**Write a docker file for java application**

🡪create a EC2 and launch

🡪install docker

🡪download sample war file->copy link as address

🡪use Wget and \*paste copy url\*

🡪then write a docker file -> vi Dockerfile and input sample war file name and path

# Use a base image with Tomcat

FROM tomcat:latest

# Remove the existing default Tomcat application

RUN rm -rf /usr/local/tomcat/webapps/\*

# Copy the static website files to the Tomcat webapps directory

COPY sample.war /usr/local/tomcat/webapps

# Expose port 8099 for accessing the Tomcat server

EXPOSE 8099

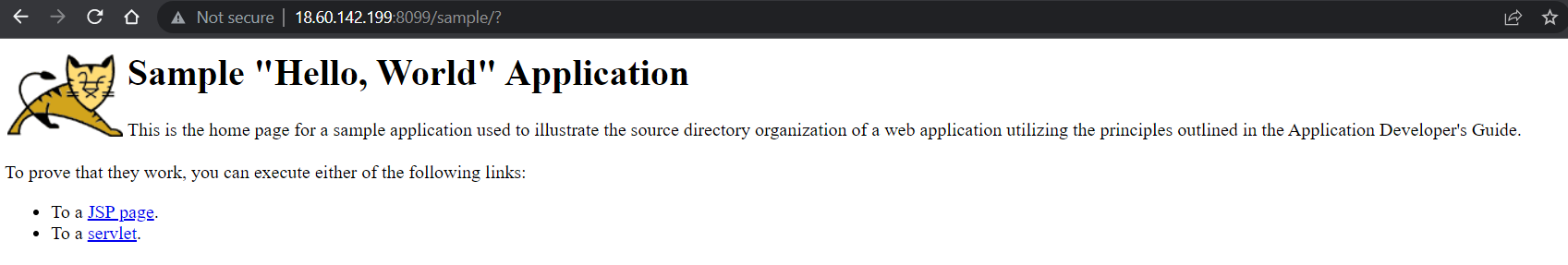
# Start Tomcat when the container starts

CMD ["catalina.sh", "run"]

🡪docker build -t my-static-website .

🡪docker run -d --name my-conta -p 8090:8090 my-static-website

🡪http://localhost:8099/sample/

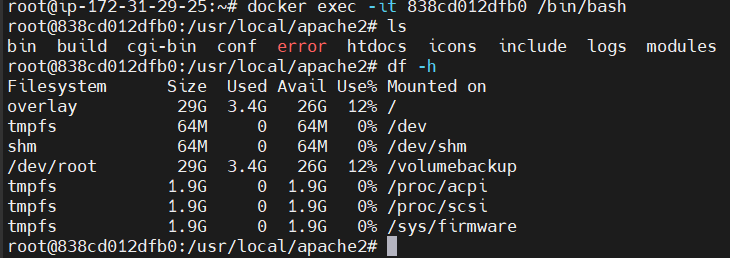
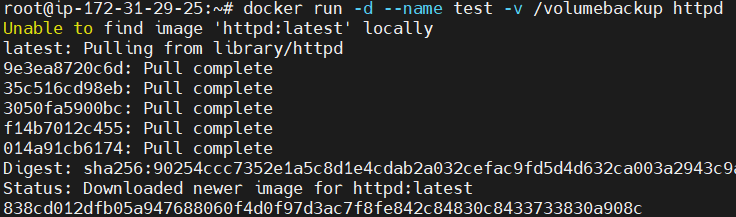


2. \***Multi-stageulti-stage Docker file**\* is a feature of Docker that allows you to define multiple stages within a single Docker file. Each stage can have its own base image and set of instructions, providing a way to build and optimize images in a more efficient and modular manner.

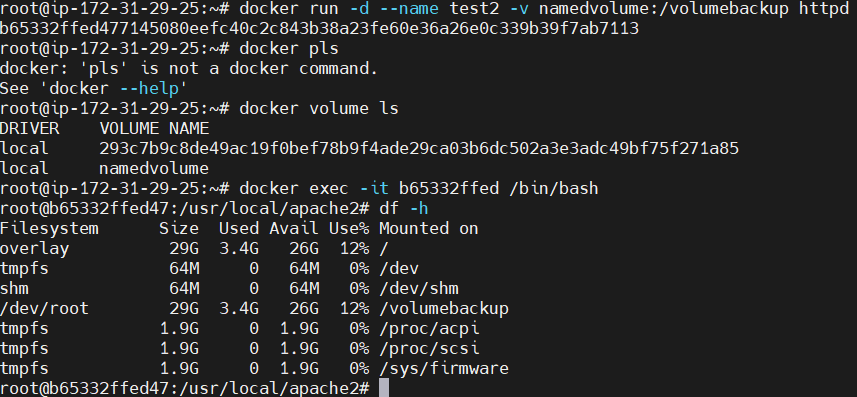
2.b.\*Alpine is a popular lightweight Linux distribution commonly used as a base image in Docker. Alpine images are significantly smaller compared to traditional Linux distributions, making them well-suited for containerization and reducing the overall image size.

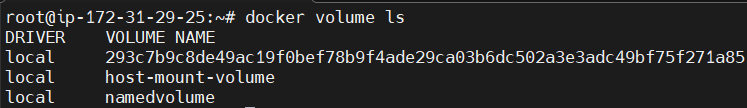
3**. Volumes of Docker**

**Anonymous volumes** Docker and are not given a specific name. They are useful when you need temporary storage or when you don't need to manage the volume explicitly. Anonymous volumes are typically used for caching or storing temporary data.



**Named volumes**: Named volumes are managed by Docker and have a specific name assigned to them. They provide an easy way to create, manage, and reference volumes in Docker. Named volumes are typically preferred for most use cases as they offer simplicity and abstraction.





**Host-mounted volumes**: you can directly mount a directory from the host machine into the container. This allows you to easily access and share files between the host and the container. Changes made on either the host or the container are immediately visible to the other.

**Docker commands:**

build image \* docker build -t my-static-website .\*

Run container \* docker run -d --name my-container -p 8090:8090 image name\*

Pull image \* docker pull \*image name\*

Login container \* docker exec -it container id /bin/bash\*

Login container and why using bin/bash terminal 🡪 to communicate with container

command to list all running container \*docker ps\*

command to list all state container \*docker ps -a\*

View container logs \*docker logs <container\_name>\*

Stop a container \*docker stop <container name>

Start a stopped container \*docker start <container name>

Pause a running container \* docker pause <container\_name>

Restart a container \* docker restart <container\_name>

Exit a container without stopping it \* Ctrl + P followed by Ctrl + Q\*

Command to rename a container \* docker rename <old\_container\_name> <new\_container\_name>

Command to free up disk space \* docker system prune -a

Command to check container resource usage \* docker stats <container\_name>

Create a network \*docker network create <network\_name>

Delete a network \*docker network rm <network name>

Command to copy a file from the local machine to a container

\* docker cp <local\_file\_path> <container\_name>:<inside container\_file\_path>\*

Command to copy a file from a container to the local machine

docker cp <container\_name>:<inside container\_file\_path> <local\_file\_path>\*

Command to create an image from a running container \* docker commit <container\_name> <new\_image\_name>

Command to create a volume \*docker volume create <volume\_name>

Command to delete a volume \*docker volume rm <volume\_name>

Command to remove unused containers and images \* docker system prune\*

**Difference between stop, pause, and kill**

docker stop: Gracefully stops a running container.

docker pause: Pauses a running container, temporarily suspending its processes.

docker kill: Sends a signal to forcefully terminate container.