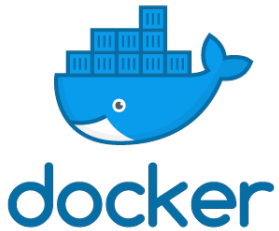
Deploying Web Apps and Web Services with Docker

Dr Mohammed Kaleem



Overview

- Introduction to Docker
  - What is it?
  - Why use it?
  - How to use it?

Docker

“Developing apps today requires so much more than writing code. Multiple languages, frameworks, architectures, and discontinuous interfaces between tools for each lifecycle stage creates enormous complexity. **Docker simplifies** and accelerates your workflow, while giving developers the freedom to innovate with their choice of tools, application stacks, and deployment environments for each project.” - Docker (2020)

What is Docker?

- In a normal virtualized environment, one or more virtual machines run on top of a physical machine using a hypervisor like Xen, Hyper-V, etc.
- Containers, on the other hand, run in userspace on top of operating systems kernel.
- You can **think of it as OS-level virtualization**. **Each container** will have its **isolated user space** and **you can run multiple containers** on a **host**, each having its own userspace.
- Each container just runs as a “thread/process” on the host operating system.

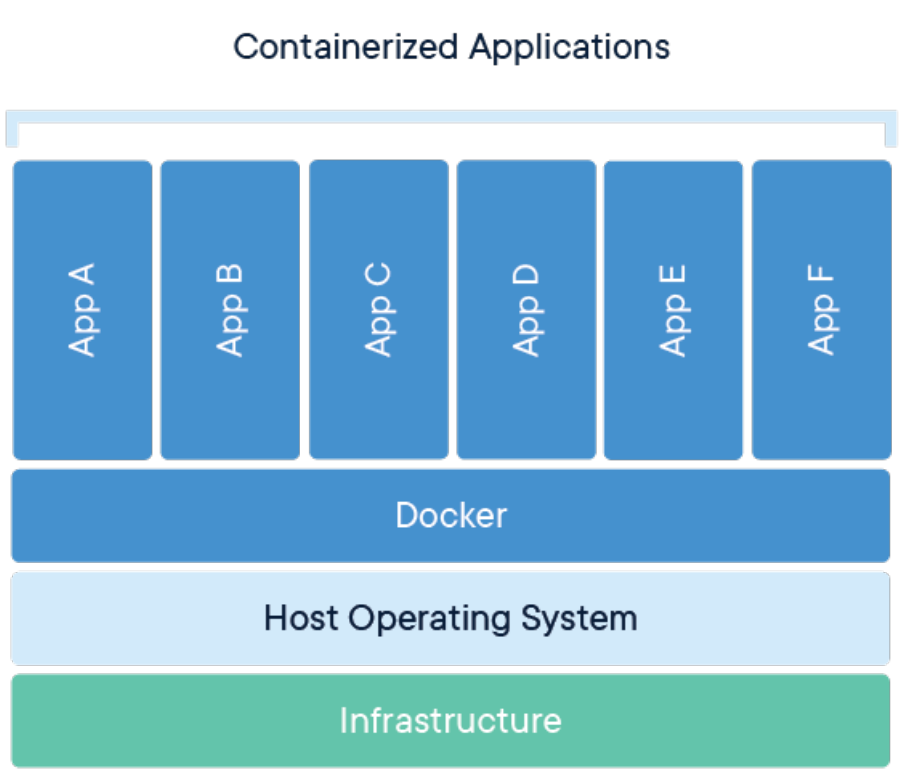
VM But NOT VM

- In a way, Docker is a **bit** like a virtual machine.
- But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer.
- This gives a significant performance boost and reduces the size of the application.

Moreover

- No need for a hypervisor, or provisioning/sharing hardware resources.
- Containers just run like any other application on the host OS.
- Use as much hardware (RAM, CPU, DISK) as required.

Instead of virtualizing hardware, containers rest on top of a single Linux instance. This means you can "*leave behind the useless 99.9 percent VM junk, leaving you with a small, neat capsule containing your application*" James Bottomley (CTO @ Parcels)



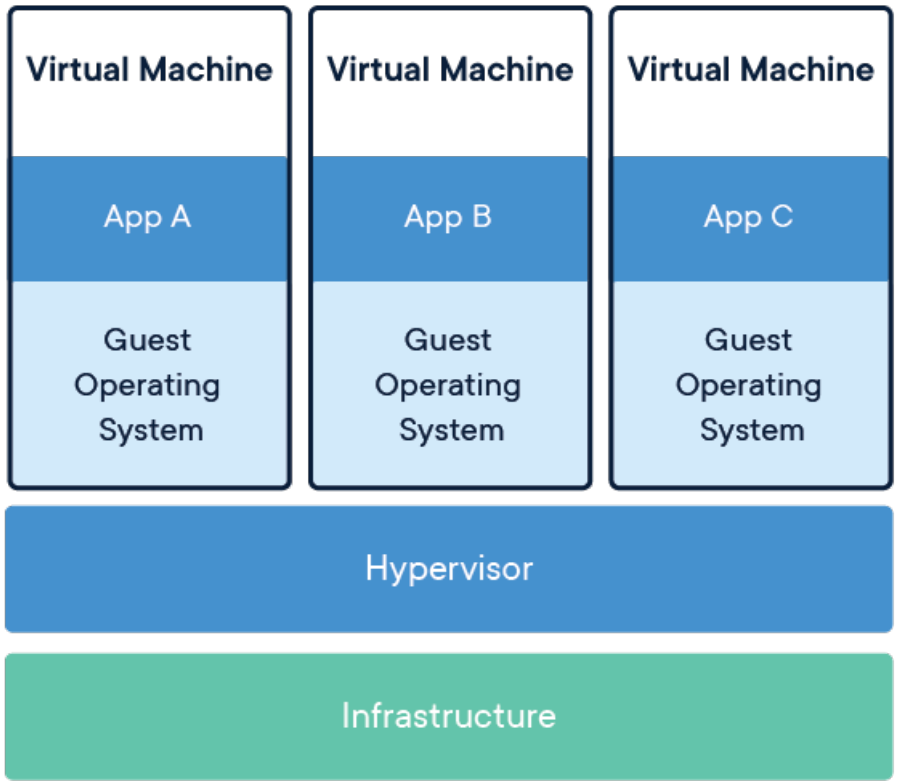


Image: https://www.docker.com/

Why use it?

Docker is a tool that is designed to benefit both developers and system administrators, making it a part of many DevOps toolchains.

For developers, it means that they can focus on writing code without worrying about the system that it will ultimately be running on.

It also allows them to get a head start by using one of thousands of programs already designed to run in a Docker container as a part of their application.

For operations staff, Docker gives flexibility and potentially reduces the number of server systems needed because of its small footprint and lower overhead.

Read more: http://opensource.com/resources/what-docker

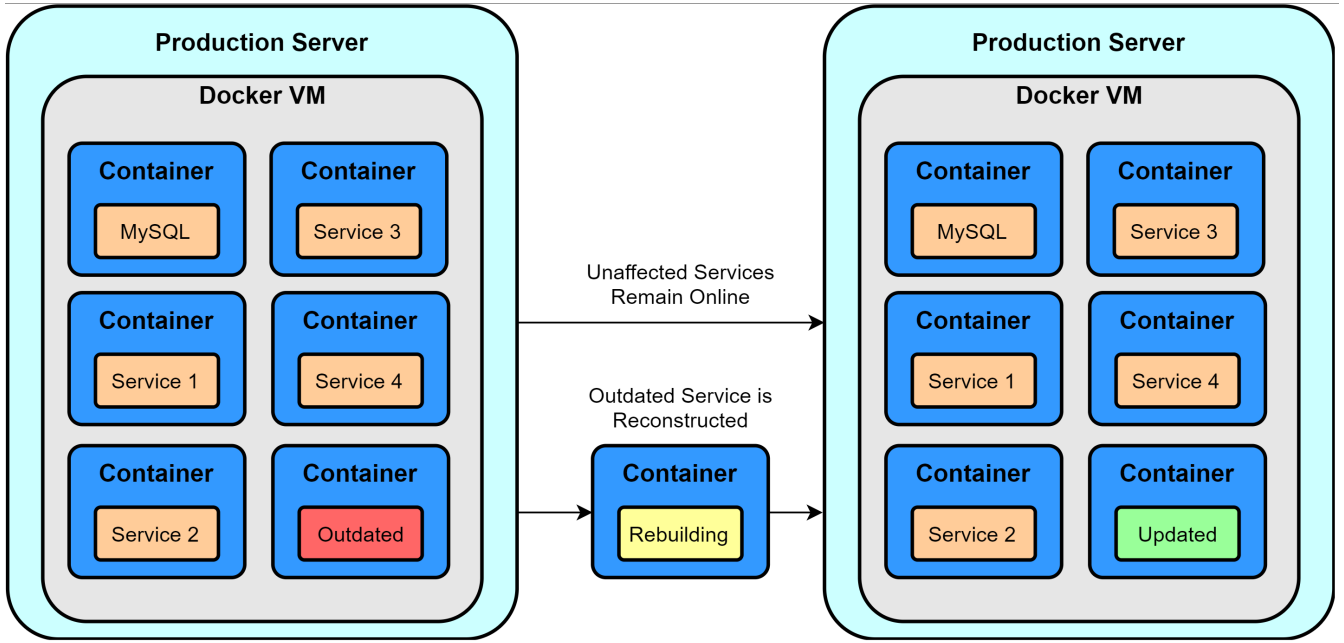
Containers

With DOCKER, you can treat containers like extremely lightweight, modular virtual machines. And you get flexibility with those containers—you can create, deploy, copy, and move them from environment to environment, which helps optimize your apps for the cloud.

https://www.redhat.com/en/topics/containers/what-is-docker

Docker and Microservices

- Docker allows each service to run in its own container
- Meaning you can take down, update/adapt and redeploy a container without taking down the whole application.

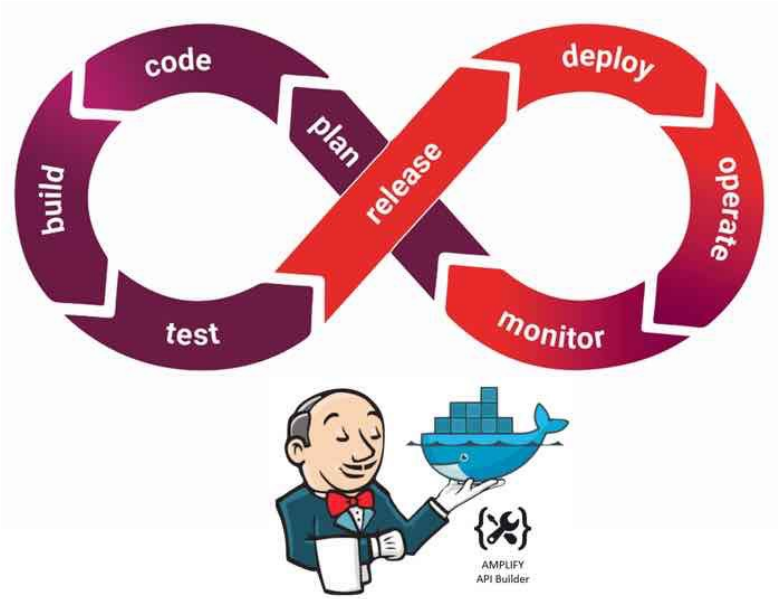


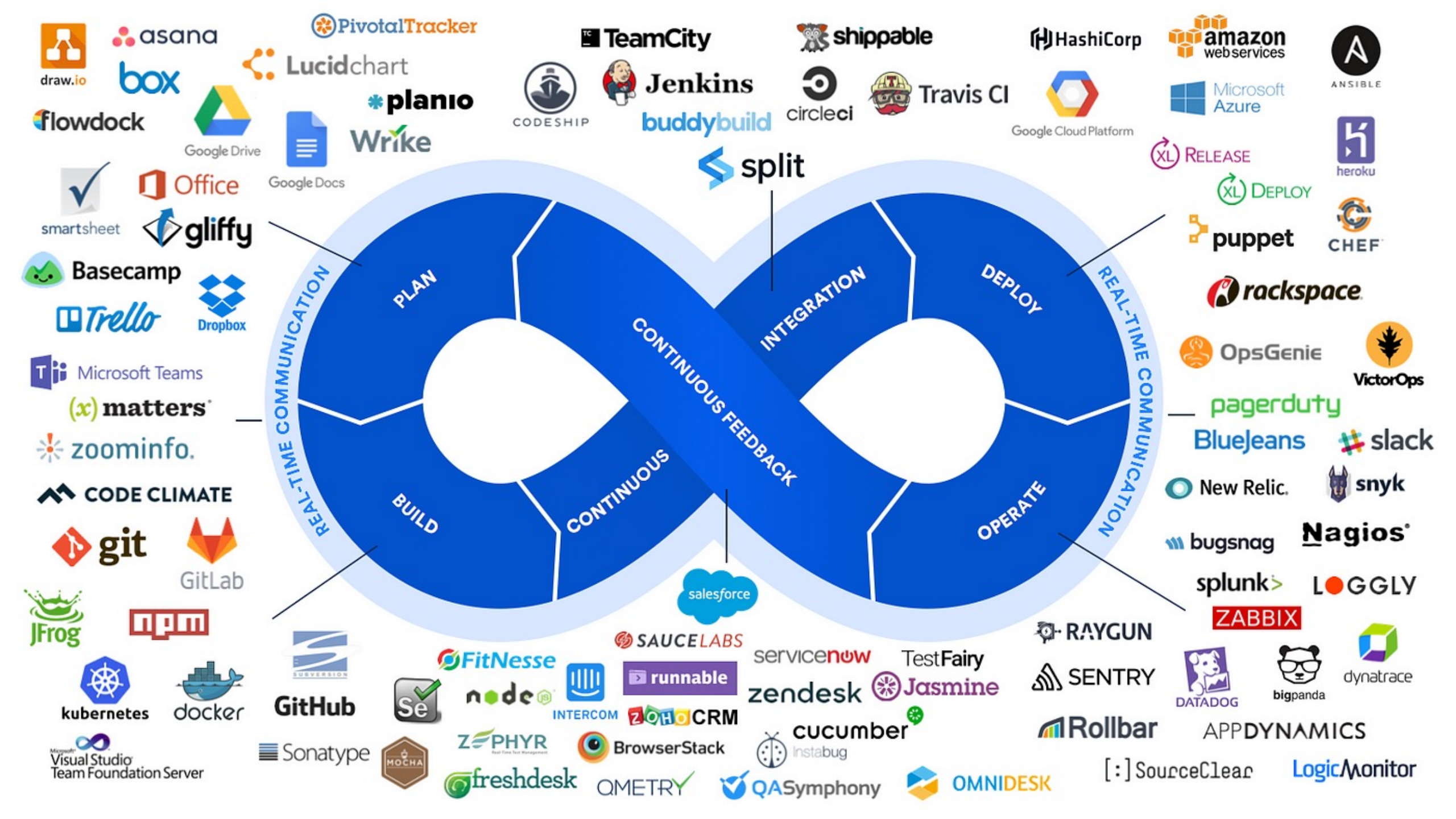
Continuous Integration and Continuous Delivery

CI/CD introduces ongoing automation and continuous monitoring throughout the lifecycle of apps, from integration and testing phases to delivery and deployment.

Docker supports the CI/CD model as docker containers can be built from a docker file (describes how a container is built) and deployed automatically tested and deployed to the cloud.

Docker + GitHub + Jenkins + AWS




PLAN
CONTINUOUS
BUILD
CONTINUOUS FEEDBACK
INTEGRATION
DEPLOY
OPERATE
REAL-TIME COMMUNICATION
REAL-TIME COMMUNICATION
draw.io
asana
PivotalTracker
Lucidchart
box
planio
flowdock
Google Drive
Google Docs
Wrike
Office
smartsheet
gliffy
Basecamp
Trello
Dropbox
Microsoft Teams
(x) matters
zoominfo.
CODE CLIMATE
git
GitLab
JFrog
npm
kubernetes
docker
Visual Studio Team Foundation Server
GitHub
Sonatype
Se
MOCHA
FitNesse
node
INTERCOM
runnable
ZOHO CRM
ZEPHYR
BrowserStack
freshdesk
QMETRY
SAUCELABS
salesforce
servicenow
TestFairy
zendesk
Jasmine
cucumber
Instabug
QASymphony
OMNIDESK
TeamCity
CODESHIP
Jenkins
buddybuild
shippable
circleci
Travis CI
split
HashiCorp
amazon web services
Google Cloud Platform
Microsoft Azure
ANSIBLE
RELEASE
heroku
DEPLOY
puppet
CHEF
rackspace
OpsGenie
VictorOps
pagerduty
BlueJeans
slack
New Relic.
snyk
bugsnag
Nagios
splunk>
LOGGLY
ZABBIX
RAYGUN
SENTRY
DATADOG
bigpanda
dynatrace
Rollbar
APPDYNAMICS
[:] SourceClear
LogicMonitor

Getting started with Docker

Install the Docker Engine

- Free (open source)
- Available for windows, linux and Mac OS.
  - In window docker runs in a Linux VM.

Creating Containers

- Containers are created from Images.
- Pick an image:
  - Literally millions to chose from.
  - Search containers here: https://hub.docker.com/
  - Some are created and maintained by major software companies/vendors
    - Microsoft
    - Ubuntu (You can create an Ubuntu server container!)
    - Oracle
    - MySQL, POSTGRES, REDIS
    - NodeJS
    - Java
    - Tomcat
- Depends on the technology stack of your application

TRY THIS IN THE LABS

MySQL Container Example

- Once docker is installed you can run the following command from the command line/terminal:

docker run --name mysql -p 3306:3306 –e MYSQL_ROOT_PASSWORD=password mysql

docker run --name [container name]-p port mappings –e environment variables image to build from

This will create a new local docker container called mysql from the official mysql image

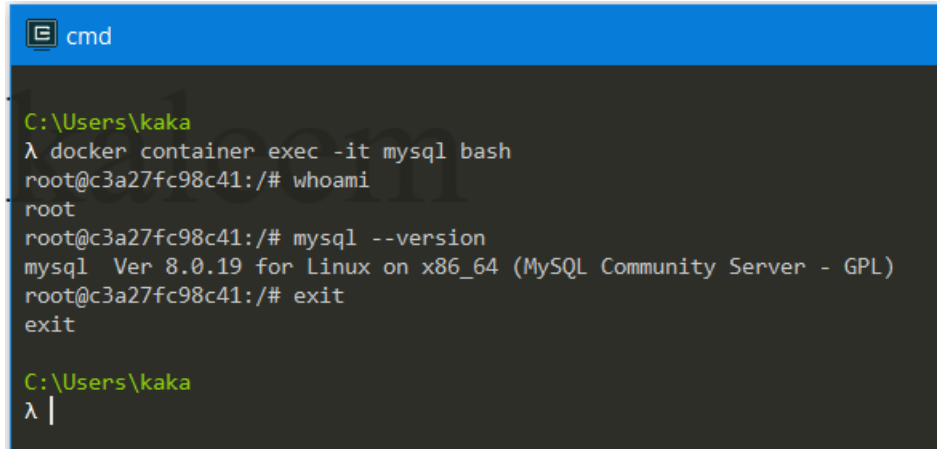
You can then start the container with:

docker start mysql

Bash/SSH into a container

Since a docker container is similar to a VM with its own file system, you can connect to it through “ssh” with the following command:

docker container exec -it [container_name] bash



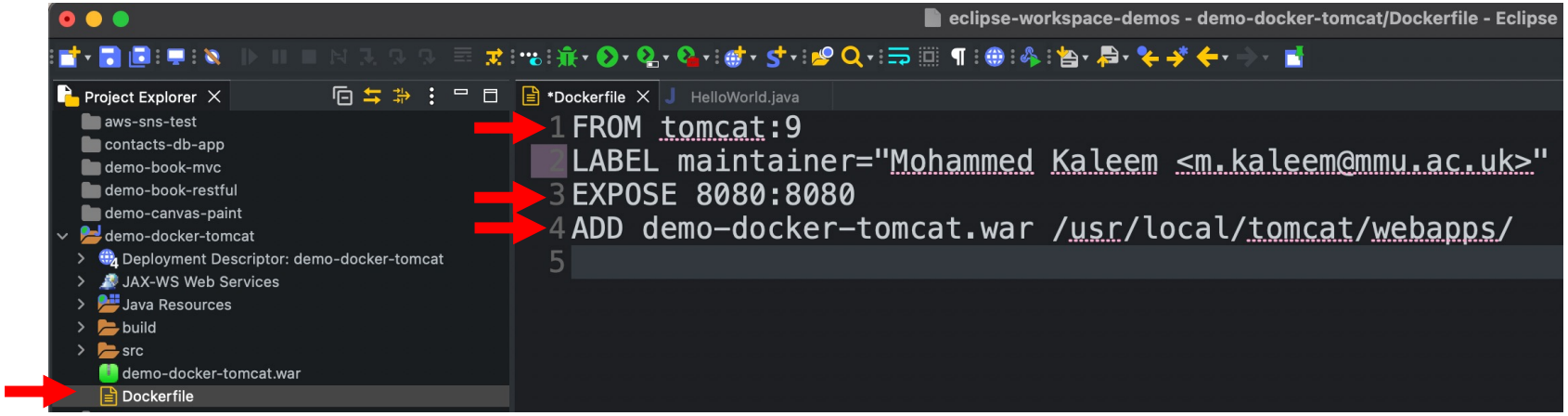
Dockerfiles

- Docker can build images automatically by reading the instructions from a Dockerfile.
- A Dockerfile is a text document that contains all the instructions to assemble an image/container.
- Using **docker build command** users can create an automated build that executes several command-line instructions on the container being built in succession.
- You can also create a Dockercompose file if you have a multi-container environment.

TRY THIS IN THE LABS

Creating a Docker Container for a Java application with a Docker file (Tomcat Web App)

- In you project create a new file called “Dockerfile” (no extension).
- **FROM**: the image you are creating the container from (tomcat:9)
- **EXPOSE**: open ports on the container so the container can communicate over the network.
- **ADD**: copy local files to the containers file system.
- Once the Dockerfile is created, run the following commands, make sure you run the commands from the folder with the docker file:

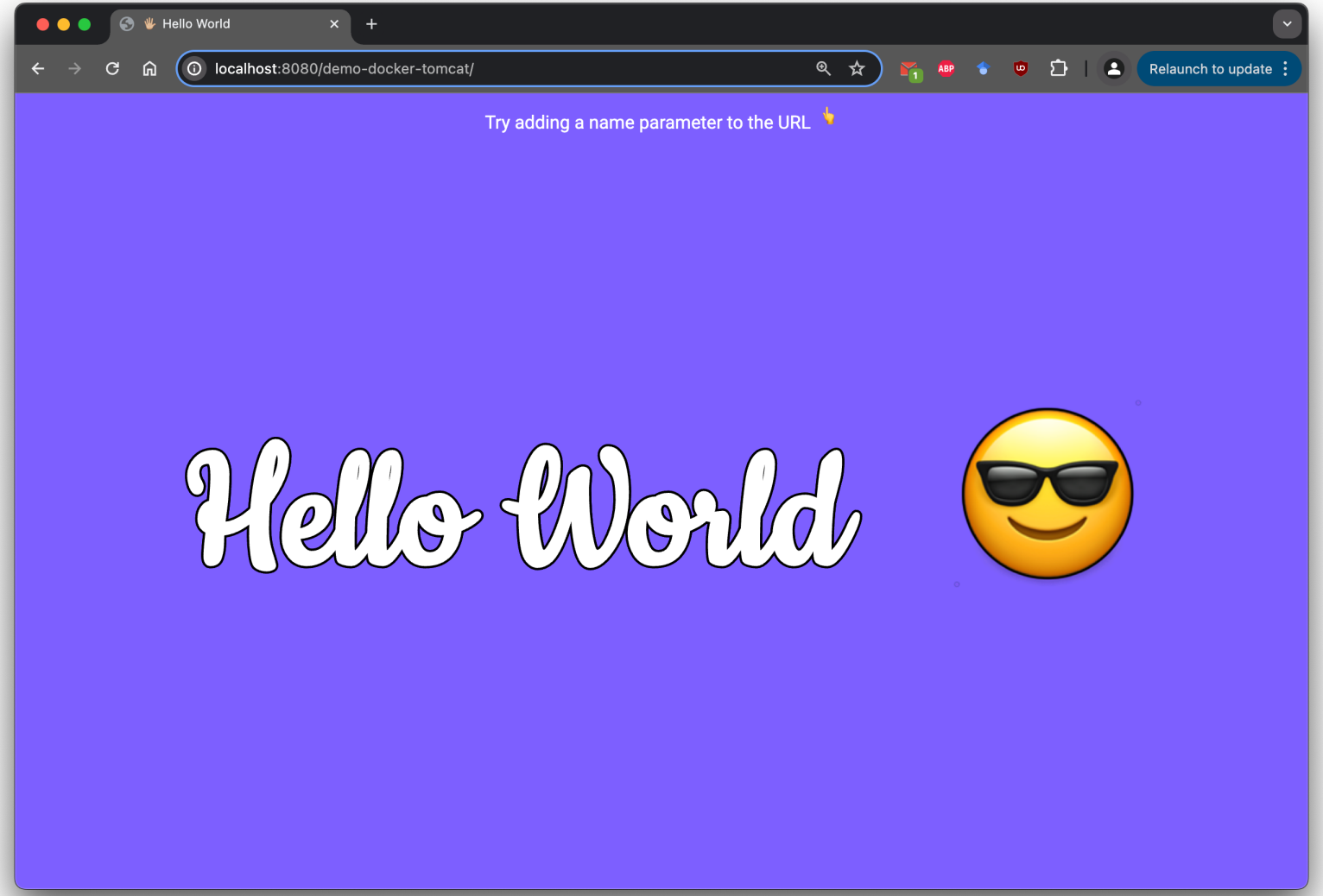


👉 `docker image build -t my-tomcat-image .`

👉 `docker container run --name –my-tomcat-server -it --publish 8080:8080 my-tomcat-image`

Test the Running Container

Served from docker container 👉



Docker Commands Cheat Sheet

List all docker containers/images

- docker container ls -a
- docker image ls -a

Delete docker container/image

- docker container rm [container]
- docker container rmi [image id]

Stop docker container

- docker stop [container_id]

Restart docker container

- docker restart [container_id]

Copy local file to docker tomcat web apps folder

- docker cp "local_file_path" [container_id]:"/usr/local/tomcat/webapps"

Start bash/ssh session with container

- docker container exec -it [container_name] bash

create docker image from docker file

- docker image build -t [name_the_image] .

create container from image name it and run it

- docker container run --name [container_name] -it --publish [LOCAL_PORT]:[CONTAINER_PORT] [name_of_image]

spin up a mysql server

- docker run --name mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=password mysql

See docker container configuration

- docker inspect [container_id]

Demo time!