

# Dockerfile Instructions with Examples and Explanations

## WHAT IS DOCKERFILE?

Dockerfile is used to build custom images. We can make use of docker instructions to create custom images

It is a declarative way of custom images

Docker instructions are as follows,

### 1. FROM

The **FROM** instruction specifies the base image for your Docker image. It is the first instruction in the Dockerfile and forms the foundation of your custom image.

Syntax: **FROM** <base-image>:<version>

Example:

```
dockerfiles > FROM > Dockerfile > ...  
1 FROM almalinux:9  
2 |  
3
```

### 2. RUN

The **RUN** instruction is used to execute commands in the image during the build process. It's often used to install software or configure the environment.

Syntax: **RUN** <command>

Example:

```
dockerfiles > RUN > Dockerfile > ...  
1 FROM almalinux:9  
2 RUN dnf install nginx -y  
3
```

### 3. CMD

The **CMD** instruction specifies the default command that will be run when a container is created from the image. Unlike **RUN**, it is executed at container runtime, not during image build.

**Syntax:** `CMD ["executable", "param1", "param2"]`

**Example:**

```
dockerfiles > CMD > Dockerfile > ...  
1 FROM almalinux:9  
2 RUN dnf install nginx -y  
3 CMD ["nginx", "-g", "daemon off;"]
```

### 4. ENTRYPOINT

The **ENTRYPOINT** instruction is similar to **CMD** but is generally used for scripts or commands that must always be executed. It allows you to configure a container that will run as an executable.

**Syntax:** `ENTRYPOINT ["executable", "param1", "param2"]`

**Example:**

```
dockerfiles > ENTRYPOINT > Dockerfile > ...  
1 # FROM almalinux:9  
2 # CMD ["ping", "google.com"]  
3  
4  
5  
6 # FROM almalinux:9  
7 # ENTRYPOINT [ "ping", "google.com" ]  
8  
9 FROM almalinux:9  
10 CMD [ "google.com" ]  
11 ENTRYPOINT [ "ping" ]
```

## 5. COPY

The **COPY** instruction copies files from the host machine to the Docker image.

**Syntax:** **COPY** <source> <destination>

**Example:**

```
dockerfiles > COPY > Dockerfile > ...  
1 FROM almalinux:9  
2 RUN dnf install nginx -y  
3 RUN rm -rf /usr/share/nginx/html/index.html  
4 COPY index.html /usr/share/nginx/html/index.html  
5 CMD ["nginx", "-g", "daemon off;"]
```

```
dockerfiles > COPY > index.html > h1  
1 <h1> Hi, This file is copied from local to image </h1>
```

## 6. ADD

The **ADD** instruction is similar to **COPY**, but it has additional features. It can extract archives and fetch files from remote URLs.

It has two extra capabilities

1. Can download files directly from internet
2. Can untar directly into image

**Syntax:** **ADD** <source> <destination>

**Example:**

```
dockerfiles > ADD > Dockerfile > ...  
1 FROM almalinux:9  
2 RUN dnf install nginx -y  
3 RUN rm -rf /usr/share/nginx/html/index.html  
4 ADD https://raw.githubusercontent.com/lakshmimungara/DevOps-notes/refs/heads/main/Terraform/  
Day-22%20session-22.txt /usr/share/nginx/html/index.html  
5 RUN chmod +x /usr/share/nginx/html/index.html  
6 # ADD sample-1.tar /tmp/  
7 CMD ["nginx", "-g", "daemon off;"]
```

## 7. LABEL

The **LABEL** instruction adds metadata to the image in key-value pairs, which can be used for descriptive or filtering purposes.

**Syntax:** LABEL <key>=<value>

**Example:**

```
dockerfiles > LABEL > Dockerfile > ...  
1 FROM almalinux:9  
2 LABEL author="lakshmi" \  
3     company="joindevops" \  
4     topic="devops" \  
5     duration="2hrs"
```

## 8. EXPOSE

The **EXPOSE** instruction informs Docker that the container will listen on specific network ports, though it does not actually publish the port to the host.

**Syntax:** EXPOSE <port>

**Example:**

```
dockerfiles > EXPOSE > Dockerfile > ...  
1 FROM almalinux:9  
2 RUN dnf install nginx -y  
3 EXPOSE 80  
4 CMD ["nginx", "-g", "daemon off;"]
```

## 9. ENV

The **ENV** instruction sets environment variables in the container that can be accessed by applications running inside the container.

**Syntax:** ENV <key>=<value>

**Example:**

```
dockerfiles > ENV > Dockerfile > ...  
1 FROM almalinux:9  
2 ENV course="devOps with AWS" \  
3     trainer="sivakumar" \  
4     duration="2hrs"
```

## 10. WORKDIR

The **WORKDIR** instruction sets the working directory for any subsequent **RUN**, **CMD**, **ENTRYPOINT**, and other instructions.

**Syntax:**        **WORKDIR** /path/to/workdir

**Example:**

```
FROM almalinux:9
RUN mkdir /tmp/docker
WORKDIR /tmp/docker
RUN pwd
RUN echo "Hello" > hello.txt
CMD ["sleep", "100"]
```

## 11. ARG

The **ARG** instruction defines build-time variables that can be passed to Docker during the build process.

**Syntax:**        **ARG** <name>=<default-value>

**Example:**

```
FROM almalinux:9
ARG course="Devops with AWS" \
    duration="2hrs"
RUN echo "course: $course, duration: $duration"
CMD ["sleep", "100"]
```

## 12. USER

The **USER** instruction sets the user to run subsequent commands during build or container execution, improving security by avoiding root usage.

**Syntax:**        **USER** <username>[:<group>]

**Example:**

```
FROM almalinux:9
RUN useradd expense
USER expense
CMD ["sleep", "100"]
```


### 13. ONBUILD

The **ONBUILD** instruction sets up a trigger that executes when the image is used as the base for another build. It is commonly used in base images to specify additional actions for future Dockerfiles.


**Syntax:** **ONBUILD** <INSTRUCTION>

**Example:**

```
FROM almalinux:9
RUN dnf install nginx -y
RUN rm -rf /usr/share/nginx/html/index.html
# This onbuild instruction runs only, when any user uses the image
ONBUILD COPY index.html /usr/share/nginx/html/index.html
CMD ["nginx", "-g", "daemon off;"]
```

```
dockerfiles > ONBUILD > test >  Dockerfile > ...
```

```
1 FROM onbuild:v1
```

```
dockerfiles > ONBUILD > test > <> index.html >  h1
```

```
1 <h1>This image is created by onbuild base</h1>
```