Hometask 5

Part 1

You should be logged into EC2 machine.

1. Pull 22.04 ubuntu image - it will serve as a **base.**

```
docker pull ubuntu:22.04
```

2. Create(run) and name the container from the ubuntu image. Bind ports between container and the host machine (EC2 machine)

```
docker run -d --name ubuntu-nginx -p 80:80 -p 443:443 ubunt
u:latest --restart=unless-stopped /bin/sh -c "while true; d
o sleep 1000; done"
```

Note: -c command is necessary.

—restart=unless-stopped or —restart=always

Makes sure that container is automatically recreated, even when host machine goes down.

Run sudo reboot and verify it works.

If you made some mistake, and you want to re-run the container, and you encounter errors, like “container name is already in use”, then just issue:

```
docker stop ubuntu-nginx
docker rm ubuntu-nginx
```

And recreate the container from image using **docker-run** command

Check if ubuntu container is running

If you run the container with **1.1** command, then the container is running in the background (-d flag tells us that).

To see the status of the containers we have running on our EC2 machine, then we need to check:

```
docker ps
```

In my case, this is showing the table, with one container running:

```
💡 docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED
STATUS          PORTS                    NAMES
2fcdc0b428f9   ubuntu:22.04   "/bin/sh -c 'while t..."   2 seconds ago
Up 2 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp   ubuntu-nginx
```

Install things inside the running container

We have running Ubuntu base container. To complete the task, and enable nginx service inside the container, we first need to install it inside the container. Just like we would do it inside our laptop running Ubuntu. Of course, we could use base image with **nginx already installed,** and we will do just like this in the future.

For now, let's:

```
docker exec -it ubuntu-nginx bash
```

This will exec an interactive shell inside a container (we take control over the container and can run commands inside)

Install nginx inside the container

Checkpoint 1. Attach screenshot of running nginx service. Use this commands as systemctl is not available in docker

```
service nginx start
service nginx status
```

Let's commit this changes into a new image ubuntu-nginx:1 See the command in practical session recording or use google.

Part 2

Modifying nginx configuration

Configuration Files

/etc/nginx/nginx.conf

- **Global Configuration**: This is the main Nginx configuration file. It includes global settings that apply to the entire server. These settings can include the user and worker processes nginx will use, logging, and basic performance tuning parameters.
- **Includes Other Files**: It often contains `include` directives to read other configuration files, such as those in the `/etc/nginx/conf.d/` directory or specific module configurations.
- **Default Server Block**: It may define a default `server` block, which is used if no other server block matches the request that nginx receives.

/etc/nginx/conf.d/default.conf

- **Virtual Hosts/Servers**: Files in `/etc/nginx/conf.d/` typically define server blocks (virtual hosts) that determine how different domains or request URIs are handled.
- **Specific Configuration**: Each `.conf` file in this directory can be used to configure specific sites or applications, making it easier to manage complex setups with multiple domains or apps.
- **Overrides and Specificity**: Settings in `conf.d/default.conf` are specific to the server block(s) defined within and can override global settings for requests that match these blocks.

Exposing nginx on port 443 (We will NEED a certificate)

Run the committed image from part 1 with some modifications:

1. use port 443 and 80 flags

2. override the command -c with this one: 'nginx -g 'daemon off;' -c /etc/nginx/nginx.conf'

Run interactive terminal like in part 1, do the required configuration:

For HTTPS to work, you need an SSL certificate. For testing, you can create a self-signed SSL certificate:

While inside a container:

```
mkdir /etc/nginx/ssl
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/nginx/ssl/nginx.key -out /etc/nginx/ssl/nginx.crt
```

You will be presented a series of prompts for creating OpenSSL certificate.

or a production environment, obtain a certificate from a trusted CA. (Certificate Authority)

2. **Configure nginx for HTTP and HTTPS**

Edit the default nginx site configuration:

```
vi or nano /etc/nginx/sites-available/default
```

Adjust the file to listen on both ports 80 and 443 and use the SSL certificate. Replace the existing server block with the following:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;
    ssl_certificate /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key /etc/nginx/ssl/nginx.key;

    root /var/www/html; #tells nginx where to look for file
s to send back to client
    index index.html index.htm index.nginx-debian.html; #ng
inx will be looking for these files in root directory

    server_name _;
```

```
    location / { #if the request ends in /, then make
        try_files $uri $uri/ =404; # look for file name ($u
ri) or catalog with name ($uri/) and return this file
    }
}
```

Set up “301” redirect 80 to 443 block.

Do docker commit with ubuntu-nginx:2. Run this newly created image, don’t specify any commands.

Checkpoint 2: Attach screenshot of docker ps command.

Checkpoint 3: Attach screenshot of nginx status command from inside docker.

Checkpoint 4: attach public ip of your instance, I need to be able to see the default nginx page on port 443 and get redirected to 443 when trying to access by port 80.

REFERENCE

Running Docker Containers

Basic Commands

Check Docker Status: To see if Docker is running on your EC2 instance:

```
sudo systemctl status docker
```

Run a Container in Interactive Mode (Foreground):

```
docker run -it --rm ubuntu #interactive mode, --rm means "c
lean up after exiting"
docker run -d --name printer ubuntu /bin/sh -c "while true;
do echo 'hello'; sleep 1; done" -> test command
```

Run a Container in Detached Mode (Background):

```
docker run -d ubuntu # -d means detached - in the background
```

This runs the container in the background.

Naming Containers:

```
docker run -d --name printer ubuntu /bin/sh -c "while true;
do echo 'hello'; sleep 1; done"
```

Flags reference:

```
-it: Runs the container in interactive mode with a terminal.
--rm: Automatically removes the container once it's stopped.
-d: Runs the container in detached mode, allowing it to run i
```

Checking Logs

Logi:

```
docker logs -f printer
```

The `-f` flag follows the log output in real-time.

Inspecting Containers

```
docker inspect <container-name>
```

Shows detailed information about the container in JSON format.
We can provide additional filters. Here we can check the container's IP address in Docker network:

```
docker inspect --format '{{.NetworkSettings.IPAddress}}' prin
```

Executing commands in running containers

Allows you to execute additional commands inside a running container.

```
docker exec -it <container-name> /bin/bash
```

Managing Docker Images and Containers

Remove Unused Docker Images:

```
docker image prune
```

Removes dangling images that are not tagged and not referenced by any container.

Stop a Container

```
docker stop <container-name>
```

Sends a TERM signal, allowing the container to clean up.

Clean Up System

```
docker system prune -a
```

Removes all unused containers, networks, images (both dangling and unreferenced), and optionally, volumes.

It should be used with caution, as it removes all images, build cache and volumes.
System often gets trashed with a lot of unused images, so it’s a good “wipe everything out” command.

Working with Volumes

Volumes are used for persisting data generated by and used by Docker containers.

They are a link point between the Docker container and the machine that hosts it.
Basic volume:

Create a Test File

```
echo "hello world" > /tmp/test.txt
```

This creates the test file in /tmp directory on the host machine. The file and all contents of /tmp directory will be shared between host machine and Docker container.

Mount Volume in Read-Only Mode:

```
docker run -it -v /tmp:/var/www:ro ubuntu # read only
```

- Mounts `/tmp` from the host to `/var/www` inside the container as read-only.

Mount Volume in Read-Write Mode:

```
docker run -it -v /tmp:/var/www:rw ubuntu
```