

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"

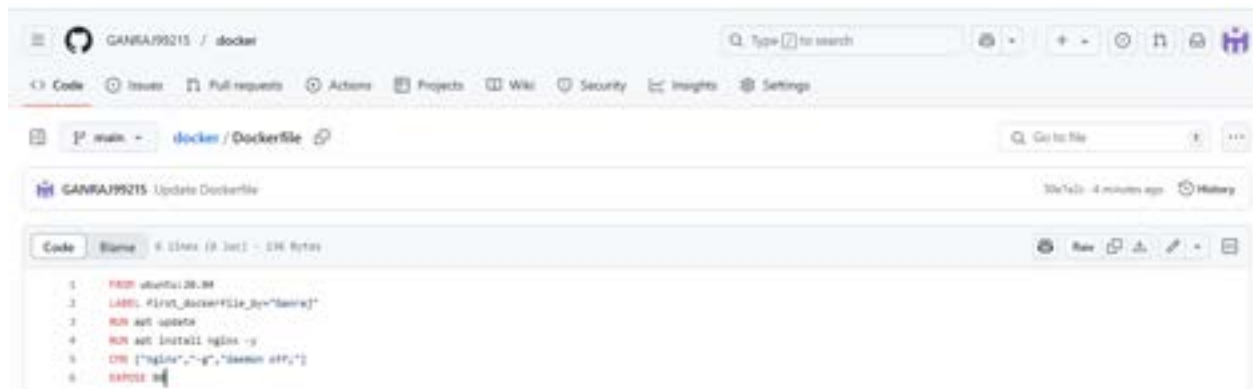
RUN apt update

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



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
```

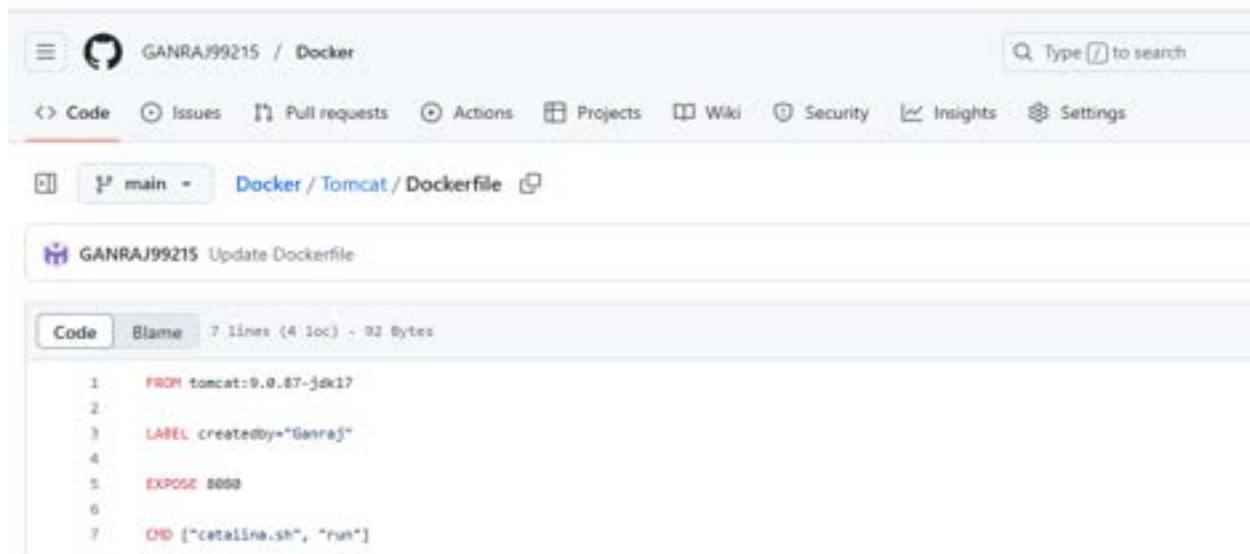
Then visit:

<http://localhost:8080>

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

2)Tomcat:-

Now we will create same for tomcat



Line

FROM tomcat:9.0.87-jdk17

LABEL createdby="Ganraj"

COPY myapp.war ...

EXPOSE 8080

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

What it Does

Uses the official Tomcat image with Java 17 pre-installed. No need to install Java, Tomcat manually.

Adds author metadata (optional, good practice).

(Optional) If you have a WAR file, copy it into the Tomcat webapps/ folder.

Exposes the Tomcat port so it can be accessed from outside the container.

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 -f Dockerfile .
[+] Building 10.3s (5/7) FINISHED
-- [internal] load build definition from Dockerfile
-- [internal] load Dockerfile from Dockerfile: 1.0s
-- [internal] load metadata for docker.io/tomcat:9.0.57-jdk8
-- [internal] load .dockerignore
-- [internal] resolve image configuration for docker.io/tomcat:9.0.57-jdk8
-- [1/7] FROM docker.io/tomcat:9.0.57-jdk8
-- [2/7] WORKDIR /opt/tomcat
-- [3/7] COPY *.war /opt/tomcat
-- [4/7] EXPOSE 8080
-- [5/7] CMD ["java", "-jar", "-Djava.class.path=/opt/tomcat/WEB-INF/classes:/opt/tomcat/WEB-INF/lib/*:.", "org.apache.catalina.startup.Bootstrap"]
-- [6/7] COMMIT docker.io/tomcat:9.0.57-jdk8
-- [7/7] PUSH docker.io/tomcat:9.0.57-jdk8
root@ip-172-31-89-53:~/docker/tomcat# docker run -d -p 8080:8080 --name tomcat-server tomcat
```

Now we will create a container using tomcat image

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

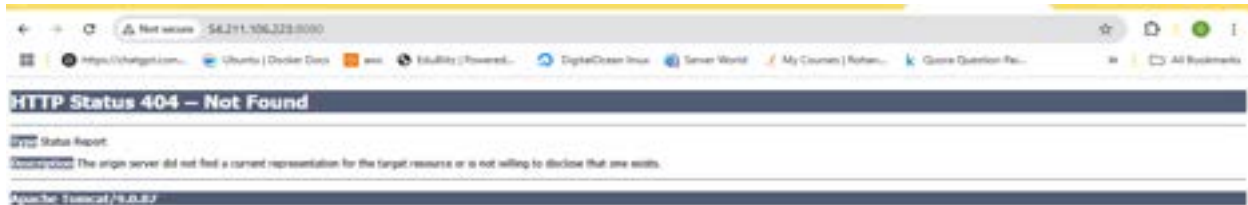
using this command we have created one container named as tomcat-server as you can see snap below.

```
root@ip-172-31-89-53:~/docker/tomcat# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
tomcat               latest             4c2e650b4775       14 months ago      425MB

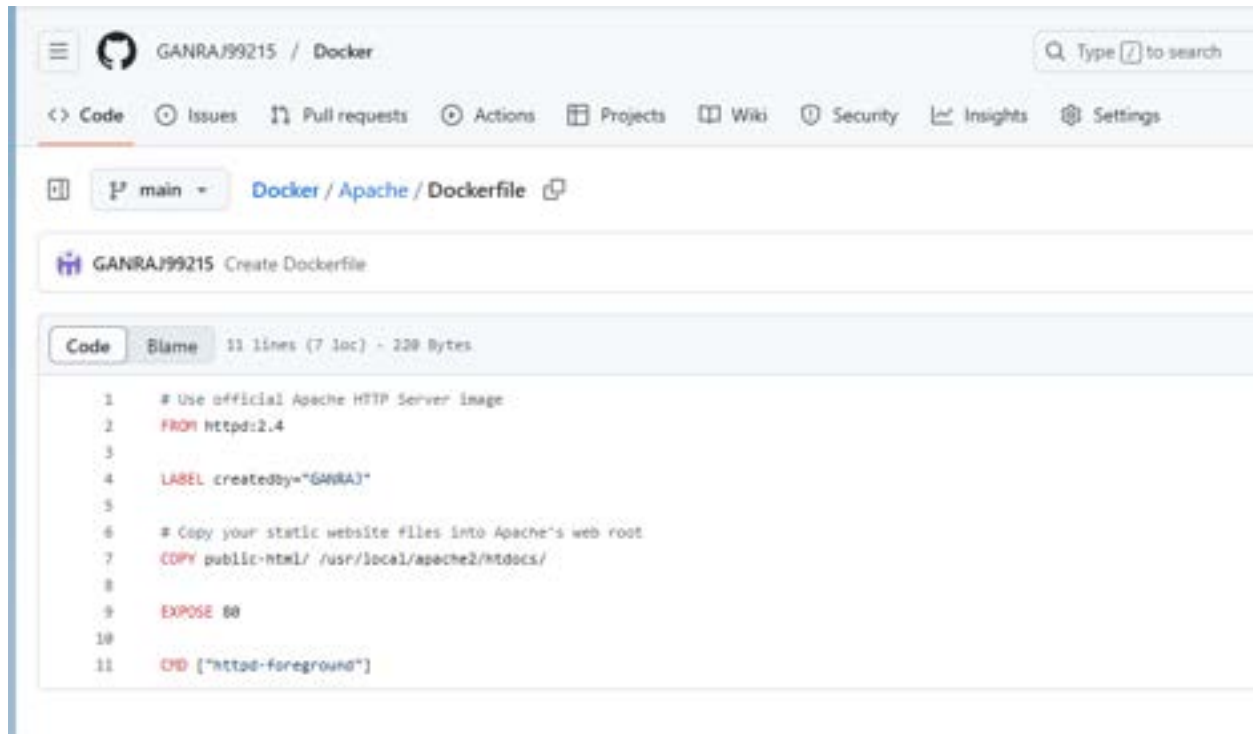
root@ip-172-31-89-53:~/docker/tomcat# docker run -d -p 8080:8080 --name tomcat-server tomcat
4470ba014f565a1572be559e9524e477463b041f3b05f3381284b244600db225

root@ip-172-31-89-53:~/docker/tomcat# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS             PORTS
4470ba014f5        tomcat             "catalina.sh run"   7 seconds ago      Up 7 seconds       0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp
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.



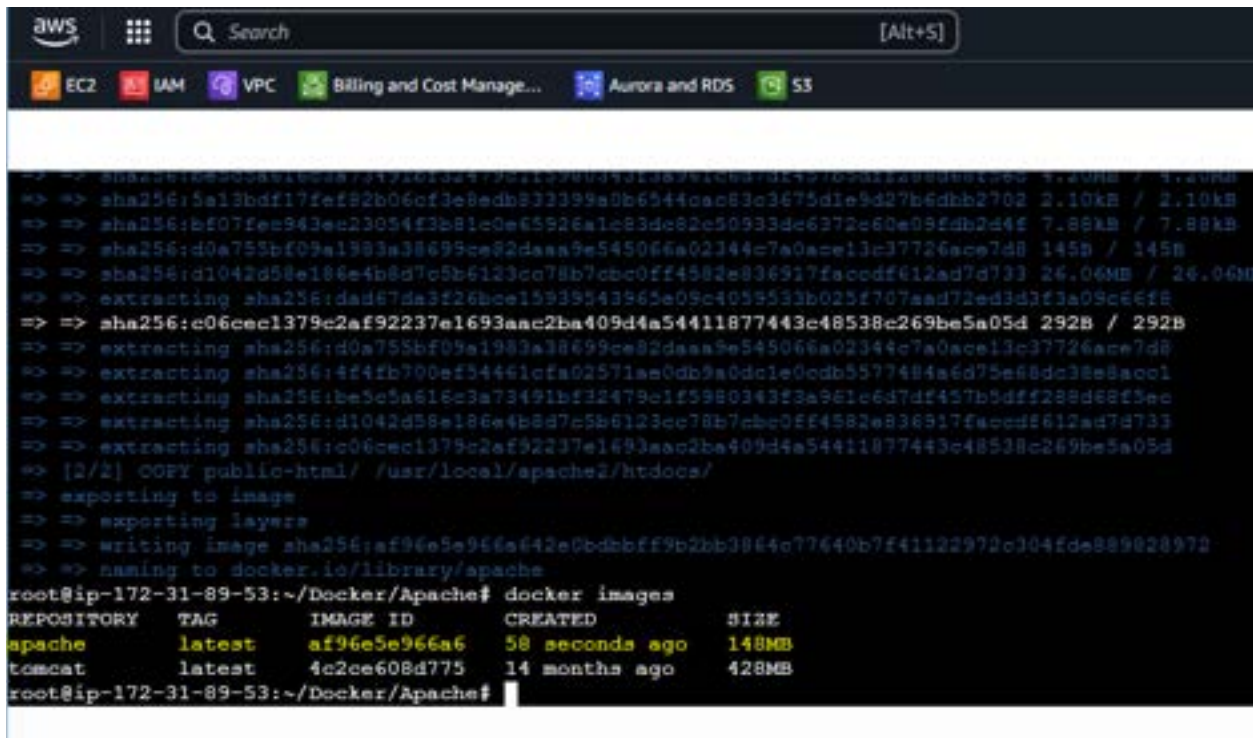
3)Apache2(httpd) :-

A screenshot of a GitHub repository page for a user named GANRAJ99215. The repository is named 'Docker'. The file 'Dockerfile' is selected, showing its code. The code is a Dockerfile for Apache2, with 11 lines of code. The code starts with a comment '# Use official Apache HTTP Server image', followed by 'FROM httpd:2.4', 'LABEL createdby="GANRAJ"', another comment '# Copy your static website files into Apache's web root', 'COPY public-html/ /usr/local/apache2/htdocs/', 'EXPOSE 80', and finally 'CMD ["httpd-foreground"]'. The interface shows the 'main' branch and a 'Create Dockerfile' button.

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 .`



```
aws
[Alt+S]
EC2 IAM VPC Billing and Cost Manage... Aurora and RDS S3

--> sha256:5e13bdf17fef82b06cf3e8edb333399a0b65440ac63c3675d1e9d27b6d8b2702 2.10kB / 2.10kB
--> sha256:b507fec943ec23054f3b81c0e65926a1c83dc82c50933dc6372c60e09fdb2d4f 7.88kB / 7.88kB
--> sha256:d0a755bf09a1983a38699ce82d8aa9e545066a02344c7a0ace13c37726ace7d8 145B / 145B
--> sha256:d1042d58e186e4b8d7c5b6123cc78b7cbc0ff4582e836917fac0df612ad7d733 26.04MB / 26.04MB
--> extracting sha256:dad67da3f26bce15939543965e09c4039533b025f707aad72ed3d3f3a09c66ff6
--> sha256:c06cec1379c2af92237e1693aac2ba409d4a54411877443c48538c269be5a05d 292B / 292B
--> extracting sha256:d0a755bf09a1983a38699ce82d8aa9e545066a02344c7a0ace13c37726ace7d8
--> extracting sha256:424fb700ef54461cfa02571a0db9a0dc1e0cbb5577484a6d75e68dc38e8aac01
--> extracting sha256:be3c5a616c3a73491bf32479c1f5980343f3a961c6d7df437b5dff288d68f5ec
--> extracting sha256:d1042d58e186e4b8d7c5b6123cc78b7cbc0ff4582e836917fac0df612ad7d733
--> extracting sha256:c06cec1379c2af92237e1693aac2ba409d4a54411877443c48538c269be5a05d
--> [2/2] COPY public-html/ /usr/local/apache2/htdocs/
--> exporting to image
--> exporting layers
--> writing image sha256:af96e5e966e442e0bd8bfff9b2bb3864c77640b7f41122972c304fd0889828972
--> naming to docker.io/library/apache
root@ip-172-31-89-53:~/Docker/Apache# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
apache latest af96e5e966e4 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`



```
aws
[Alt+S]
EC2 IAM VPC Billing and Cost Manage... Aurora and RDS S3

root@ip-172-31-89-53:~/Docker/Apache# docker run -d -p 80:80 --name httpd apache
af96e5e966e442e0bd8bfff9b2bb3864c77640b7f41122972c304fd0889828972
root@ip-172-31-89-53:~/Docker/Apache# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
af96e5e966e4 apache "httpd-foreground" 3 seconds ago Up 3 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp httpd
44708ac814d5 tomcat "catalina.sh run" 46 minutes ago Up 46 minutes 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp tomcat-server

i-071943f52aea17695 (Docker)
Public IP: 54.211.106.225 Private IP: 172.31.89.53
```



Not secure 54.211.106.225

https://chango.com... Ubuntu | Docker Docs AWS | Edabit | Powered... DigitalOcean Linux Server World | My Courses | Reth... | Quora Question Pa... | All Explan...

Welcome to my simple Apache Server in Docker!

Deployed using a clean Dockerfile from GitHub.

BY GANRAJ

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

