

**Base Containers**

A Dockerfile is the mechanism that the docker packaging model provides to automate the building the container images. Building an image from a Dockerfile is a three-step process.

1. Create a working directory.
2. Write the Dockerfile specification
3. Build the image with the docker command.

**Create a working directory.**

The Docker command can use the files in a working directory to build an image. An empty working directory should be created to keep from incorporating unnecessary files into the image. For security reasons, the root directory, /, should never be used a working directory for image builds.

**Write the Dockerfile specification.**

**# comment**

**INSTRUCTION ARGUMENTS**

**LABEL** is responsible for adding generic metadata to an image.

**MAINTAINER** is responsible for setting the Author field of the generated container image.

**RUN** execute commands in a new layer on top of the current image

**EXPOSE** indicates that the container listens on the specified network port at runtime.

**ENV** is responsible for defining environment variables that will be available to the container

**ADD** copies files from local or remote source and add them to the container’s file system.

**COPY** copies files from local or remote source and add them to the container’s file system.

**USER** specifies the username

**ENTRYPOINT** specifies the default command to execute when the container is created.

**CMD** provides the default arguments for the ENTRYPOINT instruction.

* Using CMD and ENTRYPOINT Instructions in the Dockerfile.

The ENTRYPOINT and CMD instructions have two formats:

* Using a JSON array:

**ENTRYPOINT [“command”, “param1”,” param2”]**

* Using a shell form:

**ENTRYPOINT command param1 param2**

**CMD param1 param2**

For the example: 1

The following instruction causes any container that I run to display the current time.

|  |
| --- |
| # this is a comment line  FROM centos  LABEL description=”This is a custom httpd container image.”  MAINTAINER John doe < jdoe@xyz.com”  RUN yum install -y httpd  EXPOSE 80  ENV LogLevel “info”  ADD <http://someserver.com/filename.pdf> /var/www/html  COPY ./src/ /var/www/html  USER apache  ENTRYPOINT [“/usr/sbin/httpd”]  CMD [“-D”,”FOREGROUND”] |

Example: 2

|  |
| --- |
| FROM ubuntu:18.04  LABEL NAME "sam"  RUN apt-get update && apt install -y python3-pip  COPY test.py /tmp  COPY backup.tar /tmp  WORKDIR /tmp  #ENTRYPOINT [ "python3", "./test.py" ] |

Example: 3

|  |
| --- |
| # this is a comment line  FROM centos  LABEL description="This is a custom httpd container image"  MAINTAINER sam anuragi<darwikdev@gmail.com>  RUN yum install -y httpd  EXPOSE 80  ENV LogLevel "info"  COPY index.html /var/www/html  ENTRYPOINT ["/usr/sbin/httpd"]  CMD ["-D","FOREGROUND"] |

Example: 4

|  |
| --- |
| FROM ubuntu:18.04  RUN apt-get update && apt-get install -y openssh-server \  && mkdir /var/run/sshd&& echo root:redhat | chpasswd \  && sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd\_config  # SSH login fix. Otherwise user is kicked off after login  RUN sed 's@session\s\*required\s\*pam\_loginuid.so@session optional pam\_loginuid.so@g' -i /etc/pam.d/sshd  RUN useradd -c /bin/sh -d /home/sam -m sam && echo sam:redhat | chpasswd  ENV NOTVISIBLE "in users profile"  RUN echo "export VISIBLE=now" >> /etc/profile  EXPOSE 22  CMD ["/usr/sbin/sshd", "-D"] |