Writing a docker file

First we have to know about what are arguments or the instructions used in a docker file.

The Dockerfile supports the following instructions : -

Instruction	**Description**
ADD	Add local or remote files and directories.
ARG	Use build-time variables.
CMD	Specify default commands.
COPY	Copy files and directories.
ENTRYPOINT	Specify default executable.
ENV	Set environment variables.
EXPOSE	Describe which ports your application is listening on.
FROM	Create a new build stage from a base image.
HEALTHCHECK	Check a container's health on startup.
LABEL	Add metadata to an image.
MAINTAINER	Specify the author of an image.
ONBUILD	Specify instructions for when the image is used in a build.
RUN	Execute build commands.
SHELL	Set the default shell of an image.
STOPSIGNAL	Specify the system call signal for exiting a container.
USER	Set user and group ID.
VOLUME	Create volume mounts.
WORKDIR	Change working directory.

Example :- 1)Nginx , 2)Tomcat, 3)Apache(httpd)

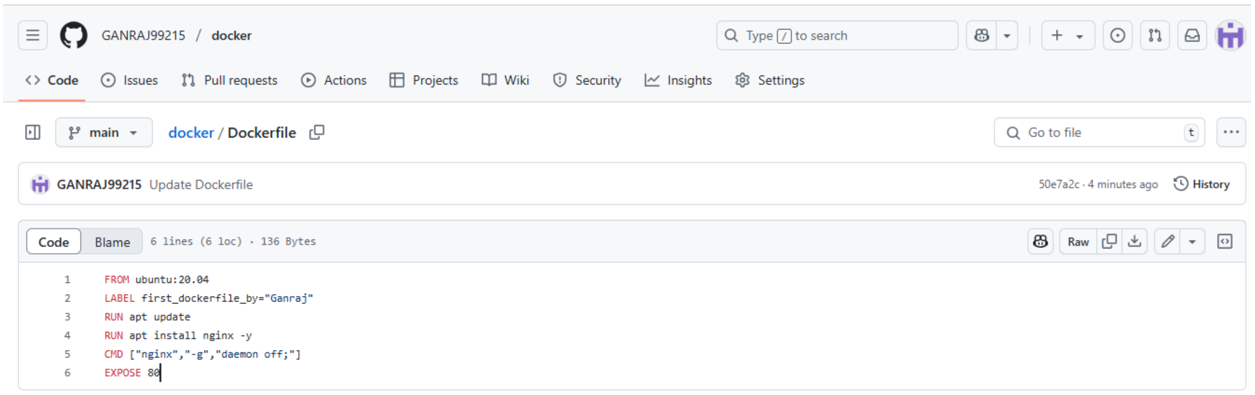
1)Nginx : -

First we will write docker file using arguments and instruction

----

```
FROM ubuntu:20.04

LABEL first_dockerfile_by="Ganraj"

RUN apt update

RUN apt install nginx -y

CMD ["nginx","-g","daemon off;"]

EXPOSE 80
```

Will save as a Dockerfile in a repo it is extension of a docker file



```
FROM ubuntu:20.04
LABEL first_dockerfile_by="Ganraj"
RUN apt update
RUN apt install nginx -y
CMD ["nginx","-g","daemon off;"]
EXPOSE 80
```

Explanation:-

FROM ubuntu:20.04

- This sets the base image for the container.
- You're starting from a clean Ubuntu 20.04 operating system (no packages, no services).

Think of it as:
"Give me a blank Ubuntu machine to build on."

`LABEL first_dockerfile_by="Ganraj"`

- Adds metadata to the image.
- Helps identify who created the Dockerfile.

Not required, but useful for documentation.

`RUN apt update`

- Updates the list of available packages from Ubuntu's servers.
- It's like running this on a real Linux machine:

```
bash

sudo apt update
```

`RUN apt install nginx -y`

- Installs the NGINX web server in the container.
- `-y` auto-confirms the installation.

Adds the program that will serve your web content.

`CMD ["nginx","-g","daemon off;"]`

- CMD tells Docker what command to run when the container starts.
- This runs NGINX in the foreground (so the container stays alive).

```
nginx -g 'daemon off;' =
```

"Start nginx and don't run it in background mode."

If you don’t use `daemon off`, the container exits after starting nginx because there's no foreground process.

```
EXPOSE 80
```

- Tells Docker that your app **listens on port 80** (default HTTP port).
- It's a **documentation hint**, not an actual port mapping.

To make it accessible, you still need to run:

```bash
docker run -p 8080:80 <image-name>
```

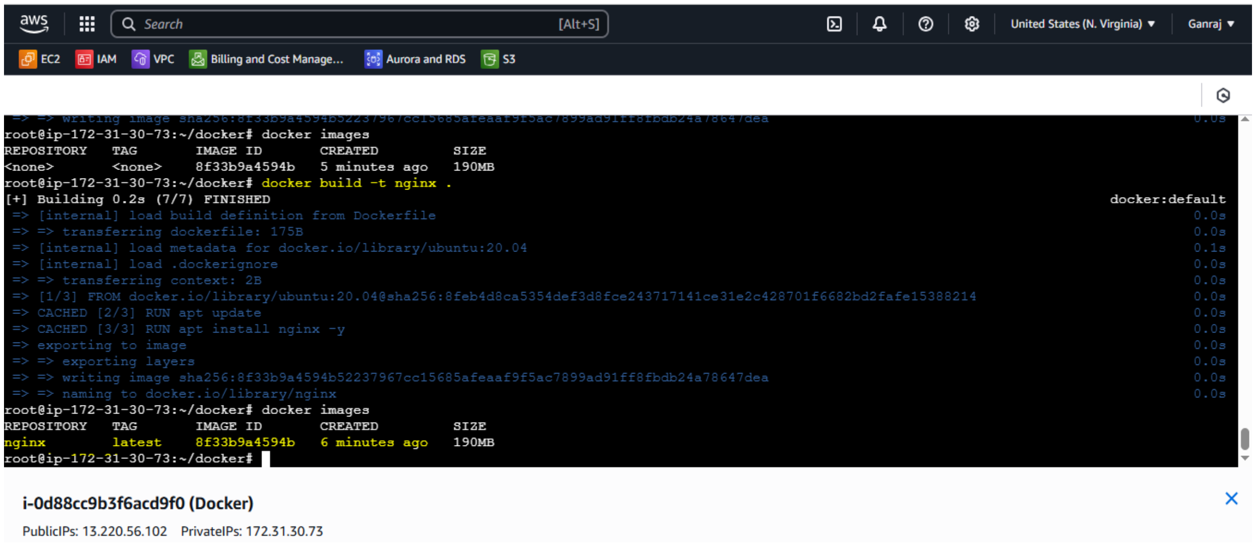
Instruction	Meaning
`FROM`	Start from Ubuntu OS
`LABEL`	Add creator info
`RUN apt update`	Get latest package list
`RUN apt install nginx -y`	Install nginx web server
`CMD`	Start nginx and keep it running
`EXPOSE`	Let Docker know we'll use port 80

How to Use This Dockerfile

```bash
docker build -t ganraj-nginx .
docker run -p 8080:80 ganraj-nginx
```

Then visit:
http://localhost:8080

Using docker build -t nginx . we are creating image from a dockerfile and -t mentions image name



```
=> => writing image sha256:8f33b9a4594b52237967cc15685afeaaf9f5ac7899ad91ff8fbdb24a78647dea                  0.0s
root@ip-172-31-30-73:~/docker# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
<none>       <none>    8f33b9a4594b   5 minutes ago   190MB
root@ip-172-31-30-73:~/docker# docker build -t nginx .
[+] Building 0.2s (7/7) FINISHED                                                                      docker:default
 => [internal] load build definition from Dockerfile                                                           0.0s
 => => transferring dockerfile: 175B                                                                           0.0s
 => [internal] load metadata for docker.io/library/ubuntu:20.04                                                0.1s
 => [internal] load .dockerignore                                                                              0.0s
 => => transferring context: 2B                                                                                0.0s
 => [1/3] FROM docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c428701f6682bd2fafe15388214   0.0s
 => CACHED [2/3] RUN apt update                                                                                0.0s
 => CACHED [3/3] RUN apt install nginx -y                                                                      0.0s
 => exporting to image                                                                                         0.0s
 => => exporting layers                                                                                        0.0s
 => => writing image sha256:8f33b9a4594b52237967cc15685afeaaf9f5ac7899ad91ff8fbdb24a78647dea                  0.0s
 => => naming to docker.io/library/nginx                                                                       0.0s
root@ip-172-31-30-73:~/docker# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
nginx        latest    8f33b9a4594b   6 minutes ago   190MB
root@ip-172-31-30-73:~/docker#
```

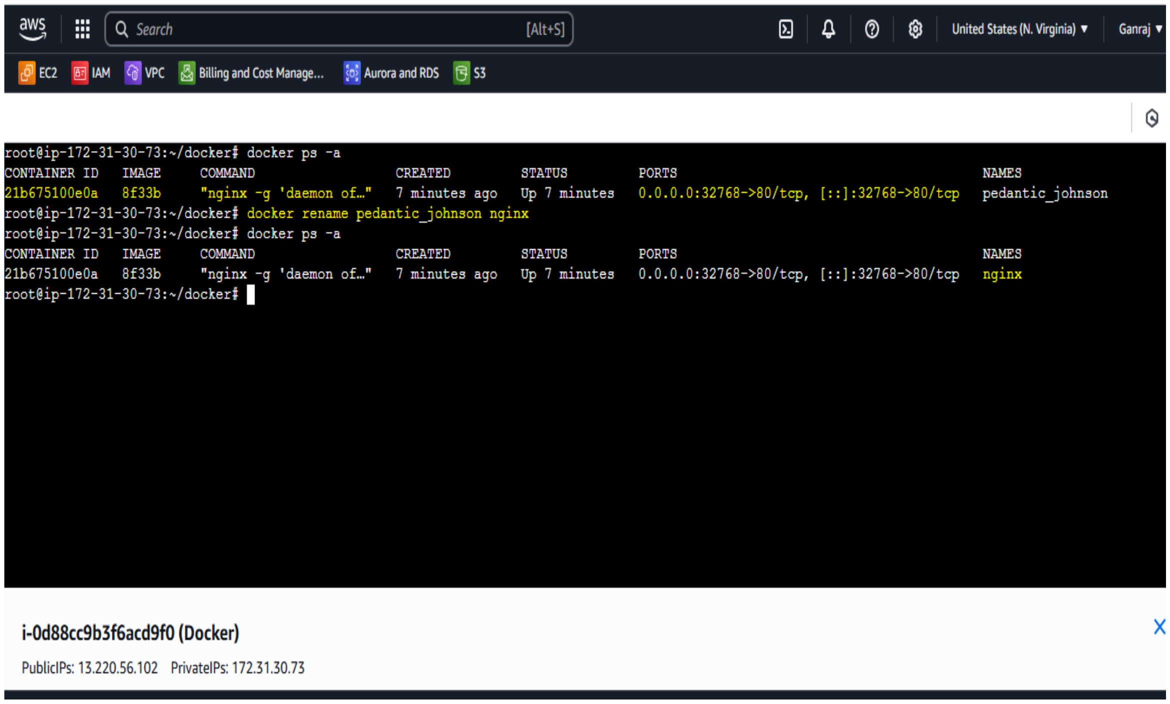
i-0d88cc9b3f6acd9f0 (Docker)

PublicIPs: 13.220.56.102 PrivateIPs: 172.31.30.73

Now we will create a docker container using our image file

docker run –d –P 8f33b

using docker rename oldname newname **u can change container name also**



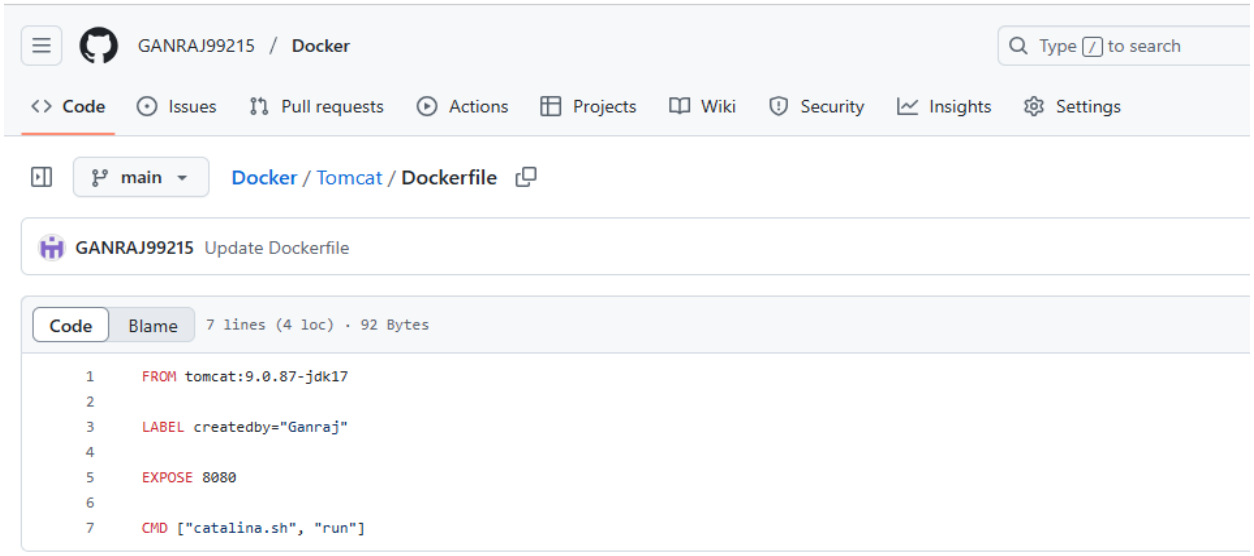
```
root@ip-172-31-30-73:~/docker# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                                     NAMES
21b675100e0a   8f33b     "nginx -g 'daemon of…"   7 minutes ago   Up 7 minutes   0.0.0.0:32768->80/tcp, [::]:32768->80/tcp   pedantic_johnson
root@ip-172-31-30-73:~/docker# docker rename pedantic_johnson nginx
root@ip-172-31-30-73:~/docker# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                                     NAMES
21b675100e0a   8f33b     "nginx -g 'daemon of…"   7 minutes ago   Up 7 minutes   0.0.0.0:32768->80/tcp, [::]:32768->80/tcp   nginx
root@ip-172-31-30-73:~/docker#
```

i-0d88cc9b3f6acd9f0 (Docker)

PublicIPs: 13.220.56.102 PrivateIPs: 172.31.30.73

2)Tomcat:-

Now we will create same for tomcat



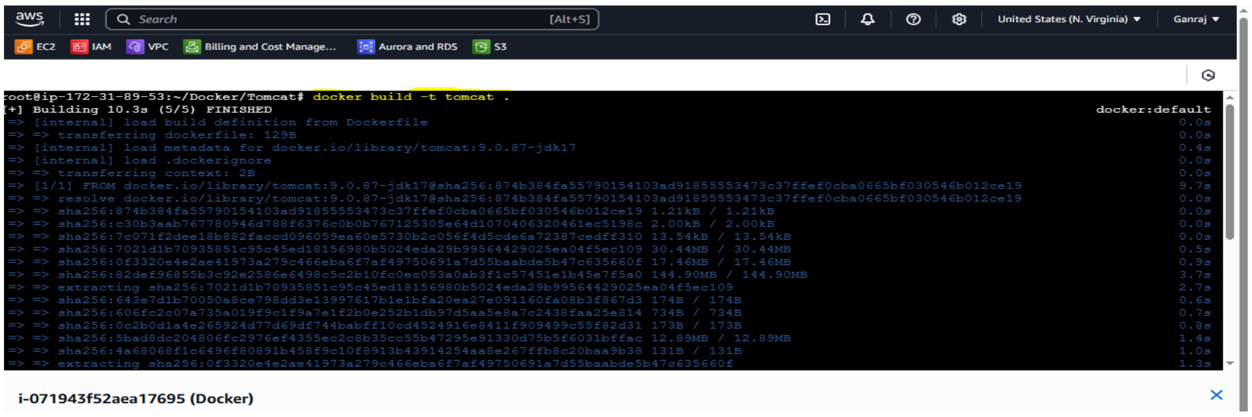
```
FROM tomcat:9.0.87-jdk17

LABEL createdby="Ganraj"

EXPOSE 8080

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

Line	**What it Does**
`FROM tomcat:9.0.87-jdk17`	Uses the official Tomcat image with Java 17 pre-installed. No need to install Java, Tomcat manually.
`LABEL createdby="Ganraj"`	Adds author metadata (optional, good practice).
`COPY myapp.war ...`	*(Optional)* If you have a WAR file, copy it into the Tomcat `webapps/` folder.
`EXPOSE 8080`	Exposes the Tomcat port so it can be accessed from outside the container.
`CMD ["catalina.sh", "run"]`	Starts the Tomcat server in foreground mode (default behavior).

---

docker build –t tomcat .

Using this command we have created one image of tomcat.



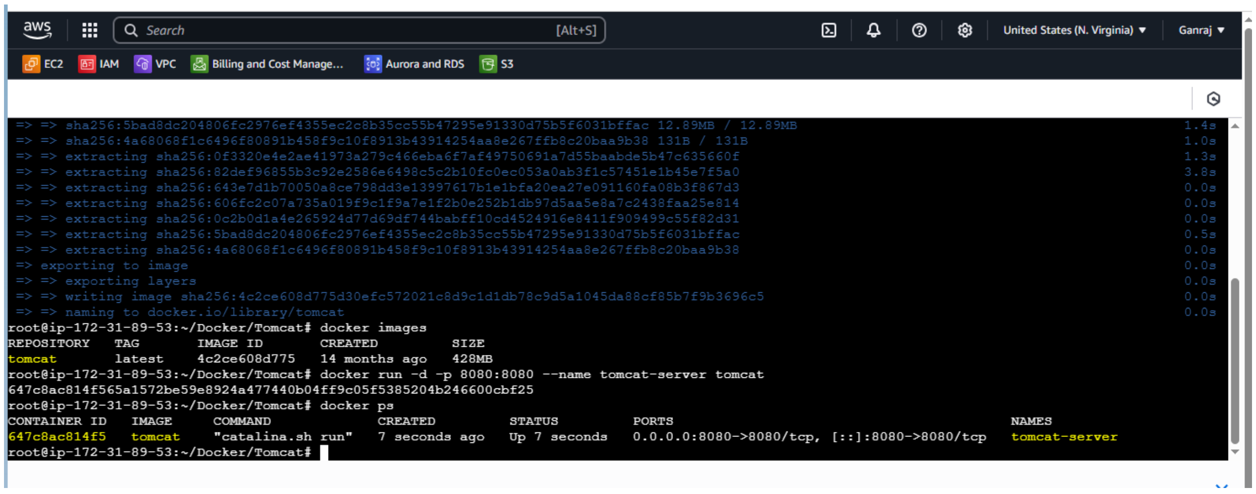
```
root@ip-172-31-89-53:~/Docker/Tomcat# docker build -t tomcat .
[+] Building 10.3s (5/5) FINISHED                                                                     docker:default
 => [internal] load build definition from Dockerfile                                                            0.0s
 => => transferring dockerfile: 129B                                                                             0.0s
 => [internal] load metadata for docker.io/library/tomcat:9.0.87-jdk17                                           0.4s
 => [internal] load .dockerignore                                                                                0.0s
 => => transferring context: 2B                                                                                  0.0s
 => [1/1] FROM docker.io/library/tomcat:9.0.87-jdk17@sha256:874b384fa55790154103ad91855553473c37ffef0cba0665bf030546b012ce19  9.7s
 => => resolve docker.io/library/tomcat:9.0.87-jdk17@sha256:874b384fa55790154103ad91855553473c37ffef0cba0665bf030546b012ce19  0.0s
 => => sha256:874b384fa55790154103ad91855553473c37ffef0cba0665bf030546b012ce19 1.21kB / 1.21kB                  0.0s
 => => sha256:c30b3aab767780946d788f6376c0b0b767125305e64d1070406320461ec5198c 2.00kB / 2.00kB                  0.0s
 => => sha256:7c071f2dee18b882faccd096059ea60e5730b2c056f4d5cde6a72387cedff310 13.54kB / 13.54kB                0.0s
 => => sha256:7021d1b70935851c95c45ed18156980b5024eda29b99564429025ea04f5ec109 30.44MB / 30.44MB                0.5s
 => => sha256:0f3320e4e2ae41973a279c466eba6f7af49750691a7d55baabde5b47c635660f 17.46MB / 17.46MB                0.9s
 => => sha256:82def96855b3c92e2586e6498c5c2b10fc0ec053a0ab3f1c57451e1b45e7f5a0 144.90MB / 144.90MB              3.7s
 => => extracting sha256:7021d1b70935851c95c45ed18156980b5024eda29b99564429025ea04f5ec109                       2.7s
 => => sha256:643e7d1b70050a8ce798dd3e13997617b1e1bfa20ea27e091160fa08b3f867d3 174B / 174B                      0.6s
 => => sha256:606fc2c07a735a019f9c1f9a7e1f2b0e252b1db97d5aa5e8a7c2438faa25e814 734B / 734B                      0.7s
 => => sha256:0c2b0d1a4e265924d77d69df744babff10cd4524916e8411f909499c55f82d31 173B / 173B                      0.8s
 => => sha256:5bad8dc204806fc2976ef4355ec2c8b35cc55b47295e91330d75b5f6031bffac 12.89MB / 12.89MB                1.4s
 => => sha256:4a68068f1c6496f80891b458f9c10f8913b43914254aa8e267ffb8c20baa9b38 131B / 131B                      1.0s
 => => extracting sha256:0f3320e4e2ae41973a279c466eba6f7af49750691a7d55baabde5b47c635660f                       1.3s
```

i-071943f52aea17695 (Docker)

Now we will create a container using tomcat image

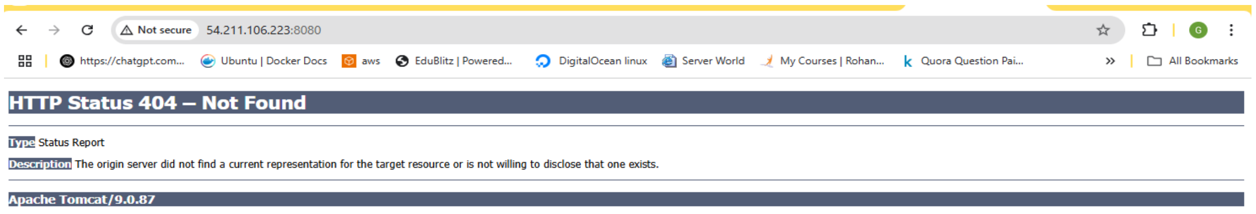
docker run -d -p 8080:8080 --name tomcat-server tomcat

using this command we have created one container named as tomcat-server as you can see snap below.



```
 => => sha256:5bad8dc204806fc2976ef4355ec2c8b35cc55b47295e91330d75b5f6031bffac 12.89MB / 12.89MB                1.4s
 => => sha256:4a68068f1c6496f80891b458f9c10f8913b43914254aa8e267ffb8c20baa9b38 131B / 131B                      1.0s
 => => extracting sha256:0f3320e4e2ae41973a279c466eba6f7af49750691a7d55baabde5b47c635660f                       1.3s
 => => extracting sha256:82def96855b3c92e2586e6498c5c2b10fc0ec053a0ab3f1c57451e1b45e7f5a0                       3.8s
 => => extracting sha256:643e7d1b70050a8ce798dd3e13997617b1e1bfa20ea27e091160fa08b3f867d3                       0.0s
 => => extracting sha256:606fc2c07a735a019f9c1f9a7e1f2b0e252b1db97d5aa5e8a7c2438faa25e814                       0.0s
 => => extracting sha256:0c2b0d1a4e265924d77d69df744babff10cd4524916e8411f909499c55f82d31                       0.0s
 => => extracting sha256:5bad8dc204806fc2976ef4355ec2c8b35cc55b47295e91330d75b5f6031bffac                       0.5s
 => => extracting sha256:4a68068f1c6496f80891b458f9c10f8913b43914254aa8e267ffb8c20baa9b38                       0.0s
 => exporting to image                                                                                           0.0s
 => => exporting layers                                                                                          0.0s
 => => writing image sha256:4c2ce608d775d30efc572021c8d9c1d1db78c9d5a1045da88cf85b7f9b3696c5                    0.0s
 => => naming to docker.io/library/tomcat                                                                        0.0s
root@ip-172-31-89-53:~/Docker/Tomcat# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
tomcat       latest    4c2ce608d775   14 months ago   428MB
root@ip-172-31-89-53:~/Docker/Tomcat# docker run -d -p 8080:8080 --name tomcat-server tomcat
647c8ac814f565a1572be59e8924a477440b04ff9c05f5385204b246600cbf25
root@ip-172-31-89-53:~/Docker/Tomcat# docker ps
CONTAINER ID   IMAGE     COMMAND             CREATED         STATUS         PORTS                                       NAMES
647c8ac814f5   tomcat    "catalina.sh run"   7 seconds ago   Up 7 seconds   0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp   tomcat-server
root@ip-172-31-89-53:~/Docker/Tomcat#
```

But as you viewed the dockerfile for tomcat we have used base image as its own (official) image which is already generated .because the tomcat base image because it saves time, reduces complexity, and follows best practices for containerized apps.



Not secure 54.211.106.223:8080

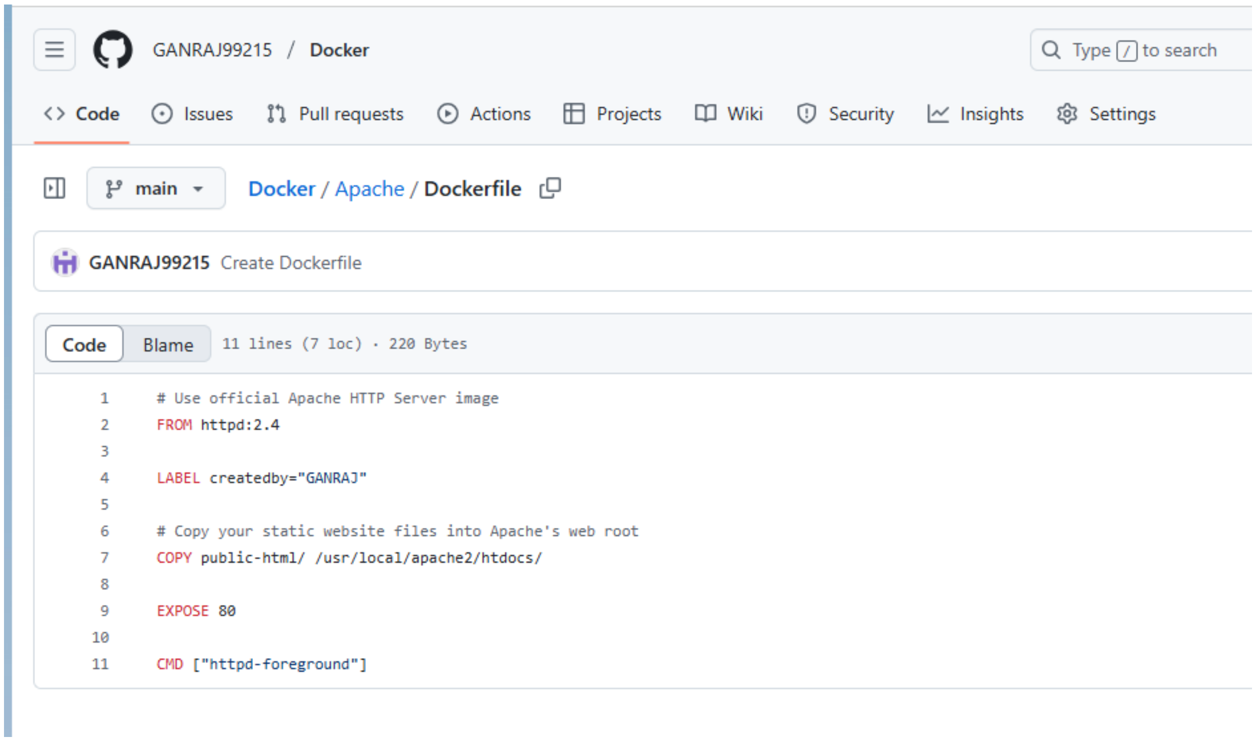
HTTP Status 404 – Not Found

Type Status Report

Description The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

Apache Tomcat/9.0.87

3)Apache2(httpd) :-

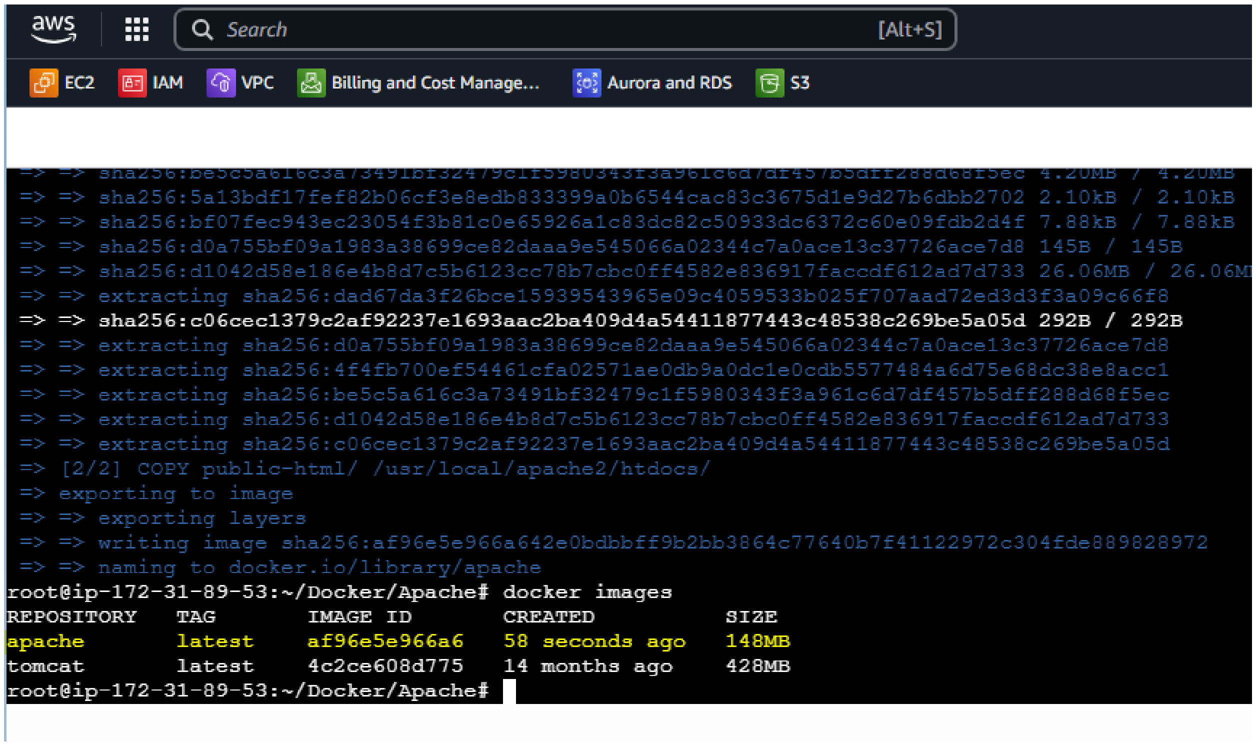


```
# Use official Apache HTTP Server image
FROM httpd:2.4

LABEL createdby="GANRAJ"

# Copy your static website files into Apache's web root
COPY public-html/ /usr/local/apache2/htdocs/

EXPOSE 80

CMD ["httpd-foreground"]
```

Line	What It Does
`FROM httpd:2.4`	Uses the official Apache server image (2.4 is the version).
`LABEL createdby="GANRAJ"`	Optional info about who created/maintains the image.
`COPY public-html/ /usr/local/apache2/htdocs/`	Puts your HTML files into Apache's default root folder.
`EXPOSE 80`	Declares port 80 as open (you still need `-p` during `docker run`).
`CMD ["httpd-foreground"]`	Keeps Apache running in the foreground — required for containers to stay alive.

Now we will build image from this docker file

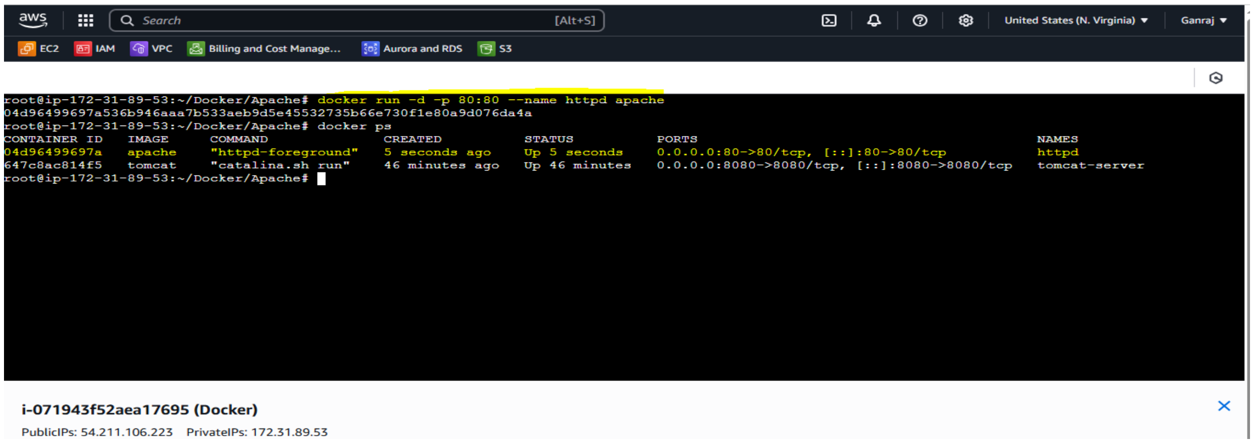
docker build –t apache .



```
=> => sha256:be5c5a616c3a73491bf32479c1f5980343f3a961c6d7df457b5dff288d68f5ec 4.20MB / 4.20MB
=> => sha256:5a13bdf17fef82b06cf3e8edb833399a0b6544cac83c3675d1e9d27b6dbb2702 2.10kB / 2.10kB
=> => sha256:bf07fec943ec23054f3b81c0e65926a1c83dc82c50933dc6372c60e09fdb2d4f 7.88kB / 7.88kB
=> => sha256:d0a755bf09a1983a38699ce82daaa9e545066a02344c7a0ace13c37726ace7d8 145B / 145B
=> => sha256:d1042d58e186e4b8d7c5b6123cc78b7cbc0ff4582e836917faccdf612ad7d733 26.06MB / 26.06M
=> => extracting sha256:dad67da3f26bce15939543965e09c4059533b025f707aad72ed3d3f3a09c66f8
=> => sha256:c06cec1379c2af92237e1693aac2ba409d4a54411877443c48538c269be5a05d 292B / 292B
=> => extracting sha256:d0a755bf09a1983a38699ce82daaa9e545066a02344c7a0ace13c37726ace7d8
=> => extracting sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1
=> => extracting sha256:be5c5a616c3a73491bf32479c1f5980343f3a961c6d7df457b5dff288d68f5ec
=> => extracting sha256:d1042d58e186e4b8d7c5b6123cc78b7cbc0ff4582e836917faccdf612ad7d733
=> => extracting sha256:c06cec1379c2af92237e1693aac2ba409d4a54411877443c48538c269be5a05d
=> [2/2] COPY public-html/ /usr/local/apache2/htdocs/
=> exporting to image
=> => exporting layers
=> => writing image sha256:af96e5e966a642e0bdbbff9b2bb3864c77640b7f41122972c304fde889828972
=> => naming to docker.io/library/apache
root@ip-172-31-89-53:~/Docker/Apache# docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
apache       latest    af96e5e966a6   58 seconds ago   148MB
tomcat       latest    4c2ce608d775   14 months ago    428MB
root@ip-172-31-89-53:~/Docker/Apache#
```

Using this docker image we will create a container httpd

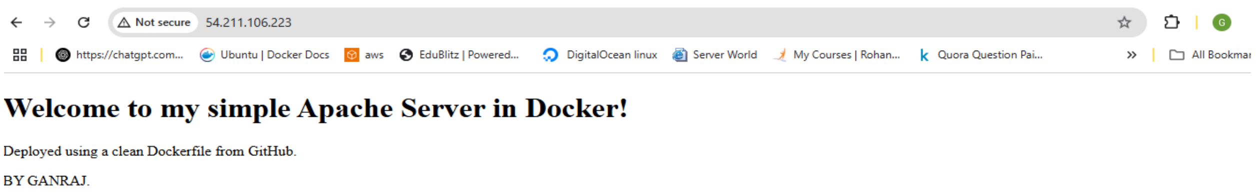
docker run –d –p 80:80 --name httpd apache



```
root@ip-172-31-89-53:~/Docker/Apache# docker run -d -p 80:80 --name httpd apache
04d96499697a536b946aaa7b533aeb9d5e45532735b66e730f1e80a9d076da4a
root@ip-172-31-89-53:~/Docker/Apache# docker ps
CONTAINER ID   IMAGE     COMMAND                CREATED          STATUS          PORTS                                         NAMES
04d96499697a   apache    "httpd-foreground"     5 seconds ago    Up 5 seconds    0.0.0.0:80->80/tcp, [::]:80->80/tcp           httpd
647c8ac814f5   tomcat    "catalina.sh run"      46 minutes ago   Up 46 minutes   0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp   tomcat-server
root@ip-172-31-89-53:~/Docker/Apache#
```

i-071943f52aea17695 (Docker)
PublicIPs: 54.211.106.223 PrivateIPs: 172.31.89.53

When you access your server using host: port you will see your server is running using container



Not secure 54.211.106.223

Welcome to my simple Apache Server in Docker!

Deployed using a clean Dockerfile from GitHub.

BY GANRAJ.