Docker installation

1)Installation

sudo apt update
sudo apt install docker.io

Explanation:Output

The command sudo apt install docker.io is used to install Docker on a Debian-based Linux distribution, such as Ubuntu. Here's the breakdown of each part of the command:
a)sudo: This is used to run the command with superuser (root) privileges. It allows you to perform administrative tasks and access system-related information.
b)apt: This is the package management tool used in Debian-based Linux distributions to install, upgrade, and manage software packages.
c)install: This is a command used with apt to indicate that you want to install a package.
d)docker.io: This is the name of the package you're installing. In Debian-based systems, the Docker package is named docker.io.
When you run sudo apt install docker.io, the command will download and install the Docker package from the official repositories of your Linux distribution. This will allow you to use Docker to create, manage, and run containers on your system.
After the installation is complete, you'll typically need to start and enable the Docker service using systemctl.

2)Start and enable the Docker service:

sudo systemctl start docker
sudo systemctl enable docker

Explanation:

The command sudo systemctl start docker is used to start the Docker service on a Linux system. Here's what each part of the command means:
a)sudo: This is used to run the command with superuser (root) privileges. It allows you to perform administrative tasks and access system-related information.
b)systemctl: This is a command-line tool used to manage and control systemd services and units on a Linux system. systemd is an initialization system and service manager that handles the startup process and management of various system services.
c)start: This is a subcommand of systemctl used to start a specified service or unit.
d)docker: This is the name of the service you're starting. In this case, it refers to the Docker service, which manages Docker containers on your system.

3)Checking docker status

sudo systemctl status docker

The command sudo systemctl status docker is used to display the status and information about the Docker service on a Linux system. Here's what each part of the command means:
a)sudo: This is used to run the command with superuser (root) privileges. It allows you to perform administrative tasks and access system-related information.
b)systemctl: This is a command-line tool used to manage and control systemd services and units on a Linux system. systemd is an initialization system and service manager that handles the startup process and management of various system services.
c)status: This is a subcommand of systemctl used to request information about the current status of a specified service or unit.
d)docker: This is the name of the service you're checking the status of. In this case, it refers to the Docker service, which manages Docker containers on your system.

4)Listing docker images and container

sudo docker images
sudo docker ps -a

Explanation:

The command sudo docker ps -a is used to list all Docker containers, including both running and stopped containers, on a Linux system. Here's what each part of the command means:
a)sudo: This is used to run the command with superuser (root) privileges. It allows you to perform administrative tasks and access system-related information.
b)docker: This is the command-line tool used to interact with Docker. It provides a set of commands for managing Docker containers and images.
c)ps: This is a subcommand of the docker command used to list Docker containers. The ps subcommand stands for "process status."
d)-a: This is a flag that modifies the behavior of the docker ps command. When used with the -a flag, the command will list all containers, both running and stopped.
When you run sudo docker ps -a, the command will display a list of all Docker containers that have been created on your system, regardless of their current status. This includes containers that are currently running as well as containers that have stopped.

5)Ubuntu in Docker

sudo docker pull ubuntu

The command docker pull ubuntu is used to download (pull) the official Ubuntu Docker image from Docker Hub. Let's break down the components and the meaning of this command:

a)docker: This is the command-line tool for interacting with Docker, which allows you to manage containers, images, networks, and more.
b)pull: This is a subcommand of the Docker CLI used to pull (download) Docker images from a registry.
c)ubuntu: This is the name of the Docker image you want to pull. In this case, it refers to the official Ubuntu image available on Docker Hub.

6)Writing docker file

Open any editor(vscode,notepad) and create file without extension.

hello.py

```
print("Welcome to  Docker")
```

Dockerfile

```
# Use the official Ubuntu  image as the base
FROM ubuntu
# Update package lists and install Python
RUN apt-get update && apt-get install -y python3
# Copy the Python script file from your host into the container
COPY hello.py /app/
# Set an environment variable (not used in this example)
ENV MY_ENV_VAR=my_value
# Specify the Python script as the default command
CMD ["python3", "/app/hello.py"]
```

Explanation:

a)FROM ubuntu
This instruction specifies that you're using the official Ubuntu image as the base image for your Docker image. Without specifying a tag (like 20.04), it will default to the "latest" tag available on Docker Hub.
b)RUN apt-get update && apt-get install -y python3
This instruction updates the package lists within the Ubuntu image and then installs Python 3 using the apt-get package manager. The -y flag automatically answers "yes" to any prompts during the installation.
c)COPY hello.py /app/
This instruction copies a file named hello.py from your host machine (the directory where the Dockerfile is located) into the /app/ directory within the container.
d)ENV MY_ENV_VAR=my_value
This instruction sets an environment variable named MY_ENV_VAR to the value my_value. This environment variable will be available within the container.
e)CMD ["python3", "/app/hello.py"]

This instruction specifies the default command to run when a container based on this image is started. In this case, it's executing the hello.py Python script using the python3 interpreter.

In summary, this Dockerfile builds an image based on the official Ubuntu image, installs Python 3, copies a Python script (hello.py) into the container, sets an environment variable, and then specifies the Python script as the default command to run when a container is launched from the image.

7)Build the Docker Image

sudo docker build -t my-python-image .

Explanation:

a)docker: This is the command-line tool for interacting with Docker, which allows you to manage containers and images.
b)build: This is a subcommand of the Docker CLI used to build Docker images.
c)-t my-python-image: The -t flag allows you to tag the image you're building with a specific name. In this case, you're tagging the image with the name my-python-image.
d). This represents the build context, which is the directory containing the Dockerfile and any other files you want to include in the image. In this case, . refers to the current directory where you're executing the docker build command.

8)sudo docker run -it my-python-image

The command docker run -it my-python-image is used to create and start a new Docker container based on the my-python-image image, and it opens an interactive terminal session (-it) within the container.
When you use -it together in the docker run command, you're indicating that you want to run the container in interactive mode with a terminal. This is commonly used when you want to interact with the container's command line, run shell commands, or work with applications that require a terminal-based interface.

Output: Welcome to Docker

9)Reflecting changes in container

hello.py

```
print("Welcome to Docker1")
print("Welcome to  Docker2")
```

sudo docker build -t my-python-image:adding .

The command you provided (sudo docker build -t my-python-image:adding .) is used to build a Docker image from the Dockerfile in the current directory. The image will be tagged with the name my-python-image and the tag adding.

sudo docker run -it my-python-image:adding
Output: Welcome to Docker
OutputWelcome to Docker

10)Removing docker images
sudo docker rmi -f image_id(-f force remove signal)

11)Deleting multiple docker images
sudo docker rmi -f image_id image_iu image_id

12)Stop the container
sudo docker stop container_id

13)If the Django server is running inside the Docker container and bound to port 8000
sudo docker run -it -p 8000:8000 project
a)sudo: This runs the command with superuser (root) privileges.
b)docker run: This command is used to create and start a new container from an image.
c)-it: These flags stand for interactive mode with a pseudo-TTY. It allows you to interact with the container's command line.
d)-p 8000:8000: This flag maps port 8000 on the host machine to port 8000 in the container. This is used for accessing services running inside the container from your host machine's web browser.
d)project: This is the name of the image you want to create a container from.
The command you've provided is using Docker to run a container from an image named project with the -it flag for interactive mode and the -p flag to map port 8000 on the host machine to port 8000 in the container.

14)For Django Project
dockerfile
```
FROM ubuntu:20.04
RUN apt-get update && apt-get install -y python3 python3-pip
COPY . .
RUN pip install -r requirements.txt
ENV MY_ENV_VAR=my_value
CMD ["python3", "manage.py", "runserver", "0.0.0.0:8000"]
```

Explanation

a)COPY . .

The command COPY . . in a Dockerfile is used to copy all the files and directories from the current directory (where the Dockerfile is located) into the specified directory in the container. In this case, it copies the contents of the current directory into the root directory (/) of the container.

b)CMD ["python3", "manage.py", "runserver", "0.0.0.0:8000"]

The command CMD ["python3", "manage.py", "runserver", "0.0.0.0:8000"] instructs Docker to run the Django development server inside the container when the container is started from the image. This allows you to access your Django application from your host machine's browser at http://127.0.0.1:8000/.

sudo docker build -t project .

sudo docker run -it -p 8000:8000 project