Day 10

Name : DHANAPAL

Date :08/08/2024

1. Mention the actions of the following commands:

1 .git remote add origin "https://github.com/dhanapalmayil/demo1.git":

This command sets the remote repository for your local Git repository. It tells Git where to push your commits. The URL should point to the repository you want to connect to.

`2.git pull origin master`

This command fetches and merges changes from the `master` branch of the remote repository named `origin` into your current branch.

`3.git push origin dev`

This command pushes your local `dev` branch to the remote repository named `origin`.

2. What are the functions of the following Docker objects and key components:

- **Dockerd**:
  - `Dockerd` is the Docker daemon. It listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. It can also communicate with other daemons to manage Docker services.

- **Dockerfile**:
  - A `Dockerfile` is a script containing a series of instructions on how to build a Docker image. Each instruction creates a layer in the image. Common instructions include `FROM`, `RUN`, `COPY`, and `CMD`.
- **Docker-compose.yaml**:
  - The `docker-compose.yaml` file defines services, networks, and volumes for a multi-container Docker application. It allows you to manage multiple containers as a single service. You can define how the containers interact and configure each container's environment.
- **Docker Registries**:
  - Docker registries are repositories for Docker images. They allow you to store and distribute Docker images. Docker Hub is the default public registry, but you can also use private registries.
- **DockerHost**:
  - DockerHost refers to the machine where Docker is installed and running. It can be a local machine, a virtual machine, or a cloud server. It hosts Docker daemons and Docker containers.

3. What is isolation in Docker containers?

Isolation in Docker containers refers to the encapsulation of applications and their dependencies within containers, which run as isolated processes in user space on the host operating system. This isolation ensures that containers do not interfere with each other or the host system. It provides several benefits:

- **Process Isolation**: Each container runs as a separate process with its own file system, network interface, and process space, isolated from other containers and the host system.
- **Resource Control**: Docker uses cgroups (control groups) to limit and prioritize resources (CPU, memory, disk I/O) for each container.
- **Namespace Isolation**: Docker uses namespaces to provide containers with their own view of the system, including process IDs (PID namespace), network (net namespace), user IDs (user namespace), and file systems (mount namespace).

Docker Examples:

1.Creating own image for python application

1. Create a Python script (app.py).
2. Create a Dockerfile to define the image.
3. Build the Docker image with docker build.
4. Run the Docker container with docker run.

1.app.py

```
# app.py

def hello_world():
    return "Hello, World!"

if __name__ == "__main__":
    print(hello_world())
```

2. Dockerfile(Image)

```
# Use an official Python runtime as a parent image

FROM python:3.8-slim



# Set the working directory in the container

WORKDIR /usr/src/app



# Copy the current directory contents into the container at /usr/src/app

COPY . .



# Run the application

CMD ["python", "./app.py"]
```

3.Build image for the application

docker build -t docker .

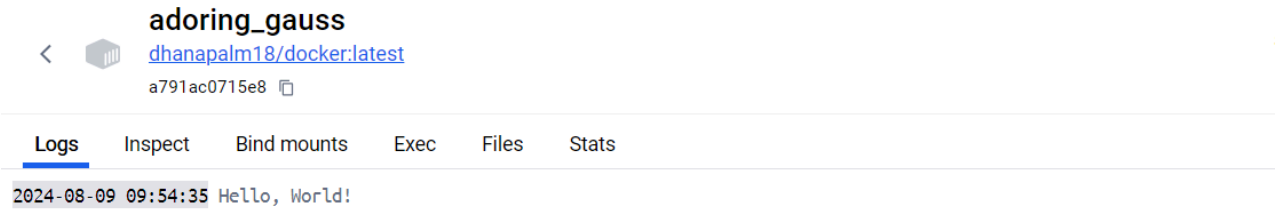
```
D:\Python\docker>docker build -t docker .
[+] Building 0.2s (8/8) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 322B                                                                                0.0s
 => [internal] load metadata for docker.io/library/python:3.8-slim                                                  0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [1/3] FROM docker.io/library/python:3.8-slim                                                                    0.0s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 473B                                                                                   0.0s
 => CACHED [2/3] WORKDIR /usr/src/app                                                                               0.0s
 => [3/3] COPY . .                                                                                                  0.0s
 => exporting to image                                                                                              0.0s
 => => exporting layers                                                                                             0.0s
 => => writing image sha256:e45ff6d35577d55807b39cab3b13f2d6cdab4ce39bd1a59370e5ed9bcba498e7                        0.0s
 => => naming to docker.io/library/docker                                                                           0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/9b6h9lyrr6xjxq9m15e6rhn6a

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

4. Run docker image using command or docker desktop

```
D:\Python\docker>docker run docker
Hello, World!
```



5. Push the created image into docker hub.

```
D:\Python\docker>docker push dhanapalm18/docker
Using default tag: latest
The push refers to repository [docker.io/dhanapalm18/docker]
baa3821afe63: Pushed
46178716d2b0: Pushed
0cfc6ead4554: Pushed
1109f5b27710: Mounted from library/python
cad3599a3016: Pushed
e7817ba7b646: Pushed
e0781bc8667f: Pushed
latest: digest: sha256:1f4d2181705e11aa903589ed6e2ef120f82cf87881b5c79a671d8fa56d708060 size: 1784
```