Build Docker Image

Rajesh Kumar

DevOps Architect

@RajeshKumarIN | www.RajeshKumar.xyz

2 Way

Modify existing
Create New

Modify Existing

```
docker run -t -i training/sinatra /bin/bash
gem install json
exit
docker commit -m "Added json gem" -a
"Rajesh Kumar" 05e15b3555dd
scmgalaxy/sinatra:v2
docker run -t -i scmgalaxy/sinatra:v2
/bin/bash
```

Create new

```
mkdir sinatra1
cd sinatra1
touch Dockerfile

FROM ubuntu:14.04
MAINTAINER Rajesh Kumar <rajesh@scmgalaxy.com>
RUN apt-get update && apt-get install -y git
RUN mkdir /tmp/bmc.txt

docker build -t scmgalxy/sinatra1:v2 .
```

Another

```
> mkdir myapp1
> cd myapp1
> vi Dockerfile

FROM ubantu:latest
MAINTAINER rajesh@scmGalaxy.com
RUN apt-get install git
RUN mkdic /src

docker build -t myapp1 .
```

Exposing Ports

> vi Dockerfile

```
FROM ubuntu:15.04
RUN apt-get update && apt-get install -y iputils-ping traceroute apache2
EXPOSE 80
ENTRYPOINT ["apache2ctl"]
CMD ["-D", "FOREGROUND"]
```

```
> docker build -t="apache-img" .
> docker run -d -p 5001:80 --name web1 apache-img
> docker ps
```

Viewing Exposed Port

- # To see the port mapping
- docker port cont-id

To run the container with udp
- docker run -d -p 5002:80/udp --name=web2 apache-image
- docker port cont-id

To map with IP address of host with container
- docker run -d -p 192.168.56.50:5003:80 --name=web3 apache-image
- docker port cont-id

Exposing Multiple Ports

> vi Dockerfile

```
FROM ubuntu:15.04
RUN apt-get update && apt-get install -y iputils-ping traceroute apache2
EXPOSE 80 500 600  700  800 900
ENTRYPOINT ["apache2ctl"]
CMD ["-D", "FOREGROUND"]
```

```
> docker build -t="apache-img" .
> docker run -d -p 5001:80 --name web1 apache-img1
> docker ps
```

- docker run -d -P --name=web4 apache-image1
- docker port web4

Command Description

ADD Copies a file from the host system onto the container
CMD The command that runs when the container starts
ENTRYPOINT
ENV Sets an environment variable in the new container
EXPOSE Opens a port for linked containers
FROM The base image to use in the build. This is mandatory and must be the first command in the file.

Command Description

MAINTAINER An optional value for the maintainer of the script
ONBUILD A command that is triggered when the image in the Dcokerfile is used as a base for another image
RUN Executes a command and save the result as a new layer
USER Sets the default user within the container
VOLUME Creates a shared volume that can be shared among containers or by the host machine
WORKDIR Set the default working directory for the container
Once you've created a Dockerfile and added all your instructions, you can use it to build an image

Reference

https://docs.docker.com/engine/tutorials/dockerimages/

Dockerfile

Plain-text

Simple format

Instructions to build image


Docerkfile

Docerkfile

FROM 123
INSTRUCTION abc
INSTRUCTION def
INSTRUCTION ghi
INSTRUCTION jkl

Docerkfile

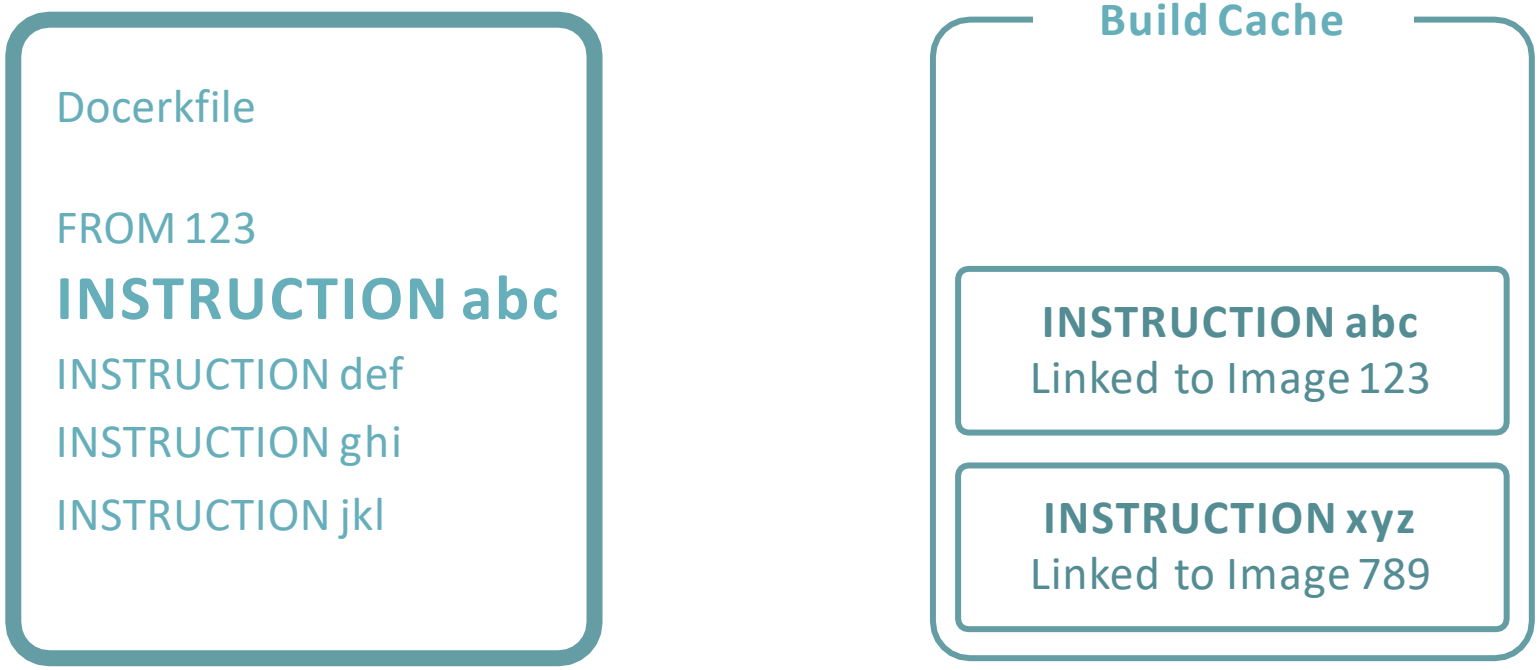
FROM 123

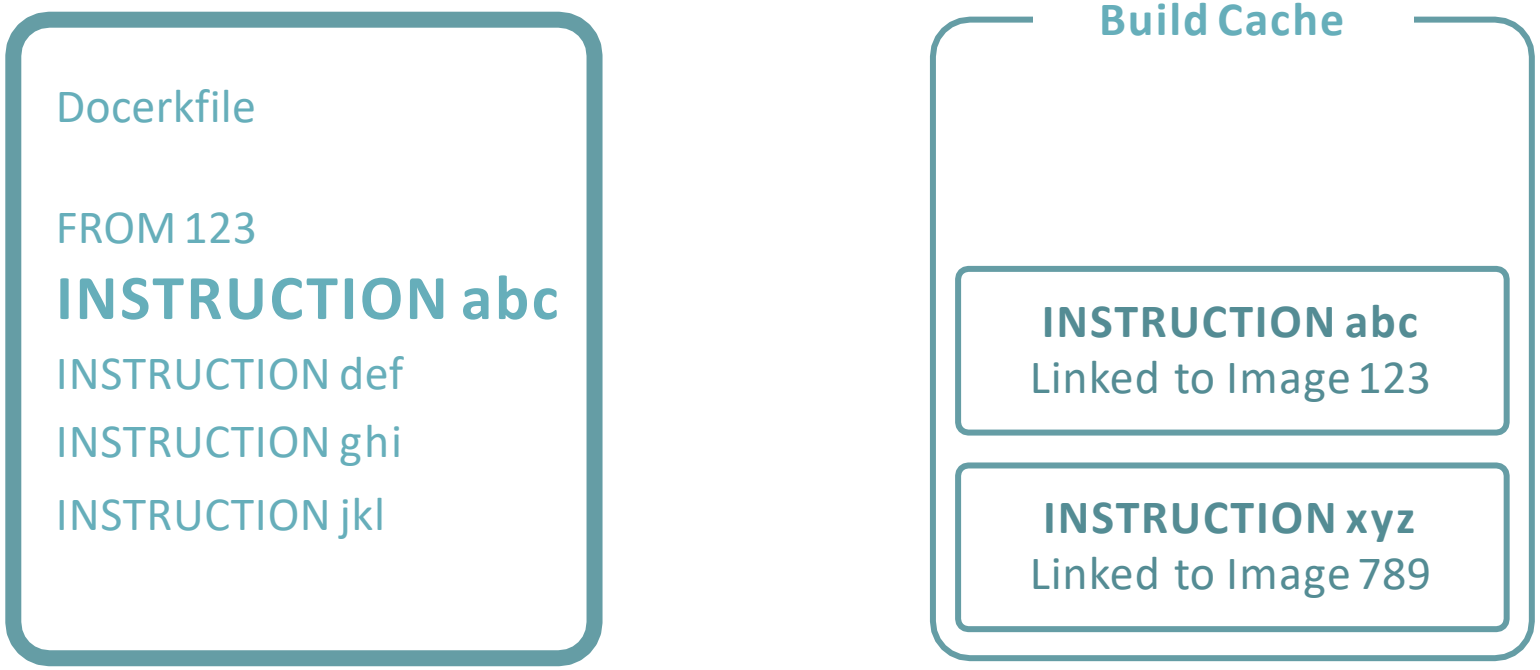
INSTRUCTION abc

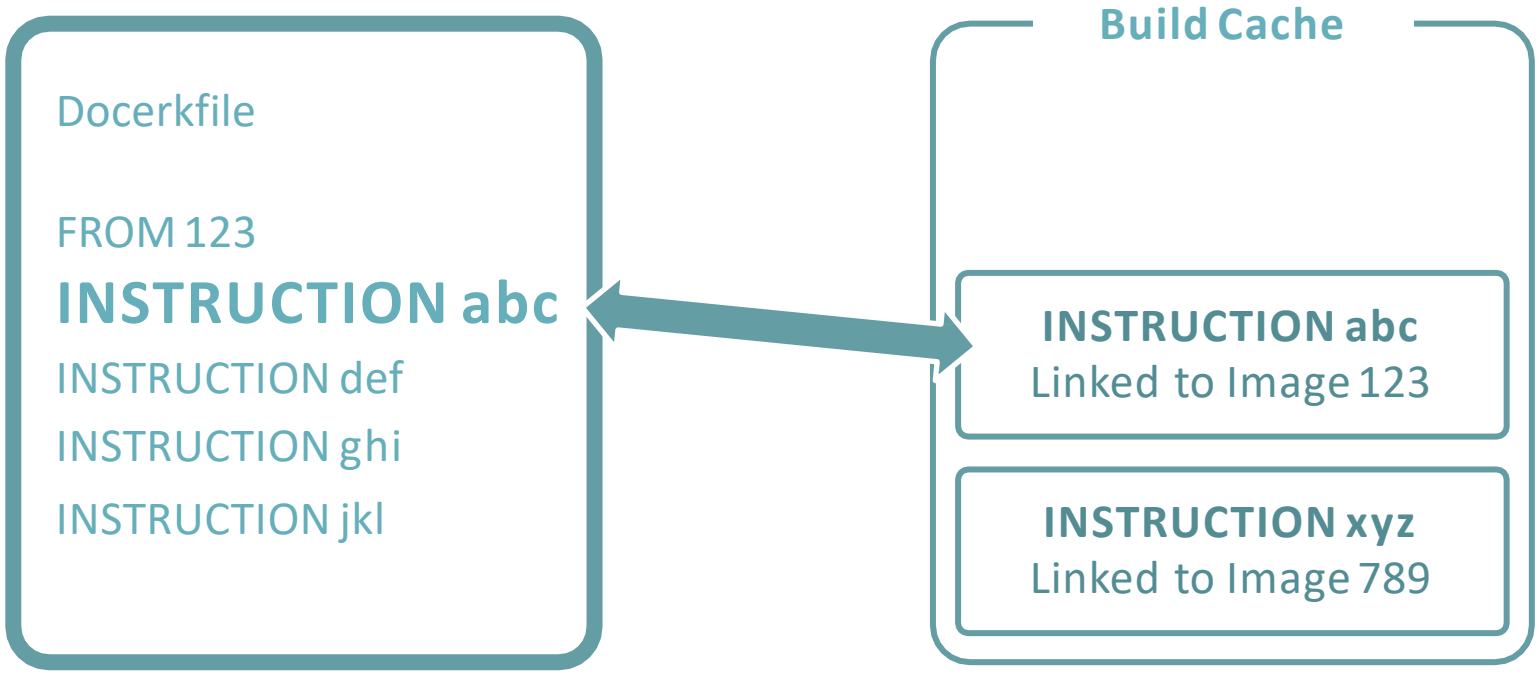
INSTRUCTION def

INSTRUCTION ghi

INSTRUCTION jkl


Docerkfile
FROM 123
INSTRUCTION abc
INSTRUCTION def
INSTRUCTION ghi
INSTRUCTION jkl
Build Cache
INSTRUCTION abc
Linked to Image 123
INSTRUCTION xyz
Linked to Image 789


Docerkfile
FROM 123
INSTRUCTION abc
INSTRUCTION def
INSTRUCTION ghi
INSTRUCTION jkl
Build Cache
INSTRUCTION abc
Linked to Image 123
INSTRUCTION xyz
Linked to Image 789


Docerkfile
FROM 123
INSTRUCTION abc
INSTRUCTION def
INSTRUCTION ghi
INSTRUCTION jkl
Build Cache
INSTRUCTION abc
Linked to Image 123
INSTRUCTION xyz
Linked to Image 789

CMD

Run-time

Run commands in containers at launch time

Equivalent of -

```
docker run <args> <command>
docker run <args> /bin/bash
```

One per Dockerfile

RUN

Build-time

Add layers to images

Used to install apps

Shell Form

Commands are expressed the same way as shell commands

Commands get prepended by “`/bin/sh -c`”

Variable expansion `etc`…

Exec Form

JSON array style -

`[“command”,“arg1”]`

Containers don’t need a shell

Avoids string munging by the shell

No shell features –

No variable expansion

No special characters (`&&, ||, >….`)

ENTRYPOINT

Can’t be overridden at run-time with normal commands – **`docker run … <command>`**

Any command at run-time is used as an argument to ENTRYPOINT

Variable expansion etc…

ADD	COPY(Recommended)
	COPY does a straight-forward, as-is copy of files and folders from the build context into the container.
The ADD instruction allows you to use a URL as the <src> parameter. When a URL is provided, a file is downloaded from the URL and copied to the <dest>. ADD http://foo.com/bar.go /tmp/main.go	COPY doesn't support URLs as a <src> argument so it can't be used to download files from remote locations. Anything that you want to COPY into the container must be present in the local build context.
ADD is the ability to automatically unpack compressed files. If the <src> argument is a local file in a recognized compression format (tar, gzip, bzip2, etc) then it is unpacked at the specified <dest> in the container's filesystem. ADD /foo.tar.gz /tmp/	Also, COPY doesn't give any special treatment to archives. If you COPY an archive file it will land in the container exactly as it appears in the build context without any attempt to unpack it.
Interestingly, the URL download and archive unpacking features **cannot** be used together. Any archives copied via URL will NOT be automatically unpacked.	COPY is really just a stripped-down version of ADD that aims to meet the majority of the "copy-files-to-container" use cases without any surprises.

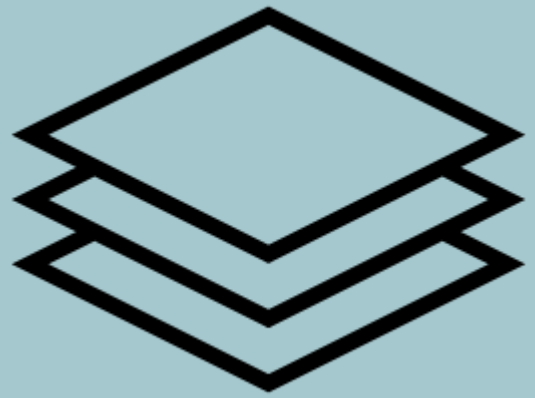
Module Recap

Build Cache

INSTRUCTION abc
Linked to Image 123

INSTRUCTION xyz
Linked to Image 789

Build Cache



RUN apt-get update
RUN apt-get install –y abc
RUN apt-get install –y def

```
RUN apt-get update \
    &&apt-get install –y \
    abc \
    def
```

Dockerfile and Layers

Dockerfile

CMD /usr/sbin/apache2ctl

ENTRYPOINT
/usr/sbin/apache2ctl

ENV var=value

VOLUMES /data

Dockerfile Instructions