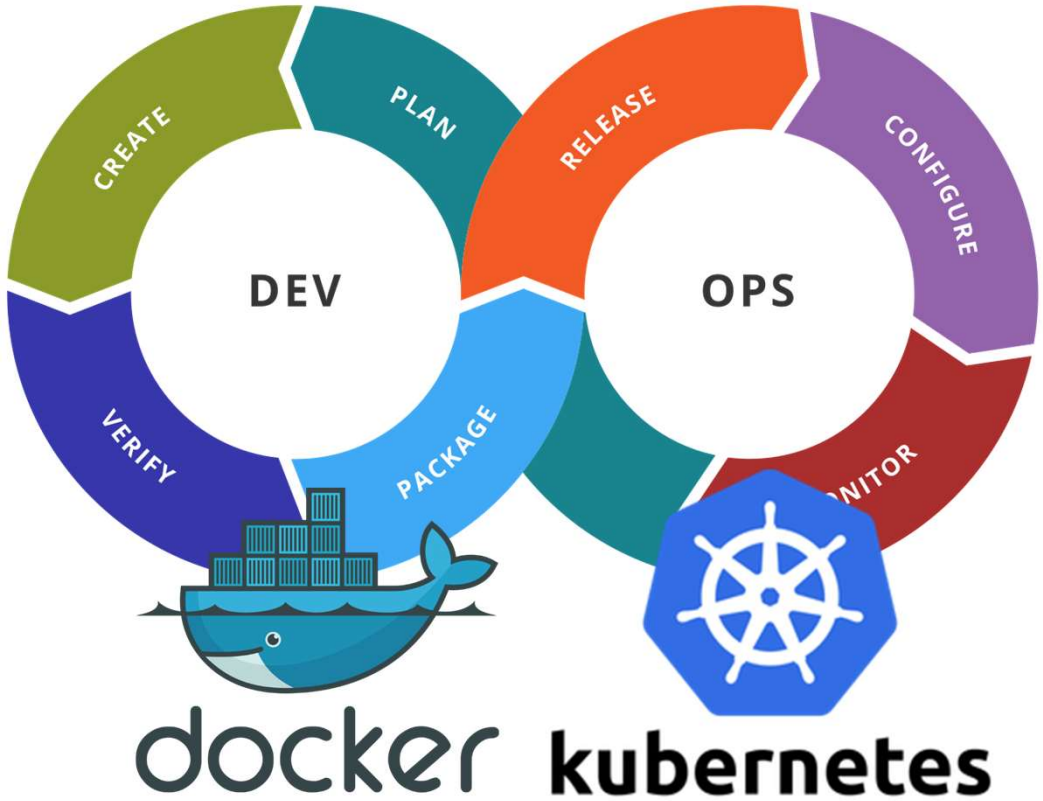


DevOp with Docker, and Kubernetes

CSS 223 Operating Systems

Chukiat Worasucheep

Learning objectives

1. Realize the meaning, benefits, and process of DevOp technology.
2. Describe benefits of Docker and its role in DevOp.
3. Describe the architecture of Docker and how to manage Docker image file.
4. Build docker image by writing a docker file.
5. Describe basic concept of and using Docker compose.
6. Describe basic concept of Kubernetes.
7. Practice basic Kubernetes usage.
8. Extend learning path for Docker and Kubernetes

Important Notice

การเรียนการสอนหัวข้อนี้ ผ่านทางสื่อออนไลน์ (Online meeting)

และมีการบันทึกภาพและเสียงเพื่อประโยชน์ทางการศึกษาต่อไปในอนาคต.

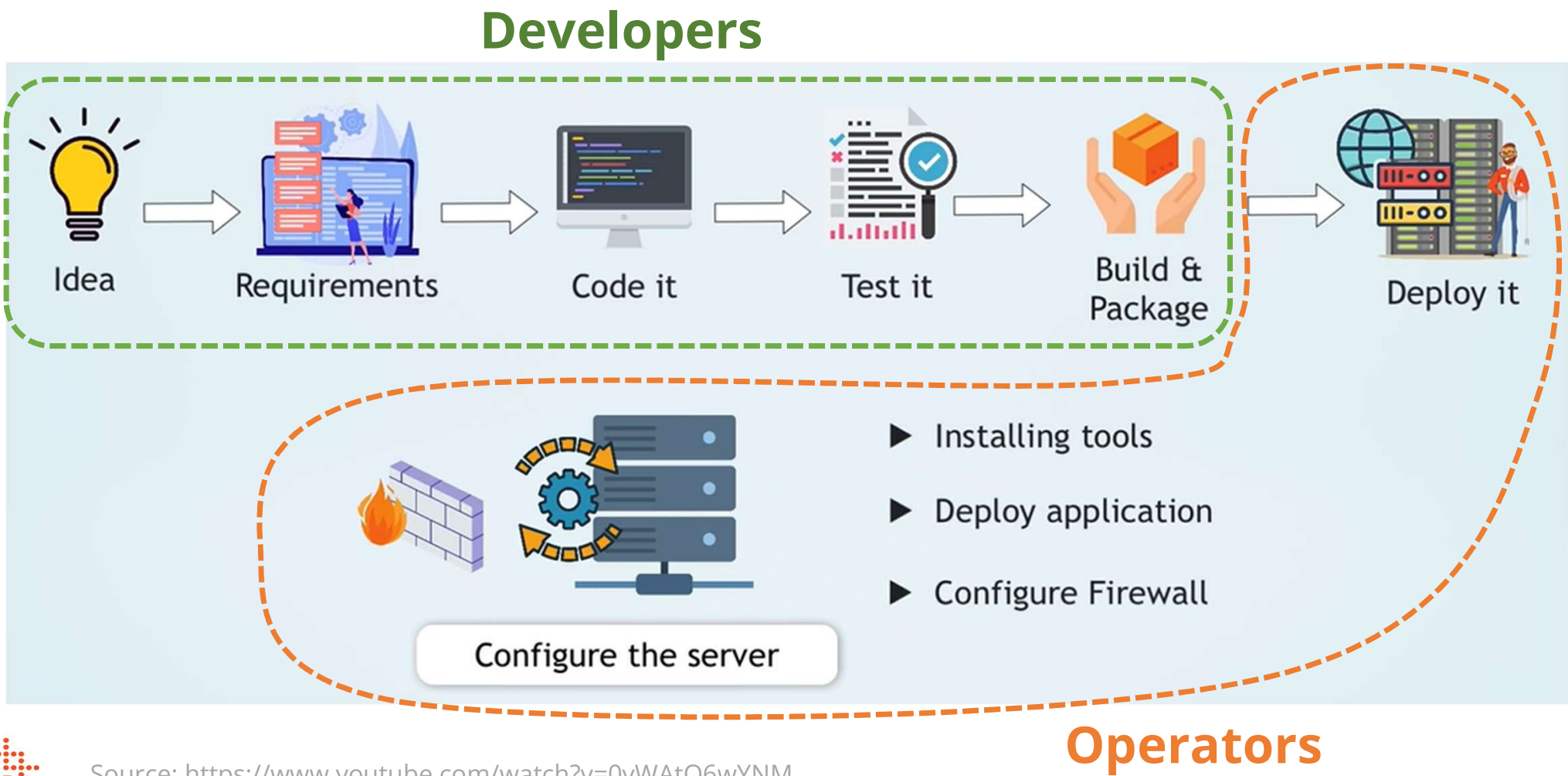
หากท่านไม่ยินยอมให้มีการเผยแพร่การบันทึกดังกล่าว ขอให้แจ้งให้ผู้สอนทราบภายใน 36 ชั่วโมง.

(พรบ. พระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล (PDPA) พ.ศ.2562)

Outline

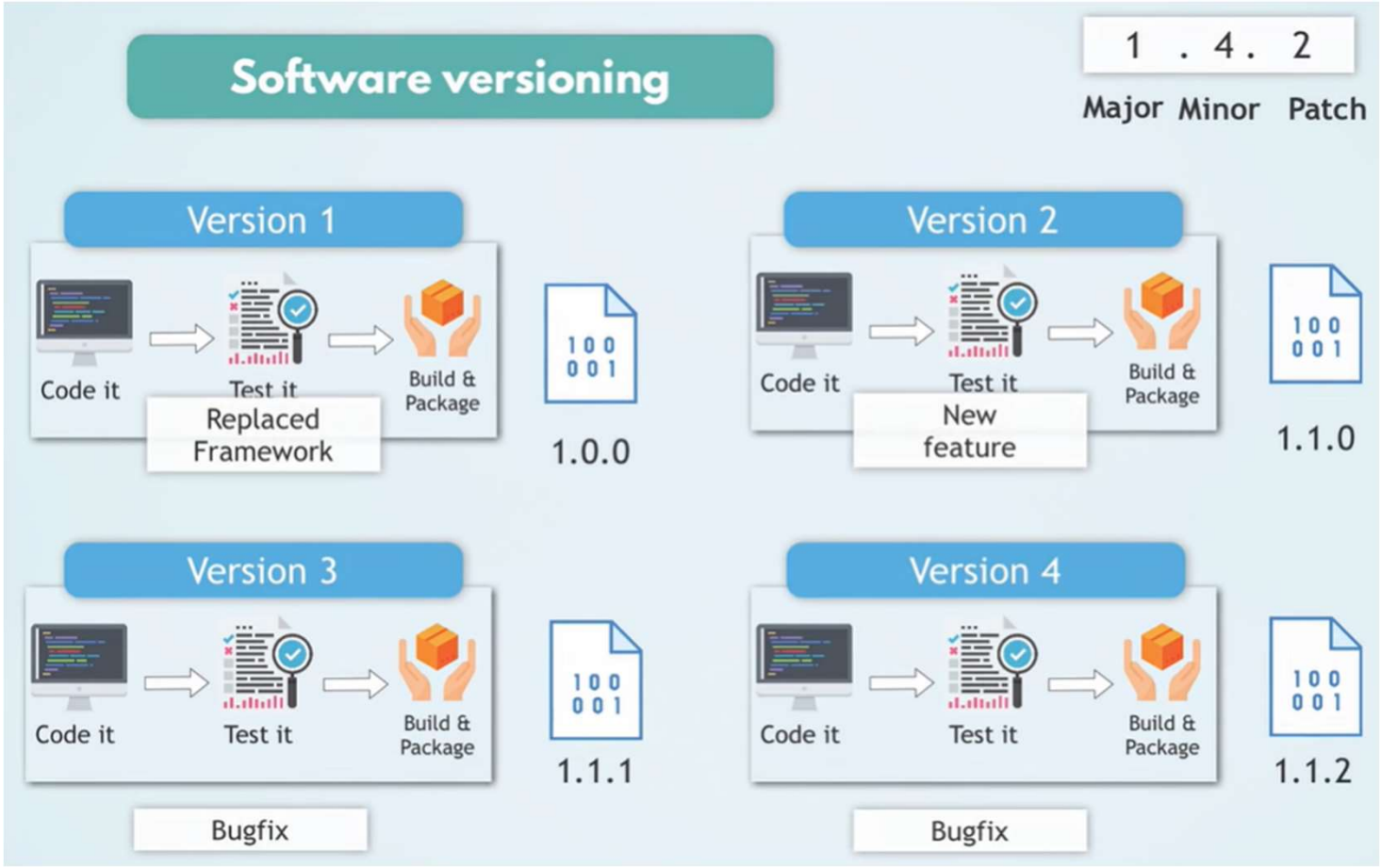
- DevOp and Software development process
- Docker
- Docker Hands-on Practice
- Build and test a dockerfile for https
- Docker compose
- Kubernetes – concepts
- K8s hands-on practice
- K8s YAML file for nginx

Generic software development and operation process



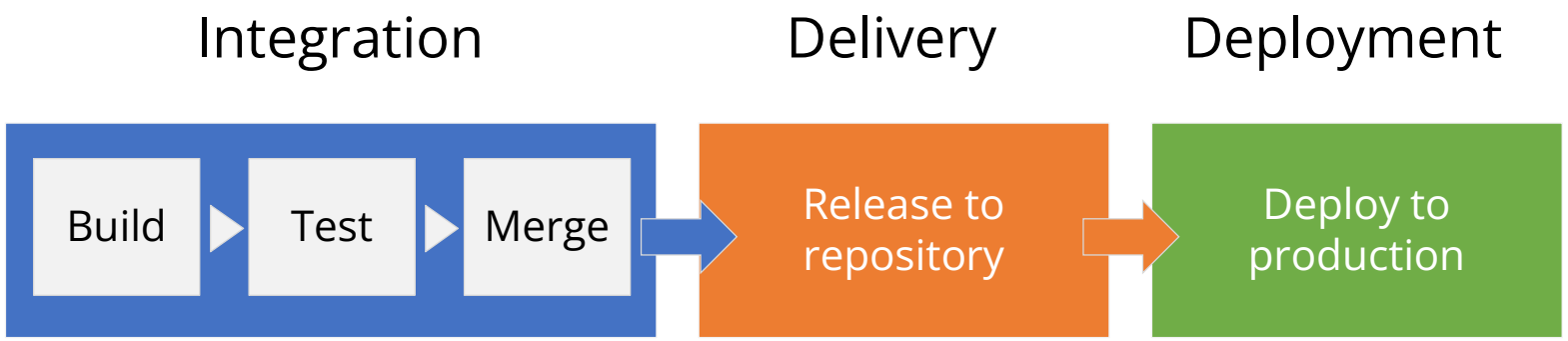
Source: https://www.youtube.com/watch?v=0yWAtQ6wYNM

Software version



Source: https://www.youtube.com/watch?v=0yWAtQ6wYNM

Integration and delivery – the Goal!



QUICKLY delivering

HIGH QUALITY code

Challenges

Developers

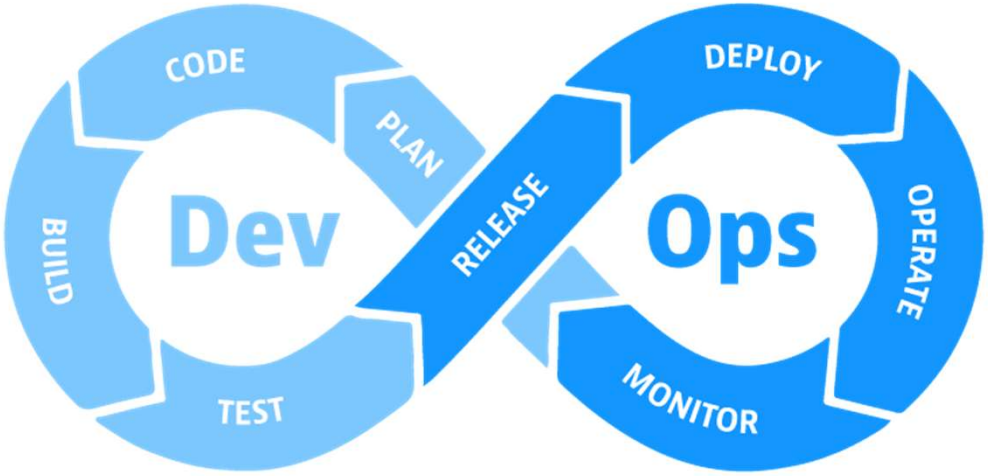
- Can’t deploy and run on the prod.
- Don’t consider where app gets deployed.
- New features fast!
- Don’t much care if app overloads the servers or has security vulnerabilities.

Operators

- Don’t know exactly how the app works.
- Maintain stability.
- Check if it’s 100% safe.
- Apps don’t burden or harm the servers.
- Apps don’t break any security measures.

Different goals → Conflicts → Delay or insecure deployment

DevOp



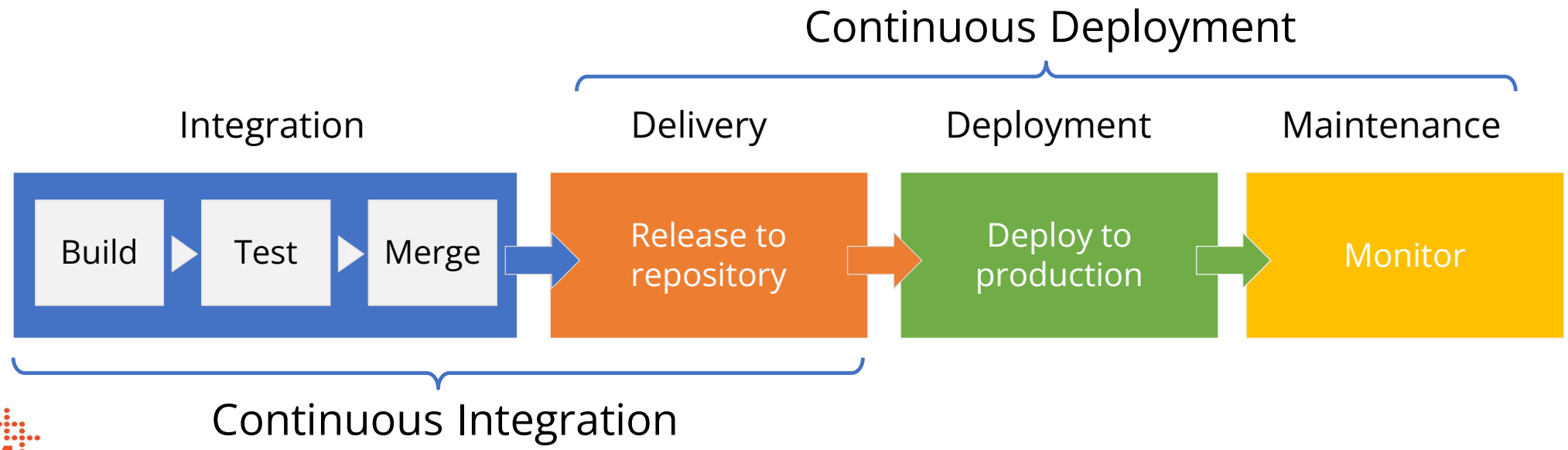
DevOps is a set of practices that combines software development (Dev) and IT operations (Ops).

The *Goal* of DevOps is to increase the speed, quality, and stability of software releases, while reducing the time it takes to diagnose and fix issues that arise.

DevOp tries to streamline and glue both development and operation together in an automate manner.

CI/CD Pipeline

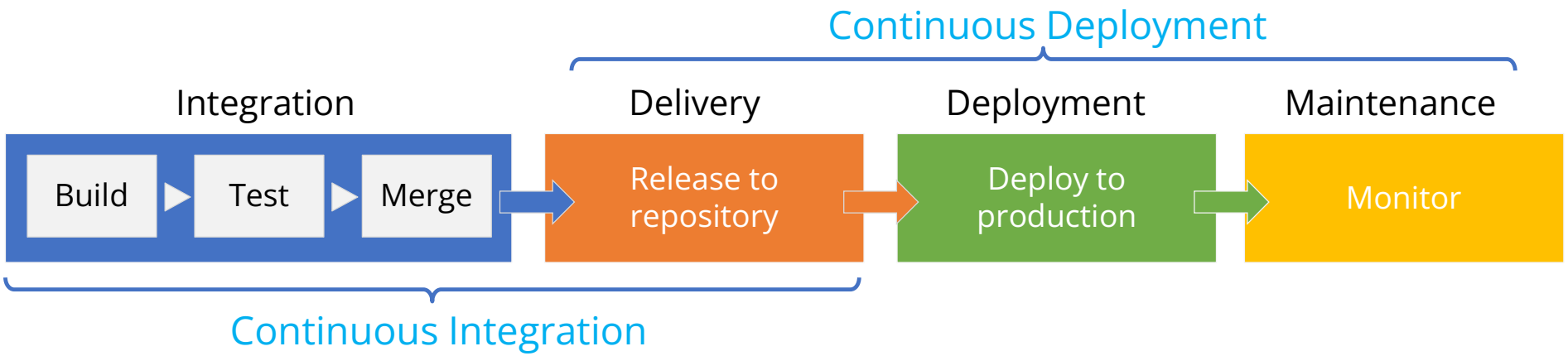
- *A continuous integration and continuous deployment (CI/CD)*
- *CI/CD pipeline* refers to the process of automatically building, testing, and deploying code changes to a production environment through the use of an *automated* pipeline.
- CD/CI is a part of DevOps.



CI/CD Pipeline

Continuous Integration (CI) is a software development practice in which all developers merge code changes in a central repository multiple times a day.

Continuous deployment (CD) means that every change that passes all stages of your production pipeline is released to your customers, with minimal human intervention, and only a failed test will prevent a new change to be deployed to production.



Main responsibilities and skill requirements of DevOp engineers

- Maintain code repository

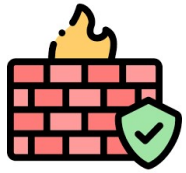
- Development process

- How to configure the applications

- Automated testing

- Linux
  - Shell commands
  - File system
  - Basic administration
  - Server management

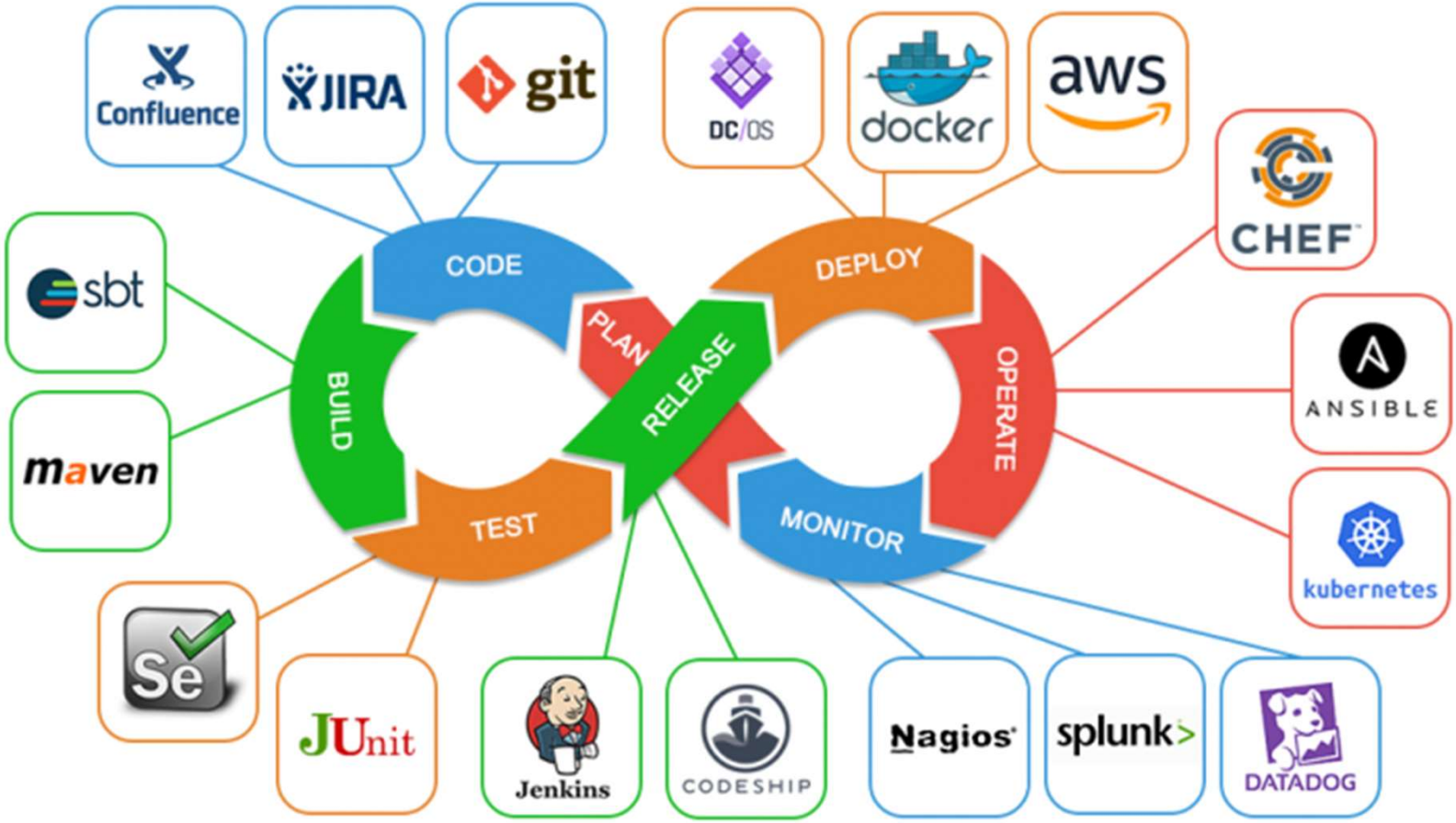
- Infrastructure
  - Network
    - HTTP/HTTPS
    - IP/DNS



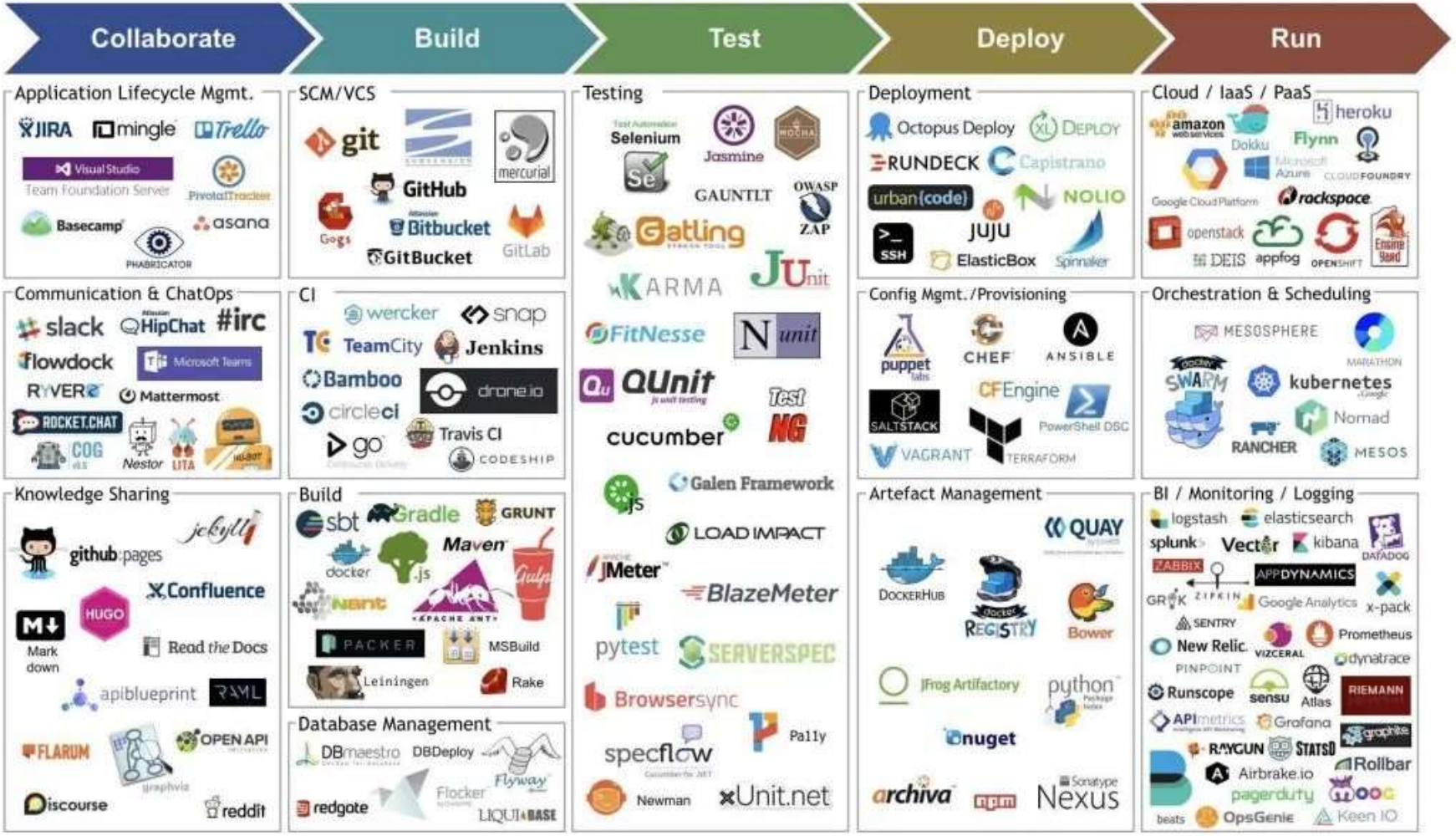
  - Firewalls/proxy servers
  - Load balancers

  - Security

Current Popular DevOp Tools



Want a more complete list?

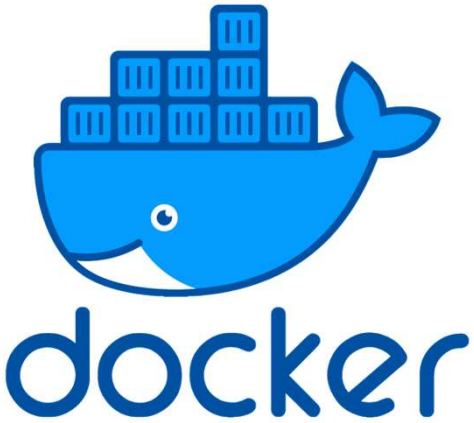


Outline

- DevOp and Software development process
- Docker
- Docker Hands-on Practice
- Build and test a dockerfile for https
- Docker compose
- Kubernetes – concepts
- K8s hands-on practice
- K8s YAML file for nginx

Introduction to Docker

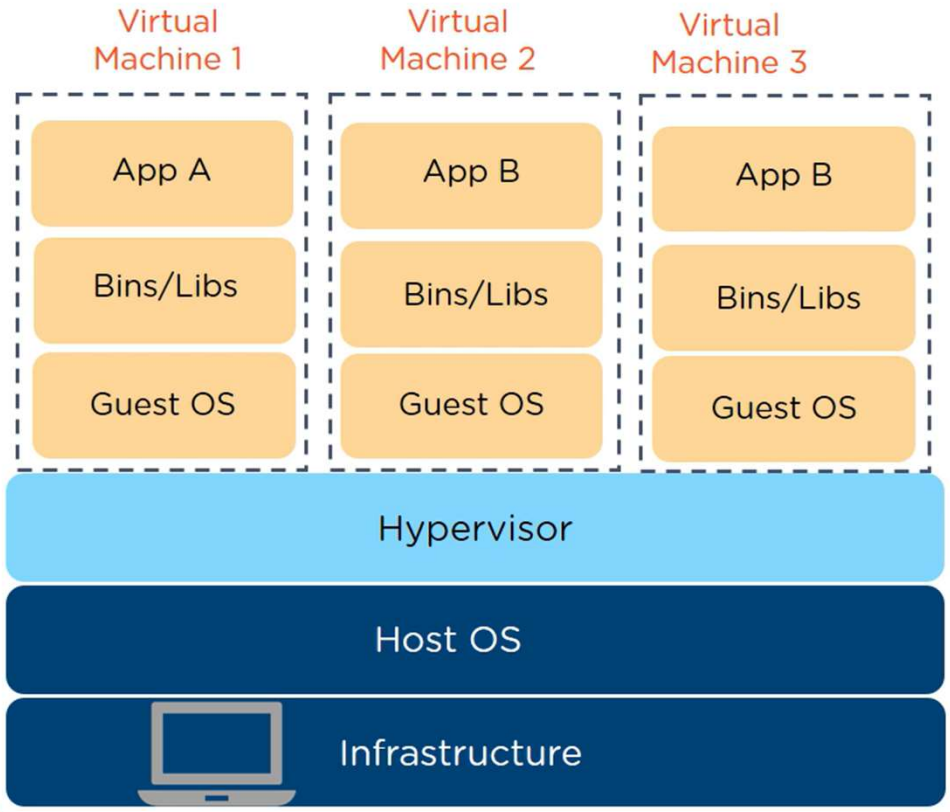
- *Docker* is a software platform that allows you to build, test, and *deploy* applications quickly.
- Docker packages software into standardized units called *containers* that have everything the software needs to run including runtime libraries, system tools, and frameworks.
- *Docker container* is a lightweight software package that includes all the dependencies (frameworks, libraries, etc.) required to execute an application.



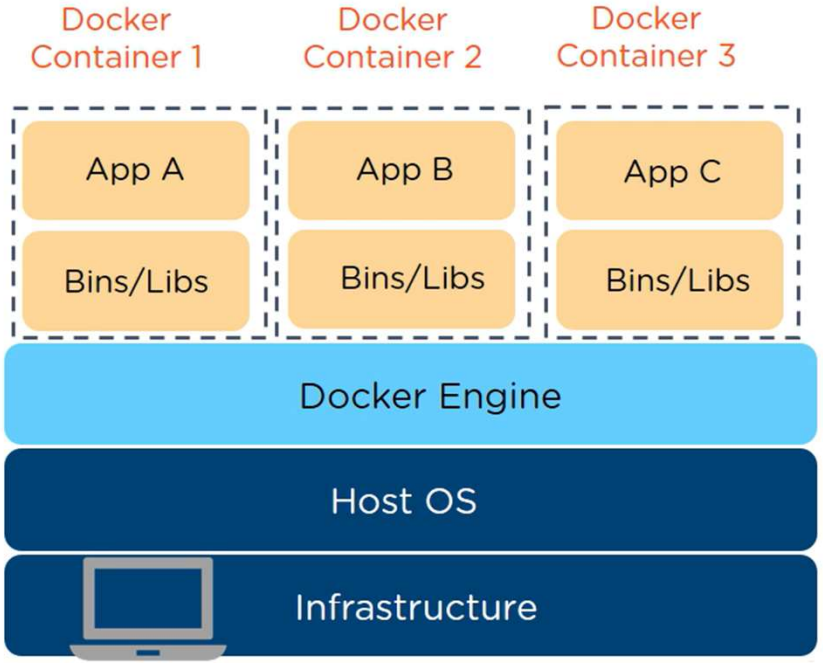
Docker containerization

Virtualization and Containerization

Virtualization e.g. VMware, VirtualBox, Proxmox, Hyper-V, Parallels, etc.



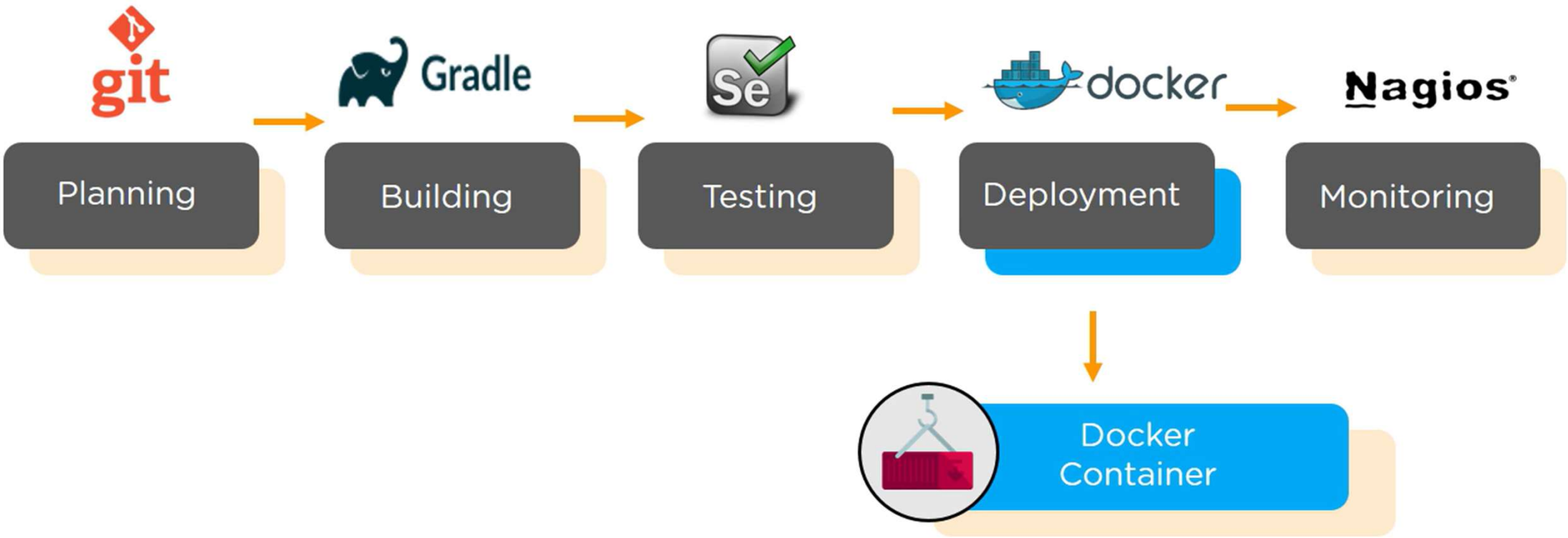
Containerization e.g. Docker



Virtual machine vs Docker

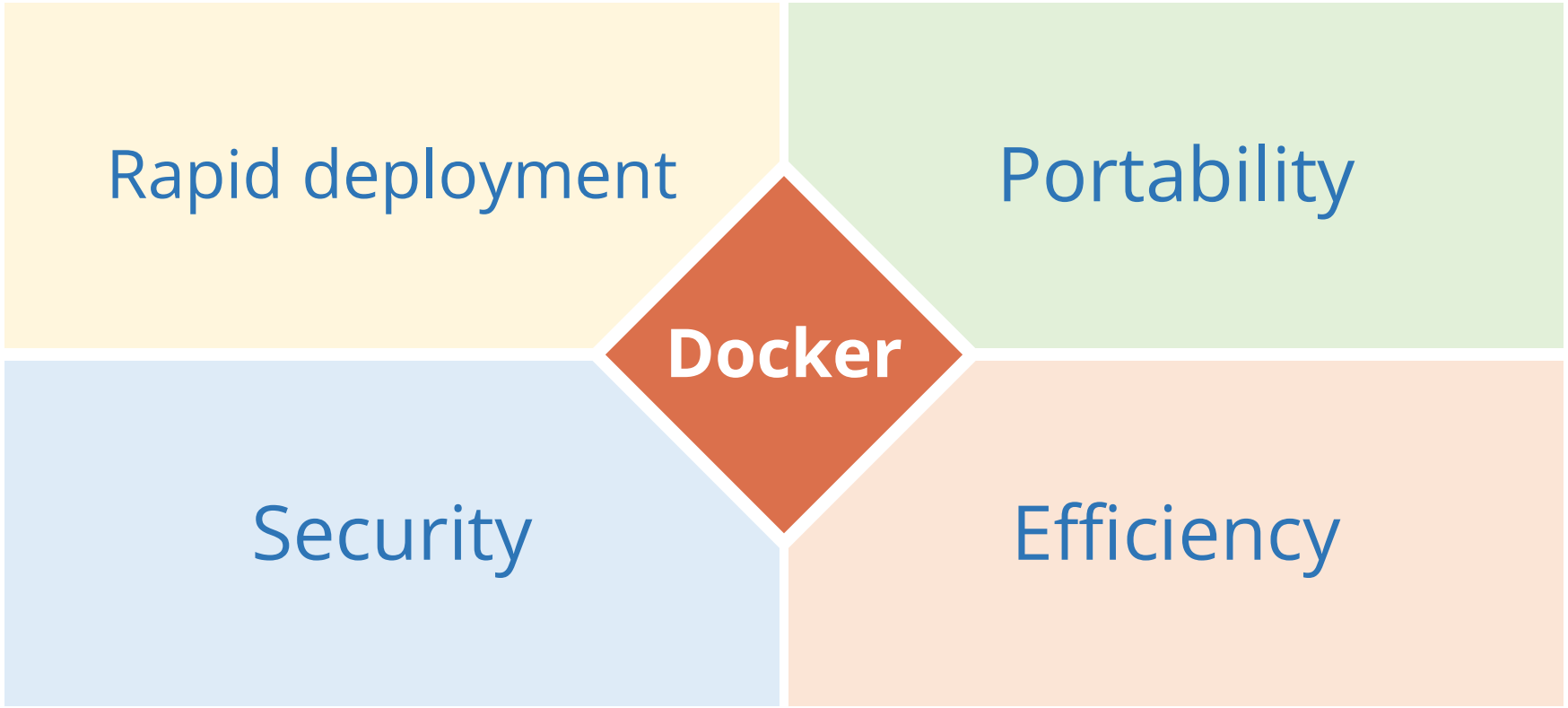
Virtual Machine		docker
	Memory usage	
	Performance	
	Portability	
	Boot-up time	

Example usage of Docker in DevOp process

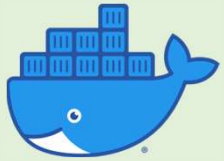


Source: https://www.simplilearn.com/tutorials/docker-tutorial/getting-started-with-docker

Benefits of using docker



Components of Docker



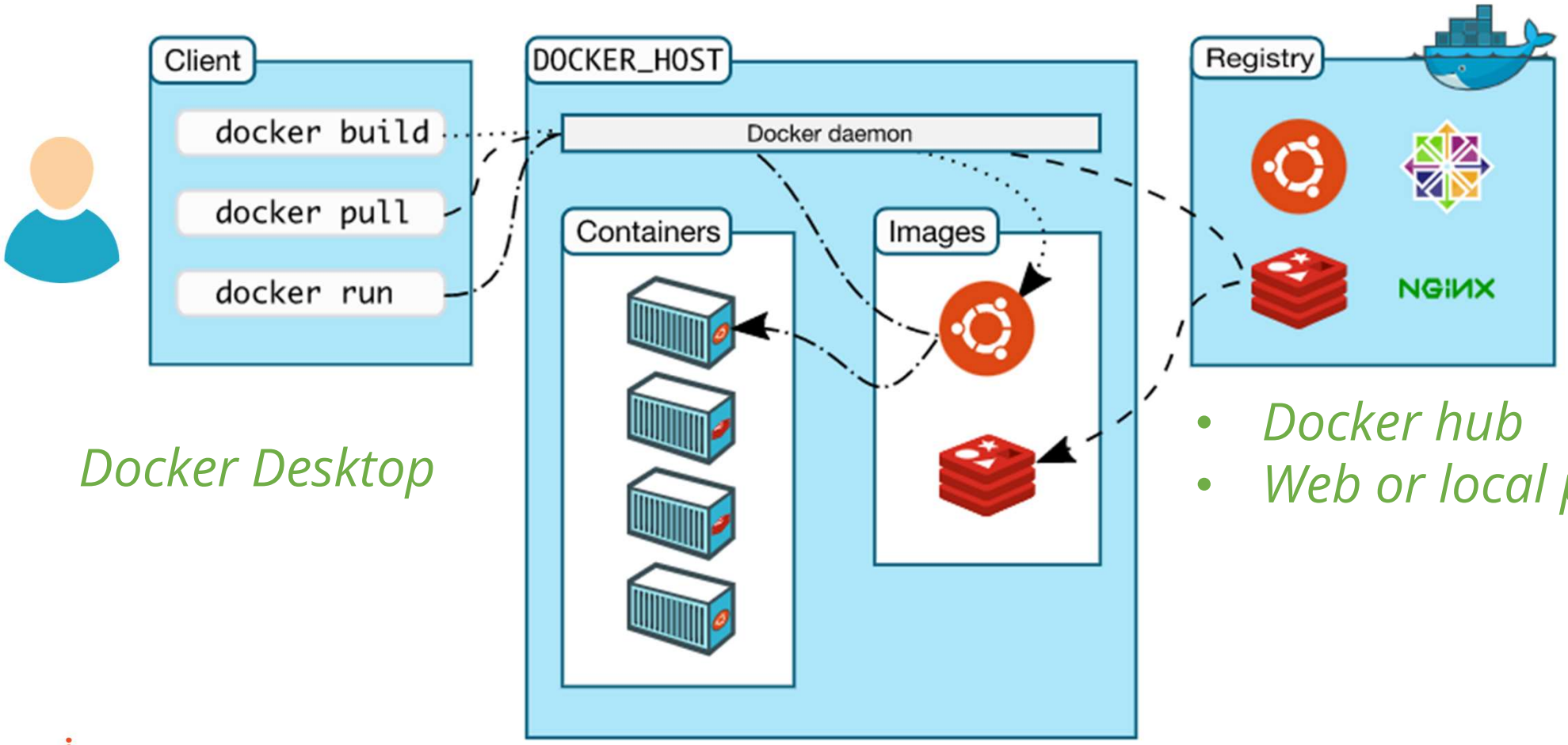
Docker client and server (daemon)

Docker images

Docker registry

Docker container

Architecture of Docker



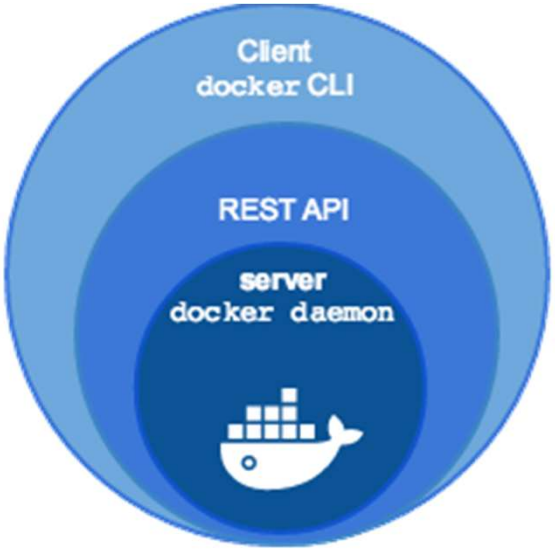
Docker Desktop

- *Docker hub*
- *Web or local premise*

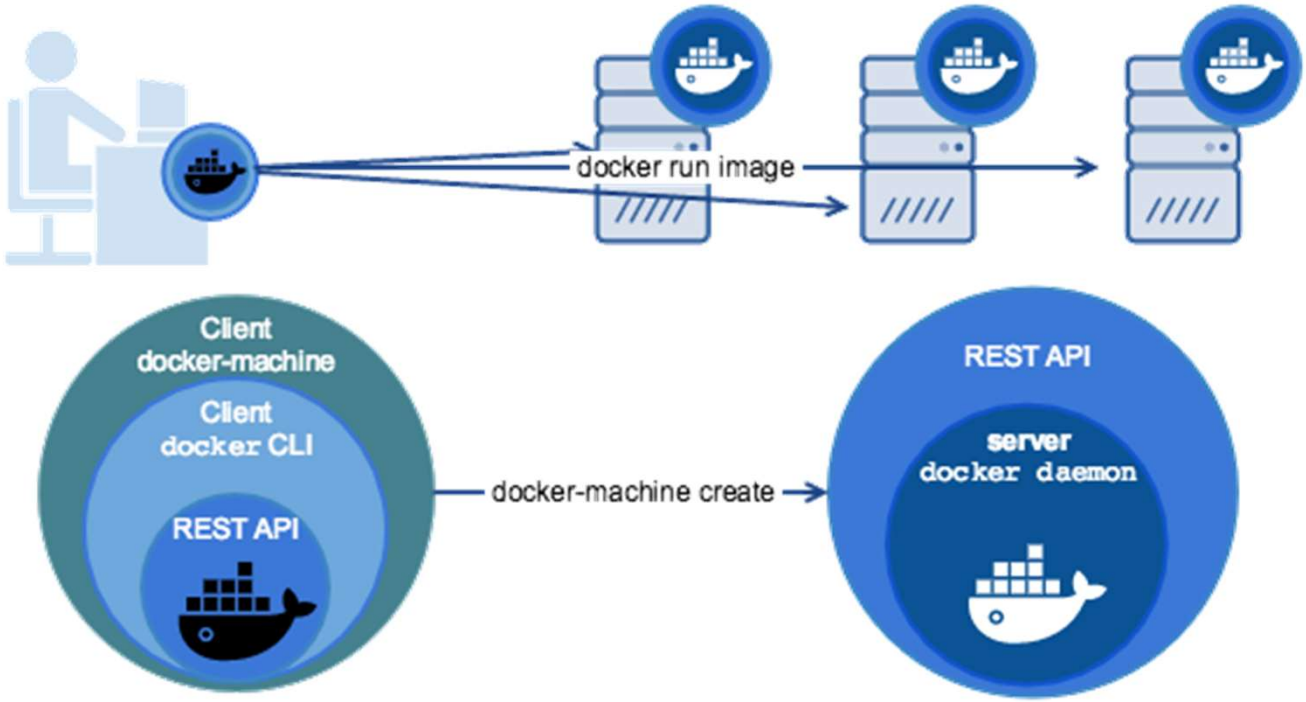
Source: https://docs.docker.com/get-started/overview/

Docker Engine vs Docker Machine

- Here, in this introduction to Docker, we'll be using the *same machine* for Docker client and server (host) with *Docker Engine*.



- Use *Docker Machine* for run remote machine or on cloud.
- More about Docker Machine: https://docs.docker.com.zh.xy2401.com/v17.09/machine/overview/

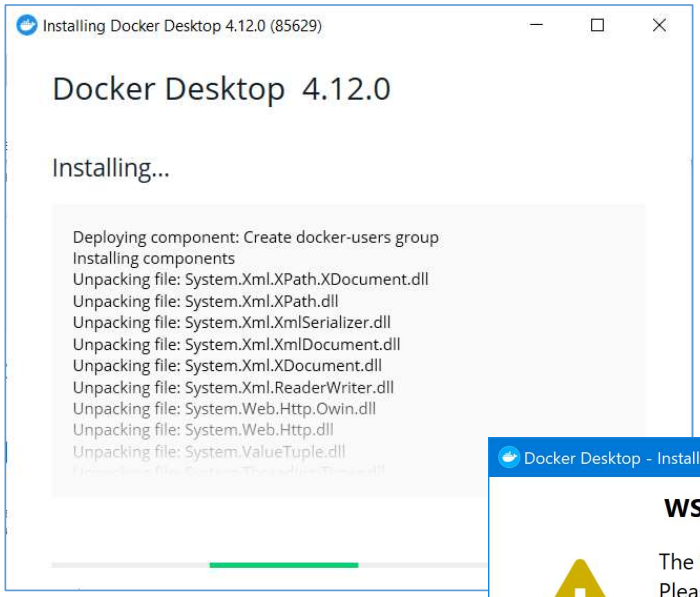


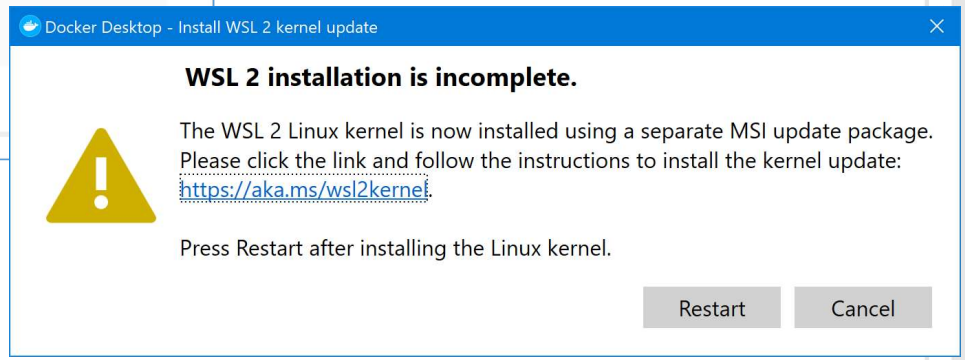
Outline

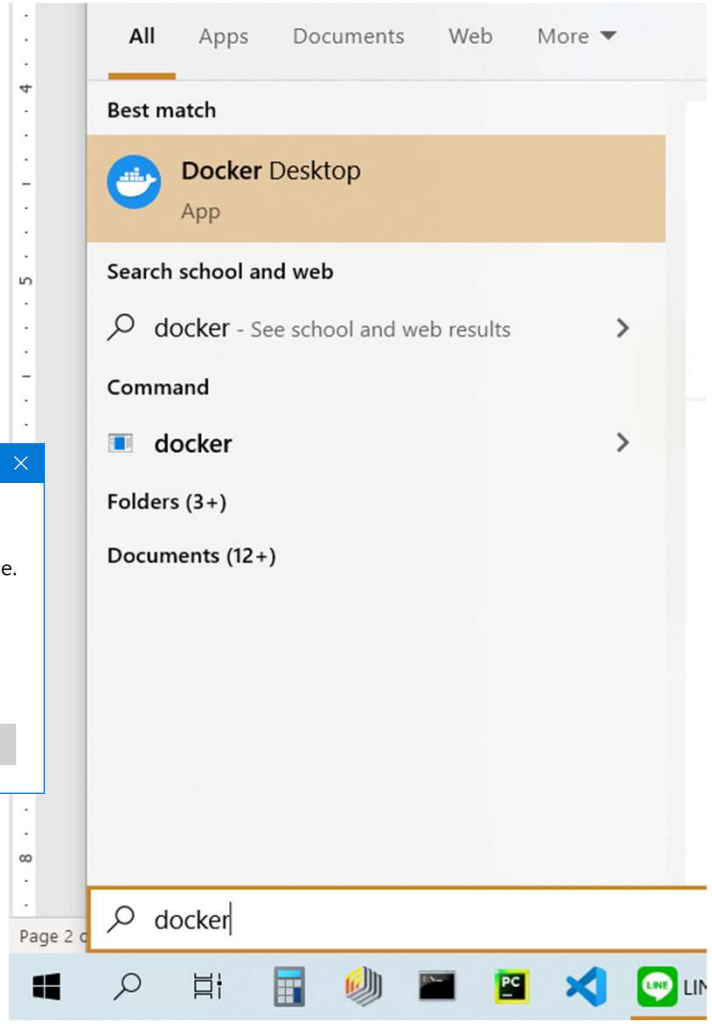
- DevOp and Software development process
- Docker
- Docker Hands-on Practice
- Build and test a dockerfile for https
- Docker compose
- Kubernetes – concepts
- K8s hands-on practice
- K8s YAML file for nginx

Docker Hands-on Practice

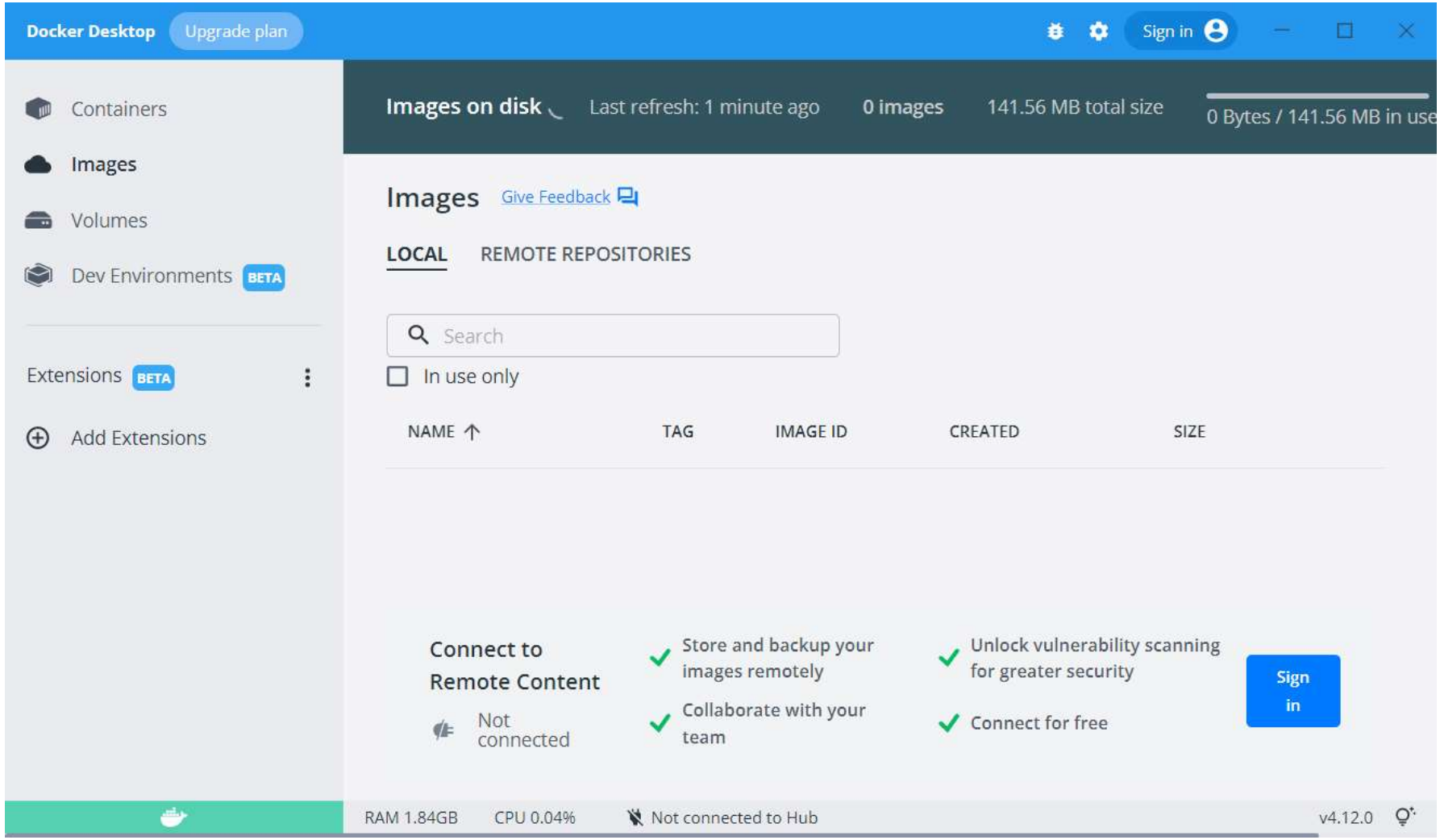
- Install Docker Desktop from web onto your machine and run it.







Docker Desktop console/dashboard



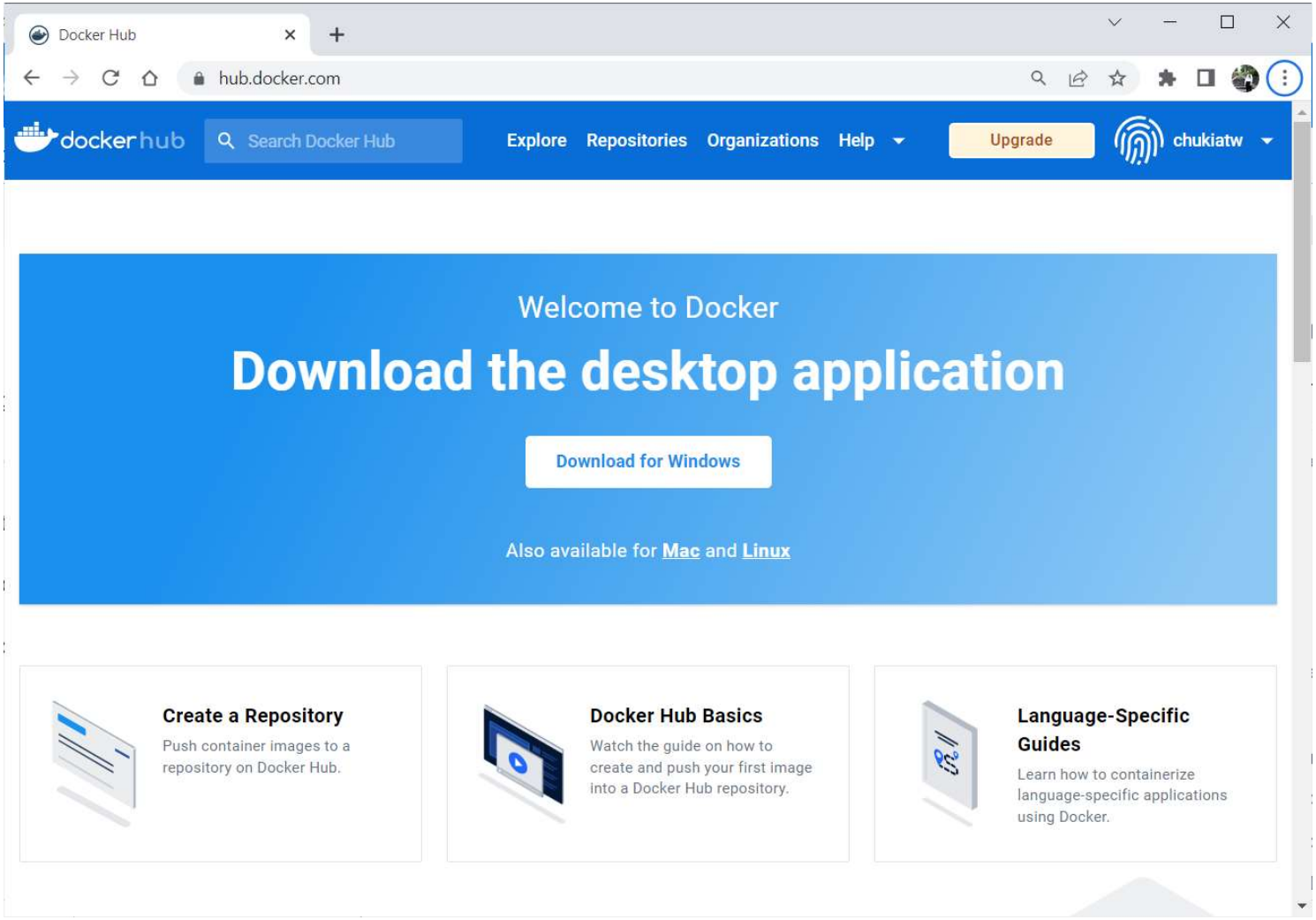
Basic Docker commands

```
$ docker run busybox

$ docker ps -a
CONTAINER ID      IMAGE     COMMAND   CREATED          STATUS                      PORTS      NAMES
305297d7a235      busybox   "uptime"  11 minutes ago   Exited (0) 11 minutes ago   distracted_gold
ff0a5c3750b9      busybox   "sh"      12 minutes ago   Exited (0) 12 minutes ago   elated_ramanujan


$ docker run -it busybox
/ # ls
bin   dev   etc   home  proc  root  sys   tmp   usr   var
/ # uptime
 05:45:21 up  5:58,  0 users,  load average: 0.00, 0.01, 0.04
/ # exit


$ docker rm 305297d7a235 ff0a5c3750b9  # with your own container id
305297d7a235
ff0a5c3750b9
```

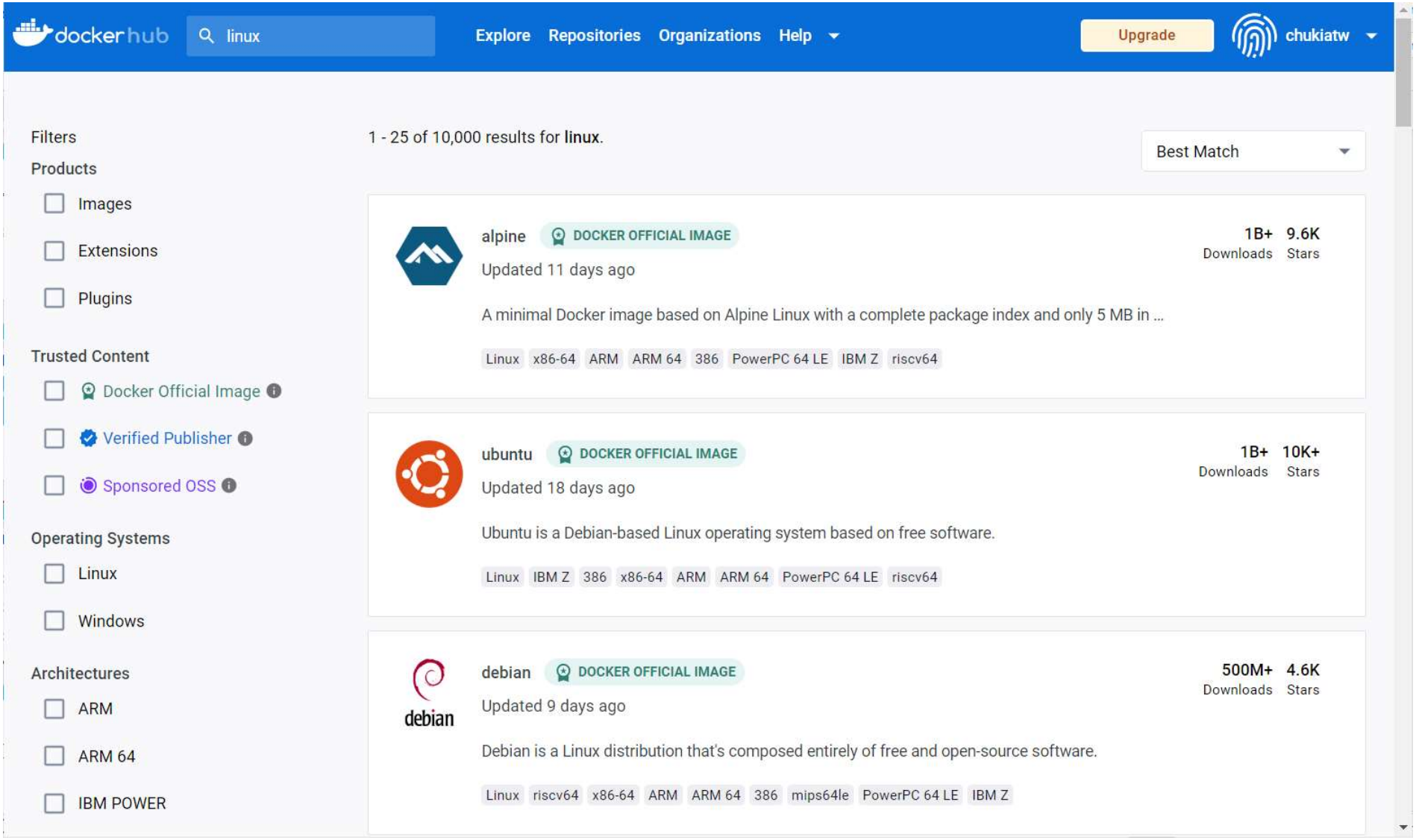
Basic Docker commands

```
$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
busybox      latest    c98db043bed9   11 days ago   1.24MB

$ docker image rm busybox
Error response from daemon: conflict: unable to remove repository reference "busybox"
(must force) - container b268c016e5bb is using its referenced image c98db043bed9

$ docker image rm -f busybox
Untagged: busybox:latest
Untagged:
busybox@sha256:20142e89dab967c01765b0aea3be4cec3a5957cc330f061e5503ef6168ae6613
Deleted: sha256:c98db043bed913c5a7b59534cbf8d976122f98b75cb00baabf8af888041e4f9d

$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
nginx        latest    2b7d6430f78d   3 weeks ago   142MB
```

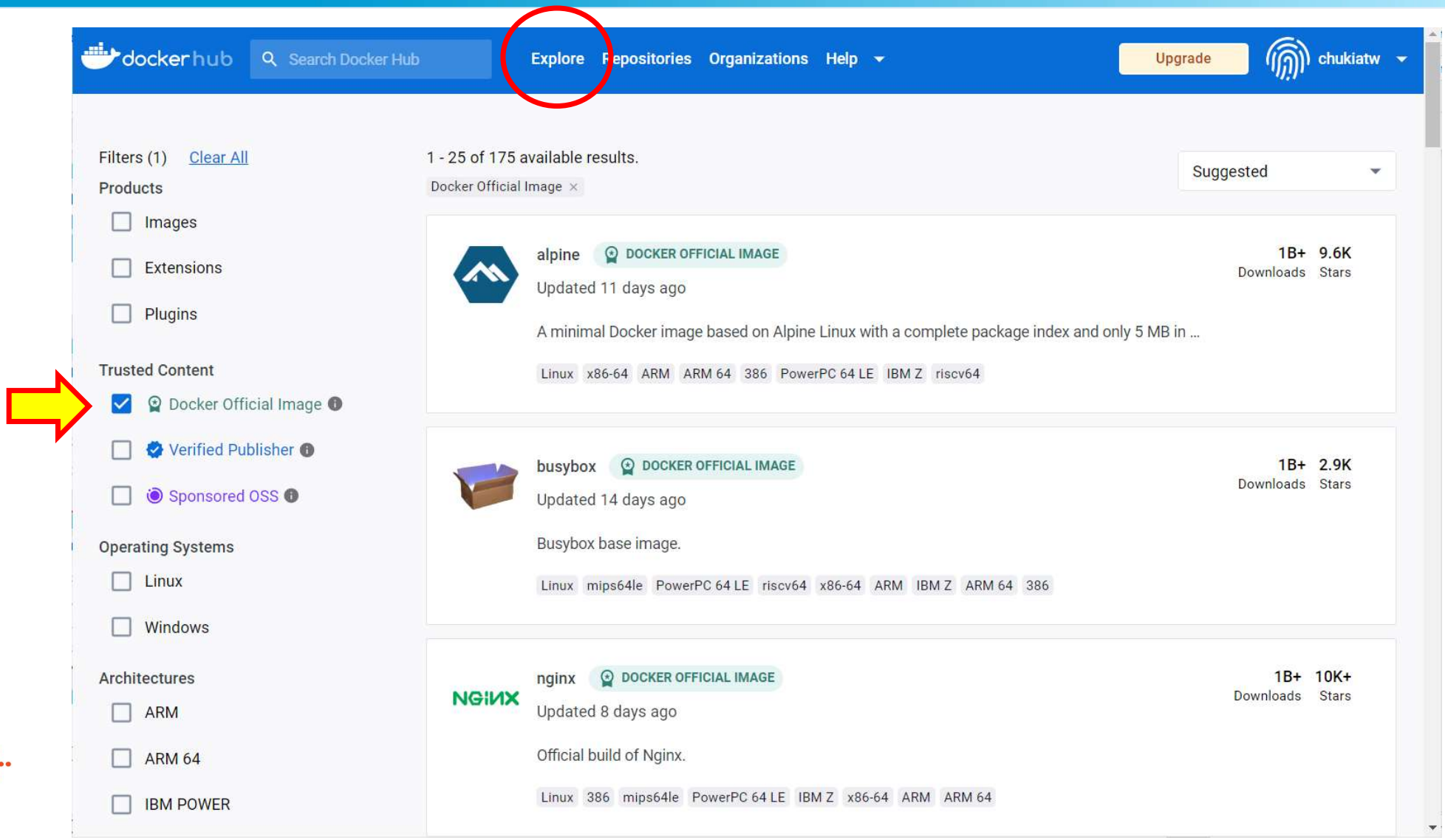
Docker Hub


Docker Hub
hub.docker.com
dockerhub
Search Docker Hub
Explore
Repositories
Organizations
Help
Upgrade
chukiatw
Welcome to Docker
Download the desktop application
Download for Windows
Also available for Mac and Linux
Create a Repository
Push container images to a repository on Docker Hub.
Docker Hub Basics
Watch the guide on how to create and push your first image into a Docker Hub repository.
Language-Specific Guides
Learn how to containerize language-specific applications using Docker.

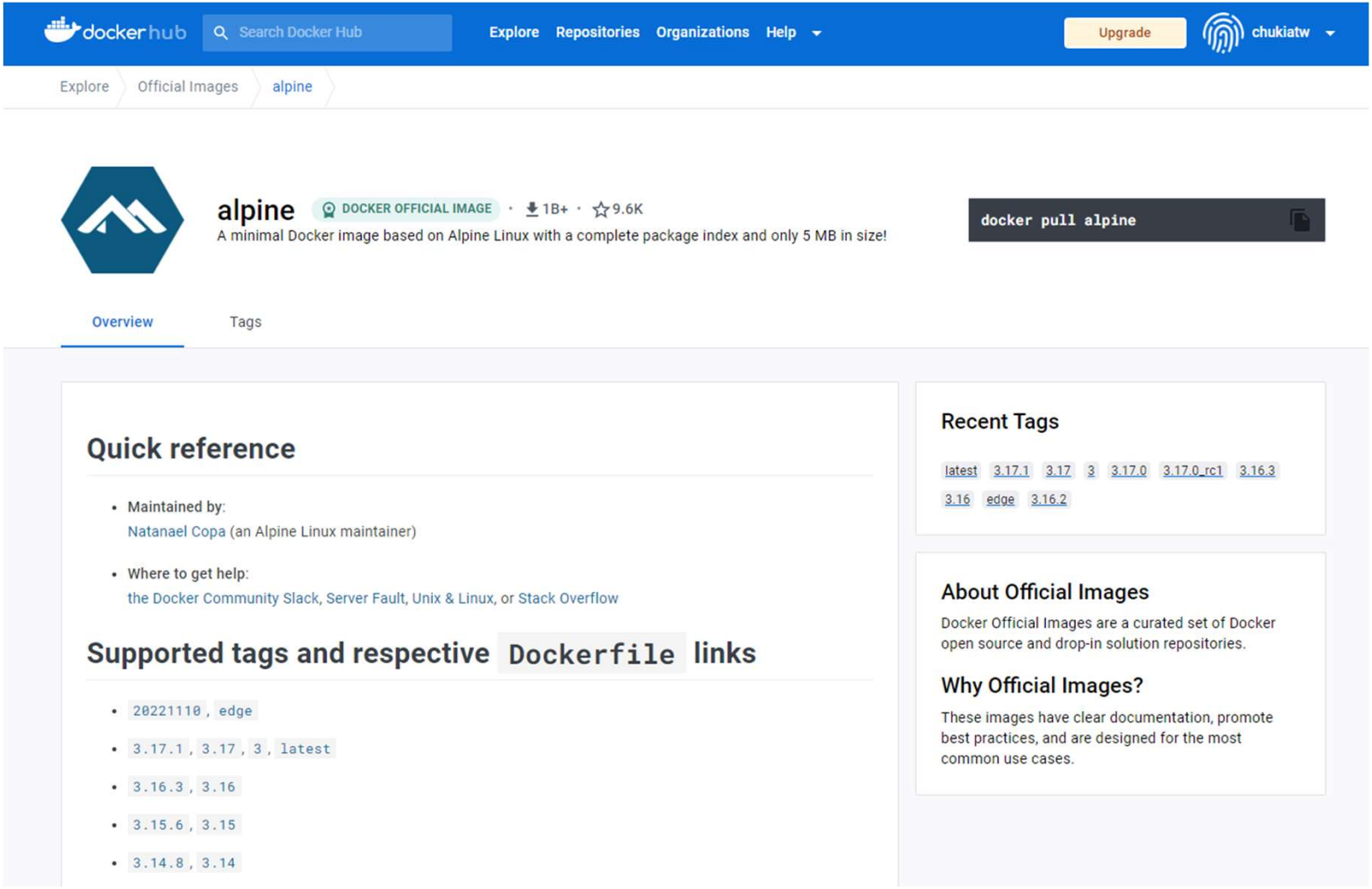
Docker Hub – search for Linux images



Explore Docker Hub



The alpine image



Ubuntu image

```
D:\>docker run -h chukiat-99 -it ubuntu bash
root@chukiat-99:/# date
Tue Sep 13 06:35:13 UTC 2022
root@chukiat-99:/# uname -a
Linux chukiat-99 5.10.16.3-microsoft-standard-WSL2 #1 SMP Fri Apr 2 22:23:49 UTC 2021 x86_64
x86_64 x86_64 GNU/Linux
root@chukiat-99:/# ls -a
.   .dockerenv  boot  etc   lib    lib64   media  opt   root  sbin  sys  usr
..  bin         dev   home  lib32  libx32  mnt    proc  run   srv   tmp  var
root@chukiat-99:/# ls -l
total 48
lrwxrwxrwx   1 root root    7 Aug 15 11:50 bin -> usr/bin
drwxr-xr-x   2 root root 4096 Apr 18 10:28 boot
drwxr-xr-x   5 root root  360 Sep 13 06:35 dev
drwxr-xr-x   1 root root 4096 Sep 13 06:35 etc
drwxr-xr-x   2 root root 4096 Apr 18 10:28 home
lrwxrwxrwx   1 root root    7 Aug 15 11:50 lib -> usr/lib
...
drwxr-xr-x  14 root root 4096 Aug 15 11:50 usr
drwxr-xr-x  11 root root 4096 Aug 15 11:53 var
```

Running Ubuntu image

```
root@chukiat-99:/# wget
bash: wget: command not found

root@chukiat-99:/# sudo
bash: sudo: command not found

root@chukiat-99:/# apt-get update
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [4732 B]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
...
Get:17 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [7291 B]
Get:18 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [3520 B]
Fetched 24.9 MB in 37s (668 kB/s)
Reading package lists... Done

root@chukiat-99:/# apt-get -y install wget sudo
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Running hooks in /etc/ca-certificates/update.d...
done.

root@chukiat-99:/# wget
wget: missing URL
Usage: wget [OPTION]... [URL]...

Try `wget --help' for more options.
```

Version of Docker

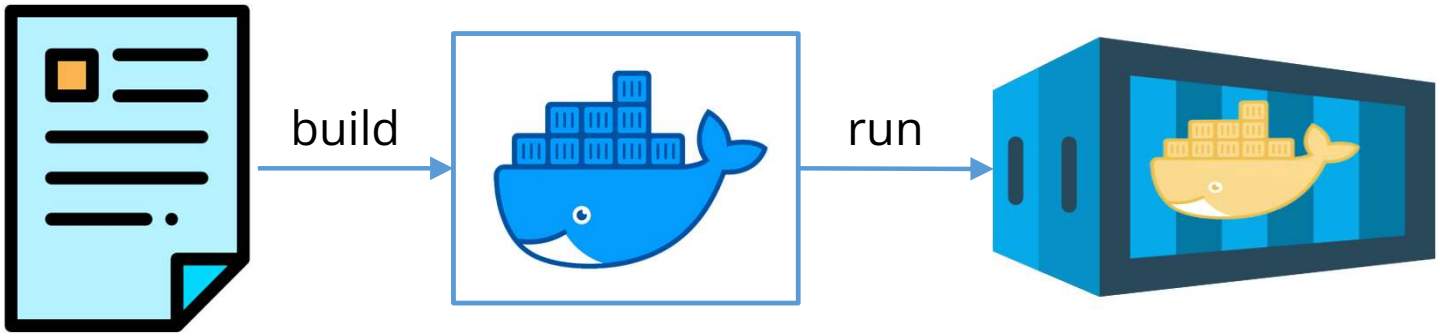
```
D:\docker>docker version
Client:
 Cloud integration: v1.0.29
 Version:           20.10.17
 API version:       1.41
 Go version:        go1.17.11
 Git commit:        100c701
 Built:             Mon Jun  6 23:09:02 2022
 OS/Arch:           windows/amd64
 Context:           default
 Experimental:      true

Server: Docker Desktop 4.12.0 (85629)
 Engine:
  Version:          20.10.17
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.17.11
  Git commit:       a89b842
  Built:            Mon Jun  6 23:01:23 2022
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.8
  GitCommit:        9cd3357b7fd7218e4aec3eae239db1f68a5a6ec6
 runc:
  Version:          1.1.4
  GitCommit:        v1.1.4-0-g5fd4c4d
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

Outline

- DevOp and Software development process
- Docker
- Docker Hands-on Practice
- Build and test a dockerfile for https
- Docker compose
- Kubernetes – concepts
- K8s hands-on practice
- K8s YAML file for nginx

Docker file

- *Dockerfile* is a simple text file what consists of sequence of instructions to *build docker images* with '*docker build*' command.

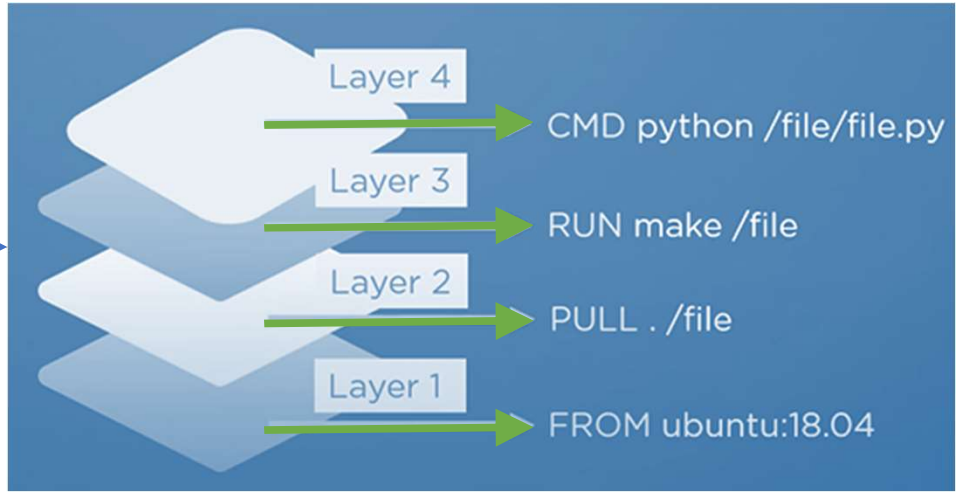



```
1  FROM python:3.5
2  RUN pip install Flask==0.11.1 redis==2.10.5
3  RUN useradd -ms /bin/bash admin
4  USER admin
5  COPY app /app
6  WORKDIR /app
7  CMD ["python", "app.py"]
```

Image layers

- Each layer is an image itself, just one without a human-assigned tag. They have auto-generated IDs though.
- Each layer stores the changes compared to the image it's based on.
- Each instruction in a Dockerfile results in a layer.

```
FROM ubuntu:18.04
PULL ./file
RUN make /file
CMD python /file/file.py
```



How to build an image from dockerfile

```
FROM ubuntu:18.04
PULL ./file
RUN make /file
CMD python /file/file.py
```

- FROM - Creates a layer from the ubuntu:18.04
- PULL - Adds files from your Docker repository
- RUN - Builds your container
- CMD - Specifies what command to run within the container

Some Dockerfile commands

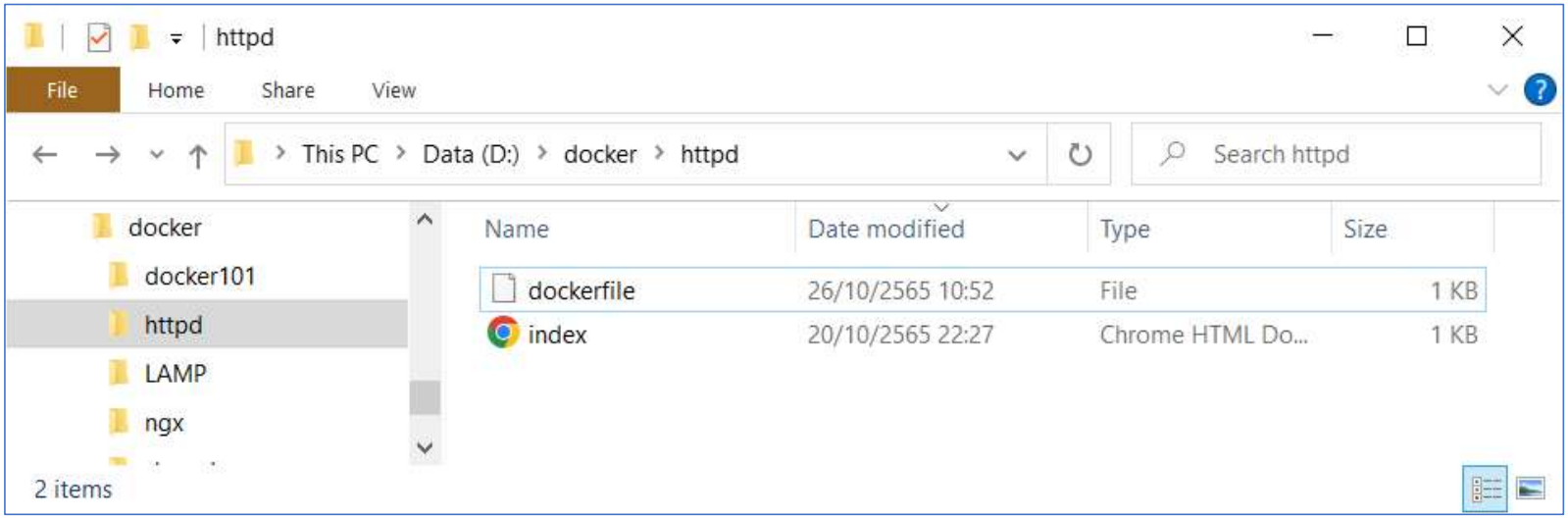
```
FROM ubuntu:18.04
PULL ./file
RUN make /file
CMD python /file/file.py
```

- FROM - Creates a layer from the ubuntu:18.04
- PULL - Adds files from your Docker repository
- RUN - Builds your container
- CMD - Specifies what command to run within the container

- COPY – Copy of files / directories from host machine to the image.
- WORKDIR – Change current working directory.
- LABEL – Add metadata to an image in a form of key-value pair.
- VOLUME – Create a mount point of a shared file folder.
- ENV – Add environment variables.
- EXPOSE – Tell the container to listen on the specified network ports at runtime.

Further study: https://docs.docker.com/engine/reference/builder

Demo building a docker image from Apache httpd



dockerfile

```
# A simple web page with Apache httpd

FROM httpd:2.4

LABEL AUTHOR=chukiatwr@gmail.com

LABEL VERSION=0.1

WORKDIR /usr/local/apache2

COPY index.html htdocs/index.html
```

Index.html

```
<!DOCTYPE html>

<html lang="en">

<head> <title>My web page in docker</title> </head>

<h1>Hello docker and my web site #1.</h1>

<p>This is a simple web page.</p>

<p>A link to wikiHow: <a
href="http://www.wikihow.com">wikiHow</a></p>

</html>
```

Build image *myhttpd*

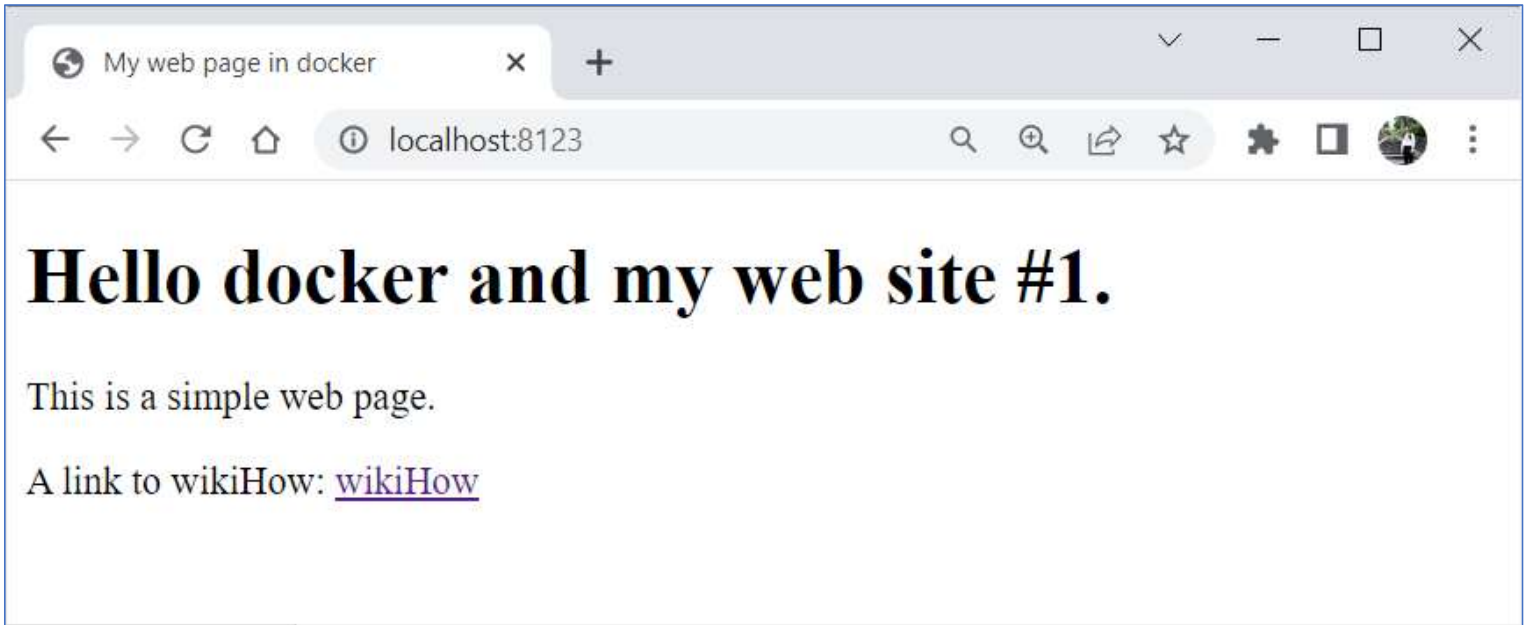
```
D:\docker\httpd>docker build -t myhttpd .
[+] Building 3.6s (2/3)
 => [internal] load build definition from Dockerfile      0.0s
 => => transferring dockerfile: 214B                      0.0s
[+] Building 3.7s (2/3)
 => [internal] load build definition from Dockerfile      0.0s
[+] Building 3.9s (2/3)
 => [internal] load build definition from Dockerfile      0.0s
 => => transferring dockerfile: 214B                      0.0s
[+] Building 4.1s (4/7)
...
[+] Building 7.3s (8/8) FINISHED
 => [internal] load build definition from Dockerfile      0.0s
 => => transferring dockerfile: 214B                      0.0s
 => [internal] load metadata for docker.io/library/httpd:2.4  3.9s
 => [1/3] FROM docker.io/library/httpd:2.4@sha256:5fa965      3.1s>
 => [2/3] WORKDIR /usr/local/apache2                      0.2s
 => [3/3] COPY index.html htdocs/index.html               0.0s
 => exporting to image                                    0.0s
 => => exporting layers                                   0.0s
 => => writing image sha256:39c0bb0d64f904654ac5119d0d10  0.0s
 => => naming to docker.io/library/myhttpd                0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

View history (layers) of the image *myhttpd*

```
D:\docker\httpd>docker history myhttpd
IMAGE          CREATED         CREATED BY                                      SIZE      COMMENT
39c0bb0d64f9   11 seconds ago  COPY index.html htdocs/index.html # buildkit    254B      buildkit.dockerfile.v0
<missing>      11 seconds ago  WORKDIR /usr/local/apache2                      0B        buildkit.dockerfile.v0
<missing>      11 seconds ago  LABEL VERSION=0.1                               0B        buildkit.dockerfile.v0
<missing>      11 seconds ago  LABEL AUTHOR=chukiatwr@gmail.com                0B        buildkit.dockerfile.v0
<missing>      24 hours ago    /bin/sh -c #(nop)  CMD ["httpd-foreground"]     0B
<missing>      24 hours ago    /bin/sh -c #(nop)  EXPOSE 80                    0B
<missing>      24 hours ago    /bin/sh -c #(nop) COPY file:c432ff61c4993ecd…   138B
<missing>      24 hours ago    /bin/sh -c #(nop)  STOPSIGNAL SIGWINCH          0B
<missing>      24 hours ago    /bin/sh -c set -eux;   savedAptMark="$(apt-m…   59.9MB
<missing>      24 hours ago    /bin/sh -c #(nop)  ENV HTTPD_PATCHES=           0B
<missing>      24 hours ago    /bin/sh -c #(nop)  ENV HTTPD_SHA256=eb397fee…   0B
<missing>      24 hours ago    /bin/sh -c #(nop)  ENV HTTPD_VERSION=2.4.54     0B
<missing>      24 hours ago    /bin/sh -c set -eux;  apt-get update;  apt-g…   4.76MB
<missing>      24 hours ago    /bin/sh -c #(nop) WORKDIR /usr/local/apache2    0B
<missing>      24 hours ago    /bin/sh -c mkdir -p "$HTTPD_PREFIX"  && chow…   0B
<missing>      24 hours ago    /bin/sh -c #(nop)  ENV PATH=/usr/local/apach…   0B
<missing>      24 hours ago    /bin/sh -c #(nop)  ENV HTTPD_PREFIX=/usr/loc…   0B
<missing>      26 hours ago    /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>      26 hours ago    /bin/sh -c #(nop) ADD file:8644a8156a07a656a…   80.5MB
```

View and run the image

```
D:\docker\httpd>docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
myhttpd       latest    39c0bb0d64f9   21 seconds ago   145MB

D:\docker\httpd>docker run -d -p8123:80 myhttpd
6de5413e33a22c0419ec1dbb55105e09fda2c96fb3d19a3cd803e0028570ebc6

D:\docker\httpd>docker container ls
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS         PORTS                    NAMES
82271a926415   myhttpd   "httpd-foreground"   3 minutes ago   Up 3 minutes   0.0.0.0:8123->80/tcp     brave_edison
```



Some notes about writing a Dockerfile

1. Use images from trusted source, e.g. Docker hub.
2. Use a minimal base image, such as Alpine Linux, instead of larger Linux distributions, and then add necessary packages as required to run your application.
3. Try to reduce the number of layers in your image by minimizing the number of separate RUN commands in your Dockerfile.
   - Use

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

   - Instead of

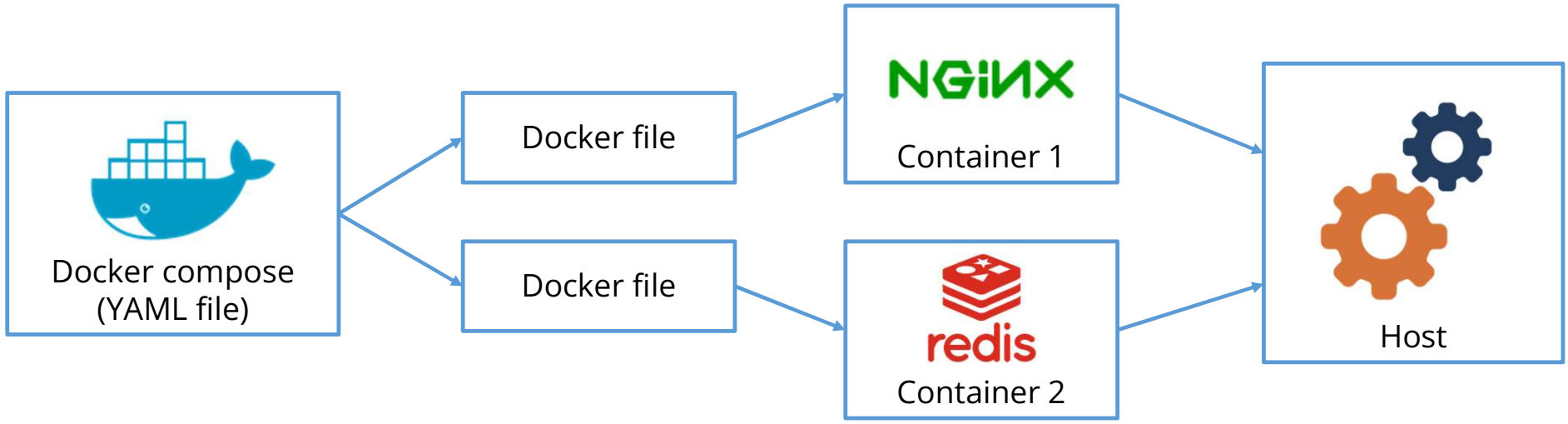
     ```
     RUN apt-get -y update
     RUN apt-get install -y python
     ```

Outline

- DevOp and Software development process
- Docker
- Docker Hands-on Practice
- Build and test a dockerfile for https
- Docker compose
- Kubernetes – concepts
- K8s hands-on practice
- K8s YAML file for nginx

Docker compose

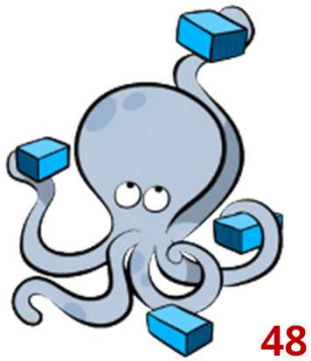
- Docker compose is used for building and running multi-container docker application on *one host*.
- With *Compose*, you use a YAML file to configure your application's services.
- Then, with a single command, you create and start all the services from your configuration (YAML file).
- *Here, each container runs in isolation, but can interact with each other on the same host.*



Installation of docker compose

- Linux
  - download the Docker Compose binary using Github's Compose repository release page.
- MS Windows
  - just have Docker Desktop for Windows installed and don't need to install Docker Compose separately.
- MacOS X
  - just have Docker Desktop for Mac installed and don't need to install Docker Compose separately.

Source: https://www.simplilearn.com/tutorials/docker-tutorial/docker-compose



Try Docker Compose

- Goal: *Run a Python code for counting web page access with Redis, in-memory data structure.*
- Install Docker Desktop which includes both Docker Engine and Docker Compose.
- Create a file called *app.py*:

```python
import time

import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times.\n'.format(count)
```

A micro web framework written in Python

- Create a *dockerfile*

```
# syntax=docker/dockerfile:1
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
CMD ["flask", "run"]
```

- Create a file called *requirements.txt*

```
flask
redis
```

Try Docker Compose

- Create a file called *docker-compose.yml*

```yaml
version: "3.9"
services:
  web:
    build: .
    ports:
      - "8000:5000"
    volumes:
      - .:/code
    environment:
      FLASK_DEBUG: True
  redis:
    image: "redis:alpine"
```

The *web* service uses an image that's built from the dockerfile in the current directory.

mounts the current project directory on the host to /code inside the container.

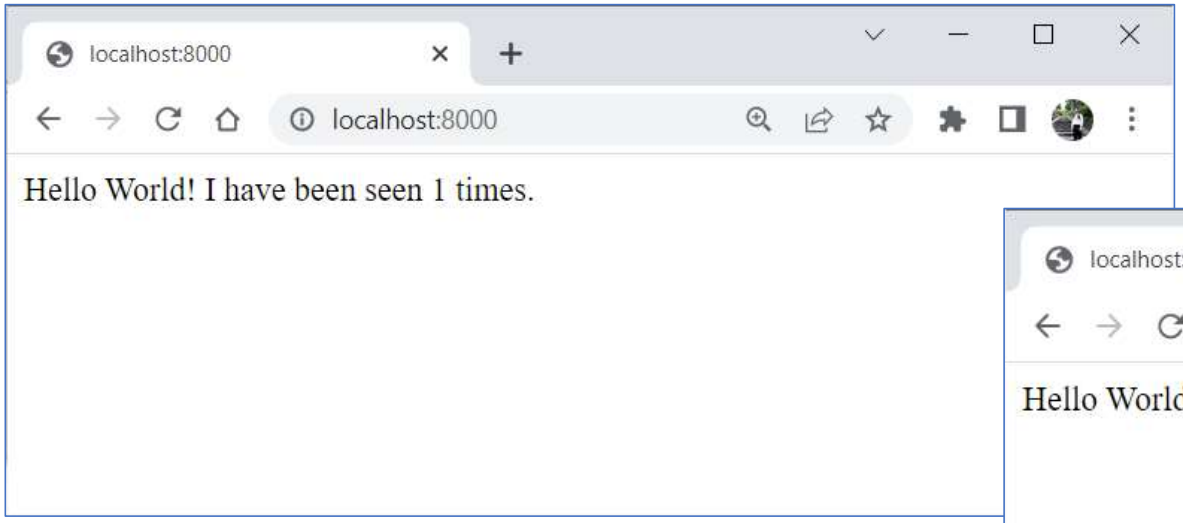
Set to tell flask to run in development mode and reload the code on change.

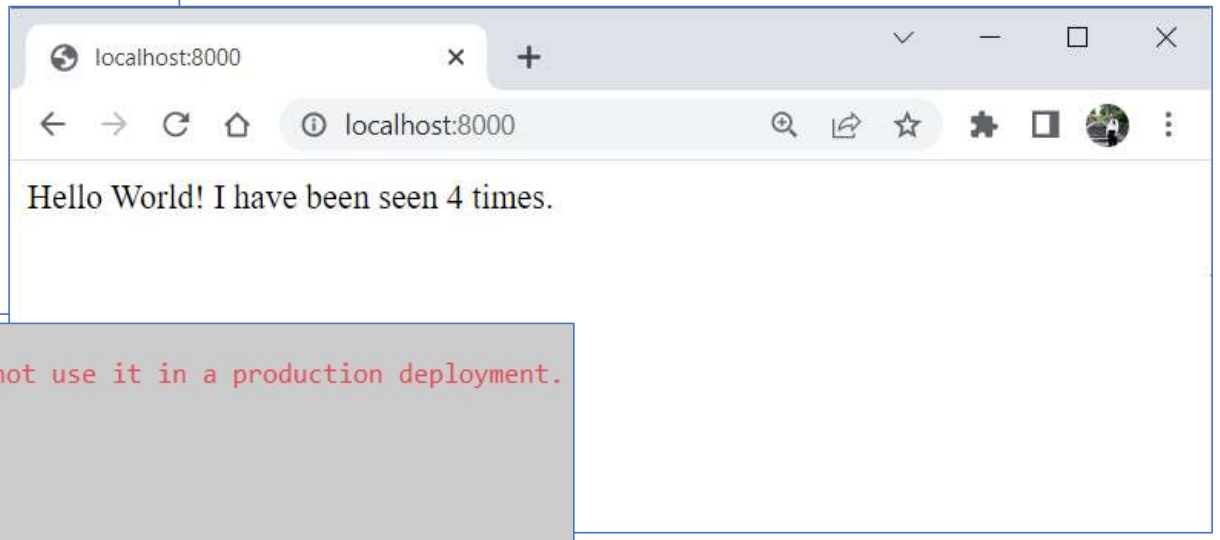
docker compose up

Command Prompt for Notebooks - docker compose up

```
D:\docker\compose>docker compose up
[+] Running 3/1
 - Network compose_default    Created                                                    0.7s
 - Container compose-web-1    Created                                                    0.0s
 - Container compose-redis-1  Created                                                    0.0s
Attaching to compose-redis-1, compose-web-1
compose-redis-1  | 1:C 30 Nov 2022 07:12:19.976 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
compose-redis-1  | 1:C 30 Nov 2022 07:12:19.976 # Redis version=7.0.5, bits=64, commit=00000000, modified=0, pid=1, just started
compose-redis-1  | 1:C 30 Nov 2022 07:12:19.976 # Warning: no config file specified, using the default config. In order to specify a
config file use redis-server /path/to/redis.conf
compose-redis-1  | 1:M 30 Nov 2022 07:12:19.977 * monotonic clock: POSIX clock_gettime
compose-redis-1  | 1:M 30 Nov 2022 07:12:19.977 * Running mode=standalone, port=6379.
compose-redis-1  | 1:M 30 Nov 2022 07:12:19.977 # Server initialized
compose-redis-1  | 1:M 30 Nov 2022 07:12:19.977 # WARNING overcommit_memory is set to 0! Background save may fail under low memory co
ndition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommi
t_memory=1' for this to take effect.
compose-redis-1  | 1:M 30 Nov 2022 07:12:19.978 * Ready to accept connections
compose-web-1    |  * Serving Flask app 'app.py'
compose-web-1    |  * Debug mode: on
compose-web-1    | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server inst
ead.
compose-web-1    |  * Running on all addresses (0.0.0.0)
compose-web-1    |  * Running on http://127.0.0.1:5000
compose-web-1    |  * Running on http://172.19.0.2:5000
compose-web-1    | Press CTRL+C to quit
compose-web-1    |  * Restarting with stat
compose-web-1    |  * Debugger is active!
compose-web-1    |  * Debugger PIN: 120-804-403
compose-web-1    | 172.19.0.1 - - [30/Nov/2022 07:12:29] "GET / HTTP/1.1" 200 -
```

Test running



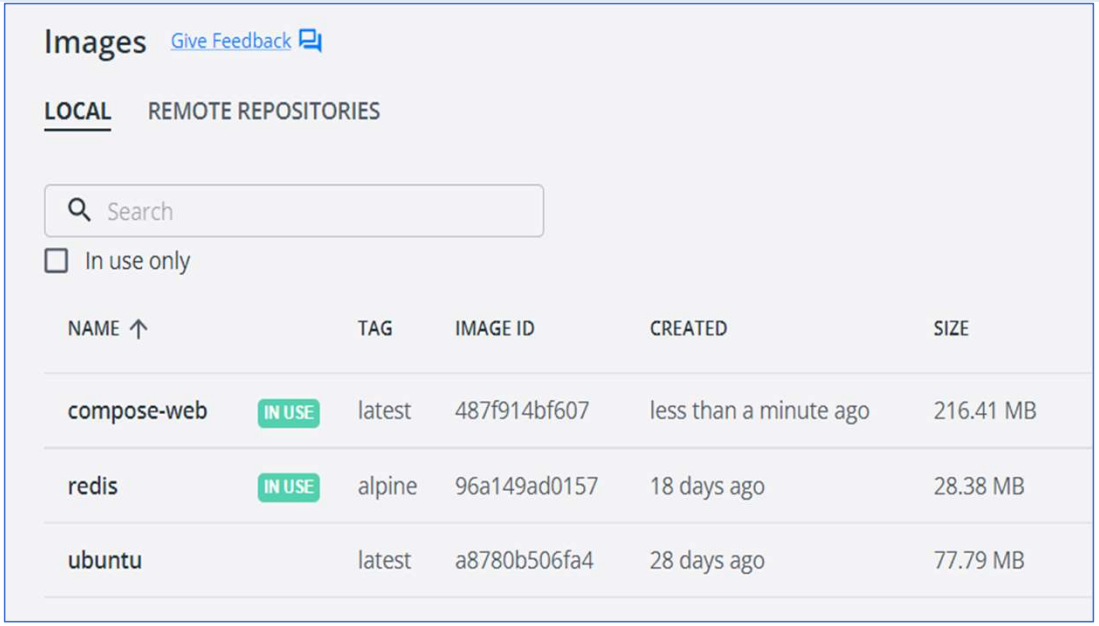


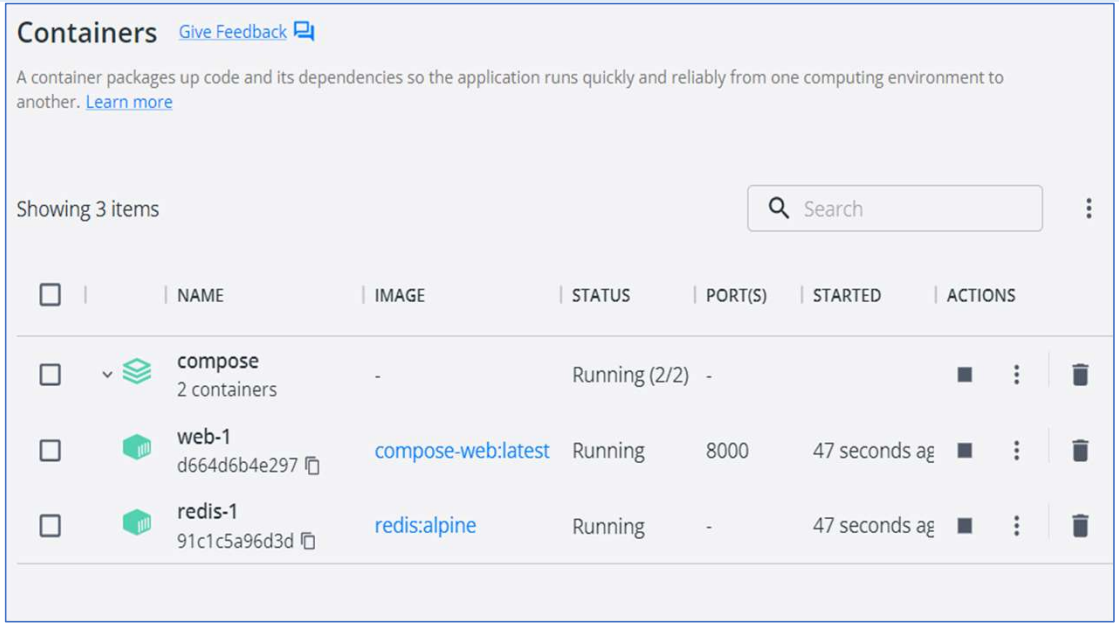

```
compose-web-1    |  * Debug mode: on
compose-web-1    | WARNING: This is a development server. Do not use it in a production deployment.
ead.
compose-web-1    |  * Running on all addresses (0.0.0.0)
compose-web-1    |  * Running on http://127.0.0.1:5000
compose-web-1    |  * Running on http://172.19.0.2:5000
compose-web-1    | Press CTRL+C to quit
compose-web-1    |  * Restarting with stat
compose-web-1    |  * Debugger is active!
compose-web-1    |  * Debugger PIN: 120-804-403
compose-web-1    | 172.19.0.1 - - [30/Nov/2022 07:12:29] "GET / HTTP/1.1" 200 -
compose-web-1    | 172.19.0.1 - - [30/Nov/2022 07:13:47] "GET / HTTP/1.1" 200 -
compose-web-1    | 172.19.0.1 - - [30/Nov/2022 07:13:48] "GET / HTTP/1.1" 200 -
compose-web-1    | 172.19.0.1 - - [30/Nov/2022 07:13:49] "GET / HTTP/1.1" 200 -
```

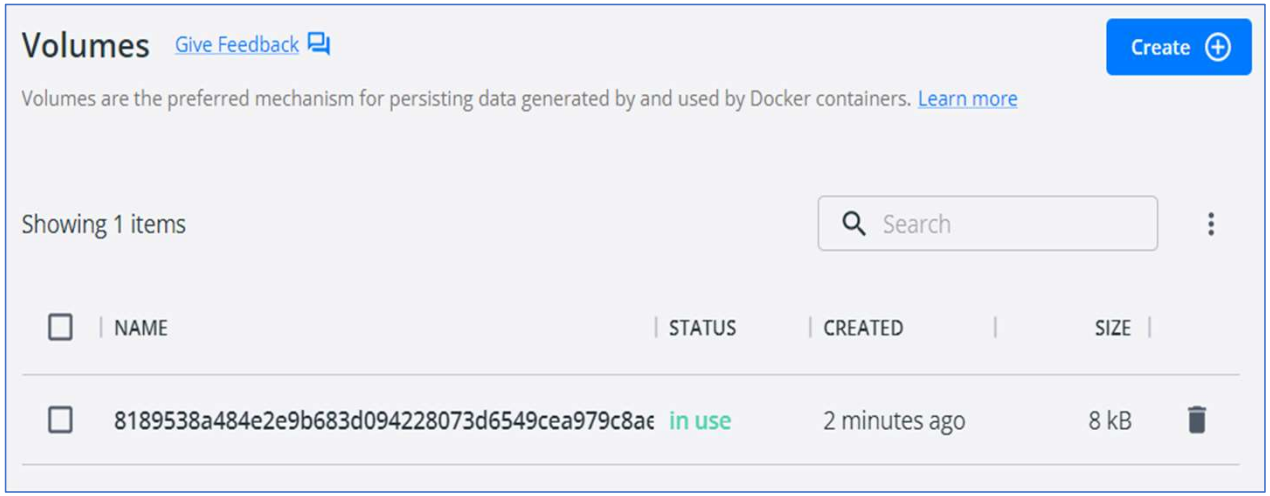
Tracing

```
D:\>docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
compose-web   latest    487f914bf607   7 minutes ago    216MB
redis         alpine    96a149ad0157   2 weeks ago      28.4MB
ubuntu        latest    a8780b506fa4   3 weeks ago      77.8MB

D:\>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS
NAMES
91c1c5a96d3d   redis:alpine   "docker-entrypoint.s…"   7 minutes ago   Up 7 minutes   6379/tcp
compose-redis-1
d664d6b4e297   compose-web    "flask run"              7 minutes ago   Up 7 minutes   0.0.0.0:8000->5000/tcp
compose-web-1

D:\>docker volume ls
DRIVER    VOLUME NAME
local     8189538a484e2e9b683d094228073d6549cea979c8ae17fca70ebe2d3c1d3bc0

D:\>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                      PORTS
NAMES
91c1c5a96d3d   redis:alpine   "docker-entrypoint.s…"   15 minutes ago   Exited (0) 8 seconds ago
compose-redis-1
d664d6b4e297   compose-web    "flask run"              15 minutes ago   Exited (0) 8 seconds ago
compose-web-1
```

Monitor from Docker Desktop







docker compose up -d

- You can pass the -d flag (for "*detached*" mode) to run your services in the background.

```
D:\docker\compose>docker compose up -d
[+] Running 7/7
 - redis Pulled                                                                  7.2s
   - ca7dd9ec2225 Pull complete                                                  1.3s
   - 83276aa4de36 Pull complete                                                  1.4s
   - 731cc432e6da Pull complete                                                  1.6s
   - 862de9590cc6 Pull complete                                                  2.6s
   - a26b23e71d57 Pull complete                                                  2.6s
   - 4b937ee5a2e0 Pull complete                                                  2.7s
[+] Building 4.4s (15/15) FINISHED
 => [internal] load build definition from dockerfile                             0.0s
 => => transferring dockerfile: 32B                                              0.0s
 => [internal] load .dockerignore                                                0.0s
```

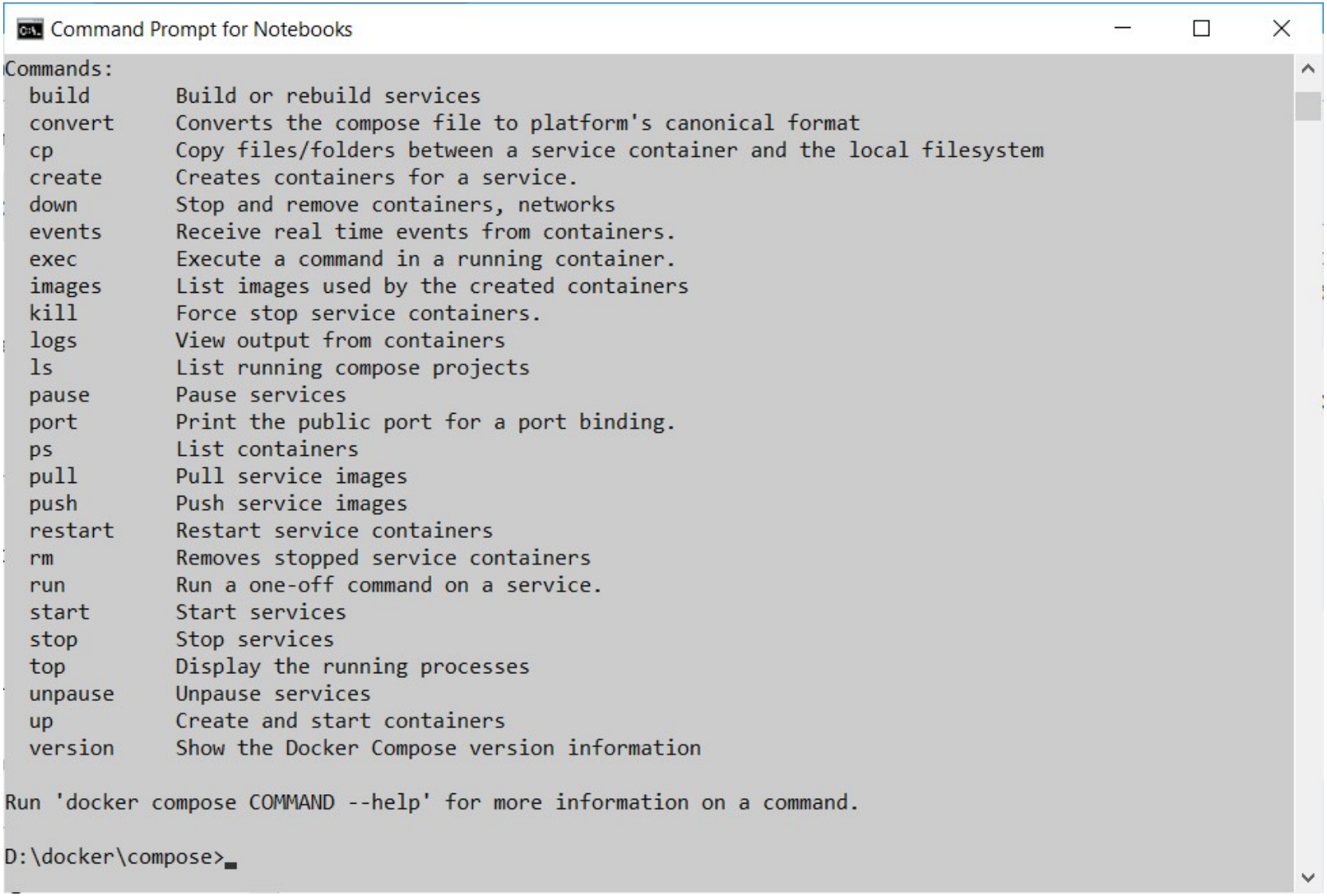
. . .

```
 => exporting to image                                                           0.0s
 => => exporting layers                                                          0.0s
 => => writing image sha256:edc77f380ee0ea078d7f21d00b5d1ffdba90c4d950aa06124a5a0d66a111e97d   0.0s
 => => naming to docker.io/library/compose-web                                   0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[+] Running 3/3
 - Network compose_default    Created                                            0.7s
 - Container compose-redis-1  Started                                            1.6s
 - Container compose-web-1    Started                                            1.9s

D:\docker\compose>
```

docker compose stop and docker compose down --volume

- Stop your services once you've finished with them.
- Remove the containers entirely, with the down command.
  Pass --volumes to also remove the data volume used by the Redis container:

docker compose --help



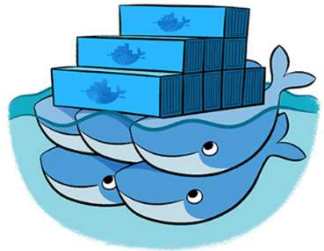
Outline

- DevOp and Software development process
- Docker
- Docker Hands-on Practice
- Build and test a dockerfile for https
- Docker compose
- Kubernetes - concepts
- K8s hands-on practice
- K8s YAML file for nginx

Docker Swarm vs Kubernetes

- *Docker Compose* is used for starting multiple Docker containers on the same host – so you don't have to start each container separately. You may do so by configuring a from within a single *YAML* file.

- *Docker swarm* is a container orchestration tool that allows you to run and connect containers on multiple hosts, for a scalable application.



- *Kubernetes* is a container orchestration tool that is similar to Docker swarm, but has a wider appeal due to its scalability, flexibility, high-availability, and ease of automation.

kubernetes

Container orchestration

- Management of multiple containers in a cluster.
  - Deploying
  - Scaling
  - Maintaining availability
  - Self-healing in case of failures.
  - Traffic management and load balancing.
- E.g. Kubernetes, Docker Swarm, Mesos, and Nomad.

Image source: https://www.musicgateway.com/blog/how-to/music-conductor

Kubernetes

- Open-source *container orchestration* tool
- Developed by Google
- Helps you manage containerized applications in different deployment environments:
  - Physical machines, virtual machines, cloud, or hybrid.

- Why needs a container orchestration tool?
  - Increased usage of containers.
  - Proper way to manage a pool of containers.
  - Trend move from monoliths to microservices.

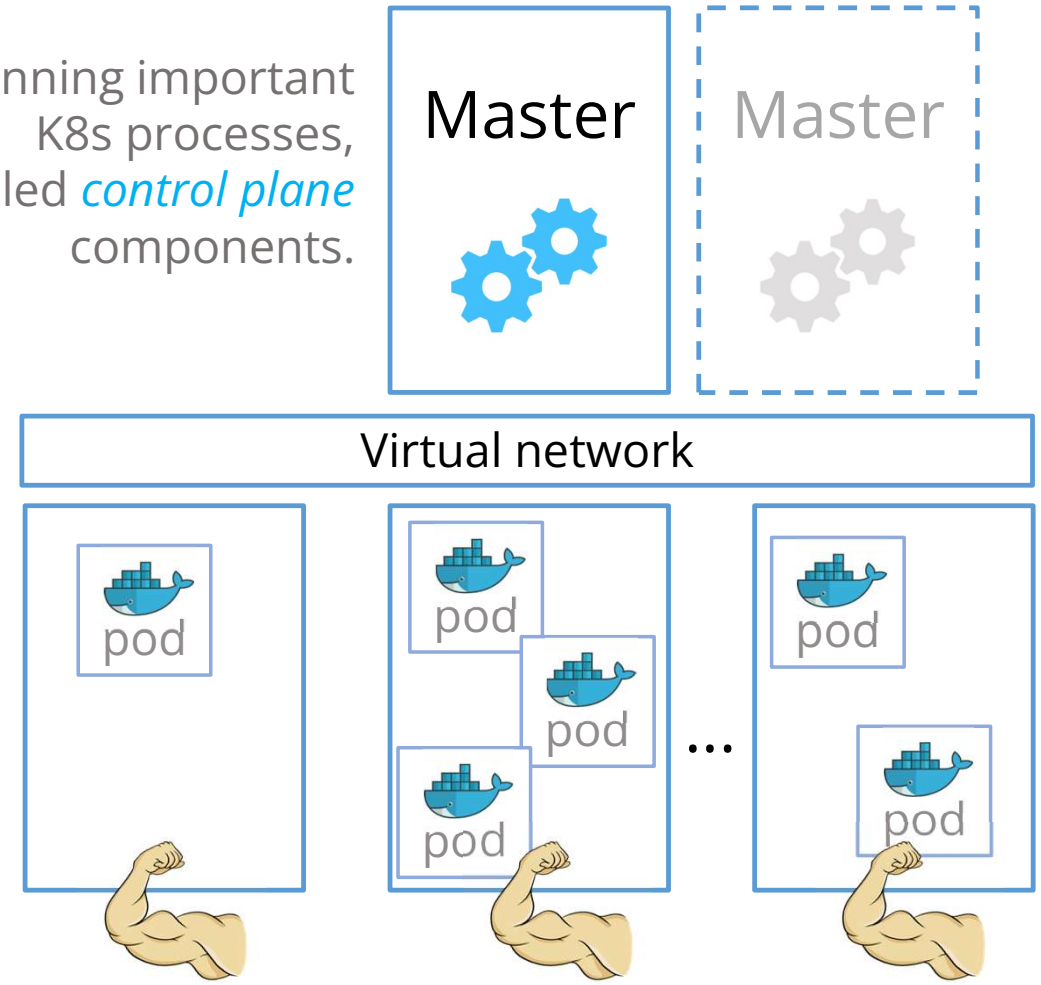
Cluster

- Cluster is a set of nodes
- Node can be a physical machine or a virtual machine (VM).
- A K8S cluster consists of a master node and at least one work node.

Kubernetes basic architecture – a cluster

Kubernetes runs your workload by placing containers into *Pods* to run on *Nodes*. A node may be a virtual or physical machine.

Running important K8s processes, called *control plane* components.



Worker node(s)

Running your application containers.

Pod is the smallest unit of K8s. It's abstraction over container.

K8s Master node and its control plane components

api

the *control plane* component that handles all API requests.

c-m

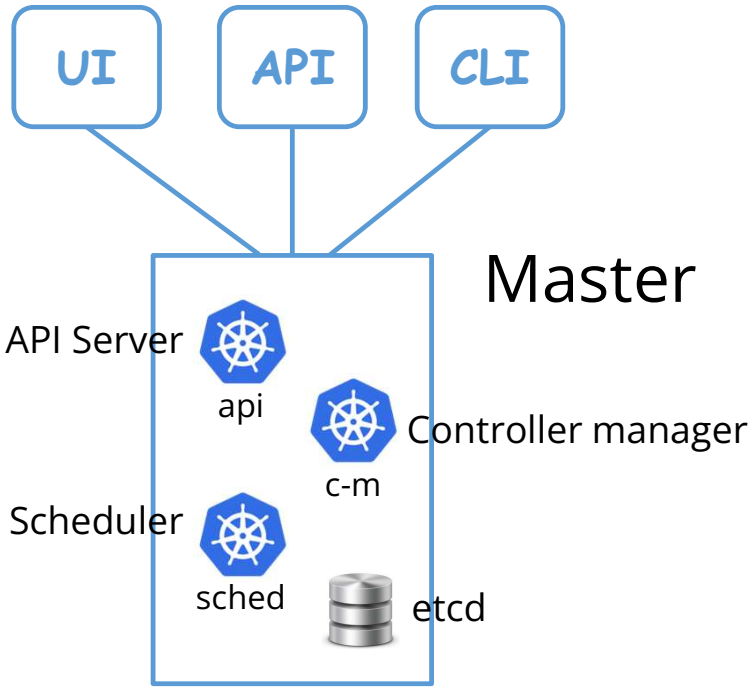
handles routine tasks in the cluster, such as replicating pods, and responding to changes in the cluster.

sched

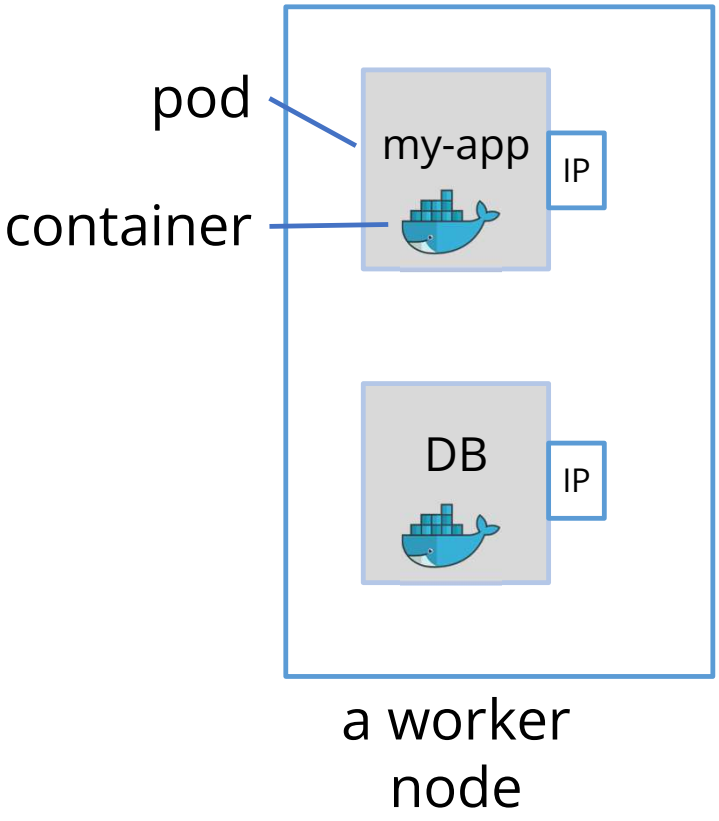
ensures *pods* placement on a node.

etcd

is a key-value store of the current states of K8s, for restoration.



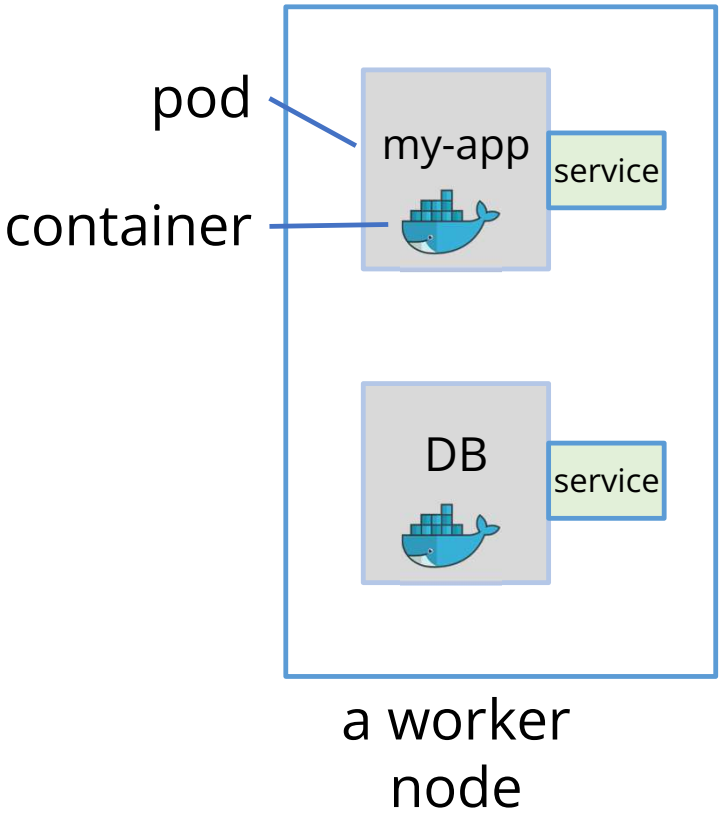
K8s Worker nodes



pod

- Smallest deployable unit of computing in K8s.
- A *Pod* is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers.
- Each pod gets its own IP address.
- New IP address on re-creation of the pod.
- So, the *service* in introduced here.

K8s Worker nodes



service

- Service is a *static IP address* that can be attached to each pod.
- Pod is ephemeral, while the service is permanent.

- *External* services
  - Open the communication to external sources e.g. web.
- *Internal* services
  - Database normally should use internal services.

Defining a *pod*

```yaml
apiVersion: v1

kind: Pod
metadata:
  name: nginx

spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

An example of a Pod which consists of a container running the image nginx:1.14.2.

This is an example of a YAML file*, a configuration file of K8s to define its components.

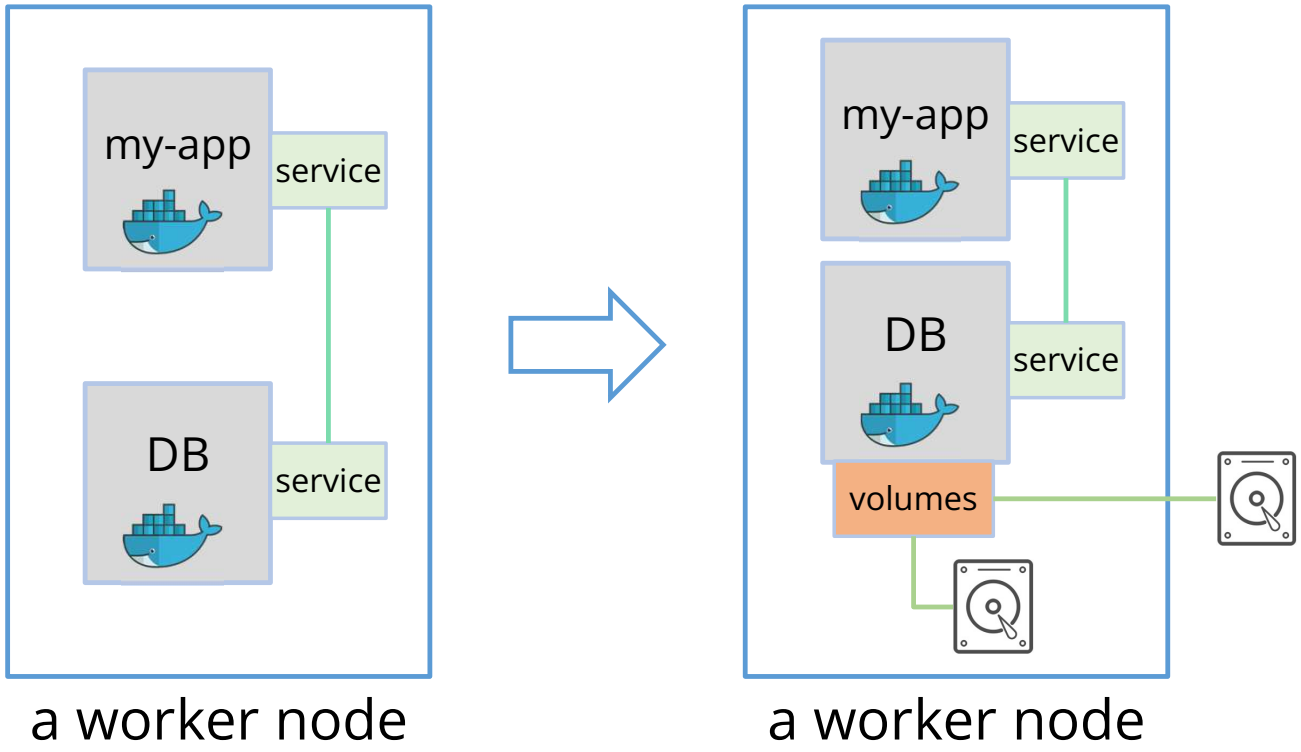
kubectl apply -f https://k8s.io/examples/pods/simple-pod.yaml

* YAML Ain't markup language.

More info is here.

Volume

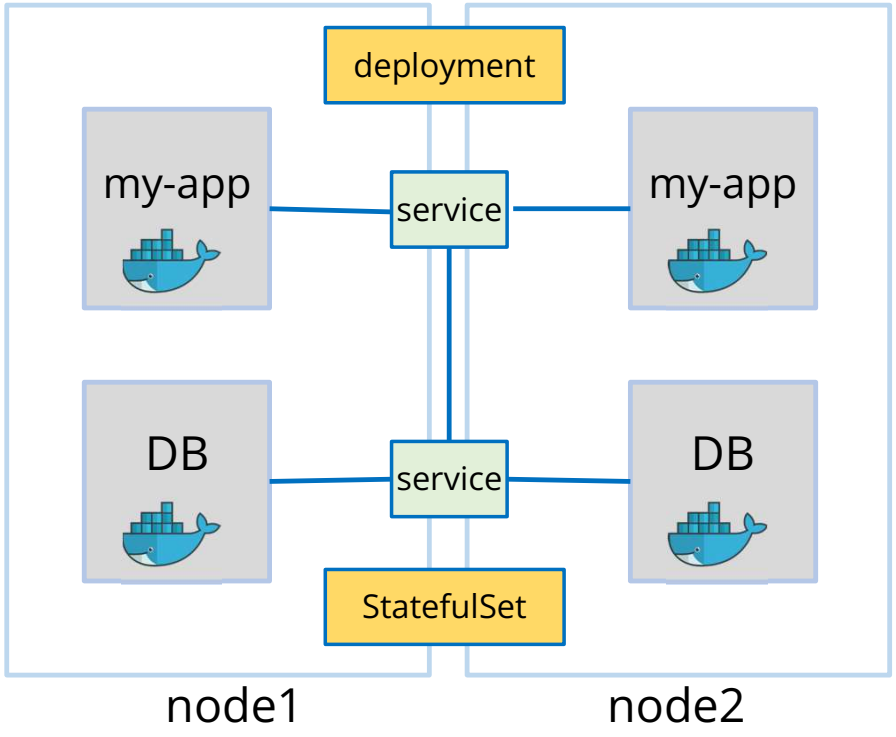
If the DB pod crashes, data are gone.



Volume

- Various forms *Persistent* data.
  - on local machine.
  - outside of the K8s cluster
  - on cloud: AWS, Azure
- K8s doesn't manage these persistent data.
  - You've to take of backing up, etc.

Deployment and StatefulSet



A *service* has 2 functionalities:

- Permanent IP
- Load balancing

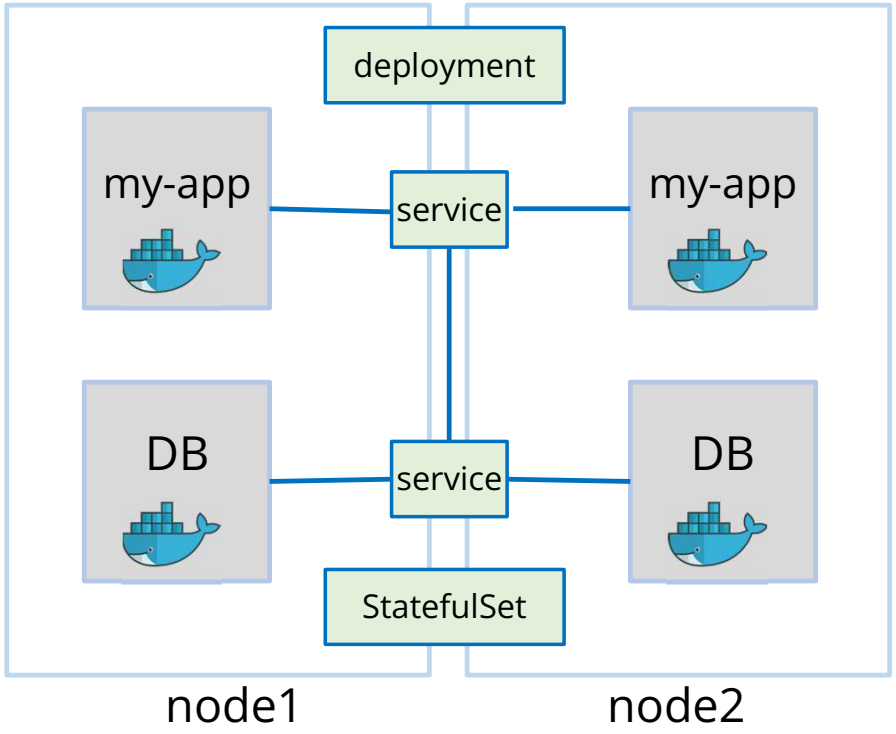
Deployment

- Blueprint for my-app pods.
- A higher abstraction of pods – can define number of *replicas*, while the *pods can't*!
- DB can't be replicated via deployment.
  - DB has states.

StatefulSet

- For replicating DB to ensure consistencies.
- Useful for MongoDB, ElasticSearch, etc.

Deployment and StatefulSet



- Deployment for stateless apps.
- StatefulSet for stateful apps or databases.

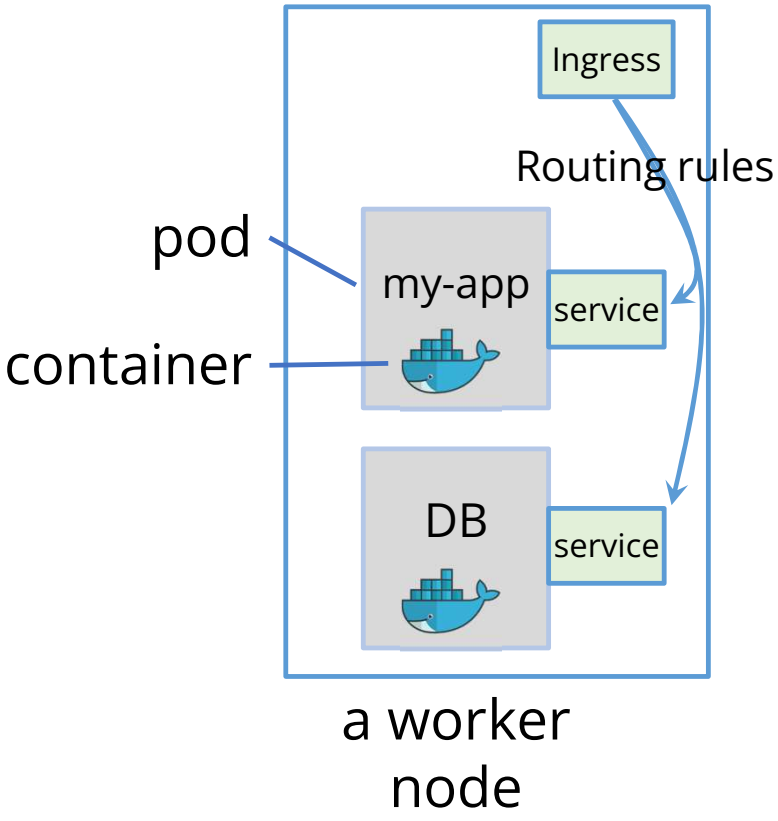
- StatefulSet is somewhat complex.
- Deploying stateful apps in K8s is complex.
- DB are often hosted outside K8s.

K8s ConfigMap

- A ConfigMap is an object used to store non-confidential data in key-value pairs.
- Pods can consume ConfigMaps as environment variables, command-line arguments, or as configuration files in a volume.
- To decouple environment-specific configuration from your container images, so that the applications are easily portable.
- ConfigMap does *not provide encryption*.
  - If the data you want to store are confidential, use a Secret rather than a ConfigMap.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: game-demo
data:
  # property-like keys; each key maps to a simple value
  player_initial_lives: "3"
  ui_properties_file_name: "user-interface.properties"

  # file-like keys
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
  user-interface.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
```

K8s Worker nodes



Ingress

- *Ingress* exposes HTTP and HTTPS routes from outside the cluster to services within the cluster.
- Ingress is a way to route incoming network traffic to the appropriate service within a cluster.
- Traffic routing is controlled by rules defined on the Ingress resource.
- To use more advanced routing rules or features like path-based routing, authentication, encryption, and rate limiting to incoming traffic.

Summary: main components of K8s

- *Pods* – abstraction of containers.
- *Service* – communication.
- *Deployment* and *StatefulSet* – replication.
- *Volumes* – for data storage definition.
- *ConfigMap* and *Secrets* – External configuration.
- *Ingress* – route traffic into clusters.

Outline

- DevOp and Software development process
- Docker
- Docker Hands-on Practice
- Build and test a dockerfile for https
- Docker compose
- Kubernetes - concepts
- K8s hands-on practice
- K8s YAML file for nginx

Tools: minikube and kubectl

- *minikube* is a tool that helps you run *a single-node Kubernetes cluster locally* on your machine.
  - Minikube is a way to set up a cluster on your local machine to test and experiment with.
  - Primarily used for development and testing purposes.

- *kubectl* is a command-line tool for *interacting* with and *configure* a K8s cluster.
  - kubectl is necessary to interact with the cluster
  - To deploy, inspect, update, and manage services running on a K8s cluster.

Install minikube

- Go to minikube to install it.

minikube

1 Installation

Click on the buttons that describe your target platform. For other architectures, see the release page for a complete list of minikube binaries.

Operating system	Linux	macOS	**Windows**
Architecture	**x86-64**		
Release type	**Stable**	Beta	
Installer type	**.exe download**	Windows Package Manager	Chocolatey

To install the latest minikube **stable** release on **x86-64 Windows** using **.exe download**:

1. Download and run the installer for the latest release.
   Or if using `PowerShell`, use this command:

```
New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force
Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes
```

2. Add the `minikube.exe` binary to your `PATH`.
   Make sure to run PowerShell as Administrator.

```
$oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Mac
if ($oldPath.Split(';') -inotcontains 'C:\minikube'){ `
  [Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [Enviro
}
```

If you used a terminal (like powershell) for the installation, please close the terminal and reopen it before running minikube.

Check if the minikube has been installed

```
Microsoft Windows [Version 10.0.19044.2486]
(c) Microsoft Corporation. All rights reserved.

D:\notebooks>minikube
minikube provisions and manages local Kubernetes clusters optimized for development workflows.

Basic Commands:
  start            Starts a local Kubernetes cluster
  status           Gets the status of a local Kubernetes cluster
  stop             Stops a running local Kubernetes cluster
  delete           Deletes a local Kubernetes cluster
  dashboard        Access the Kubernetes dashboard running within the minikube cluster
  pause            pause Kubernetes
  unpause          unpause Kubernetes

Images Commands:
  docker-env       Provides instructions to point your terminal's docker-cli to the Docker Engine insi
(Useful for building docker images directly inside minikube)
  podman-env       Configure environment to use minikube's Podman service
  cache            Manage cache for images
  image            Manage images

Configuration and Management Commands:
  addons           Enable or disable a minikube addon
  config           Modify persistent configuration values
  profile          Get or list the current profiles (clusters)
  update-context   Update kubeconfig in case of an IP or port change
```

Start the minikube with 2 nodes: master node and a worker node

```
D:\notebooks>minikube start -n=2
* minikube v1.28.0 on Microsoft Windows 10 Home Single Language 10.0.19044 Build 19044
* Automatically selected the docker driver
* Using Docker Desktop driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.25.3 preload ...
    > preloaded-images-k8s-v18-v1...:  385.44 MiB / 385.44 MiB  100.00% 31.12 M
    > gcr.io/k8s-minikube/kicbase:  386.27 MiB / 386.27 MiB  100.00% 6.21 MiB p
    > gcr.io/k8s-minikube/kicbase:  0 B [_____________________________] ?% ? p/s 21s
* Creating docker container (CPUs=2, Memory=2200MB) ...
* Preparing Kubernetes v1.25.3 on Docker 20.10.20 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass

* Starting worker node minikube-m02 in cluster minikube
* Pulling base image ...
* Creating docker container (CPUs=2, Memory=2200MB) ...
* Found network options:
  - NO_PROXY=192.168.49.2
  - no_proxy=192.168.49.2
```

minikube status

```
D:\docker>minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

minikube-m02
type: Worker
host: Running
kubelet: Running
```

```
D:\docker>minikube status
minikube
type: Control Plane
host: Stopped
kubelet: Stopped
apiserver: Stopped
kubeconfig: Stopped

minikube-m02
type: Worker
host: Stopped
kubelet: Stopped

D:\docker>kubectl get nodes
Unable to connect to the server: dial tcp 127.0.0.1:53938: connectex: No connection could be made because the target machine actively refused it.
```

In case the minikube has been stopped!

```
D:\docker>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS          PORTS
NAMES
cb02ae271b01   gcr.io/k8s-minikube/kicbase:v0.0.36   "/usr/local/bin/entr…"   13 hours ago   Up
38 minutes   127.0.0.1:62772->22/tcp, 127.0.0.1:62773->2376/tcp, 127.0.0.1:62775->5000/tcp,
127.0.0.1:62776->8443/tcp, 127.0.0.1:62774->32443/tcp   minikube-m02
ba0e7e394a03   gcr.io/k8s-minikube/kicbase:v0.0.36   "/usr/local/bin/entr…"   13 hours ago   Up
38 minutes   127.0.0.1:62635->22/tcp, 127.0.0.1:62636->2376/tcp, 127.0.0.1:62633->5000/tcp,
127.0.0.1:62634->8443/tcp, 127.0.0.1:62637->32443/tcp   minikube
```

Check status with kubectl

```
D:\docker>kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
minikube       Ready    control-plane   13h   v1.25.3
minikube-m02   Ready    <none>          39m   v1.25.3

D:\docker>kubectl get pods -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS       AGE
kube-system   coredns-565d847f94-4l7nr           1/1     Running   1 (40m ago)    13h
kube-system   etcd-minikube                      1/1     Running   1 (40m ago)    13h
kube-system   kindnet-pq5wz                      1/1     Running   1 (40m ago)    13h
kube-system   kindnet-zcpg4                      1/1     Running   1 (39m ago)    13h
kube-system   kube-apiserver-minikube            1/1     Running   1 (40m ago)    13h
kube-system   kube-controller-manager-minikube   1/1     Running   1 (40m ago)    13h
kube-system   kube-proxy-7jjpf                   1/1     Running   1 (40m ago)    13h
kube-system   kube-proxy-cc9bw                   1/1     Running   1 (39m ago)    13h
kube-system   kube-scheduler-minikube            1/1     Running   1 (40m ago)    13h
```

Create a deployment of nginx with 3 replicas

```
D:\docker>kubectl delete deployment mynginx
deployment.apps "mynginx" deleted

D:\docker>kubectl create deployment mynginx --image nginx:latest --replicas 3
deployment.apps/mynginx created

D:\docker>kubectl get pods
NAME                         READY   STATUS              RESTARTS        AGE
mynginx-55b5b97fd6-rnd89     0/1     ContainerCreating   0               13s
mynginx-55b5b97fd6-sqgsh     0/1     ContainerCreating   0               13s
mynginx-55b5b97fd6-wsc5h     1/1     Running             0               13s

D:\docker>kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
mynginx   2/3     3            2           34s
```

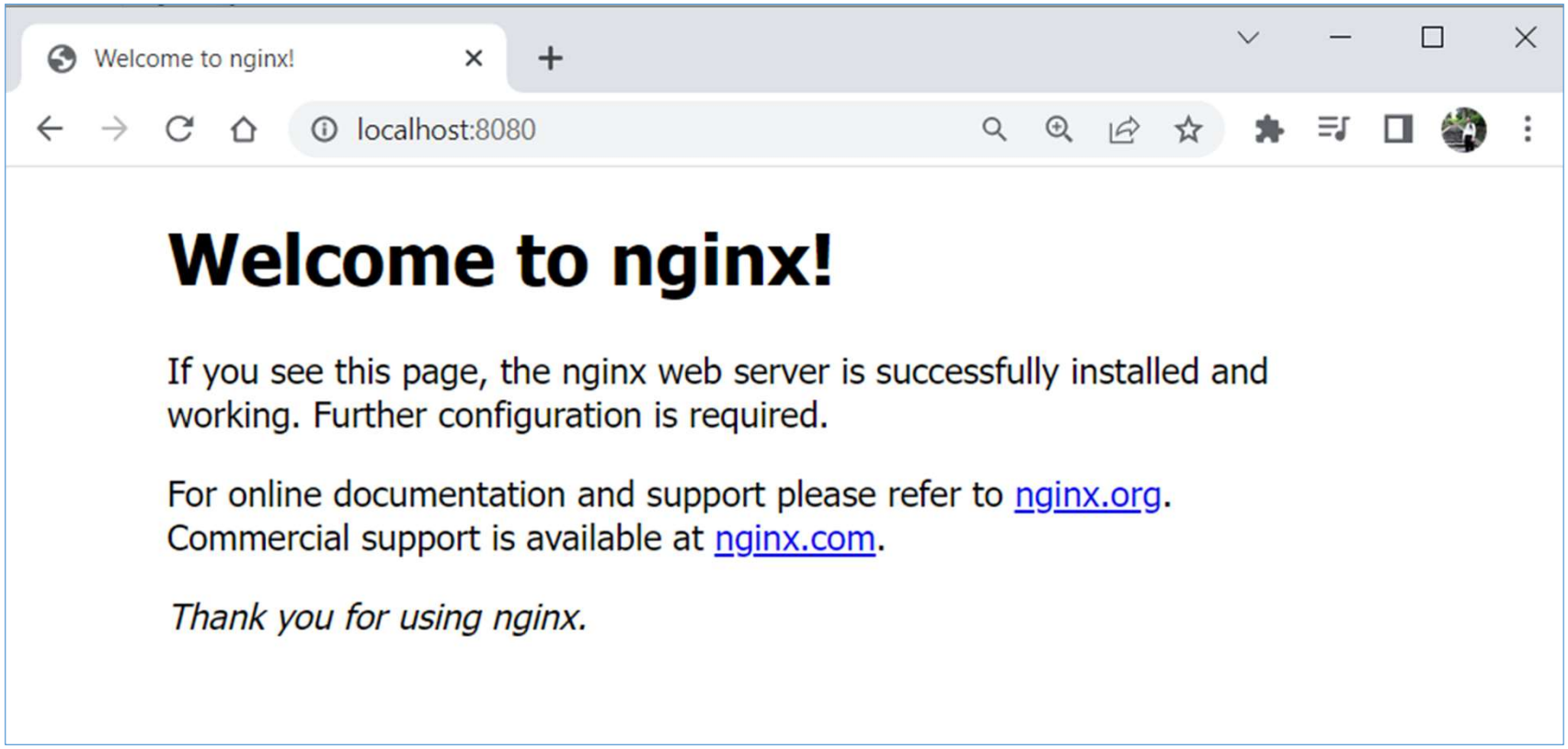
Create the nginx deployment to 5 replicas

```
D:\docker>kubectl scale deployment mynginx --replicas 5
deployment.apps/mynginx scaled

D:\docker>kubectl get pods
NAME                         READY   STATUS              RESTARTS   AGE
mynginx-55b5b97fd6-lr5v9     0/1     ContainerCreating   0          3s
mynginx-55b5b97fd6-rnd89     1/1     Running             0          51s
mynginx-55b5b97fd6-rrdqp     0/1     ContainerCreating   0          3s
mynginx-55b5b97fd6-sqgsh     0/1     ContainerCreating   0          51s
mynginx-55b5b97fd6-wsc5h     1/1     Running             0          51s

D:\docker>kubectl get pods
NAME                         READY   STATUS              RESTARTS   AGE
mynginx-55b5b97fd6-lr5v9     1/1     Running             0          23s
mynginx-55b5b97fd6-rnd89     1/1     Running             0          71s
mynginx-55b5b97fd6-rrdqp     1/1     Running             0          23s
mynginx-55b5b97fd6-sqgsh     1/1     Running             0          71s
mynginx-55b5b97fd6-wsc5h     1/1     Running             0          71s
```

Expose mynginx's port 80 to external port 8080

```
D:\docker>kubectl expose deployment/mynginx --port 80
service/mynginx exposed

D:\docker>kubectl get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   24m
mynginx      ClusterIP   10.100.137.111   <none>        80/TCP    21s

D:\docker>kubectl port-forward service/mynginx 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
Handling connection for 8080
```

Test accessing mynginx with port 8080



Monitor K8s's components

```
D:\docker>kubectl get deployments,pods,services
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mynginx       5/5     5            5           16m

NAME                              READY   STATUS    RESTARTS   AGE
pod/mynginx-55b5b97fd6-lr5v9      1/1     Running   0          15m
pod/mynginx-55b5b97fd6-rnd89      1/1     Running   0          16m
pod/mynginx-55b5b97fd6-rrdqp      1/1     Running   0          15m
pod/mynginx-55b5b97fd6-sqgsh      1/1     Running   0          16m
pod/mynginx-55b5b97fd6-wsc5h      1/1     Running   0          16m

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   38m
service/mynginx      ClusterIP   10.100.137.111   <none>        80/TCP    14m
```

Delete pod, service or deployment

```
D:\docker>kubectl delete service mynginx
service "mynginx" deleted


D:\docker>kubectl delete deployment nginx
deployment.apps "nginx" deleted
```

Outline

- DevOp and Software development process
- Docker
- Docker Hands-on Practice
- Build and test a dockerfile for https
- Docker compose
- Kubernetes - concepts
- K8s hands-on practice
- K8s YAML file for nginx

K8s YAML file

- A Kubernetes (K8s) YAML file is a configuration file written in YAML (YAML Ain't Markup Language) format.
- It is used to define and create objects such as pods, deployments, services, and others in a Kubernetes cluster.
- The file contains a set of key-value pairs that define the properties and behavior of the object.
- The file can be used with the **`kubectl`** command to create, update or delete the object in the cluster.
- The file can also be used to define multiple objects in the same file using the **---** separator between each object.

Write a yaml file to configure K8s components

```yaml
apiVersion: v1
kind: Service

metadata:
  name: my-nginx-svc
  labels:
    app: nginx

spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: nginx

---

apiVersion: apps/v1
kind: Deployment
```

continue

```yaml
apiVersion: apps/v1
kind: Deployment

metadata:
  name: my-nginx
  labels:
    app: nginx

spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

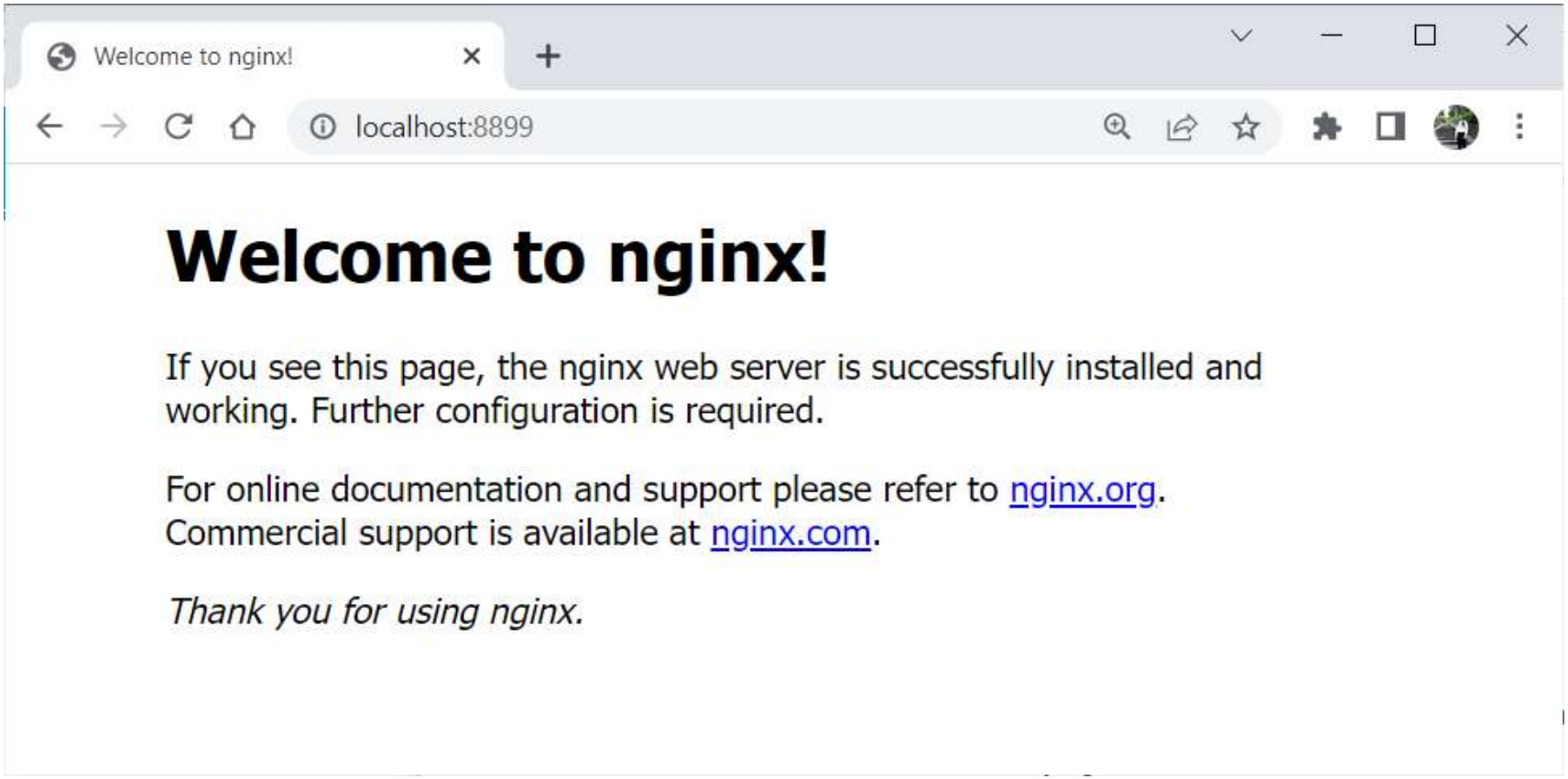
Run K8s to use nginx with the yaml file

```
D:\docker\kubernetes>kubectl apply -f nginx-app.yaml
service/my-nginx-svc created
deployment.apps/my-nginx created

D:\docker\kubernetes>kubectl get deployments,pods,services
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/my-nginx      3/3     3            3           60s

NAME                                READY   STATUS    RESTARTS   AGE
pod/my-nginx-7fb96c846b-cb5c5       1/1     Running   0          60s
pod/my-nginx-7fb96c846b-g5wc4       1/1     Running   0          60s
pod/my-nginx-7fb96c846b-rvlzh       1/1     Running   0          60s

NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)          AGE
service/kubernetes      ClusterIP      10.96.0.1        <none>         443/TCP          2m12s
service/my-nginx-svc    LoadBalancer   10.111.234.181   <pending>      80:31771/TCP     60s
```

Run K8s to use nginx with the yaml file

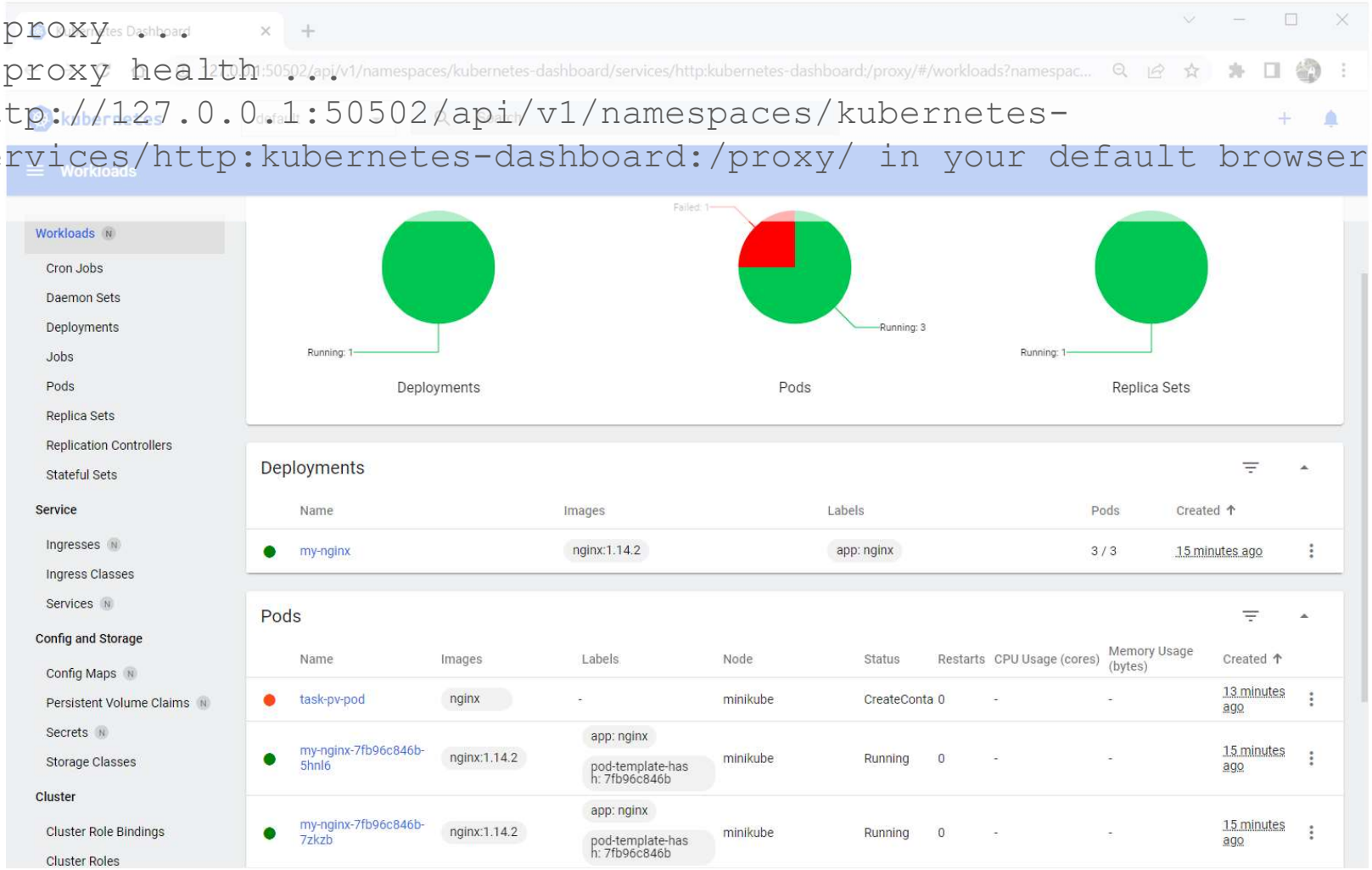
```
D:\docker\kubernetes>kubectl get pods -A
NAMESPACE     NAME                                    READY   STATUS    RESTARTS          AGE
default       my-nginx-7fb96c846b-cb5c5               1/1     Running   0                 3m21s
default       my-nginx-7fb96c846b-g5wc4               1/1     Running   0                 3m21s
default       my-nginx-7fb96c846b-rvlzh               1/1     Running   0                 3m21s
kube-system   coredns-565d847f94-bg6fk                1/1     Running   0                 4m18s
kube-system   etcd-minikube                           1/1     Running   0                 4m30s
kube-system   kube-apiserver-minikube                 1/1     Running   0                 4m30s
kube-system   kube-controller-manager-minikube        1/1     Running   0                 4m30s
kube-system   kube-proxy-566bh                        1/1     Running   0                 4m18s
kube-system   kube-scheduler-minikube                 1/1     Running   0                 4m30s
kube-system   storage-provisioner                     1/1     Running   1 (3m57s ago)     4m28s

D:\docker\kubernetes>kubectl autoscale deployment/my-nginx --min=1 --max=4
horizontalpodautoscaler.autoscaling/my-nginx autoscaled

D:\docker\kubernetes>kubectl port-forward deployment/my-nginx 8899:80
Forwarding from 127.0.0.1:8899 -> 80
Forwarding from [::1]:8899 -> 80
Handling connection for 8899
Handling connection for 8899
```

Test the nginx with localhost on port 8899



minikube dashboard

```
D:\docker\kubernetes>minikube dashboard
* Verifying dashboard health ...
* Launching proxy ...
* Verifying proxy health ...
* Opening http://127.0.0.1:50502/api/v1/namespaces/kubernetes-
dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...
```



More to learn

- Kubernetes Fundamentals https://kubebyexample.com/learning-paths/kubernetes-fundamentals/what-kubernetes-3-minutes
- A simple example for deployment https://k8s-examples.container-solutions.com/examples/Deployment/Deployment.html
- Linux essentials https://kubebyexample.com/learning-paths/linux-essentials/what-are-linux-open-source-software-and-distribution
- Youtube: Kubernetes Tutorial https://www.youtube.com/watch?v=yznvWW_L7AA

What we've learnt today

- DevOp and Software development process
- Docker
- Docker Hands-on Practice
- Build and test a dockerfile for https
- Docker compose
- Kubernetes - concepts
- K8s hands-on practice
- K8s YAML file for nginx

Inside K8s master and worker nodes