Docker -Yml file :-

A Docker Compose file is a YAML configuration file used to define and run multi-container Docker applications. It allows you to specify the services, networks, and volumes for your application's containers, along with their configurations and dependencies. Docker Compose simplifies the process of managing complex applications by providing a single file to define the entire application stack, making it easier to deploy and maintain containerized applications across different environments.

How it is connect:

yml file is most commanly used for to deploy multi container application. Multi container that mean you have frontend, backend,db and so on other thing in you application using dockerfile you deploy only one but using docker compose file you can deploy whole application.

In order to do that first you have know about what is version, service, netwok, volume

* Version: Specifies the version of the Docker Compose file syntax being used. This helps ensure compatibility with the Docker Compose tool and its features.
* Services: Defines the various containers that make up the application. Each service represents a containerized component of the application, such as a web server, a database, or an API.
* Service Configuration: For each service, you specify its configuration settings, including:
* Image: The Docker image to use for the service.
* Ports: Port mappings to expose the service externally, if needed.
* Volumes: Mount points for persistent data storage or sharing files between the host and container.
* Environment Variables: Environment variables passed to the container.
* Dependencies: Dependencies on other services, ensuring they start in the correct order.
* Healthchecks: Configuration for checking the health of the service.
* Other settings: Such as restart policies, resource constraints, etc.
* Networks: Optionally, you can define custom networks to isolate services or control network communication between containers.
* Volumes: Optionally, you can define named volumes to manage persistent data storage independently of container lifecycle.

here is one sample example for a application which have frontend, backend and db

version: '3.8' # Version of Docker Compose syntax

services: # Definition of services in the application

web: # Service named 'web'

image: nginx:latest # Docker image to use for this service

ports:

- "8080:80" # Port mapping, host:container

volumes:

- ./html:/usr/share/nginx/html # Mounting a volume

db: # Service named 'db'

image: mysql:5.7 # Docker image for MySQL 5.7

environment:

MYSQL\_ROOT\_PASSWORD: example # Environment variable for MySQL root password

app: # Service named 'app'

build: ./app/Dockerfile # Build context for Dockerfile

ports:

- "3000:3000" # Port mapping for a Node.js application

depends\_on:

- db # Dependence on the 'db' service

In this yml file I have 3 service

1. web service : we are pulling the image from docker hub and running in the port 8080 and doing volume mounting.

2. db: in db we are using mysql 5.7 defining the enviroment and passing our password to login into it.

3. app: this will be backend in this we are building the image and running into 3000 port and it is depend on the db

In GhostQA we have 2 yml files:-

1.deploy.yml

2.docker-compose.yml

How to use it:

In order to use compose file first we have open the cmd and we have change our dicrectory where the yml file is located and then we have to run the "docker compose up" command it will start the containers.