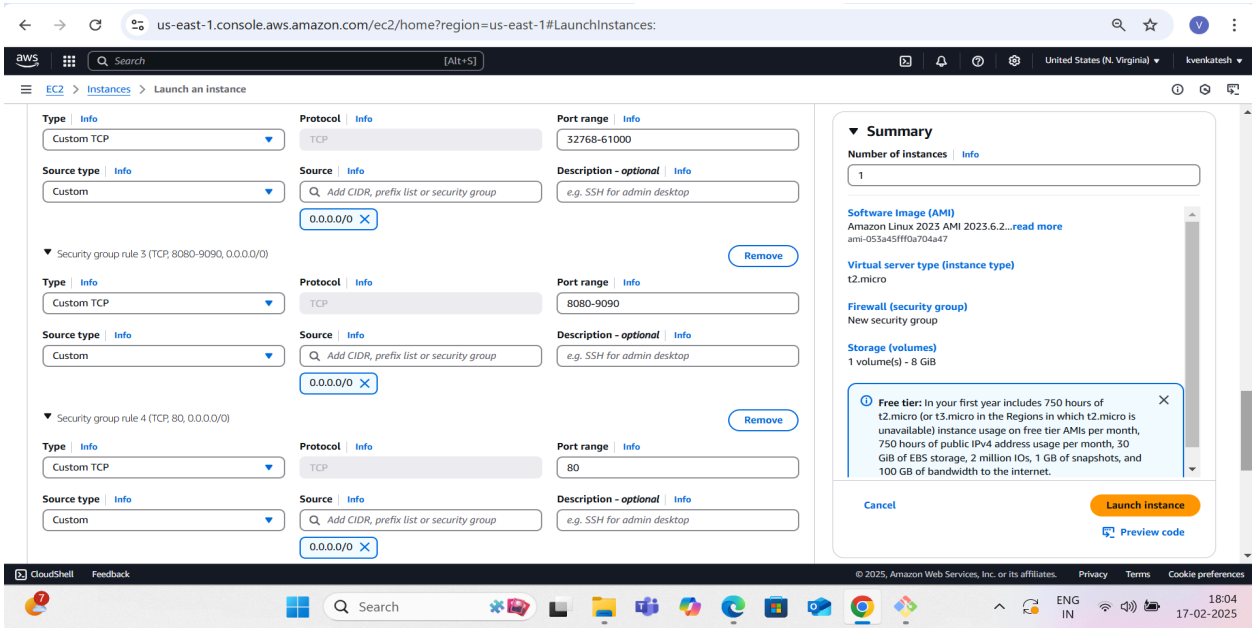


Now add this ports to run

In port range - 8080-9090
Source - 0.0.0.0/0

Port range - 32768-61000
Source - 0.0.0.0/0

Port range- 80
Source - 0.0.0.0/0



Now give **15 GB** or **20 GB**
Click in launch instance

Check the Docker First
Command: docker

Update your system
Command: sudo apt update -y

Now install Docker
Command: curl -fsSL https://get.docker.com -o get-docker.sh && sh get-docker.sh

Now check docker
Command: docker

When installing Docker, you can use the command "**curl fsSL https://get.docker.com -o get-docker.sh**" to download the installation script, followed by executing the script using the "sh" command. This process is quick and easy, and once Docker is installed, you can enjoy its many benefits, such as the ability to easily create and run containers for your applications.

Command: docker info

Command: docker image ls
Command: docker container ls

To run the docker related work, without sudo access, add the current user to the docker group.

Command: sudo usermod -aG <group> <user-name>

Note: group = docker
User-name = ubuntu

Command: sudo usermod -aG docker ubuntu

Command: logout

The command above allows you to add a user to a specific group. To be more precise, the "**sudo usermod**" command is used to modify a user's account details, with "**-aG**" specifying the group to which the user will be added. In the example provided, the group is "docker" and the user is "**ubuntu**". Once again ssh back into docker host and start executing docker commands.

Command: docker info

Command: docker image ls

Command: docker container ls

● **docker info**: This will display system-wide information about the Docker installation on your machine, including the version number, the number of containers and images, and the status of your containers.

● **docker image ls**: This will show a list of all the images that are currently stored on your machine. You can use this command to find the image you need for your application.

● **docker container ls**: This will display a list of all the containers that are currently running on your machine. You can use this command to check the status of your containers and to manage them as needed.

Setup Containers

Setup Nginx Container

To get started with using Nginx in a Docker container, you can begin by pulling the Nginx image from the Docker Hub registry using the command:

Check the image is available or not
Command: docker image ls

If image is not available pull the image now
Command: docker pull nginx

Now check the image
Command: docker image ls

To run an Nginx server using Docker containerization, you can use the command,

command: docker container run nginx

This command will pull the Nginx image from the Docker Hub registry and create a container instance of it.

Command: docker container ls

Command: sudo systemctl status nginx

Command: sudo ss -ntpl

Command: docker container inspect c13ddeded047

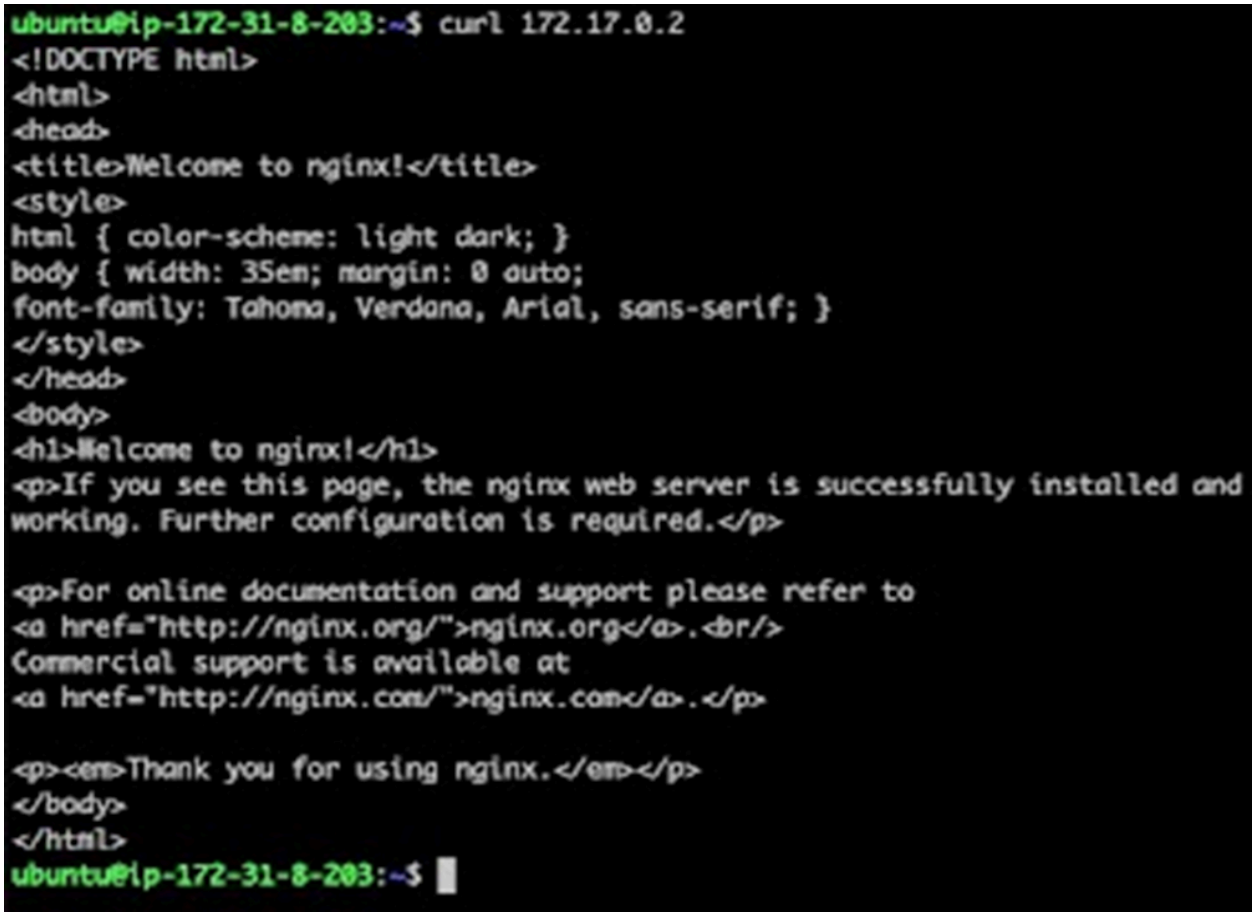
To inspect the details of a specific Docker container, you can use the command "**docker container inspect**". This command will provide you with information such as the container's ID, name, status, and configuration settings. In addition, you can also view the container's network settings, including its IP address and the ports it is listening on.

By using this command, you can gain a deeper understanding of how your Docker containers are configured and running, which can be useful for troubleshooting issues.

Test Web Server
Browse - http://public-ip

curl 172.17.0.2

In order to retrieve data from a server, you can use the command " **curl** " followed by the IP address of the server. This will initiate a request to the server, which will then respond with the requested data. Once the data has been retrieved, it can be saved to a local file or displayed on the screen,



Setup Containers

Container Modes

● When using Docker, there are two modes to run containers: mode and detached mode. interactive

● **Interactive mode** allows you to interact with the container's shell directly.

● **Detached mode** runs the container in the background, which is useful for running long-term processes without needing to keep a terminal open.

Interactive/Foreground Mode

Command: docker container run nginx

Command: docker container run -it nginx

Detached/Background Mode

Command: docker container run -dt nginx

The **-dt** flag indicates that the container should be started in meaning that it will run in the detached mode, background and not take up your terminal session.

Command: docker container ls

Command: docker container ls -a

When running the "docker container ls" command, you will see a list of currently running containers. This can be useful to quickly check which containers are currently active. However, if you want to see all containers, including those that are not currently running, you can run the "docker container ls -a" command instead.

Manage Containers

Command: docker container rm d1424ba58ee4

In order to remove a container in Docker, you can use the command "**docker container rm**". It is important to note that this command is irreversible, so be sure that you want to permanently remove the container before executing the command.

If you want to remove multiple containers at once, you can specify their names or IDs in a space-separated list after the "**docker container rm**" command.
Command: docker container stop d1424ba58ee4
hjkimu
Command: docker container start d1424ba58ee4

When working with Docker, you have the ability to start and stop containers as needed. By using the docker container start command, you can start a previously stopped container and resume its operations. This can be useful in situations where you need to temporarily shut down a container for maintenance or updates.

Conversely, the docker container stop command allows you to stop a running container and free up the resources it was using. This can be helpful in situations where you need to quickly stop a container due to a problem or to make changes to its configuration.