Experiment No: 3 B

Title :- Docker Container Environment.

objectives :-

1. Understand about basic concept of Docker.
2. Creating Account on Docker Container Environment.

Problem statement :-

Create Docker container environment

Outcomes :-

Student will be able to,

1. Understand the concept of Docker Environment.
2. create account on Docker container Environment.

Software & Hardware Requirements :-

Software :- Browser, Docker Container environment.

~~PRE-Requistes~~ :-

1. NVIDIA GPU :- Sure that you have a system with an NVIDIA GPU installed.

2. ~~NV~~NVIDIA Docker : Install NVIDIA Docker on your System. The installation step can vary based on your operating System. Refer to the offical documentation for installation instructions.

Theory :-

Cker is an open paltform for developing, shipping and running applications. Docker enables you to Separate your application from your infrastructure so you can deliver Software quickly with Docker, you can manage your infrastructure in the Same way you manage your application.

The Docker Platform

Docker provide the ability to package and run an application in a loosly isolated environment called a conteiner. The isolation and Security lets you to run away Containers Simulteously on a given host. Container are lightweight and contain everythink needed to run the application, so you don't need to rely on whats installed on the host.

Docker provide tooling and a platform to manage the lifecycle of your Container.

- Develop your application and its Supporting Components using Container.
- The Container become the unit for destrubuting and testing your application
- When your ready, deploy your applicatio into your production environment, as a Container or an orchestrated Services. This work the Same wether your production environment is an local data Centre, a cloud provider, or a hybrid of the two.

Uses of Docker :-

1. Fast, Consistent delivery of your applications.

Docker Stoamlines the development lifecycle by allowing developers to work in Standaized environments using local Containers which provide your application and Services. Containers are greate for Continuous integration and Continous delivery workFlows.

Consider the Following example Scenario:

- Your developers write code locally and Share their work with their colleagues using Docker Containers.
- They use Docker to push their application into a teast environment and run automated and manual tests.
- When developers find bugs, ~~their~~ they Can Fix them in the development environment and redeploy them to the test environment For testing and Validation.
- When testing is Complete, getting the Fix to the Customer is a simple as pushing the update image to the production environment.

Responsive deployment and Scaling

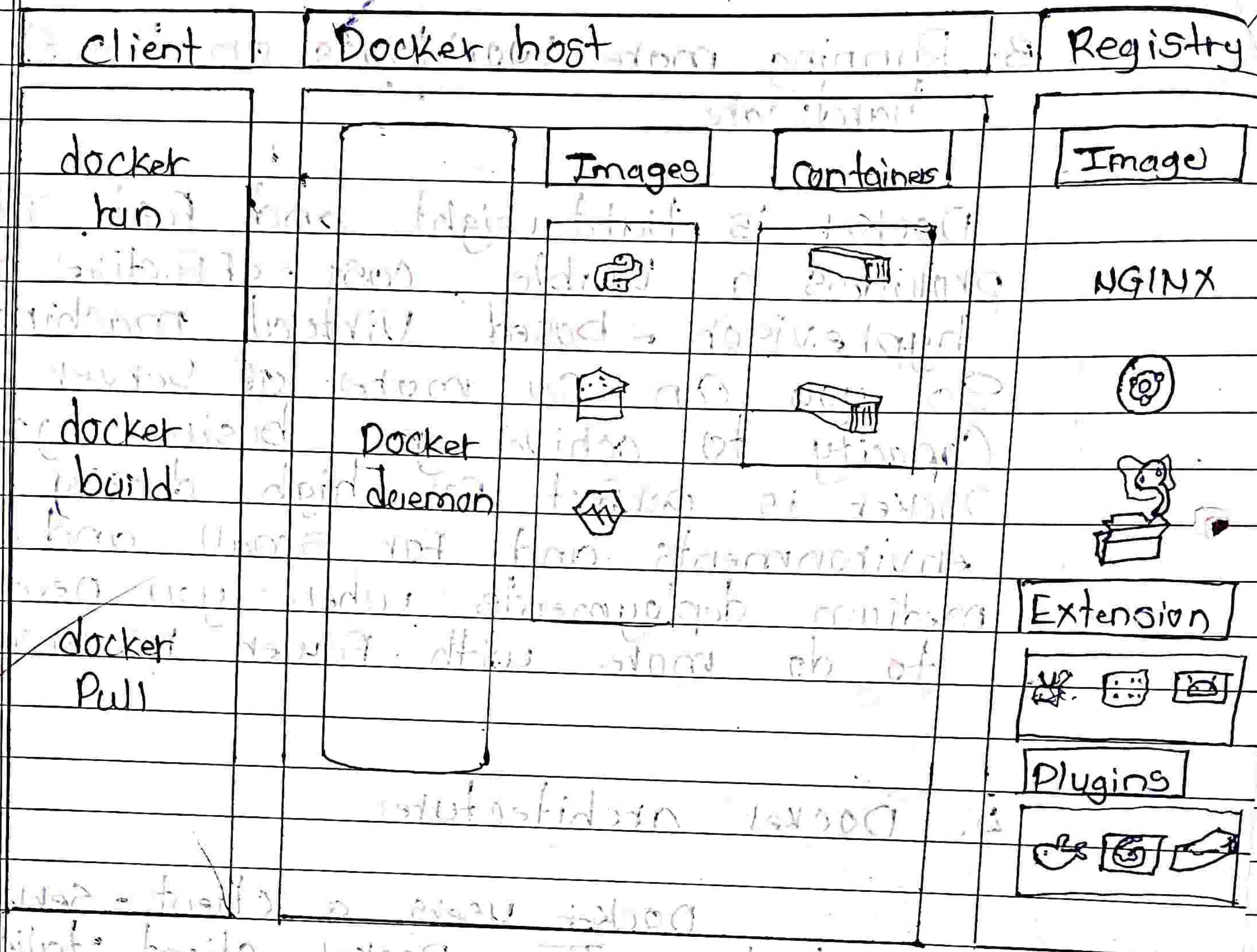
Dockeres Container based platform allow For highly portable workloads. Docker Containers Can run on a developer's local laptop, on physical or virtual machines in a data centre, on cloud providers, or in a mixture of environment.

3. Running more workloads on the Same hardware

Docker is lightweight and Fast. It provides a Viable, cost-effective to hyprevisor - based Virtual machine, So you Can use more of Server Capacity to achive your business goals. Docker is perfect for high density environments and For small and medium deployments where you need to do more with Fewer resource

4. Docker architecture

Docker users a client-server architecture. The Docker client talk to the Docker daemon, which does the heavey lifting of building, running and distrabuting your docker Container. The docker Client and daemon Can run on the same System, or you Can Connect a docker client to a remote docker daemon. The docker Client and daemon Communicate using a REST API, over UNIX Socket or a Network interface.



5. The Docker daemon

The Docker daemon listen for docker API Request and manage Docker object Such as images, containers, networks and volums. A daemon can also Communicate with other daemon to manage Docker Services.

6. The Docker client

The Docker Client is the primary way that many Docker users interact with Docker. When you use Command Such as docker run, the client sent these commands to dockered, which Carries them out.

7. Docker DeskTOP

Docker DeskTOP is an easy-to-install application for Mac, Window or linux environment that enable you to build and share Containerized application and microservices. Docker Desktop include the docker daemon, the docker client, Docker compose.

8. Docker registries

A Docker registry Stores Docker image. Docker Hub is a public registry that anyone Can use and Docker look For image on Docker Hub by default. You Can even run your own private registry.

9. Docker objects

When you use Docker, you are creating and using image, containers, network, volume, pluging and other object. This section is a brief overview of some of those objects.

10. Image

An image is a read-only template with instruction for creating a docker container. often an image is based on another image, with some additional customization. for example you may build an image which is based on ~~another image~~ ubuntu image. but install the Apche web server and your application, as well as the configuration details needed to make your application run.

You might create your own image or you might only use those created by other and published in a registry.

11. Containers:

A Container is a runnable instance OF an image. You can create, start. Stop, move or delect a container using the Docker API and CLI. You can connect a Container to one or more networks, attach strage to it, or even create a new image based on its current State.

By default, a container is relatively well isolated From other container and its host machines. You can control how isolate a container's network, storage or other underlying Subsystem are From other container or From the host machine.

Execution Steps :-

1. Install Docker - IF you haven't already, install docker on your system. You can Find installation instruction For your Specific os on the Docker websites.

2. Verify NVIDIA Driver -

Ensure that you have the NVIDIA GPU driver installed on your System. You can check the driver Version using the following Commad :

bash

```
• nvidia-smi
```

- Pull a Docker Image :-

Choose a Docker image that include the necessary tools and libaries for your Specific use case. NVIDIA provides official CUDA image that are GPU-enabled.

bash :

```
• docker pull nvidia/cuda:11.0-base
```

- Create a DockerFile -

IF you need to Cutomize the docker image, create a DockerFile in your project directory.

DockerFile :

```
# Use an official CUDA runtime as a
  parent image
From nvidia/cuda:11.0-base
# Install additional Package
RUN apt-get update && apt-get install -y \
  Package 1
  Package 2
```

```
# Set the Working directory.
  WORKDIR /app

# Copy your application code into the
    Container
  COPY . / app

# Specify the Command to run on
  Container startup
  CMD ["/bin / bash"]
```

- Build the Docker Image - IF you created a custom DockerFile, build the ~~bo~~ docker using the Following command, replacing my_Custom_image with your ~~desired~~ image name.

bash

- docker build -t my_custom_image .

- Run a Docker Container - Start a Docker Container From the image you pulled or build. You can use the nvidia-docker Command to enable GPU access within the container.

bash

- docker run --gpus all -it my-custom -image

Replace my_custom_image with the name of your Docker image.

- Access the Container -
You be inside the running container with the access to the GPU. You can execute command, run GPU-accelerated application, or perform any other task as needed.

- Exit the Container -
To exit the container and return to your host System, use the exit command

- Cleanup :- When your done with the Container you can stop and remove it using the Following Commands

bash

9. docker stop <container-id-or-name>
10. docker rm <container_id-or_name>
11. Replace <container-id-or-name> with the actual container ID or name

Conclusion / Analysis :-

Hence, We have created a Github account, created a new repository, intialized a local Git repository and pushed your code to Github. You can Continue to use Git Command to manage your Code, make Changes, and Collaborate with others on Github.