accenture technology

Learning and Knowledge Management

Module 5: Managing Images

Module Objectives

At the end of this module, you will be able to:

- Describe build images by committing changes in containers and writing Dockerfile
- Explain how to Dockerize an application
- Explore options to manage collaborations in private repository

Topic List

- Building Images
- Dockerizing Applications
- Managing Repositories

Topic List

Building Images

Dockerizing Applications

Managing Repositories

Build Images (1)

Layers Functionality: Images Layers

- Each Docker image references a list of read-only layers
- Layers are stacked on top of each other to form a base for a container's root filesystem

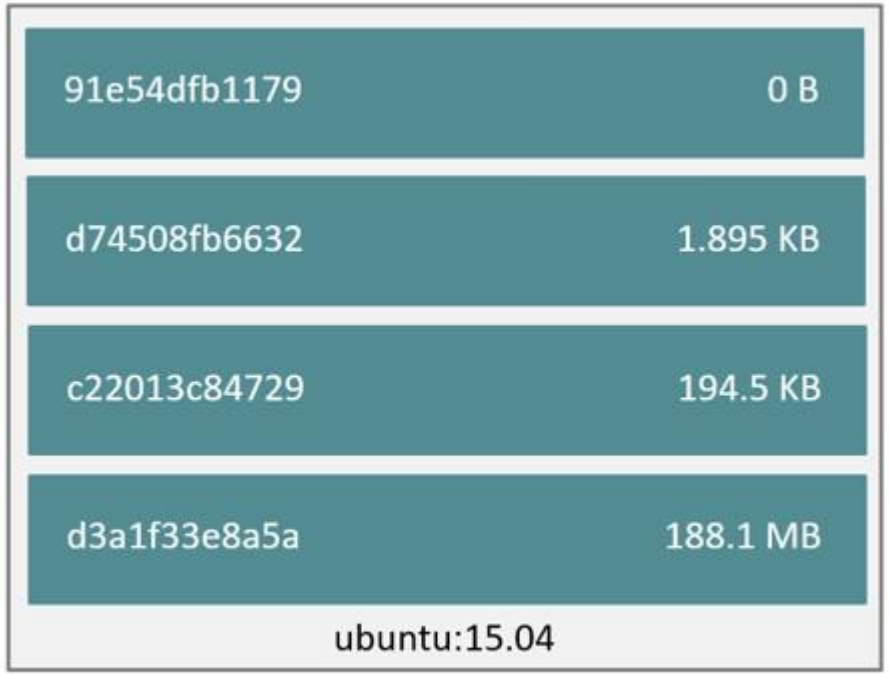


Image courtesy: Docker
https://docs.docker.com

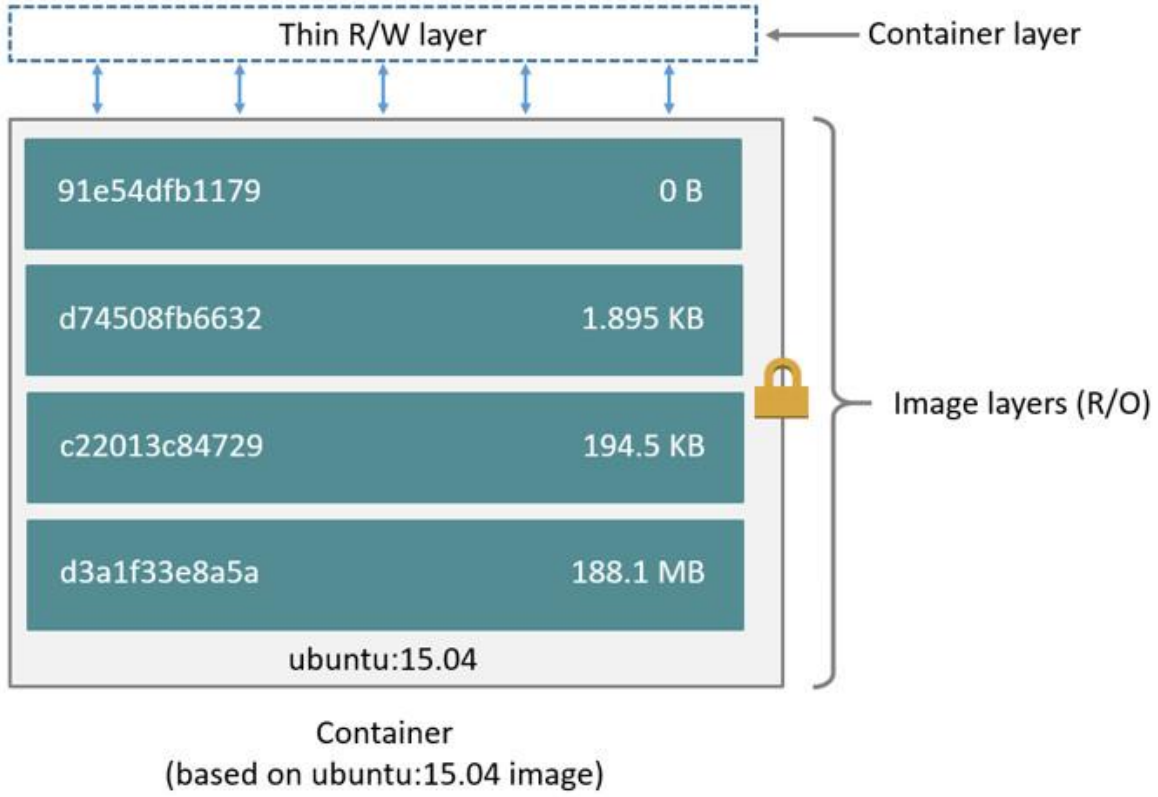
Build Images (2)

Layers Functionality: Container Layer

Container has a new, thin, writable layer on top of the underlying stack called the “container layer”

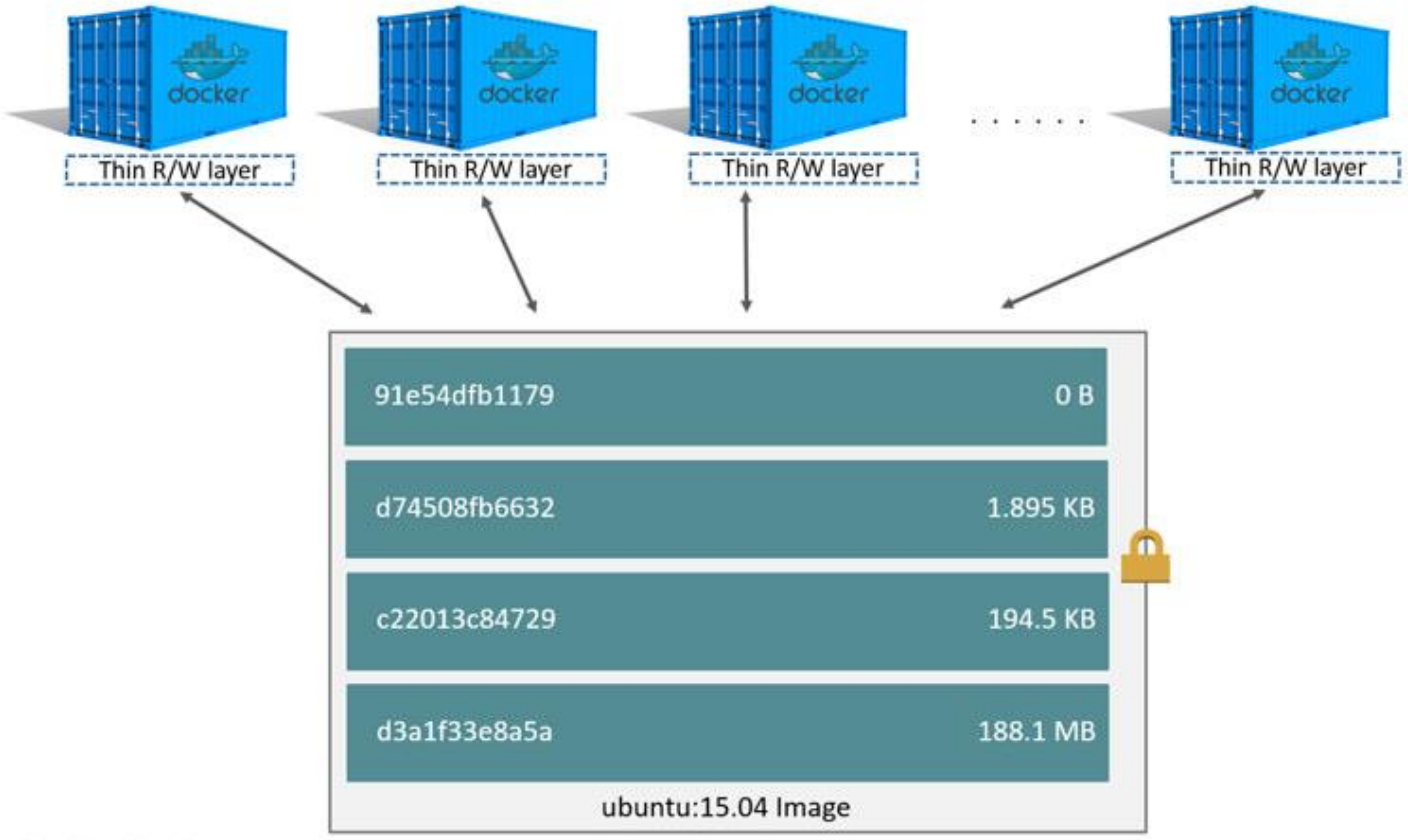
Changes made to the running container are written to this layer

Example: writing new files, modifying existing files, and deleting files



Build Images (3)

Layers Functionality: Sharing Layers Between Images



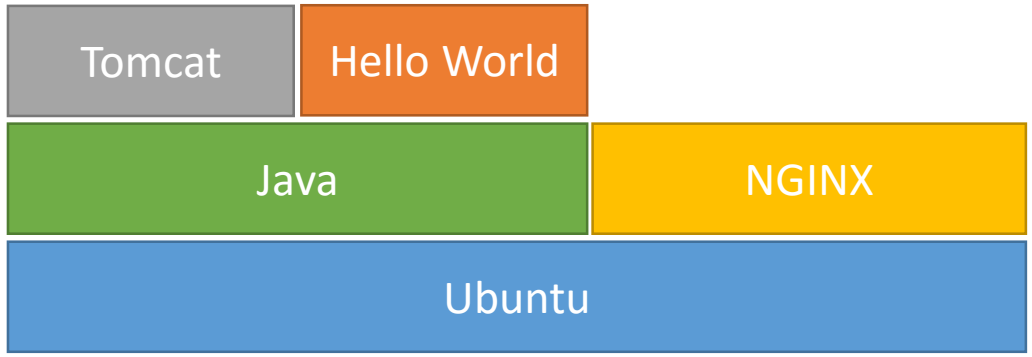
Build Images (4)

Layers Functionality: Sharing Layers Between Images

Images share the layers with each other; speeds up the transfer of image

Optimizes disk and memory usage

Parent image which is already available need not to be downloaded



Build Images (5)

Copy-on-write Strategy

Processes share common instance of the data

If one process needs to modify or write to the data

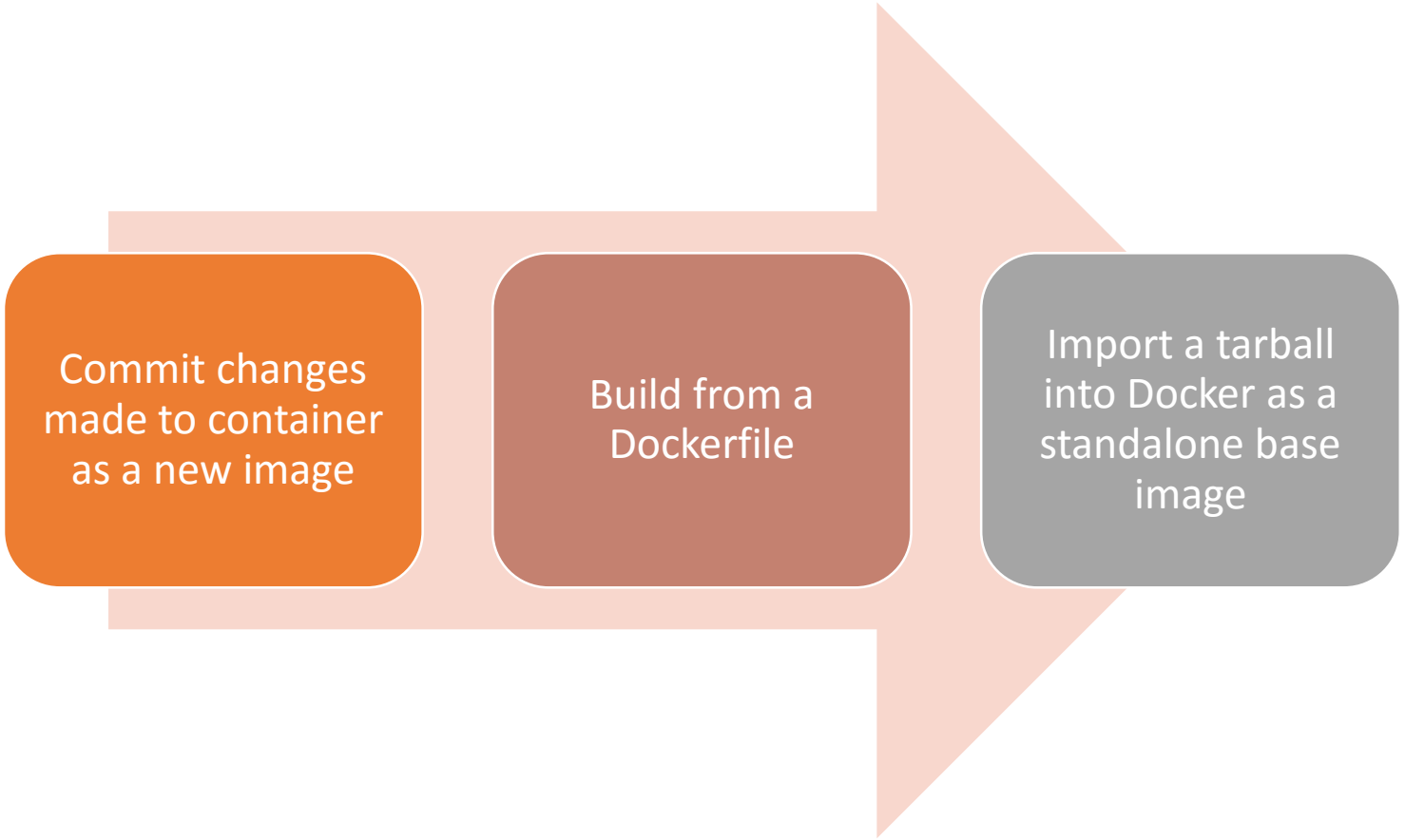
- Make a copy of the data

Only that specific process has access to the data copied

- All the other processes continue to use the original data

Build Images (6)

Building Images: An Overview



Build Images (7)

Commit Changes

Description
Create a new image from container's file changes or settings
Ensures interactive way of building image

Usage
docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]

Example
docker run -i -t ubuntu:14.04 bash
apt-get update (to refresh the packages)
apt-get install -y wget vim (install wget and vim)
exit
docker commit <containerID> <yourName>/myapp:1.0

Build Images (9)

Compare Changes

Description

- Inspect changes to files or directories on a container’s filesystem
- List the changed files and directories in a container's filesystem since the container was created.

Usage

docker diff

Example

docker diff <containerID>

Build Images (8)

Commit Changes: Options

Name	Description
--author , -a	Author (e.g., “John Hannibal Smith hannibal@a-team.com”)
--change , -c	Apply Dockerfile instruction to the created image
--message , -m	Commit message
--pause , -p	Pause container during commit

Exercise 18

Let’s practice what we have learned so far!

Perform the following steps to commit changes in Docker:

- docker commit <containerID> <yourName>/myapp:1.0
- docker images
- docker run -i -t <yourName>/myapp:1.0 bash
- which vim (verify vim installation)
- which wget (verify wget installation)
- touch test
- ls
- exit
- docker ps -a
- docker diff <containerID>
- docker commit <containerID> <yourName>/myapp:1.1

Build Images (10)

Dockerfile

- Docker can build images automatically by reading the instructions from a Dockerfile
- A Dockerfile is a text document that contains all the commands to assemble an image
- Using docker build users can create an automated build that executes several command-line instructions in succession
- Using docker is more effective than committing the changes

Build Images (11)

Dockerfile Commands

FROM

- Sets the Base Image for subsequent instructions
- A valid Dockerfile must have FROM as its first instruction
- The image can be any valid image

RUN

- Will execute any commands in a new layer on top of the current image and commit the results
- The resulting committed image will be used for the next step in the Dockerfile

Build Images (12)

Dockerfile: Multiple Commands In Single RUN

Use && to concatenate multiple RUN commands; this will reduce the number of layers

Example:

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

Note: The balance between readability of the Dockerfile and minimizing the number of layers it uses is a must.

Build Images (13)

Build Command

Description
Builds Docker images from a Dockerfile and a “context”
A build’s context is the files located in the specified PATH or URL

Usage
docker build [OPTIONS] PATH | URL | -
Option -t for specifying Name and optionally a tag in the ‘name:tag’ format

Example
docker build -t myimage

Exercise

Let's practice what we have learned so far!

Perform the following steps to build image in Docker:

- root@ubuntu: /home/docker5/myimage#vi Dockerfile
- Content in Dockerfile:
  - ✓ FROM ubuntu:14.04
  - ✓ RUN apt-get update
  - ✓ RUN apt-get install -y wget
- docker build -t myimage . (. means use Dockerfile from current directory)
- docker images
- docker run myimage which wget (run and check installation of wget)

Build Images (14)

.dockerignore File

- Before sending the context to the docker daemon CLI
  - ✓ Looks for a file named .dockerignore in the root directory of the context
  - ✓ Modifies context to exclude files and directories that match patterns in it
- This avoids unnecessary files and directories in images
- .dockerignore file is optional

Build Images (15)

Build Cache

Description

Build an image from a Dockerfile

- During the process of building an image Docker
- Will step through the instructions in Dockerfile
- Execute each in the order specified
- Will look for an existing image in its cache that it can reuse
- Rather than creating a new (duplicate) image

Usage

- Use the --no-cache=true option to avoid use of cache
- It is used while specifying build command
- docker build [OPTIONS] PATH | URL | -

Exercise 20

Let's practice what we have learned so far!

Perform the following steps to build cache in Docker:

- docker build -t myimage . (observe the use of cache)
- modify Dockerfile to install vim
  - ✓ FROM ubuntu14:04
  - ✓ RUN apt-get update
  - ✓ RUN apt-get install -y wget
  - ✓ RUN apt-get install -y vim
- docker build -t myimage:1.0 .
- Edit the Dockerfile and change the sequence of wget and vim
  - ✓ FROM ubuntu14:04
  - ✓ RUN apt-get update
  - ✓ RUN apt-get install -y vim
  - ✓ RUN apt-get install -y wget
- docker build -t myimage:1.0 .

Build Images (16)

Build History

Description
Show the history of an image

Usage
docker history [OPTIONS] IMAGE

Example
docker history myimage:1.0

Build Images (17)

Build History: Options

Name	Description
--format	Pretty-print images using a Go template
--human , -H	Print sizes and dates in human readable format
--no-trunc	Don’t truncate output
--quiet , -q	Only show numeric IDs

Exercise 21

Let’s practice what we have learned so far!

Perform the following steps to build history in Docker:

- docker history myimage:1.0
- note the imageID of the layer with apt-get update command
- change the Dockerfile to combine vim and wget (RUN apt-get install -y wget vim)
- docker history myimage:1.0

Topic List

Building Images

Dockerizing Applications

Managing Repositories

Dockerizing Applications (1)

An Overview

Dockerizing an **application** is the procedureof transforming an **application** to run within a Docker container. If the container has already been created, you can add the code in the container and deploy it.

Exercise 22

Let's practice what we have learned so far!

Perform the following steps to dockerize application:

- touch Dockerfile
- add following contents
  - ✓ FROM java:7
  - ✓ COPY HelloWorld.java /
  - ✓ RUN javac HelloWorld.java
  - ✓ ENTRYPOINT ["java", "HelloWorld"]
- build the image → docker build -t javahelloworld:1.0
- docker run javahelloworld:1.0

Dockerizing Applications (2)

CMD Command

Description

- CMD provide defaults for an executing container
- There can only be one CMD instruction in a Dockerfile

Usage

The CMD has three forms:

- CMD["executable","param1"," param2"] (exec form, preferred)
- CMD command param1 param2 (shell form)
- CMD ["param1","param2"] (as default parameters to ENTRYPOINT)

Example

CMD [“ping”, “127.0.0.1”, “-c”, “30”]

Exercise 23

Let’s practice what we have learned so far!

Perform the following steps to dockerize applications using CMD:

- Add following line to end of Dockerfile
  - ✓ CMD [“ping”, “127.0.0.1”, “-c”, “30”]
- docker build -t <yourname>/myimage:1.0 .
- docker run <yourname>/myimage:1.0
- docker run <yourname>/myimage:1.0 echo “Hello World” (cmd is overridden)

Dockerizing Applications (2)

Entrypoint

Description

An ENTRYPOINT allows to configure a container that will run as an executable

Usage

ENTRYPOINT has two forms:

- ENTRYPOINT ["executable", "param1", "param2"] (exec form, preferred)
- ENTRYPOINT command param1 param2 (shell form)

Example

ENTRYPOINT [“ping”, “127.0.0.1”, “-c”, “30”]

Exercise 24

Let’s practice what we have learned so far!

Perform the following steps to dockerize applications using ENTRYPOINT:

- Change the Dockerfile to
  - ✓ ENTRYPOINT [“ping”] (instead of CMD)
- docker build -t <yourname>/myimage:1.0 .
- docker run <yourname>/myimage:1.0
- docker run <yourname>/myimage:1.0 127.0.0.1

Dockerizing Applications (2)

Cmd Vs. Entrypoint

Description

- Both CMD and ENTRYPOINT instructions define what command gets executed when running a container
- Dockerfile should specify at least one of CMD or ENTRYPOINT

Usage

When using the container as an executable CMD:

- As a way of defining default arguments for an ENTRYPOINT
- For executing an ad-hoc command in a container

CMD will be overridden when running the container with alternative arguments

Example

NEED AN EXAMPLE HERE

Exercise 25

Let’s practice what we have learned so far!

Perform the following steps to dockerize applications using CMD and ENTRYPOINT:

- Change the Dockerfile to
  - ✓ CMD [“127.0.0.1”]
  - ✓ ENTRYPOINT [“ping”, “-c”, “30”]
- docker build -t <yourname>/myimage:1.0 .
- docker run <yourname>/myimage:1.0
- docker run <yourname>/myimage:1.0 localhost

Dockerizing Applications (2)

ADD Command

Description

The ADD instruction copies:

- New files, directories or remote file URLs
- From source and
- Adds them to the image filesystem destination path

Usage

ADD has two forms:

- ADD <src>... <dest>
- ADD ["<src>",... "<dest>"] (for paths containing whitespace)

Example

ADD hom* /mydir

Dockerizing Applications (2)

COPY Command

Description

The COPY instruction copies:

- New files, directories
- From source and
- Adds them to the image filesystem destination path

Usage

COPY has two forms:

- COPY <src>... <dest>
- COPY ["<src>",... "<dest>"] (For paths containing whitespace)

Example

COPY hom* /mydir/

Dockerizing Applications (2)

Working Directory

Description

- The WORKDIR instruction sets the working directory
- If the WORKDIR doesn't exist, it will be created even if it's not used in any subsequent Dockerfile instruction

Usage

- Used by RUN, CMD, ENTRYPOINT, COPY and ADD instructions
- WORKDIR /path/to/workdir

Example

WORKDIR/home/root/javahelloworld

Exercise 26

Let's practice what we have learned so far!

Perform the following steps to specify working directory:

- add following command in Dockerfile after COPY command (before RUN)
  - ✓ WORKDIR /home/root/javahelloworld
  - ✓ Add following command to Dockerfile after WORKDIR
    - ➢ RUN mkdir bin
- docker build -t javahelloworld:1.0
- docker run javahelloworld:1.0
- Look inside the container
  - ✓ docker run -it --entrypoint bash javahelloworld:1.0
  - ✓ ls

Dockerizing Applications (2)

Setting Environment

Description

- The ENV instruction sets the environment variable <key> to the value <value>
- This value will be in the environment of all “descendant” Dockerfile commands

Usage

The ENV instruction has two forms:

- ENV <key> <value>
- ENV <key>=<value> ...

Example

ENV JAVA_HOME "/usr/lib/jvm/open-jdk"

Topic List

Building Images

Dockerizing Applications

Managing Repositories

Managing Repositories

Repositories On Docker Hub

Docker Hub repositories let you share images with co-workers, customers, or the Docker community at large

Images can be pushed to a Docker Hub repository

If the source code for image is on GitHub or Bitbucket

"Automated build" facility can be used to update repository

Managing Repositories

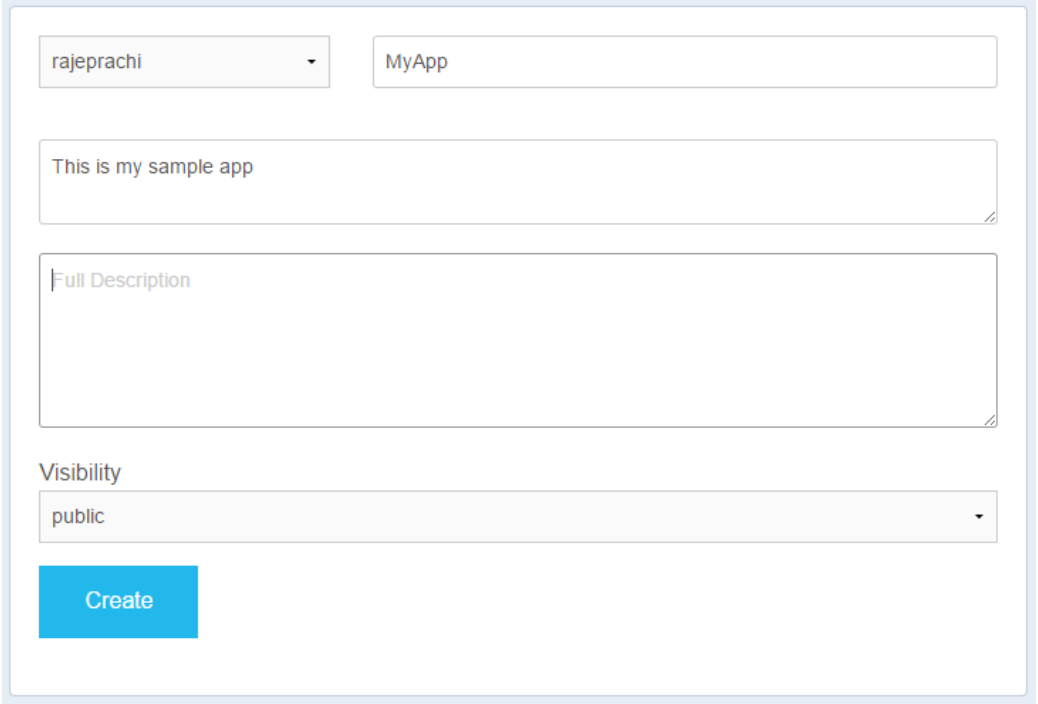
Creating New Repository

Login to hub.docker.com

Click on “Create Repository +” button

Name must be same as image name you built

Provide name and other details



Managing Repositories

Login To Hub From CLI

Description
- Log in to a Docker registry
- Default server is hub.docker.com

Usage
- docker login[OPTIONS] [SERVER]
- Options -e (email), -u (username), -p (password)

Managing Repositories

Tagging Image

Description

docker tag

Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Usage

docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]

Example

docker tag helloworldapp:1.0 <username>/helloworldapp:1.0

Managing Repositories

Pushing Image To Registry

Description

- docker push
- Push an image or a repository to a registry

Usage

docker push [OPTIONS] NAME[:TAG]

Example

docker push <username>/helloworldapp:1.0

Exercise 27

Let’s practice what we have learned so far!

Perform the following steps to push image to registry:

- docker login
- docker tag javahelloworld:1.0 <username>/javahelloworld:1.0
- docker push <username>/javahelloworld:1.0

  verify on hub

Managing Repositories

Collaborators

A collaborator is one who has been given access to a private repository

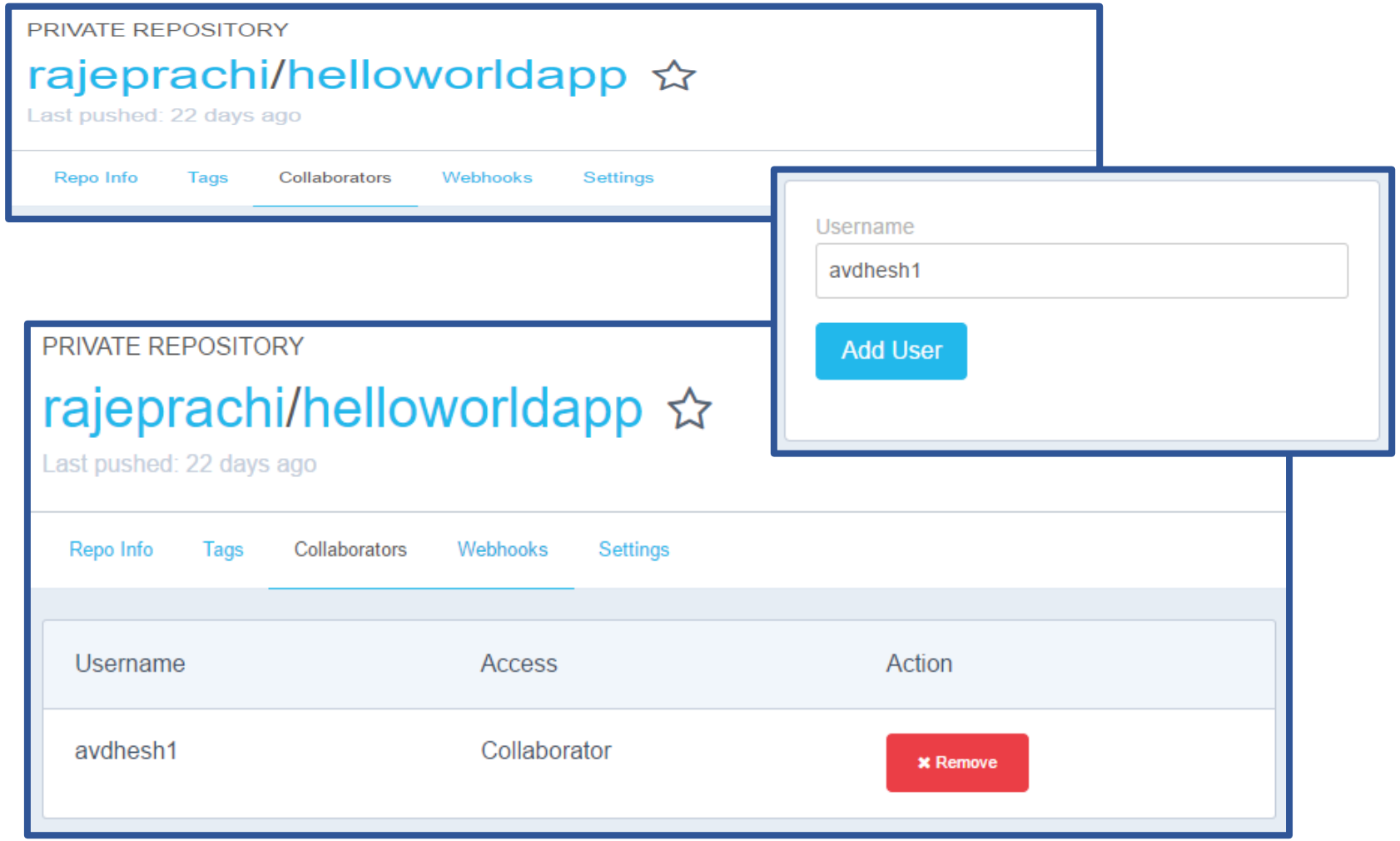
Once designated, they can push and pull to your repositories. They will not be allowed to perform any administrative tasks

Administrative tasks include:

- Deleting the repository
- Changing its status from private to public
- Adding new collaborator

Managing Repositories

Adding Collaborators



Exercise 28

Let’s practice what we have learned so far!

Perform the following steps to add a collaborator:

- docker pull <xyz user>/myapplication:1.0

Managing Repositories

Deleting Local Images

Description
docker rmi
Remove one or more images

Usage
- docker rmi [OPTIONS] IMAGE [IMAGE...]
- Option -f for Force removal of the image

Example
docker rmi <username>/myapplication:1.0

Exercise 29

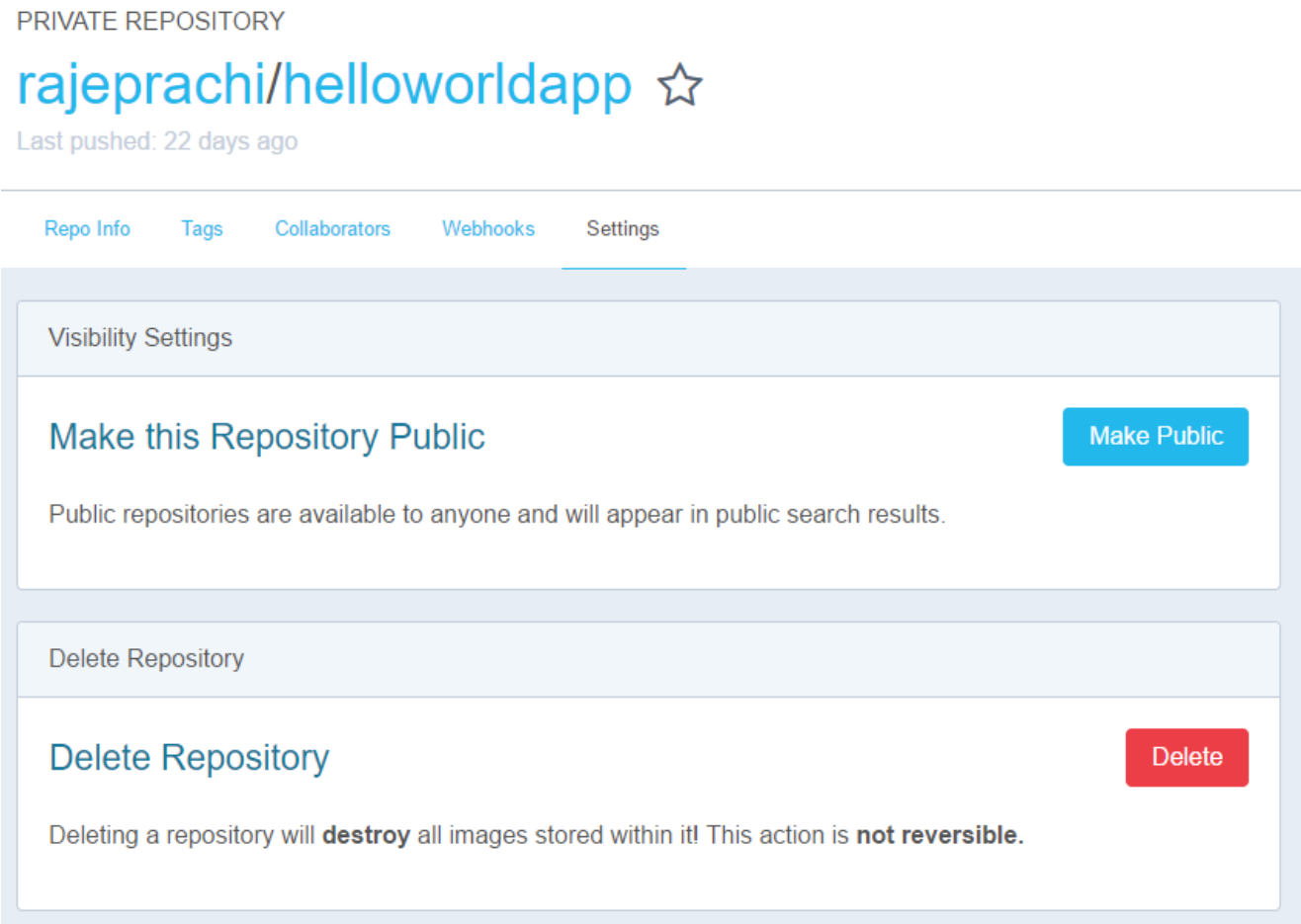
Let’s practice what we have learned so far!

Perform the following steps to delete local images:

- docker rmi <username>/myapplication:1.0

Managing Repositories

Deleting Repositories



Module Summary

Now, you should be able to:

- Describe build images by committing changes in containers and writing Dockerfile
- Explain how to Dockerize an application
- Explore options to manage collaborations in private repository

Course Summary

Now, you should be able to:

- Describe infrastructure automation, concept of evolving and immutable infrastructure, virtual machines and containers
- Explain Docker containers, its features, components and benefits of Docker
- Explain the Docker architecture and it's components
- Explain Docker installation—codes before and during installation and during testing
- Explain container processes and the key tasks associated with it; managing containers
- Describe Docker image and its attributes; build images by committing changes in containers
- Explain how to Dockerize an application

Thank You