

Image Save and Load

Image Save and Load

docker.io
Docker Hub

```
▶ docker image save alpine:latest -o alpine.tar
```

```
▶ docker image load -i alpine.tar  
bee9f30bc1f: Loading layer [=====>] 5.862MB/5.862MB  
Loaded image: alpine:latest
```

```
▶ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alpine	latest	a187dde48cd2	4 weeks ago	5.6MB

Import and Export Operations

```
▶ docker export <container-name> > testcontainer.tar
```

```
▶ docker image import testcontainer.tar newimage:latest  
sha256:8090b7da236bb21aa2e52e6e04dff4b7103753e4046e15457a3daf6dfa723a12
```

```
▶ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
newimage	latest	8090b7da236b	2 minutes ago	5.6MB
alpine	latest	a187dde48cd2	4 weeks ago	5.6MB

Lab: from image create tar file then from that tar file create a image

```
rajiv@server-A:~$ docker image save alpine:latest -o rajiv.tar  
rajiv@server-A:~$ ls  
awscli2.zip Desktop docker Documents Downloads Music Pictures Public rajiv1.pem rajiv_n_virgina.pem rajiv.tar snap Templates Videos  
rajiv@server-A:~$ docker image load -i rajiv.tar  
Loaded image: alpine:latest  
rajiv@server-A:~$ |
```

Lab: create a tar file from a container then launch a container from that tar file

```

rajiv@server-A:~$ docker container run -itd --name=test alpine
b7404c2d82bd21317ca14eaf2a028c967893fcc994d1d44e94d77d721cd14072
rajiv@server-A:~$ docker container ls -l
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
b7404c2d82bd   alpine    "/bin/sh" 10 seconds ago  Up 10 seconds           test
rajiv@server-A:~$ docker export test >rajiv.tar
rajiv@server-A:~$ ls
awscli2.zip  Desktop  docker  Documents  Downloads  Music  Pictures  Public  rajiv1.pem  rajiv_n_virgina.pem  rajiv.tar  snap  Templates  Videos
rajiv@server-A:~$ docker image import rajiv.tar rajiv:latest
sha256:f0b7a6a23e88beccc9f11ecca390b2d388df550e76fc1263e177aedeeaa3c9138
rajiv@server-A:~$ docker images
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
rajiv         latest    f0b7a6a23e8b  9 seconds ago  7.33MB
result-app    latest    14b88973d259  4 days ago  264MB

```

Building Images Using Commit

Docker Container Commit

```
docker run -d --name httpd httpd
```

```
docker exec -it httpd bash
```

```
root@3484d738:/# cat > htdocs/index.html
```

```
Welcome to my custom web application
```

```
docker container commit -a "Ravi" httpd customhttpd
```

```
docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
customhttpd	latest	adac0f56a7df	5 seconds ago	138MB
httpd	latest	417af7dc28bc	8 days ago	138MB




Image-customhttpd

docs.docker.com/reference/

ভূমি উন্নয়ন কর || Dhaka North City C... || ping form outside || subnetmask CIDR || iCloud Drive || ভূমি উন্নয়ন কর || Dhaka North City C... || ping form outside || subnetmask CIDR

docker docs Guides Manuals **Reference** Samples FAQ

Reference documentation

Command-line reference ^

Docker CLI (docker) ^

Docker run reference

Use the Docker command line

docker (base command)

docker attach

docker build

docker builder ^

docker buildx ^

docker checkpoint ^

docker commit

docker compose ^

docker config ^

docker container ^

docker context ^


docker cp

docker create

Reference documentation


This section includes the reference documentation for the Docker platform's various APIs, CLIs, drivers and specifications, and file formats.

File formats



Dockerfile


Defines the contents and startup behavior of a single container.



Compose file


Defines a multi-container application.

Command-line interfaces (CLIs)




Docker CLI

The main Docker CLI, includes all `docker` commands.



Compose CLI

The CLI for Docker Compose, for building and running multi-container applications.



Daemon CLI (dockerd)

Persistent process that manages containers.

Lab: edit a website from a running container and then create a image and run a container

First

```
#docker image pull centos:7
#docker container create -it --name=rajivweb centos:7
#docker container ls -l
#docker container start rajivweb
#docker container ls -l
#docker container exec -it rajivweb /bin/bash
Now run some command in this container
```

```
yum update -y
yum install httpd -y
echo "Hello world" > /var/www/html/index.html
exit
```

```
#docker container ls
#docker container stop rajivweb
#docker container ls -l
#docker container commit -a "rajiv siddiqui" -c 'CMD ["httpd", "-D", "FOREGROUND"]'
rajivweb rajivweb2023:v1
#docker image ls
#docker container run -itd -p 80:80 rajivweb2023:v1
#docker container ls -l
```

Difference between Save vs Load vs Export vs Import vs commit

save= create a tar file form an image

Load= create an image form that tar file

export= create an tar from a running container

import= create an image from a running container tar file

commit= edit a running container and create an image from that running container

Create our own images:

How to create my own image?

Dockerfile

```
FROM Ubuntu

RUN apt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

1. OS - Ubuntu
2. Update apt repo
3. Install dependencies using apt
4. Install Python dependencies using pip
5. Copy source code to /opt folder
6. Run the web server using "flask" command

```
docker build . -f Dockerfile -t rajivsiddiqui/my-custom-app
```

```
docker push rajivsiddiqui/my-custom-app
```

Docker Registry

Dockerfile

INSTRUCTION

ARGUMENT

Dockerfile

```
FROM Ubuntu
```

Start from a base OS or another image

```
RUN apt-get update && apt-get -y install python
```

Install all dependencies

```
RUN pip install flask flask-mysql
```

```
COPY . /opt/source-code
```

Copy source code

```
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

Specify Entrypoint

Layered architecture

```
docker build . -f Dockerfile -t rajivsiddiqui/my-custom-app
```

Dockerfile

```
FROM Ubuntu
```

```
RUN apt-get update && apt-get -y install python
```

```
RUN pip install flask flask-mysql
```

```
COPY . /opt/source-code
```

```
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

Layer 1. Base Ubuntu Layer

120 MB

Layer 2. Changes in apt packages

306 MB

Layer 3. Changes in pip packages

6.3 MB

Layer 4. Source code

229 B

Layer 5. Update Entrypoint with "flask" command

0 B

```
root@osboxes:/root/simple-webapp-docker # docker history rajivsiddiqui/my-app
IMAGE          CREATED          CREATED BY          SIZE      COMMENT
1a45ba829f10   About an hour ago /bin/sh -c #(nop)  ENTRYPOINT ["/bin/sh" "...  0B
37d37ed8fe99   About an hour ago /bin/sh -c #(nop)  COPY file:29b92853d73898...  229B
d6aaebf8ded0   About an hour ago /bin/sh -c pip install flask flask-mysql  6.3MB
e4c055538e60   About an hour ago /bin/sh -c apt-get update && apt-get insta...  306MB
ccc7a11d65b1   2 weeks ago      /bin/sh -c #(nop)  CMD ["/bin/bash"]  0B
<missing>      2 weeks ago      /bin/sh -c mkdir -p /run/systemd && echo '...  7B
<missing>      2 weeks ago      /bin/sh -c sed -i 's/^#\s*(deb.*universe\...  2.76kB
<missing>      2 weeks ago      /bin/sh -c rm -rf /var/lib/apt/lists/*  0B
<missing>      2 weeks ago      /bin/sh -c set -xe && echo '#!/bin/sh' >...  745B
<missing>      2 weeks ago      /bin/sh -c #(nop)  ADD file:39d3593ea220e68...  120MB
```

Docker build output

```
root@osboxes:/root/simple-webapp-docker # docker build .
Sending build context to Docker daemon 3.072kB
Step 1/5 : FROM ubuntu
--> ccc7a11d65b1
Step 2/5 : RUN apt-get update && apt-get install -y python python-setuptools python-dev
--> Running in a7840dbfad17
Get:1 http://archive.ubuntu.com/ubuntu xenial InRelease [247 kB]
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Get:4 http://security.ubuntu.com/ubuntu xenial-security/universe Sources [46.3 kB]
Get:5 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Get:6 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [440 kB]
Step 3/5 : RUN pip install flask flask-mysql
--> Running in a4a6c9190ba3
Collecting flask
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)
Collecting flask-mysql
  Downloading Flask_MySQL-1.4.0-py2.py3-none-any.whl
Removing intermediate container a4a6c9190ba3
Step 4/5 : COPY app.py /opt/
--> e7cdab17e782
Removing intermediate container faaaaf63c512
Step 5/5 : ENTRYPOINT FLASK_APP=/opt/app.py flask run --host=0.0.0.0
--> Running in d452c574a8bb
--> 9f27c36920bc
Removing intermediate container d452c574a8bb
Successfully built 9f27c36920bc
```

Build Cache

Build Cache

Dockerfile

```
FROM ubuntu

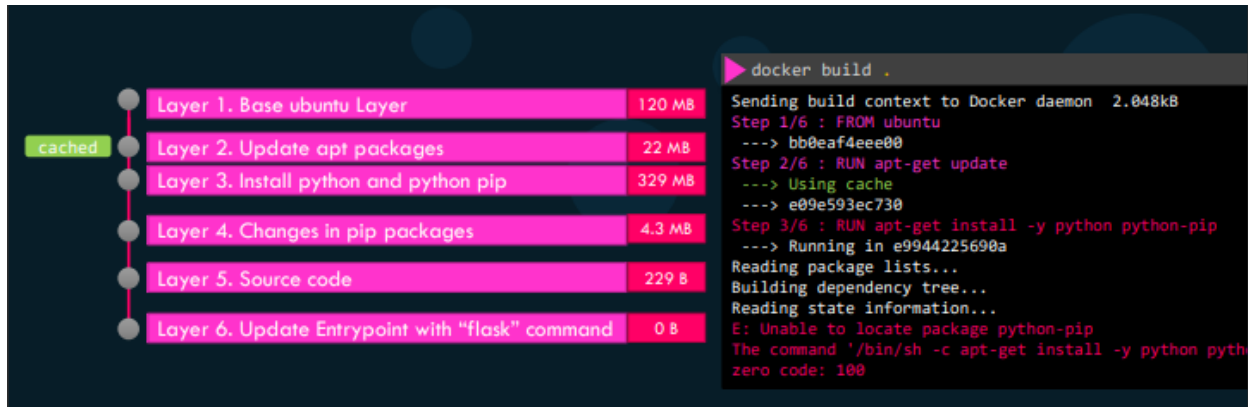
RUN apt-get update
RUN apt-get install -y python python3-pip

RUN pip3 install flask

COPY app.py /opt/source-code

ENTRYPOINT flask run
```

Layer 1. Base ubuntu Layer	120 MB
Layer 2. Update apt packages	22 MB
Layer 3. Install python and python pip	329 MB
Layer 4. Changes in pip packages	4.3 MB
Layer 5. Source code	229 B
Layer 6. Update Entrypoint with "flask" command	0 B



Lab: build a image from docker file and push it docker repository

Create 2 file

Dockerfile

index.html

Dockerfile

```

FROM centos:7
MAINTAINER Rajiv<rajivsiddiqui@gmail.com>
RUN yum update -y
RUN yum install httpd -y
COPY index.html /var/www/html/
ENTRYPOINT ["/usr/sbin/httpd","-D","FOREGROUND"]

```

index.html

Hello world

Now run the following command to build the image

```

#docker image build -t web2 .
#docker run -itd -p 82:80 web2
#docker image tag web2:latest rajivsiddiqui/web-httpd-centos
#docker images
#docker push rajivsiddiqui/web-httpd-centos

```

Use tomcat version as a argument variable

git lab url for this docker file

```

FROM centos:7
ARG tomcat_version=8.5.6
RUN yum install -y epel-release java-1.8.0-openjdk.x86_64 wget
RUN groupadd tomcat && mkdir /opt/tomcat
RUN useradd -s /bin/nologin -g tomcat -d /opt/tomcat tomcat
WORKDIR /
RUN wget https://archive.apache.org/dist/tomcat/tomcat-
8/v$tomcat_version/bin/apache-tomcat-$tomcat_version.tar.gz
RUN tar -zxvf apache-tomcat-$tomcat_version.tar.gz -C /opt/tomcat --strip-
components=1
RUN cd /opt/tomcat && chgrp -R tomcat conf
RUN chmod g+rw /opt/tomcat/conf && chmod g+r /opt/tomcat/conf/*
RUN chown -R tomcat /opt/tomcat/logs/ /opt/tomcat/temp /opt/tomcat/webapps
/opt/tomcat/work
RUN chgrp -R tomcat /opt/tomcat/bin && chgrp -R tomcat /opt/tomcat/lib && chmod
g+rw /opt/tomcat/bin && chmod g+r /opt/tomcat/bin/*

```

```
WORKDIR /opt/tomcat/webapps
#RUN wget https://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/sample.war
EXPOSE 8080
CMD ["/opt/tomcat/bin/catalina.sh","run"]
```

```
#docker image build -t rajivsiddiqui/tomcat:v2 --build-arg tomcat_version=8.5.8 .
```

Build Context

Build Context

Dockerfile

```
FROM ubuntu
RUN apt-get update
RUN apt-get install python
RUN pip install flask
RUN pip install flask-mysql
COPY . /opt/source-code
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

Docker Daemon

Docker CLI

/opt/my-custom-app

```
docker build . -t my-custom-app
```

```
docker build /opt/my-custom-app
```

Build Context

Dockerfile

```
FROM ubuntu
RUN apt-get update
RUN apt-get install python
RUN pip install flask
RUN pip install flask-mysql
COPY . /opt/source-code
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

Docker Daemon

Docker CLI

/var/lib/docker/tmp/docker-builderxxxxx

/opt/my-custom-app

app.py

```
docker build . -t my-custom-app
```

```
docker build /opt/my-custom-app
```

Sending build context to Docker daemon 2.048kB
Step 1/7 : FROM ubuntu

.dockerignore

```
.dockerignore
```

```
tmp  
logs  
build
```

```
▶ docker build . -t my-custom-app
```

```
▶ docker build /opt/my-custom-app
```

```
Sending build context to Docker daemon 2.048kB  
Step 1/7 : FROM ubuntu
```

Docker Daemon

/var/lib/docker/tmp/docker-builderxxxx

Docker CLI

/opt/my-custom-app

app.py

tmp

logs

build

.dockerignore

Build Context

```
▶ docker build . -t my-custom-app
```

```
▶ docker build /opt/my-custom-app
```

```
Sending build context to Docker daemon 2.048kB  
Step 1/7 : FROM ubuntu
```

```
▶ docker build https://github.com/myaccount/myapp
```

```
▶ docker build https://github.com/myaccount/myapp#<branch>
```

```
▶ docker build https://github.com/myaccount/myapp:<folder>
```

```
▶ docker build -f Dockerfile.dev https://github.com/myaccount/myapp
```

Docker Daemon

/var/lib/docker/tmp/docker-builderxxxx

Docker CLI

/opt/my-custom-app

Image Cache

Build Cache

Dockerfile

```
FROM ubuntu

RUN apt-get update
RUN apt-get install -y python python3-pip

RUN pip3 install flask

COPY app.py /opt/source-code

ENTRYPOINT flask run
```

Layer 1. Base ubuntu Layer	120 MB
Layer 2. Update apt packages	22 MB
Layer 3. Install python and python pip	329 MB
Layer 4. Changes in pip packages	4.3 MB
Layer 5. Source code	229 B
Layer 6. Update Entrypoint with "flask" command	0 B

Build Cache

	Layer 1. Base ubuntu Layer	120 MB
cached	Layer 2. Update apt packages	22 MB
	Layer 3. Install python and python pip	329 MB
	Layer 4. Changes in pip packages	4.3 MB
	Layer 5. Source code	229 B
	Layer 6. Update Entrypoint with "flask" command	0 B

```
▶ docker build .
Sending build context to Docker daemon 2.048kB
Step 1/6 : FROM ubuntu
--> bb0eaf4eee00
Step 2/6 : RUN apt-get update
--> Using cache
--> e09e593ec730
Step 3/6 : RUN apt-get install -y python python-pip
--> Running in e9944225690a
Reading package lists...
Building dependency tree...
Reading state information...
E: Unable to locate package python-pip
The command '/bin/sh -c apt-get install -y python pyth
zero code: 100
```

Build Cache

Dockerfile

```
FROM ubuntu

RUN apt-get update
RUN apt-get install -y python python3-pip

RUN pip3 install flask flask-mysql

COPY [app.py] /opt/source-code

ENTRYPOINT flask run
```

	Layer 1. Base ubuntu Layer	120 MB
cached	Layer 2. Update apt packages	22 MB
cached	Layer 3. Install python and python pip	329 MB
invalid	Layer 4. Changes in pip packages	4.3 MB
invalid	Layer 5. Source code	229 B
invalid	Layer 6. Update Entrypoint with "flask" command	0 B

1. Compare instructions in Dockerfile
2. Compare checksums of files in ADD or COPY

Build Cache - Cache Busting

Dockerfile

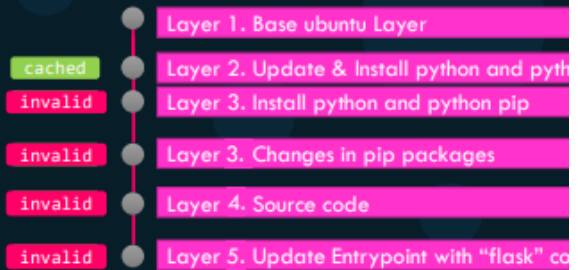
```
FROM ubuntu

RUN apt-get update && apt-get install -y \
    python python3-pip python-dev

RUN pip3 install flask flask-mysql

COPY app.py /opt/source-code

ENTRYPOINT flask run
```



Build Cache - Version Pinning

Dockerfile

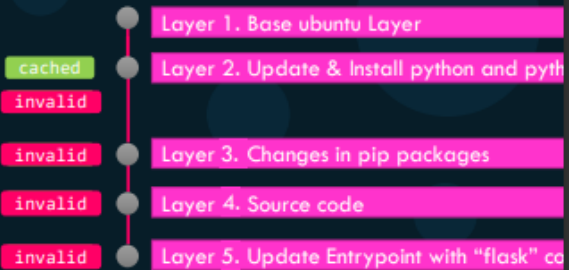
```
FROM ubuntu

RUN apt-get update && apt-get install -y \
    python \
    python-dev \
    python3-pip=20.0.2

RUN pip3 install flask flask-mysql

COPY app.py /opt/source-code

ENTRYPOINT flask run
```



Build Cache

Dockerfile

```
FROM ubuntu
```

```
RUN apt-get update && apt-get install -y \  
    python \  
    python-dev \  
    python3-pip=20.0.2
```

cached

```
RUN pip3 install flask flask-mysql
```

cached

```
COPY app.py /opt/source-code
```

invalid

```
ENTRYPOINT flask run
```

Build Cache

Dockerfile

```
FROM ubuntu
```

```
COPY app.py /opt/source-code
```

invalid

```
RUN apt-get update && apt-get install -y \  
    python \  
    python-dev \  
    python3-pip=20.0.2
```

invalid

```
RUN pip3 install flask flask-mysql
```

invalid

```
ENTRYPOINT flask run
```

Copy and ADD

Difference between COPY and ADD

Dockerfile

```
FROM centos:7
COPY /testdir /testdir
```

Dockerfile

```
FROM centos:7
ADD /testdir /testdir
```

Dockerfile

```
FROM centos:7
ADD app.tar.xz /testdir
```

Dockerfile

```
FROM centos:7
ADD http://app.tar.xz /testdir
RUN tar -xJf /testdir/app.tar.xz -C /tmp/app
RUN make -C /tmp/app
```

Copy or ADD?

Dockerfile

```
FROM centos:7
COPY /testdir /testdir
```

Dockerfile

```
FROM centos:7
ADD /testdir /testdir
```

Dockerfile

```
FROM centos:7
ADD app.tar.xz /testdir
```

Dockerfile

```
FROM centos:7
RUN curl http://app.tar.xz \
  | tar -xJ /testdir/file.tar.xz \
  && yarn build \
  && rm /testdir/file.tar.xz
```

Dockerfile

```
FROM centos:7
ADD http://app.tar.xz /testdir
RUN tar -xJf /testdir/app.tar.xz -C /tmp/app
RUN make -C /tmp/app
```

Docker CMD vs Entrypoint

```
▶ docker run ubuntu
```

```
▶ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
--------------	-------	---------	---------	--------	-------

```
▶ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
45aacca36850	ubuntu	"/bin/bash"	43 seconds ago	Exited (0) 41 seconds ago	

```

# Install Nginx.
RUN \
    add-apt-repository -y ppa:nginx/stable && \
    apt-get update && \
    apt-get install -y nginx && \
    rm -rf /var/lib/apt/lists/* && \
    echo "\ndaemon off;" >> /etc/nginx/nginx.conf && \
    chown -R www-data:www-data /var/lib/nginx

# Define mountable directories.
VOLUME ["/etc/nginx/sites-enabled", "/etc/nginx/certs", "/etc/nginx/conf.d"]

# Define working directory.
WORKDIR /etc/nginx

# Define default command.
CMD ["nginx"]

```

```

ARG MYSQL_SERVER_PACKAGE_URL=https://repo.mysql.com/yum/mysql-8.0-community/docker
ARG MYSQL_SHELL_PACKAGE_URL=https://repo.mysql.com/yum/mysql-tools-community/el/7

# Install server
RUN rpmkeys --import https://repo.mysql.com/RPM-GPG-KEY-mysql \
    && yum install -y $MYSQL_SERVER_PACKAGE_URL $MYSQL_SHELL_PACKAGE_URL libpq \
    && yum clean all \
    && mkdir /docker-entrypoint-initdb.d

VOLUME /var/lib/mysql

COPY docker-entrypoint.sh /entrypoint.sh
COPY healthcheck.sh /healthcheck.sh
ENTRYPOINT ["/entrypoint.sh"]
HEALTHCHECK CMD /healthcheck.sh
EXPOSE 3306 33060
CMD ["mysqld"]

```

```

# Pull base image.
FROM ubuntu:14.04

# Install.
RUN \
    sed -i 's/# \(\.*multiverse$\)/\1/g' /etc/apt/sources.list && \
    apt-get update && \
    apt-get -y upgrade && \
    apt-get install -y build-essential && \
    apt-get install -y software-properties-common && \
    apt-get install -y byobu curl git htop man unzip vim wget && \
    rm -rf /var/lib/apt/lists/*

# Add files.
ADD root/.bashrc /root/.bashrc
ADD root/.gitconfig /root/.gitconfig
ADD root/.scripts /root/.scripts

# Set environment variables.
ENV HOME /root

# Define working directory.
WORKDIR /root

# Define default command.
CMD ["bash"]

```

```
▶ docker run ubuntu [COMMAND]
```

```
▶ docker run ubuntu sleep 5
```



```
FROM Ubuntu
```

```
CMD sleep 5
```

```
CMD command param1
```

```
CMD sleep 5
```

```
CMD ["command", "param1"]
```

```
CMD ["sleep", "5"]
```

```
CMD ["sleep 5"]
```



```
▶ docker build -t ubuntu-sleeper .
```

```
▶ docker run ubuntu-sleeper
```



3

FROM Ubuntu

CMD sleep 5

```
docker run ubuntu-sleeper sleep 10
```

Command at Startup: sleep 10

FROM Ubuntu

ENTRYPOINT ["sleep"]

```
docker run ubuntu-sleeper 10
```

Command at Startup: sleep 10

```
docker run ubuntu-sleeper
```

```
sleep: missing operand  
Try 'sleep --help' for more information.
```

Command at Startup:sleep

FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]

```
docker run ubuntu-sleeper
```

```
sleep: missing operand  
Try 'sleep --help' for more information.
```

Command at Startup: sleep 5

```
docker run ubuntu-sleeper 10
```

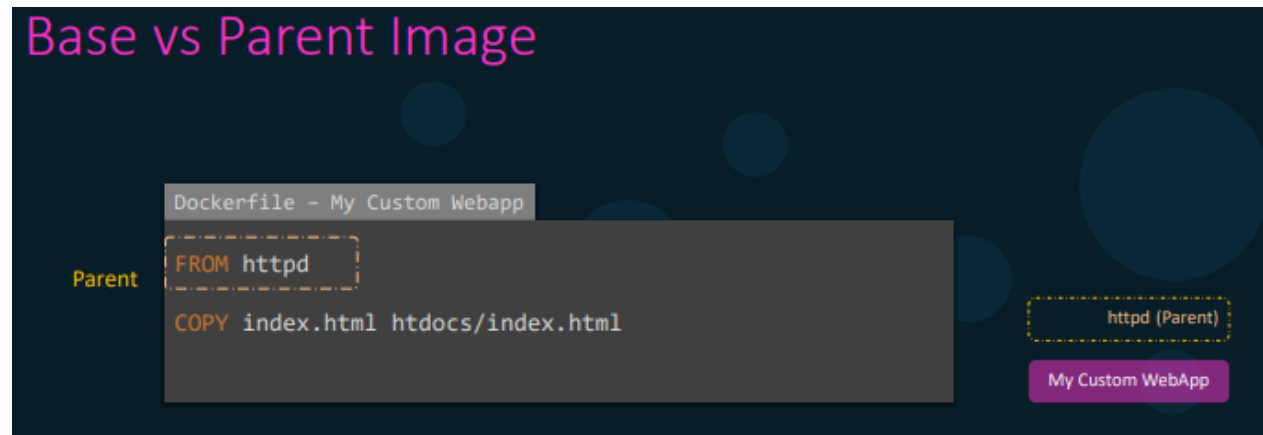
Command at Startup: sleep 10

```
docker run --entrypoint sleep2.0 ubuntu-sleeper 10
```

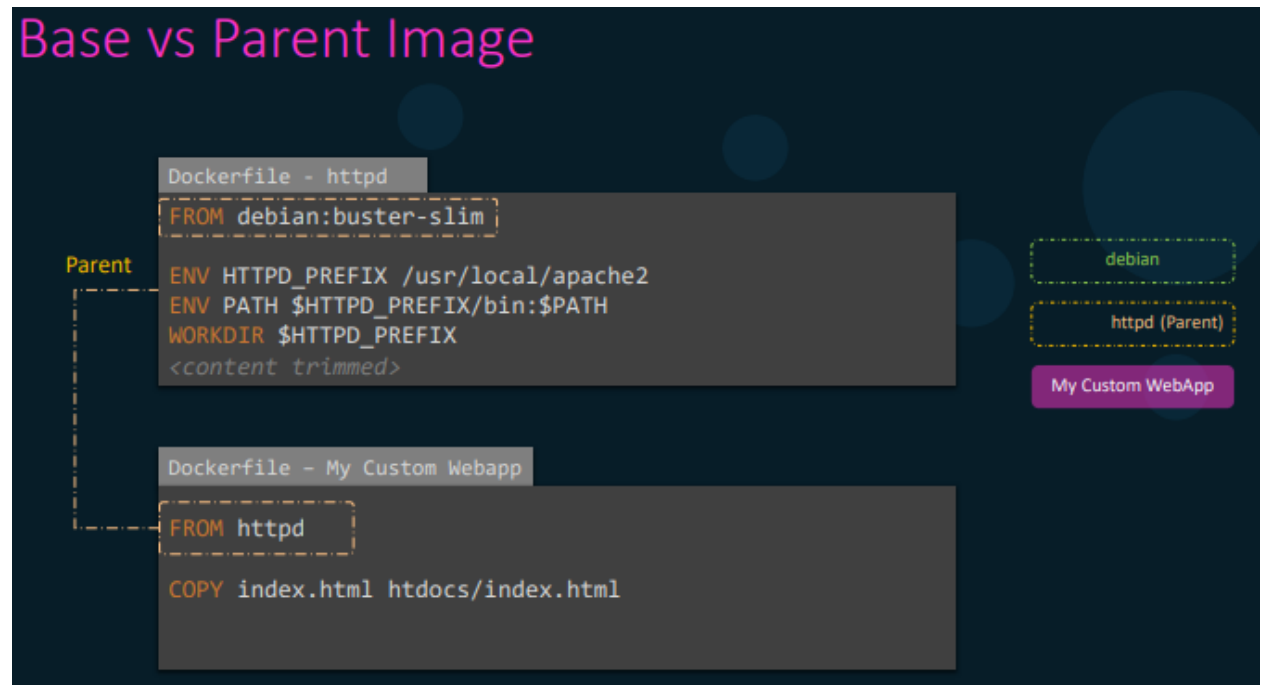
Command at Startup:sleep2.0 10

Docker Base Image

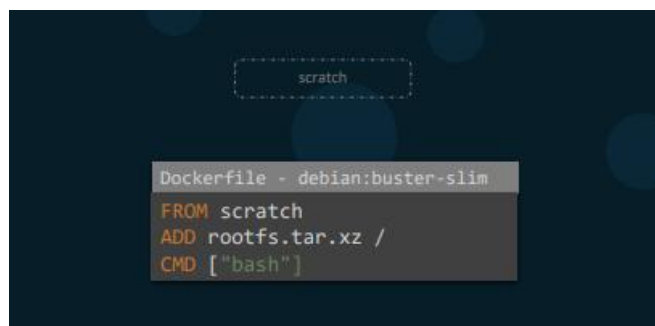
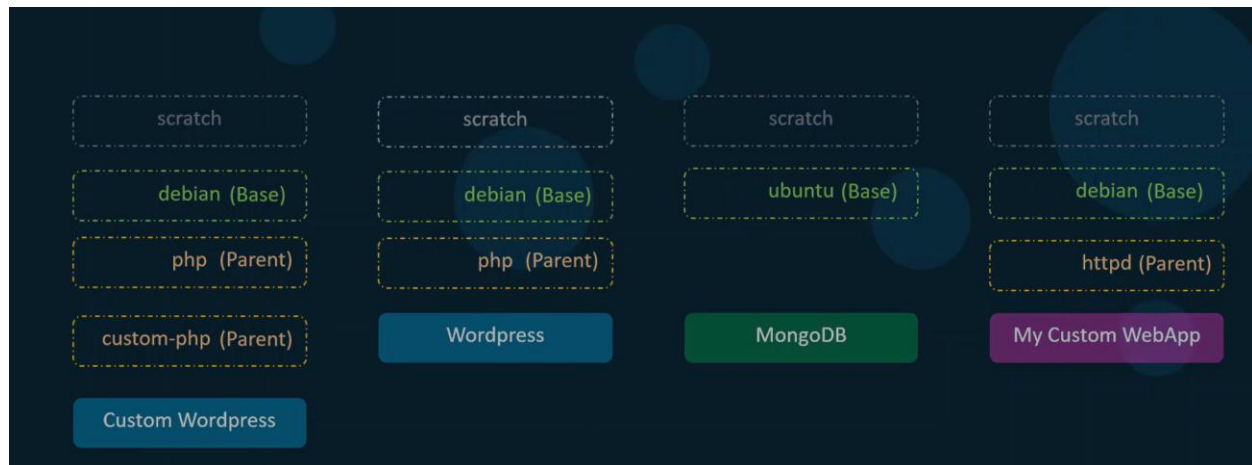
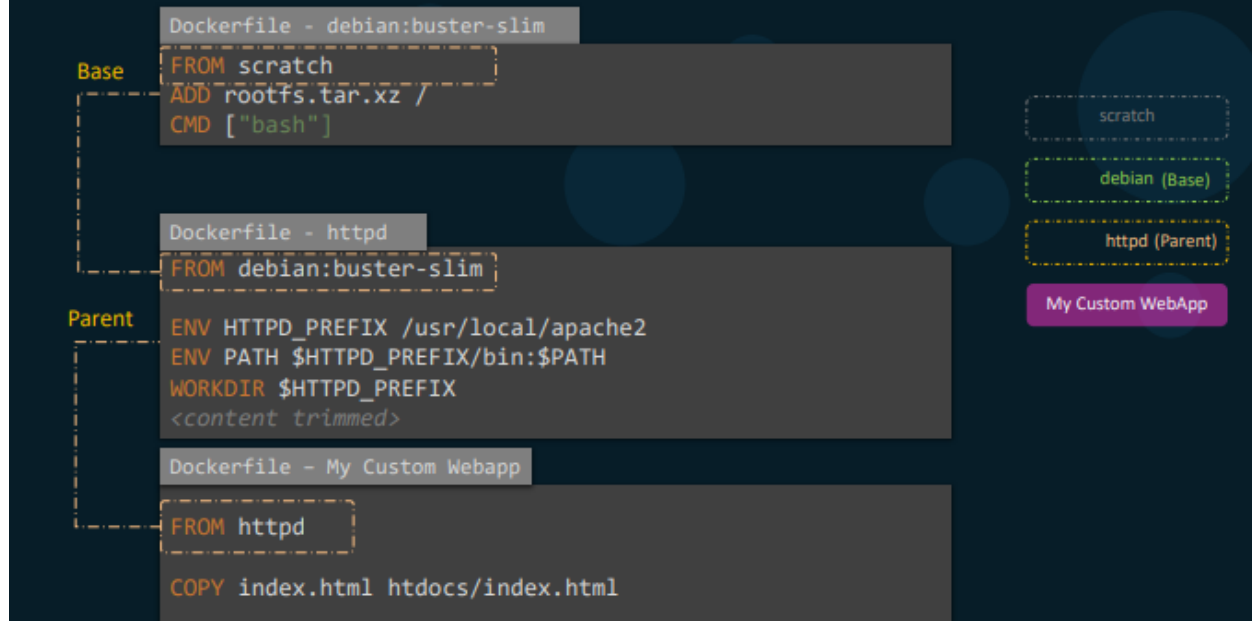
Base vs Parent Image



Base vs Parent Image



Base vs Parent Image



Multi Stage Build

Multi-stage builds

1. Build
Stage 0

Stage 1

3. Extract build from first image

3. Containerize for Production

```
Dockerfile
FROM node AS builder
COPY . .
RUN npm install
RUN npm run build

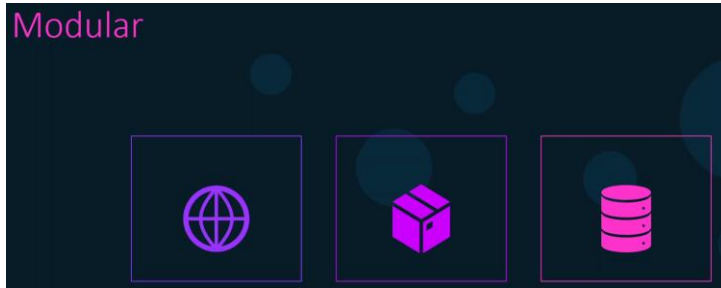
FROM nginx
COPY --from=builder dist /usr/share/nginx/html

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

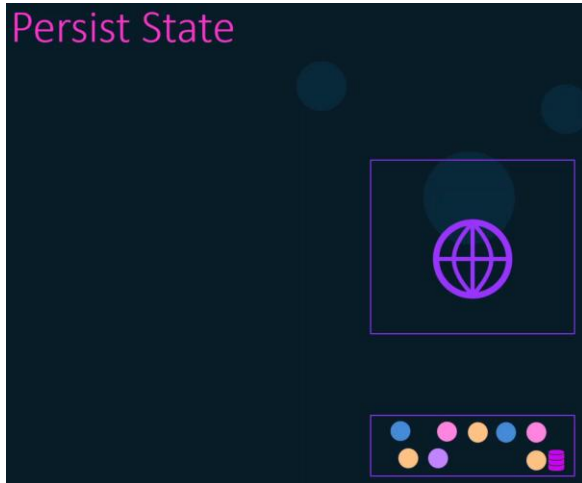
```
▶ docker build -t my-app .
▶ docker build --target builder -t my-app .
```

Best practice when create an image

Modular



Persist State



Slim/Minimal Images

1. Create slim/minimal images
2. Find an official minimal image that exists
3. Only install necessary packages
4. Maintain different images for different environments:
 - Development – debug tools
 - Production - lean
5. Use multi-stage builds to create lean production ready images.
6. Avoid sending unwanted files to the build context