#SOSC23

CONTAINERS 101

DIEGO CIANGOTTINI - INFN PERUGIA

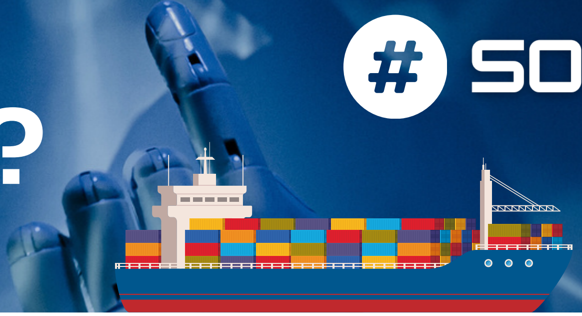
imdciangot

diego-ciangottini-8a03b98a

ciangottini@infn.it

INFN

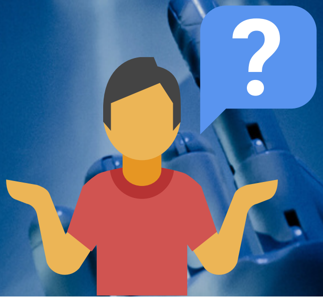
What are containers?



Containers are a form of **virtualization technology** that allows you to **package an application** and its dependencies together, **isolating it from the underlying system**

- Key concepts:
  - **Isolation:** Containers provide a secure and isolated environment for applications, preventing conflicts with other applications or the host system.
  - **Portability:** Containers can run consistently across different environments, making it easy to develop and deploy applications.
  - **Efficiency:** Containers are efficient in terms of resource usage, as they share the host OS kernel.

Why do we care?

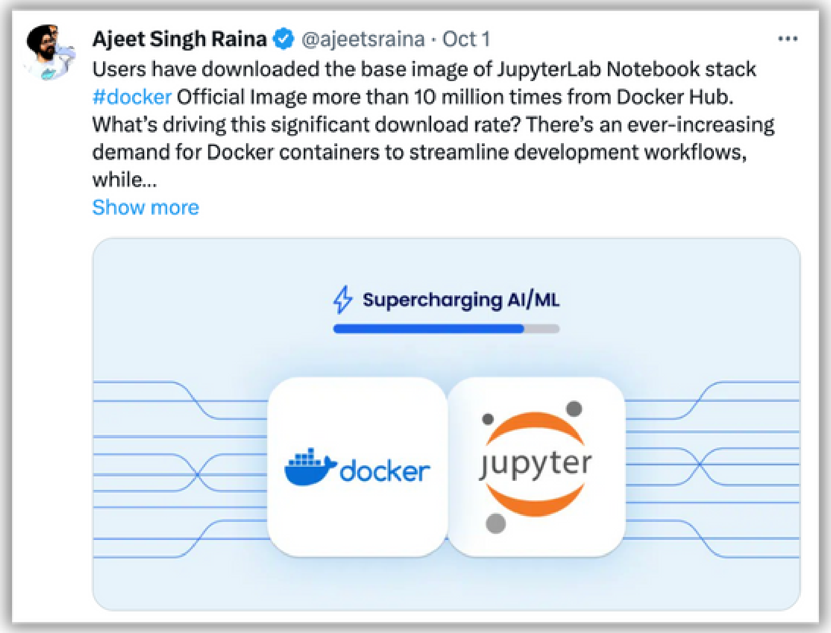


#SOSC23

(Data) science ecosystem speaks container –> unavoidable

The cloud paradigm for resource provisioning is based on containers

In addition you get:

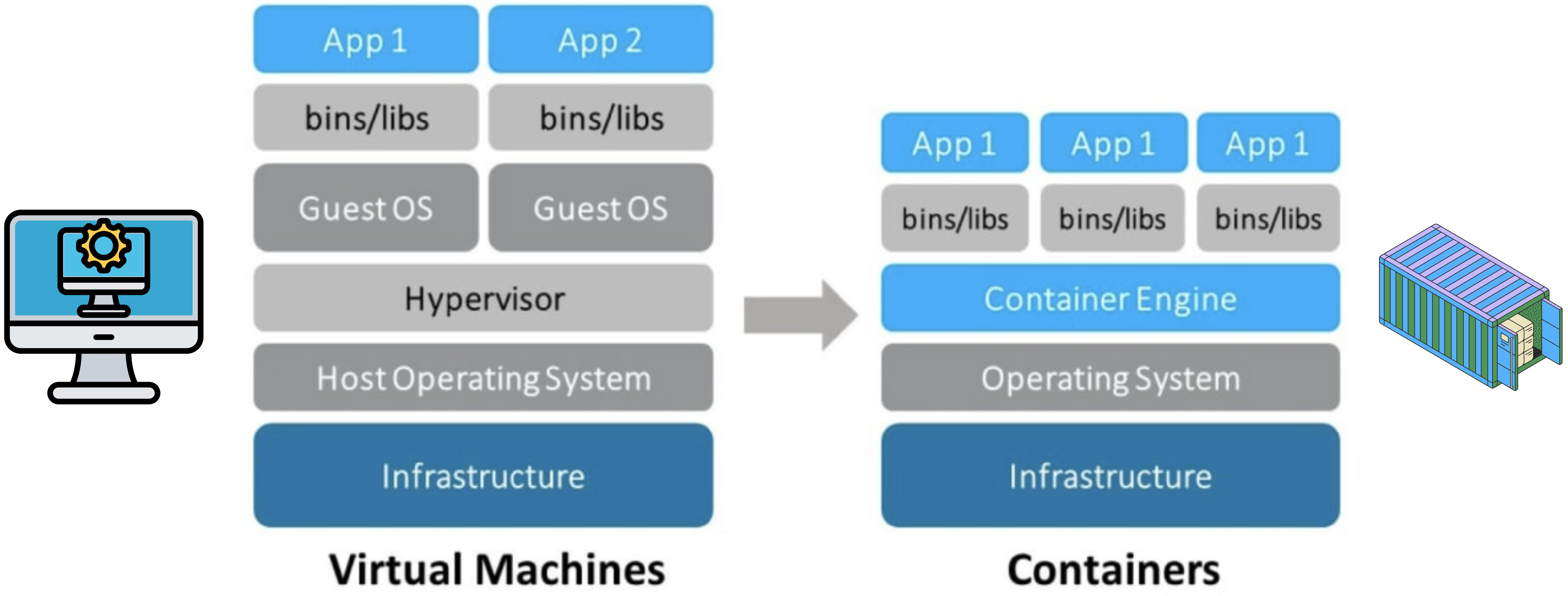

Ajeet Singh Raina @ajeetsraina · Oct 1
Users have downloaded the base image of JupyterLab Notebook stack #docker Official Image more than 10 million times from Docker Hub. What's driving this significant download rate? There's an ever-increasing demand for Docker containers to streamline development workflows, while...
Show more

- **Reproducibility:**
    - bring and share your software with ease
- **Manage and scale**
    - your code will be able to leverage a wide set of opportunities on ~any cloud provider

Virtual Machines vs Containers

#SOSC23

The Containers work on the concept of OS-level virtualization, i.e. the **kernel's ability to make multiple isolated environments** on a single host.



#SOSC23

Pros and cons

VIRTUAL MACHINE

- Provide strong isolation and offer flexibility in choosing different operating systems.
- Heavier, slower to start, and consume more resources due to their independent OS.

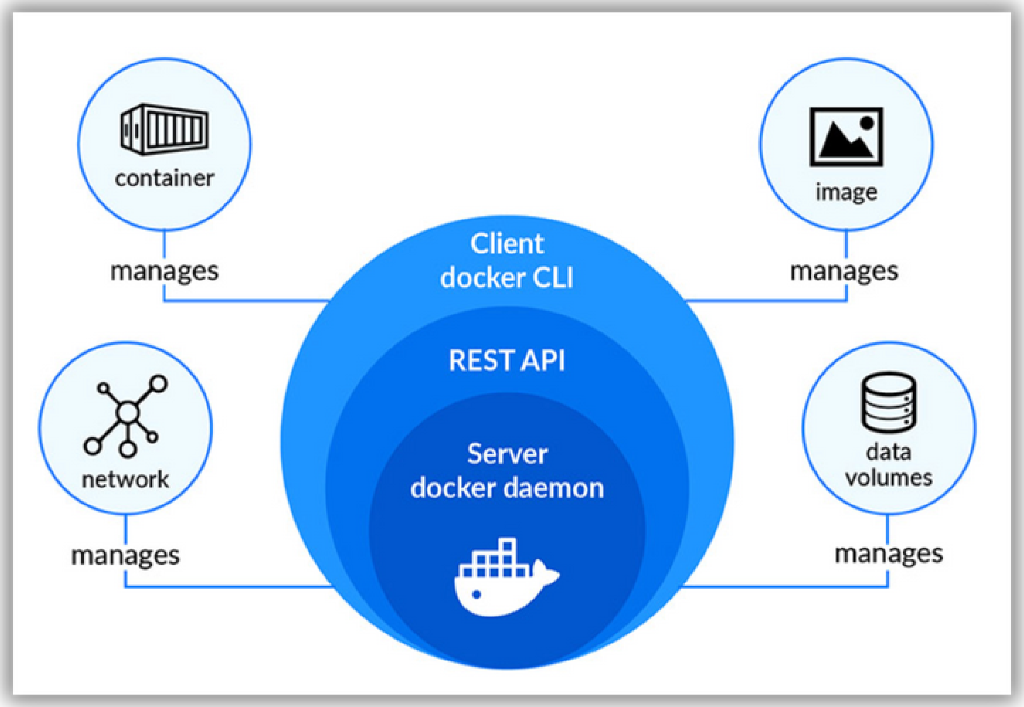
CONTAINER

- Lightweight and efficient, sharing the host OS kernel, faster startup and efficient resource usage.
- Limited OS compatibility and less isolation compared to VMs

Docker

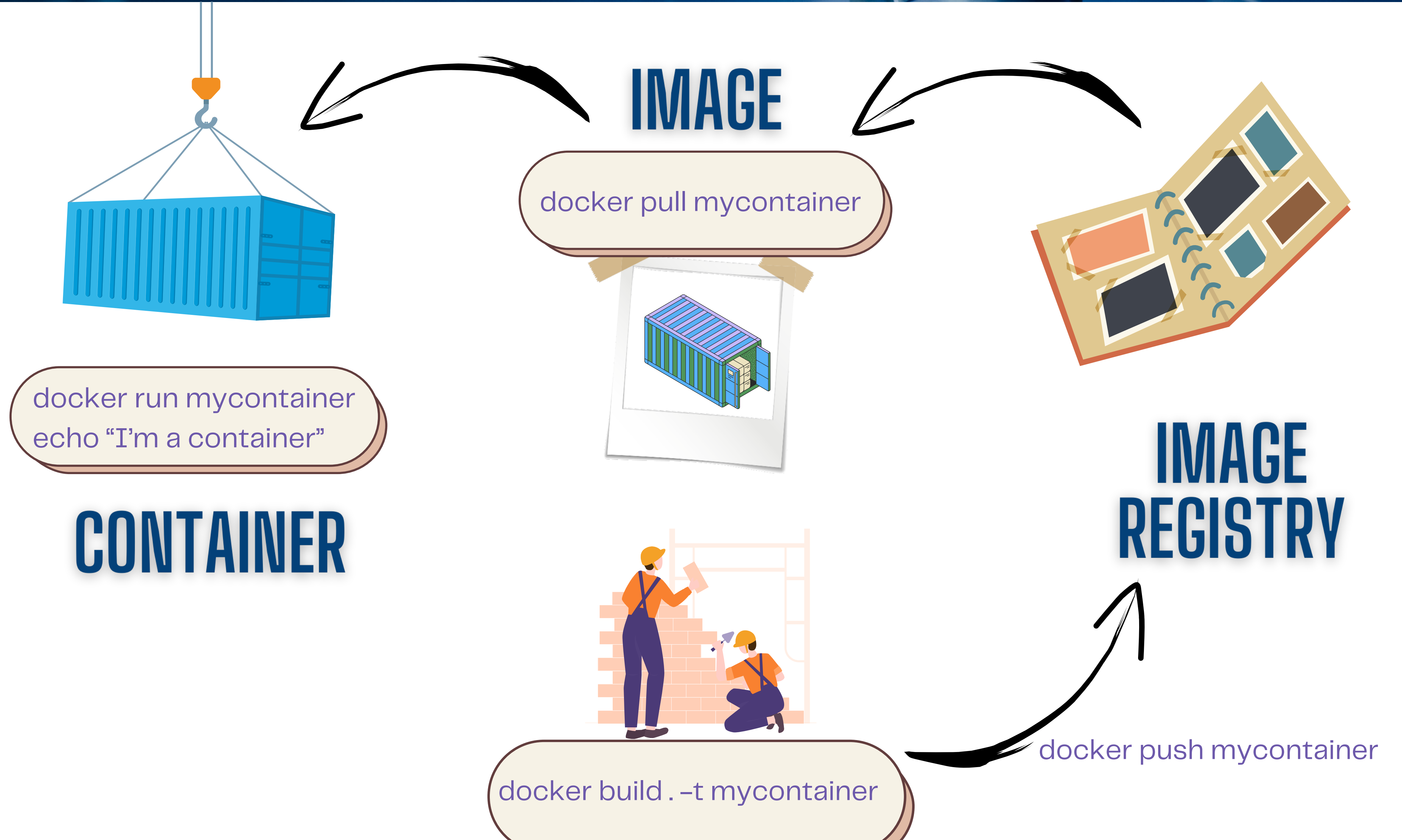
Docker is an open source platform for building, deploying, and managing containerized applications

A client-server architecture:



Ecosystem components

#SOSC23



```
docker pull mycontainer
```

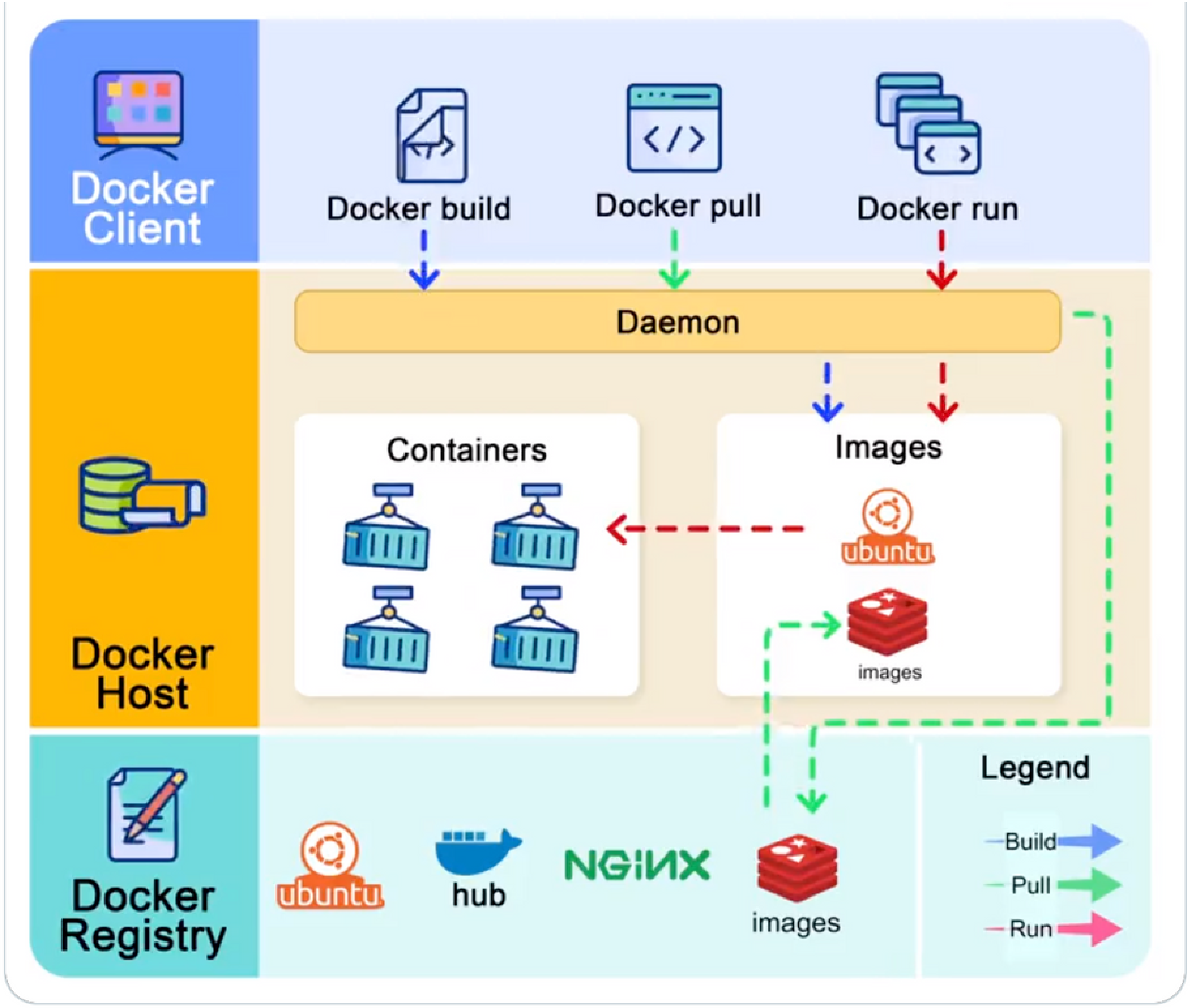
```
docker run mycontainer
echo "I'm a container"
```

```
docker build . -t mycontainer
```

```
docker push mycontainer
```

We will learn the docker way

#SOSC23

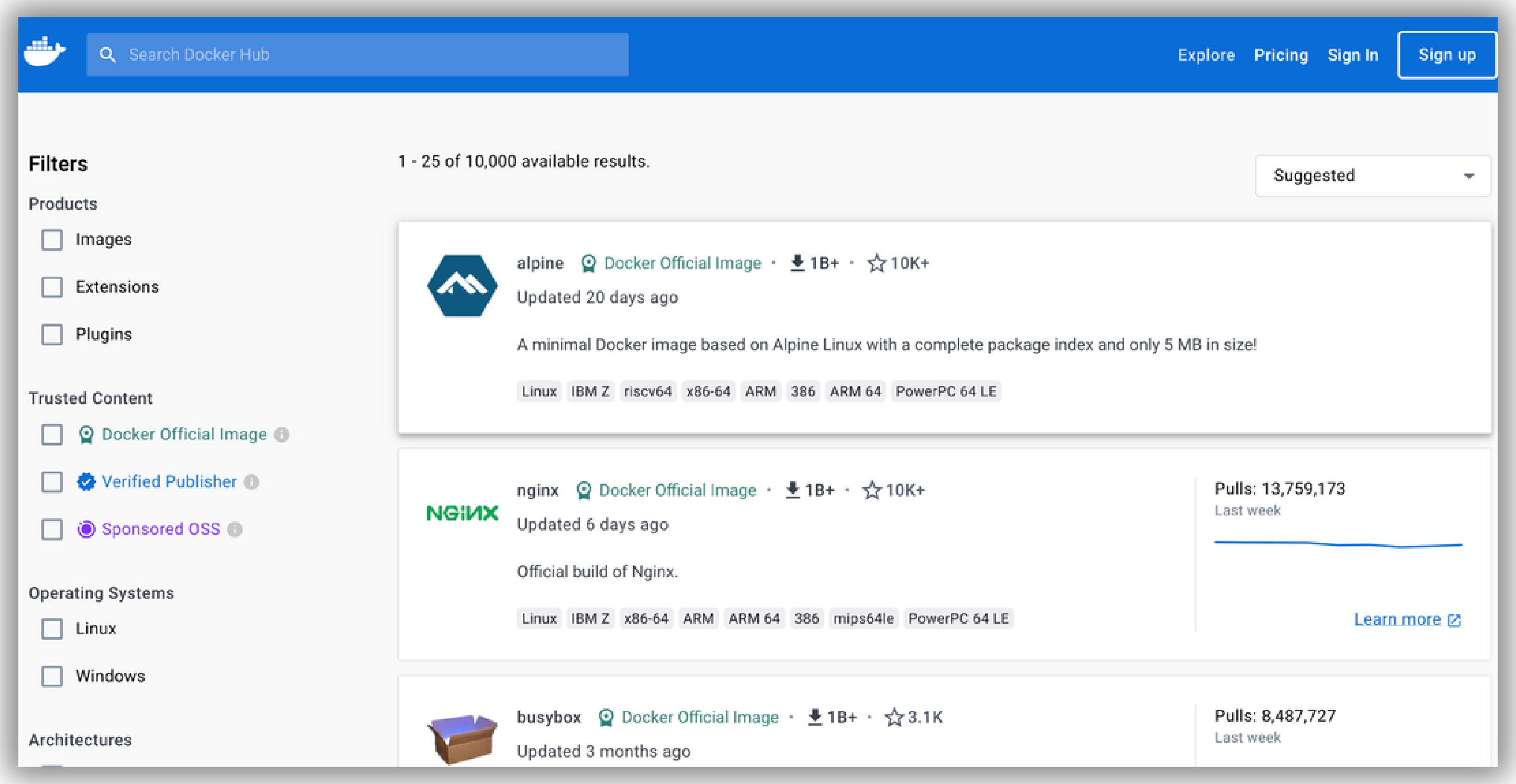


DockerHUB

#SOSC23

Default registry if you use docker CLI
(free with some limitation on space and traffic)

https://hub.docker.com/



Search Docker Hub

Explore Pricing Sign In Sign up

Filters

Products
- Images
- Extensions
- Plugins

Trusted Content
- Docker Official Image
- Verified Publisher
- Sponsored OSS

Operating Systems
- Linux
- Windows

Architectures

1 - 25 of 10,000 available results.

Suggested

alpine · Docker Official Image · 1B+ · 10K+
Updated 20 days ago
A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!
Linux IBM Z riscv64 x86-64 ARM 386 ARM 64 PowerPC 64 LE

nginx · Docker Official Image · 1B+ · 10K+
Updated 6 days ago
Official build of Nginx.
Linux IBM Z x86-64 ARM ARM 64 386 mips64le PowerPC 64 LE
Pulls: 13,759,173
Last week
Learn more

busybox · Docker Official Image · 1B+ · 3.1K
Updated 3 months ago
Pulls: 8,487,727
Last week

Other registries

Other public registries worth mentioning (free to use):

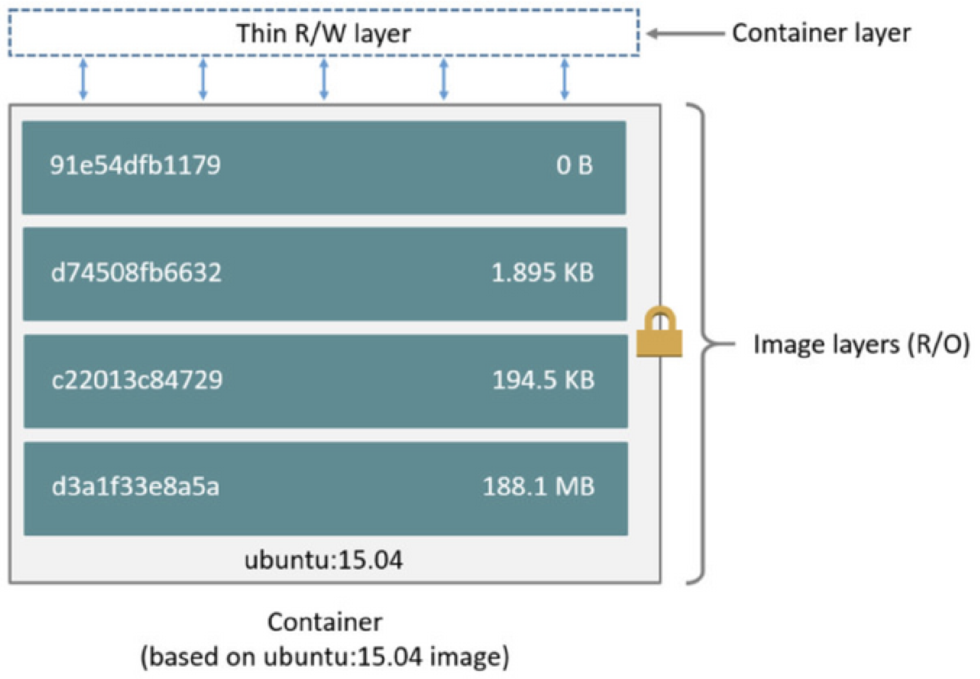
- **GithubContainerRegistry** (ghcr.io)
  - fully integrate in Github CI/CD for images autobuild

Private registries (on-prem)

- **GitLab Container Registry** is tightly integrated with GitLab CI's workflow, with minimal setup.
- **Harbor** (CNCF Graduated project) is an open source registry that secures artifacts with policies and role-based access control, ensures images are scanned and free from vulnerabilities, and signs images as trusted.

Layer by layer

- A Docker Image consists of read-only layers built on top of each other.
- Docker uses the **Union File System** (UFS) to build an image.
- The image is shared across containers.
- Each time Docker launches a container from an image, it adds a thin writable layer, known as the **container layer**, which stores all changes to the container throughout its runtime.

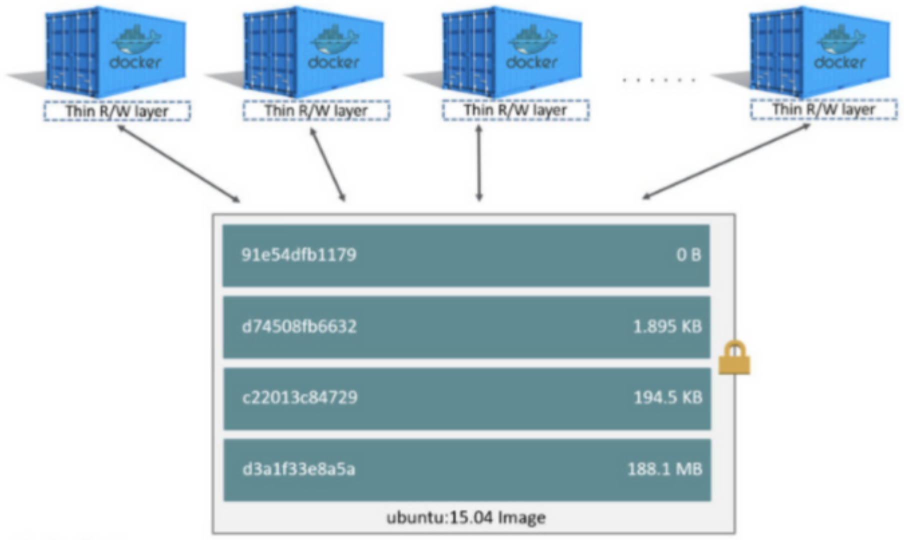


One image many containers

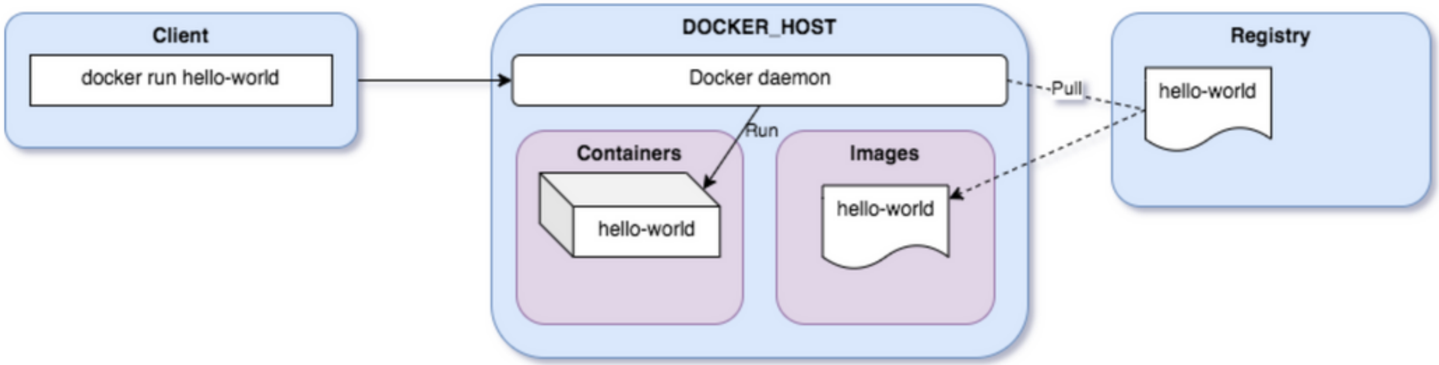
Each container has its own writable container layer, and all changes are stored in this container layer.

Multiple containers can share access to the same underlying image and yet have their own data state.

When the container is deleted, the writable layer is also deleted. The underlying image remains unchanged.



Run a container



HANDS-ON TIME

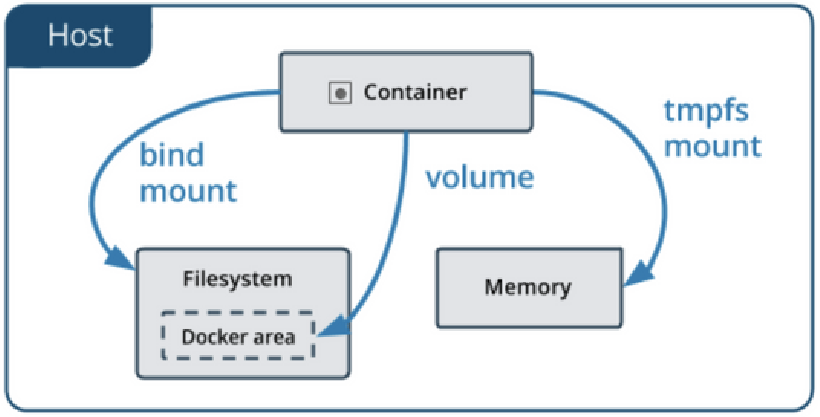
[Link to first hands-on]()

Persist data with volumes

#SOSC23

Docker provides the following options for containers to store files in the host machine, so that the files are persisted even after the container stops

- ❖ volumes
- ❖ bind mounts
- ❖ tmpfs



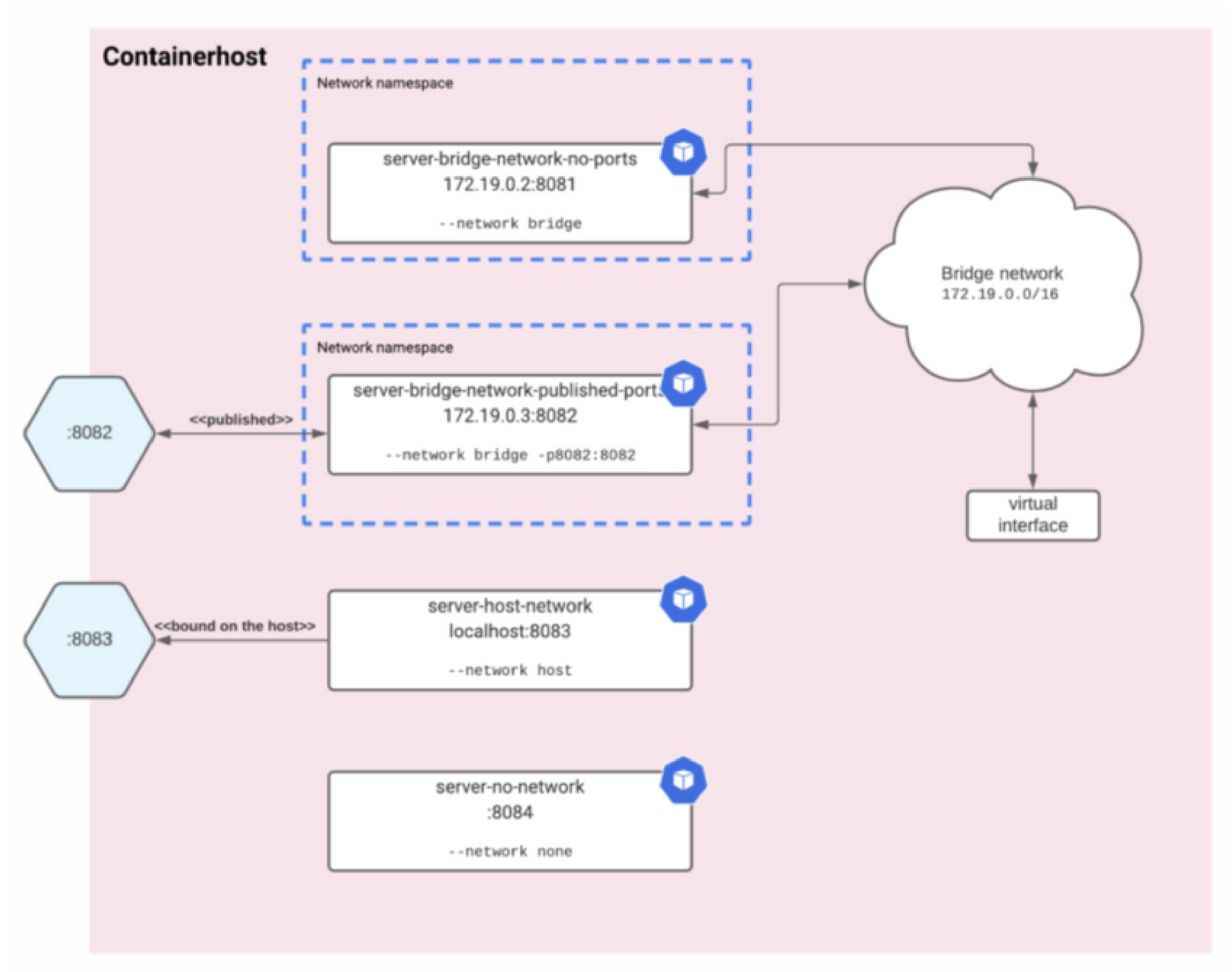
HANDS-ON TIME

Link to volume exercise

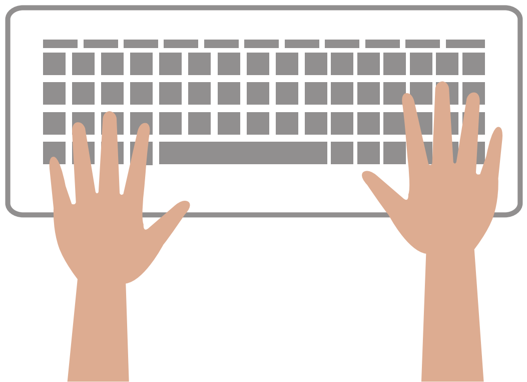
Docker networking

#SOSC23

[Link to network exercise]



HANDS-ON TIME



#SOSC23

MANAGING CONTAINERS

INFN

Graphical Interface

#SOSC23

Portainer

portainer.io

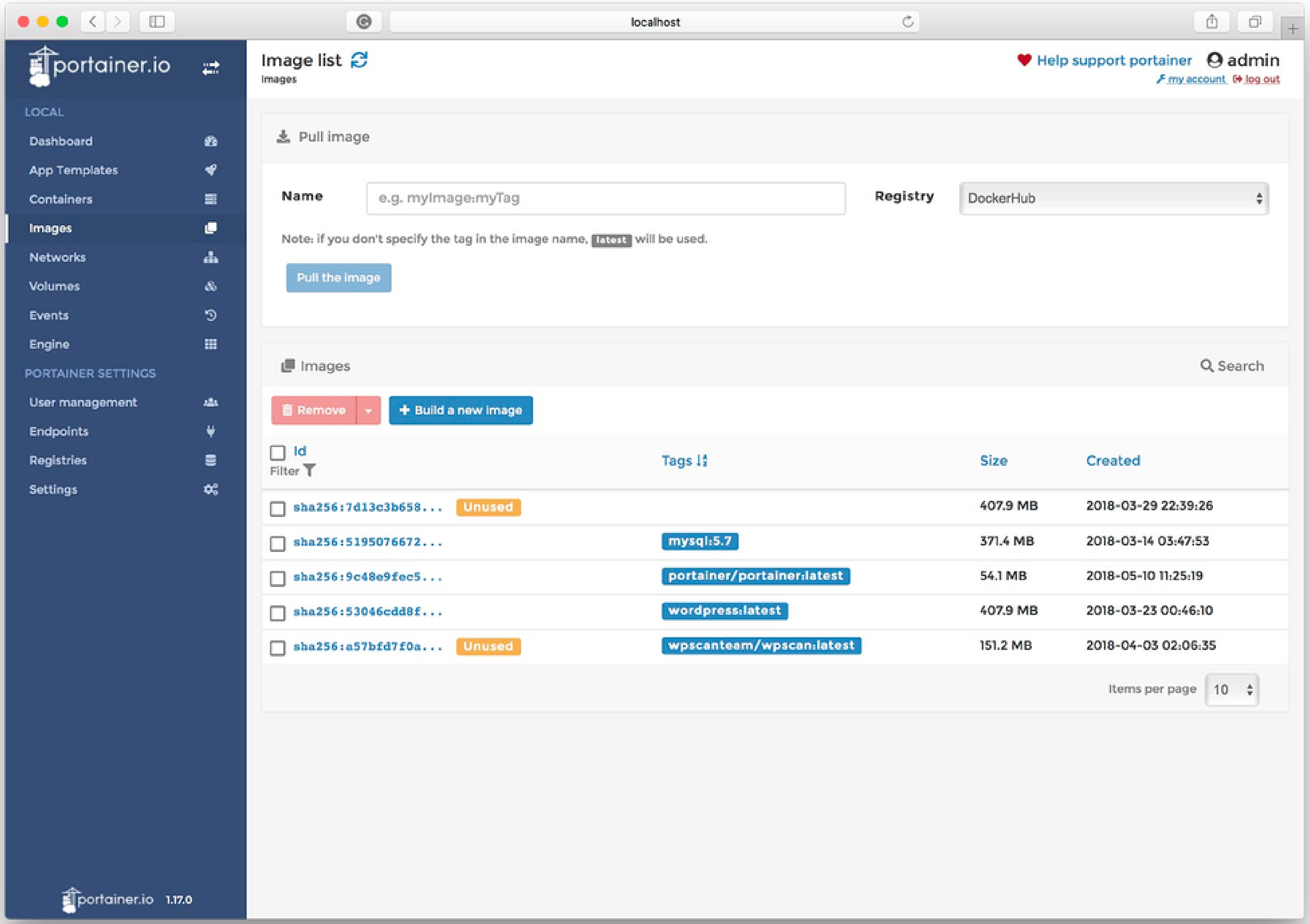


Image list
Images

Pull image

Name: e.g. myImage:myTag   Registry: DockerHub

Note: if you don't specify the tag in the image name, latest will be used.

Images

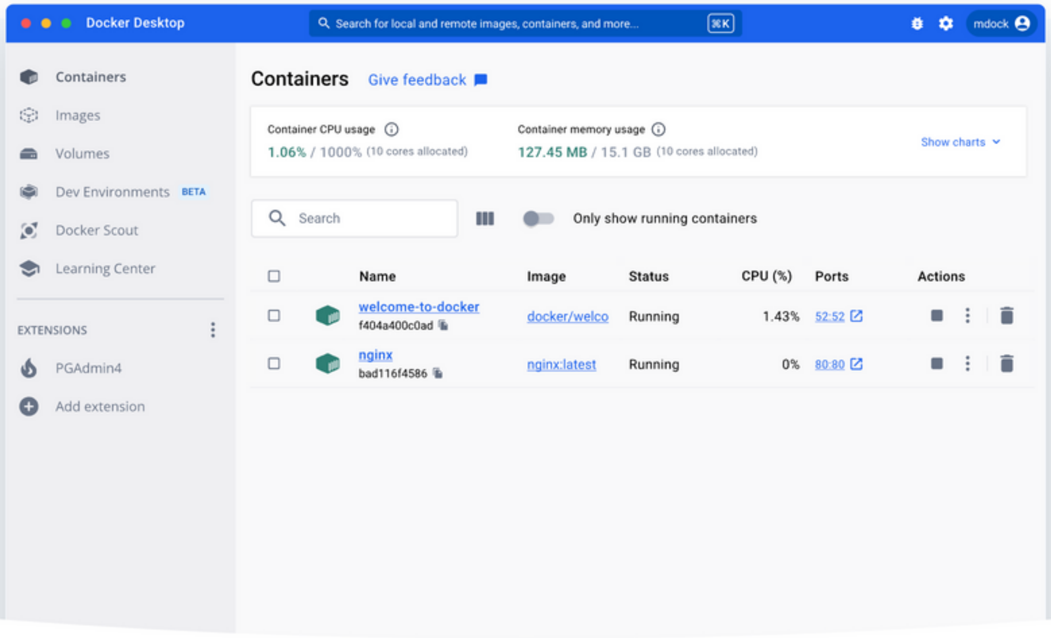
Id	Tags	Size	Created
sha256:7d13c3b658... Unused		407.9 MB	2018-03-29 22:39:26
sha256:5195076672...	mysql:5.7	371.4 MB	2018-03-14 03:47:53
sha256:9c48e9fec5...	portainer/portainer:latest	54.1 MB	2018-05-10 11:25:19
sha256:53046cdd8f...	wordpress:latest	407.9 MB	2018-03-23 00:46:10
sha256:a57bfd7f0a... Unused	wpscanteam/wpscan:latest	151.2 MB	2018-04-03 02:06:35

Items per page 10

Docker on your laptop

#SOSC23

Docker desktop



Docker Desktop

Search for local and remote images, containers, and more... ⌘K

mdock

Containers
Images
Volumes
Dev Environments BETA
Docker Scout
Learning Center

EXTENSIONS

PGAdmin4
Add extension

Containers Give feedback

Container CPU usage
1.06% / 1000% (10 cores allocated)

Container memory usage
127.45 MB / 15.1 GB (10 cores allocated)

Show charts

Search

Only show running containers

	Name	Image	Status	CPU (%)	Ports	Actions
☐	welcome-to-docker f404a400c0ad	docker/welco	Running	1.43%	52:52	
☐	nginx bad116f4586	nginx:latest	Running	0%	80:80	

What's Docker Desktop?

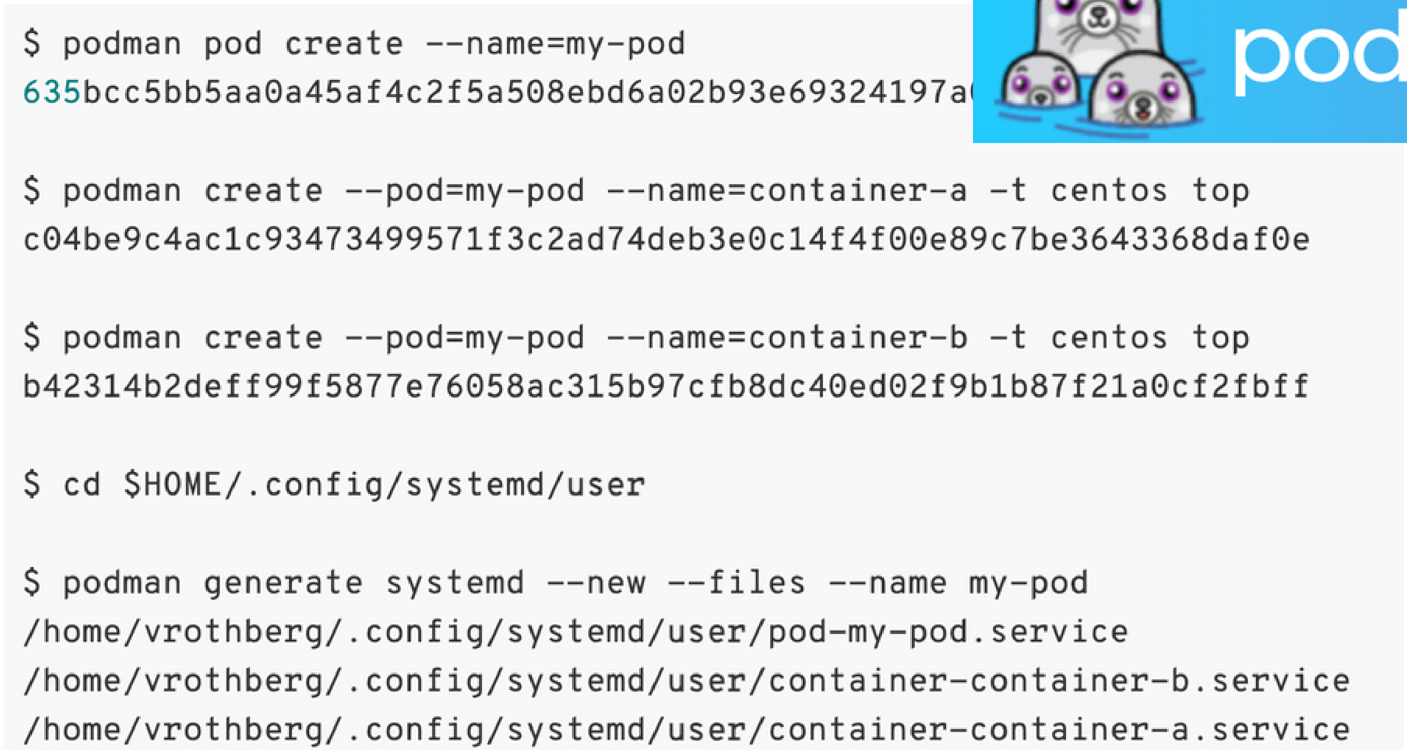
The fastest way to containerize applications

Docker Desktop is secure, out-of-the-box containerization software offering developers and teams a robust, hybrid toolkit to build, share, and run applications anywhere.

Container on systemd with Podman

Podman is an alternative to Docker engine for running containers

WHEN –> it provides features particularly thought for managing long running services



```
$ podman pod create --name=my-pod
635bcc5bb5aa0a45af4c2f5a508ebd6a02b93e69324197a

$ podman create --pod=my-pod --name=container-a -t centos top
c04be9c4ac1c93473499571f3c2ad74deb3e0c14f4f00e89c7be3643368daf0e

$ podman create --pod=my-pod --name=container-b -t centos top
b42314b2deff99f5877e76058ac315b97cfb8dc40ed02f9b1b87f21a0cf2fbff

$ cd $HOME/.config/systemd/user

$ podman generate systemd --new --files --name my-pod
/home/vrothberg/.config/systemd/user/pod-my-pod.service
/home/vrothberg/.config/systemd/user/container-container-b.service
/home/vrothberg/.config/systemd/user/container-container-a.service
```

#SOSC23

BUILD YOUR FIRST IMAGE

INFN

Build you first container image

#SOSC23

[Link to build image – part 1]

[Link to build image – part 2]

HANDS-ON TIME

Sneak peak to...

the image we prepared for you

#SOSC23

We customized an official image with interactive framework (JupyterLab) + all majors data science python packages

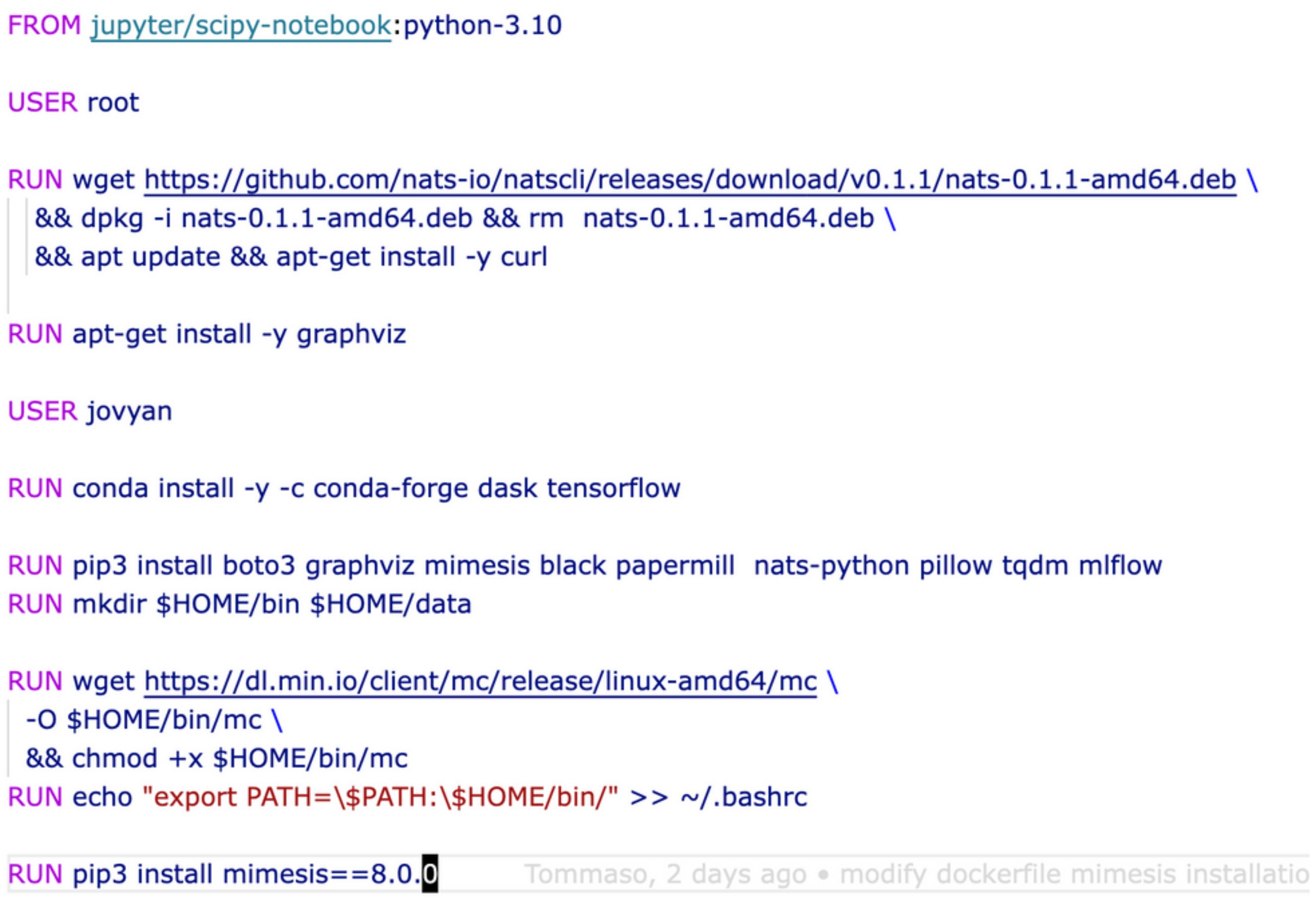
jupyter/docker-stacks

Ready-to-run Docker images containing Jupyter applications

206 Contributors · 17 Used by · 8k Stars · 3k Forks

jupyter/docker-stacks: Ready-to-run Docker images containing Jupyter applications

Ready-to-run Docker images containing Jupyter applications - GitHub - jupyter/docker-stacks: Ready-to-run Docker images containing Jupyter applications


```
FROM jupyter/scipy-notebook:python-3.10

USER root

RUN wget https://github.com/nats-io/natscli/releases/download/v0.1.1/nats-0.1.1-amd64.deb \
  && dpkg -i nats-0.1.1-amd64.deb && rm  nats-0.1.1-amd64.deb \
  && apt update && apt-get install -y curl

RUN apt-get install -y graphviz

USER jovyan

RUN conda install -y -c conda-forge dask tensorflow

RUN pip3 install boto3 graphviz mimesis black papermill  nats-python pillow tqdm mlflow
RUN mkdir $HOME/bin $HOME/data

RUN wget https://dl.min.io/client/mc/release/linux-amd64/mc \
 -O $HOME/bin/mc \
 && chmod +x $HOME/bin/mc
RUN echo "export PATH=\$PATH:\$HOME/bin/" >> ~/.bashrc

RUN pip3 install mimesis==8.0.0        Tommaso, 2 days ago • modify dockerfile mimesis installatio
```

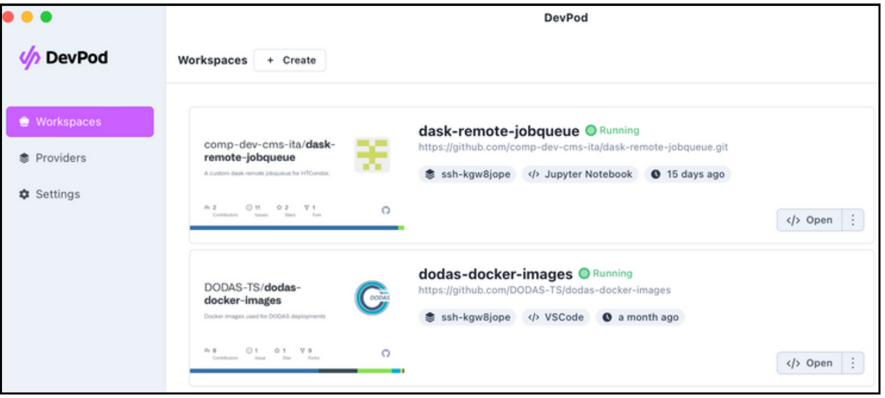
DevContainer

#SOSC23

Share your code as you write it

DevPod is a new user-friendly tool for managing dev environment directly from your code repository!

Based on devcontainer.json allows you to effectively store your development environment for later use and sharing



comp-dev-cms-ita/**dask-remote-jobqueue**

A custom dask remote jobqueue for HTCondor

dask-remote-jobqueue Running

https://github.com/comp-dev-cms-ita/dask-remote-jobqueue.git

ssh-kgw8jope · Jupyter Notebook · 15 days ago

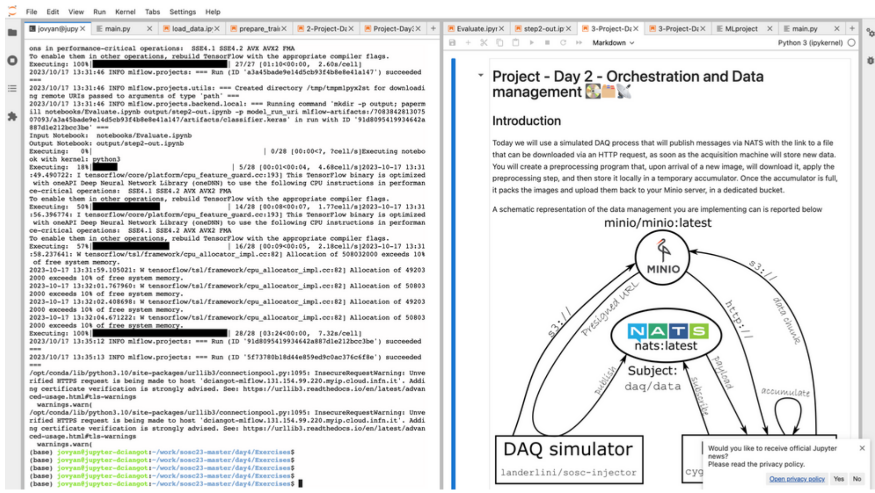
DODAS-TS/**dodas-docker-images**

Docker images used for DODAS deployments

dodas-docker-images Running

https://github.com/DODAS-TS/dodas-docker-images

ssh-kgw8jope · VSCode · a month ago



```
ons in performance-critical operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Executing: 100%|██████████████████████████████| 27/27 [01:10<00:00,  2.60s/cell]
2023/10/17 13:31:46 INFO mlflow.projects: === Run (ID 'a3a45bade9e14d5cb93f4b8e8e41a147') succeeded
===
2023/10/17 13:31:46 INFO mlflow.projects.utils: === Created directory /tmp/tmpmlpyx2st for downloadi
ng remote URIs passed to arguments of type 'path' ===
2023/10/17 13:31:46 INFO mlflow.projects.backend.local: === Running command 'mkdir -p output; paperm
ill notebooks/Evaluate.ipynb output/step2-out.ipynb -p model_run_uri mlflow-artifacts:/708384281307
507093/a3a45bade9e14d5cb93f4b8e8e41a147/artifacts/classifier.keras' in run with ID '91d8095419934642a
887d1e212bcc3be' ===
Input Notebook:  notebooks/Evaluate.ipynb
Output Notebook: output/step2-out.ipynb
Executing:   0%|                                        | 0/28 [00:00<?, ?cell/s]Executing notebo
ok with kernel: python3
Executing:  18%|█████                          | 5/28 [00:01<00:04,  4.68cell/s]2023-10-17 13:31
:49.490722: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized
 with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performan
ce-critical operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Executing:  50%|███████████████               | 14/28 [00:08<00:07,  1.77cell/s]2023-10-17 13:31
:56.396774: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized
 with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performan
ce-critical operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Executing:  57%|█████████████████             | 16/28 [00:09<00:05,  2.18cell/s]2023-10-17 13:31
:58.237641: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 508032000 exceeds 10%
 of free system memory.
2023-10-17 13:31:59.105021: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 49203
2000 exceeds 10% of free system memory.
2023-10-17 13:32:01.767960: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 50803
2000 exceeds 10% of free system memory.
2023-10-17 13:32:02.408698: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 49203
2000 exceeds 10% of free system memory.
2023-10-17 13:32:04.671222: W tensorflow/tsl/framework/cpu_allocator_impl.cc:82] Allocation of 50803
2000 exceeds 10% of free system memory.
Executing: 100%|██████████████████████████████| 28/28 [03:24<00:00,  7.32s/cell]
2023/10/17 13:35:12 INFO mlflow.projects: === Run (ID '91d8095419934642a887d1e212bcc3be') succeeded
===
2023/10/17 13:35:13 INFO mlflow.projects: === Run (ID '5f73780b18d44e859ed9c0ac376c6f8e') succeeded
===
/opt/conda/lib/python3.10/site-packages/urllib3/connectionpool.py:1095: InsecureRequestWarning: Unve
rified HTTPS request is being made to host 'dciangot-mlflow.131.154.99.220.myip.cloud.infn.it'. Addi
ng certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advan
ced-usage.html#tls-warnings
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/urllib3/connectionpool.py:1095: InsecureRequestWarning: Unve
rified HTTPS request is being made to host 'dciangot-mlflow.131.154.99.220.myip.cloud.infn.it'. Addi
ng certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advan
ced-usage.html#tls-warnings
  warnings.warn(
(base) jovyan@jupyter-dciangot:~/work/sosc23-master/day4/Exercises$
(base) jovyan@jupyter-dciangot:~/work/sosc23-master/day4/Exercises$
(base) jovyan@jupyter-dciangot:~/work/sosc23-master/day4/Exercises$
(base) jovyan@jupyter-dciangot:~/work/sosc23-master/day4/Exercises$
(base) jovyan@jupyter-dciangot:~/work/sosc23-master/day4/Exercises$
```

Project - Day 2 - Orchestration and Data management 📀🗂️📡

Introduction

Today we will use a simulated DAQ process that will publish messages via NATS with the link to a file that can be downloaded via an HTTP request, as soon as the acquisition machine will store new data. You will create a preprocessing program that, upon arrival of a new image, will download it, apply the preprocessing step, and then store it locally in a temporary accumulator. Once the accumulator is full, it packs the images and upload them back to your Minio server, in a dedicated bucket.

A schematic representation of the data management you are implementing can is reported below

Would you like to receive official Jupyter news?
Please read the privacy policy.

Suggested reference course

https://youtu.be/pg19Z8LL06w?si=W5QXikaYfqkFYjIf

#SOSC23

CONTAINER ORCHESTRATION

DIEGO CIANGOTTINI - INFN PERUGIA

imdciangot diego-ciangottini-8a03b98a

ciangottini@infn.it

INFN

What if you need multiple containers?

What if your application requires multiple containers executed at once?

Surely, multiple “docker run“ commands can be used:

```
docker container run -d -p 3000:80 -v host_path1:container_path1 -e ENV_VAR=VAL1 container_image1
docker container run -d -p 3001:80 -v host_path2:container_path2 -e ENV_VAR=VAL2 container_image2
docker container run -d -p 3002:80 -v host_path3:container_path3 -e ENV_VAR=VAL3 container_image3
docker container run -d -p 3003:80 -v host_path4:container_path4 -e ENV_VAR=VAL4 container_image4
```

A script could be even written to implement “start”, “restart”, “stop” commands.

What if we update only a single container and we want to restart only that one?

Docker compose

#SOSC23

Docker Compose is a tool for defining and running multi-container Docker application

- YAML file to configure your app's services
- With a single command, you create and start all the services of your application

```
docker compose up         #initialize and start the application environment
docker compose start      #start the app environment
docker compose restart    #restart the created app environment
docker compose stop       #stop the app environment
docker compose down       #stop and destroy the app environment
```

UX in a nutshell

Docker Compose is a tool for defining and running multi-container Docker application

```
user@vm:~/prova_1$ cat docker-compose.yaml
version: "3.8"
services:
  db:
    image: influxdb:2.1
    volumes:
    - db_data:/var/lib/influxdb2.1:rw
    ports:
    - "8086:8086"
    networks:
    - db_net

volumes:
  db_data:

networks:
  db_net:
```

```
user@vm:~/prova_1$ docker compose up -d
✔ db 10 layers [⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿]        0B/0B   Pulled 39.4s
[+] Running 3/3
✔ Network prova_1_db_net      Created
✔ Volume "prova_1_db_data"    Created
✔ Container prova_1-db-1      Started

user@vm:~/prova_1$ docker container ps
CONTAINERID   IMAGE            NAMES
Be72e5..      influxdb:2.1     prova_1_db_1

user@vm:~/prova_1$ docker volume ls
DRIVER  VOLUME NAME
local   46f4a..
local   prova_1_db_data

user@vm:~/prova_1$ docker network ls
NETWORK ID     NAME              DRIVER       SCOPE
51039c996454   bridge            bridge       local
85ffa41af96f   host              host         local
d6f329122060   none              null         local
ec31eefd60d5   prova_1_db_net    bridge       local
```

Docker compose network

- By default Compose sets up a single network for your app.
- Each container for a service joins the default network and is both **reachable** by other containers on that network and **discoverable** by them at a hostname identical to the container name.
- If you make a configuration change to a service and run `docker compose up` to update it, the old container is removed and the new one joins the network under a **different IP** address but the **same name**.
- Use **container names** and port to connect services
- You can specify your own networks with the top-level networks key.

```
networks:
  front:
    driver: bridge
    driver_opts:
      com.docker.network.enable_ipv6: "true"
    ipam:
      driver: default
      config:
        - subnet: 172.16.238.0/24
        gateway: 172.16.238.1
        - subnet: "2001:3984:3989::/64"
        gateway: "2001:3984:3989::1"
```

Self-healing

#SOSC23

Health Checks

```
docker volume create influxdb-storage
export INFLUXDB_USERNAME=admin
export INFLUXDB_PW=admin
docker container run --name influxdb
           -p 8086:8086 \
           -v influxdb-storage:/var/lib/influxdb \
           -e INFLUXDB_DB=db0 \
           -e INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME} \
           -e INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PW} \
           --health-cmd='curl -f http://localhost:8086||exit 1' \
           --health-start-period=30s \
           --health-interval= 30s \
           --health-timeout=10s \
           --health-retries=4 \
            influxdb:2.1
```

```
version: '3.8'
services:
  influxdb:
    image: influxdb:2.1
    ports:
      - '8086:8086'
    volumes:
      - influxdb-storage:/var/lib/influxdb
    environment:
      - INFLUXDB_DB=db0
      - INFLUXDB_ADMIN_USER=${INFLUXDB_USERNAME}
      - INFLUXDB_ADMIN_PASSWORD=${INFLUXDB_PASSWORD}
    healthcheck:
      test: "curl --fail http://localhost:8086 || exit 1"
      start_period: 30s
      interval: 30s
      timeout: 10s
      retries: 4
```

Homeworks

#SOSC23

- Setup a jupyterlab with persistent volume, starting from the image created yesterday

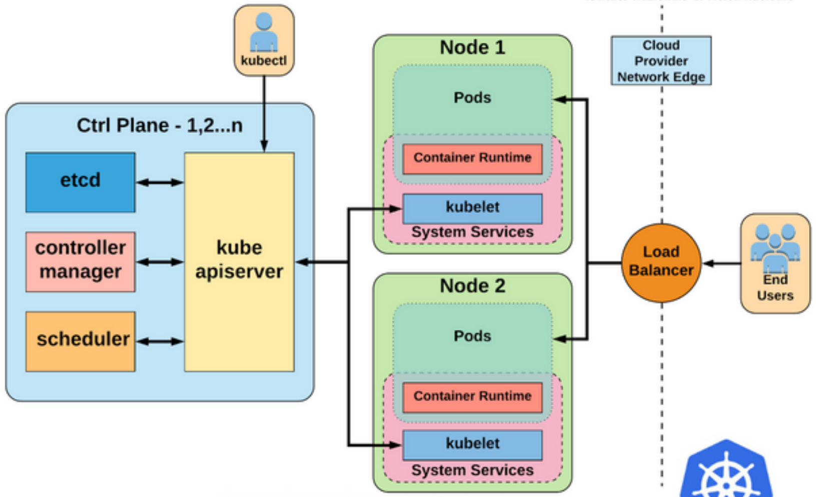
Multiple containers... On multiple machine!

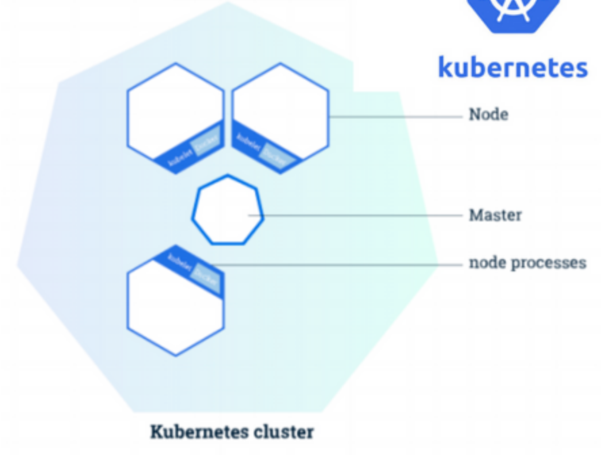
#SOSC23

Kubernetes

Kubernetes is an **open-source platform that coordinates a highly available cluster of computers that are connected to work as a single unit**. It is backed by Google and RedHat.

- **Applications** need to be **containerized**.
- Kubernetes automates the distribution and scheduling of application containers across a cluster in a fairly efficient way.
- A Kubernetes cluster can be deployed on either physical or virtual machines.





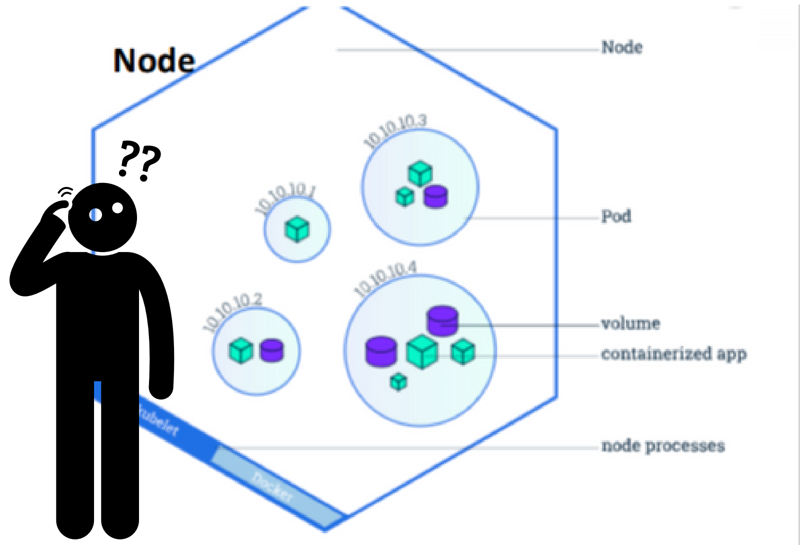
Kubernetes firs citizen: the POD

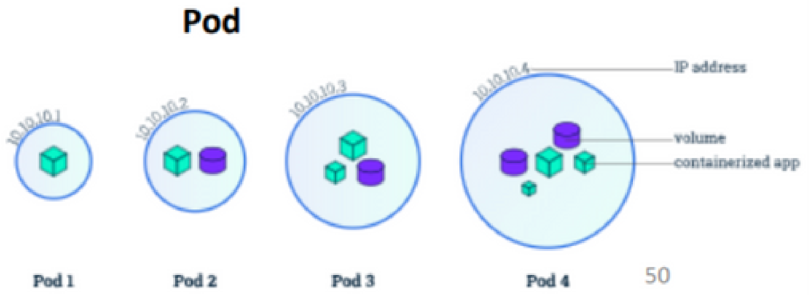
#SOSC23

A docker compose services equivalent

Be aware that NO ports are accessible without the component on the next slide!

- A Pod is the basic building block of Kubernetes. It represents a running process on your cluster.
- A Pod encapsulates an application container, storage resources, a unique network IP, and options that govern how the container(s) should run.
- Pods that run a single container. The “one -container - per -Pod” model is the most common Kubernetes use case; in this case, you can think of a Pod as a wrapper around a single container, and Kubernetes manages the Pods rather than the containers directly.
- Pods that run multiple containers that need to work together. A Pod might encapsulate an application composed of multiple co -located containers that are tightly coupled and need to share resources. The Pod wraps these containers and storage resources together as a single manageable entity.





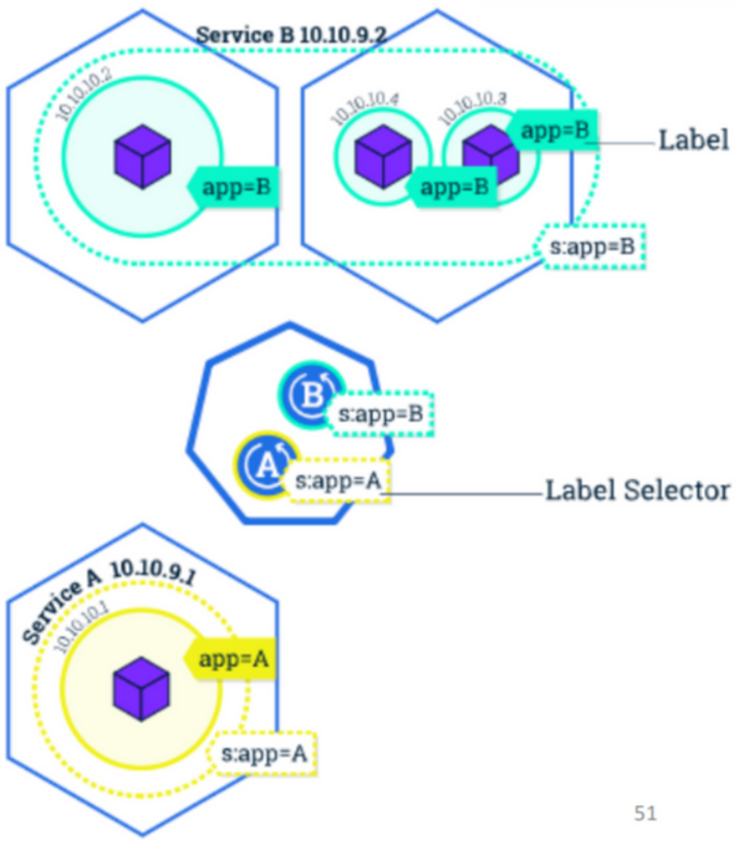
Devil is in the... network!

#SOSC23

Kubernetes service --> expose your application:

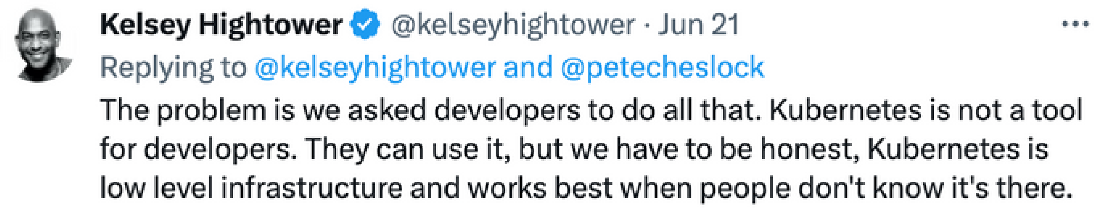
- pod to pod/cluster wide
- world-wide

- A Kubernetes Service is an abstraction which defines a logical set of Pods and a policy by which to access it.
- Although each Pod has a unique IP address, those IPs are not exposed outside the cluster without a Service. Services allow your applications to receive traffic.
- Services match a set of Pods using labels and selectors, a grouping primitive that allows logical operation on objects in Kubernetes.
- Labels are key/value pairs attached to objects and can be used in any number of ways:
  - Designate objects for development, test, and production
  - Embed version tags
  - Classify an object using tags



Why do I need to care

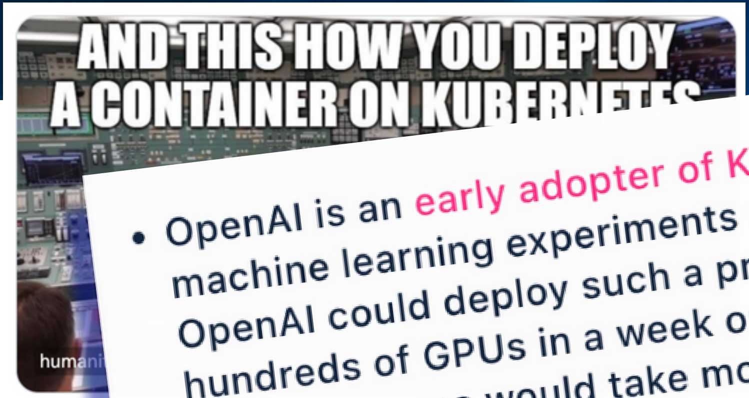
#SOSC23


Kelsey Hightower @kelseyhightower · Jun 21
Replying to @kelseyhightower and @petecheslock
The problem is we asked developers to do all that. Kubernetes is not a tool for developers. They can use it, but we have to be honest, Kubernetes is low level infrastructure and works best when people don't know it's there.

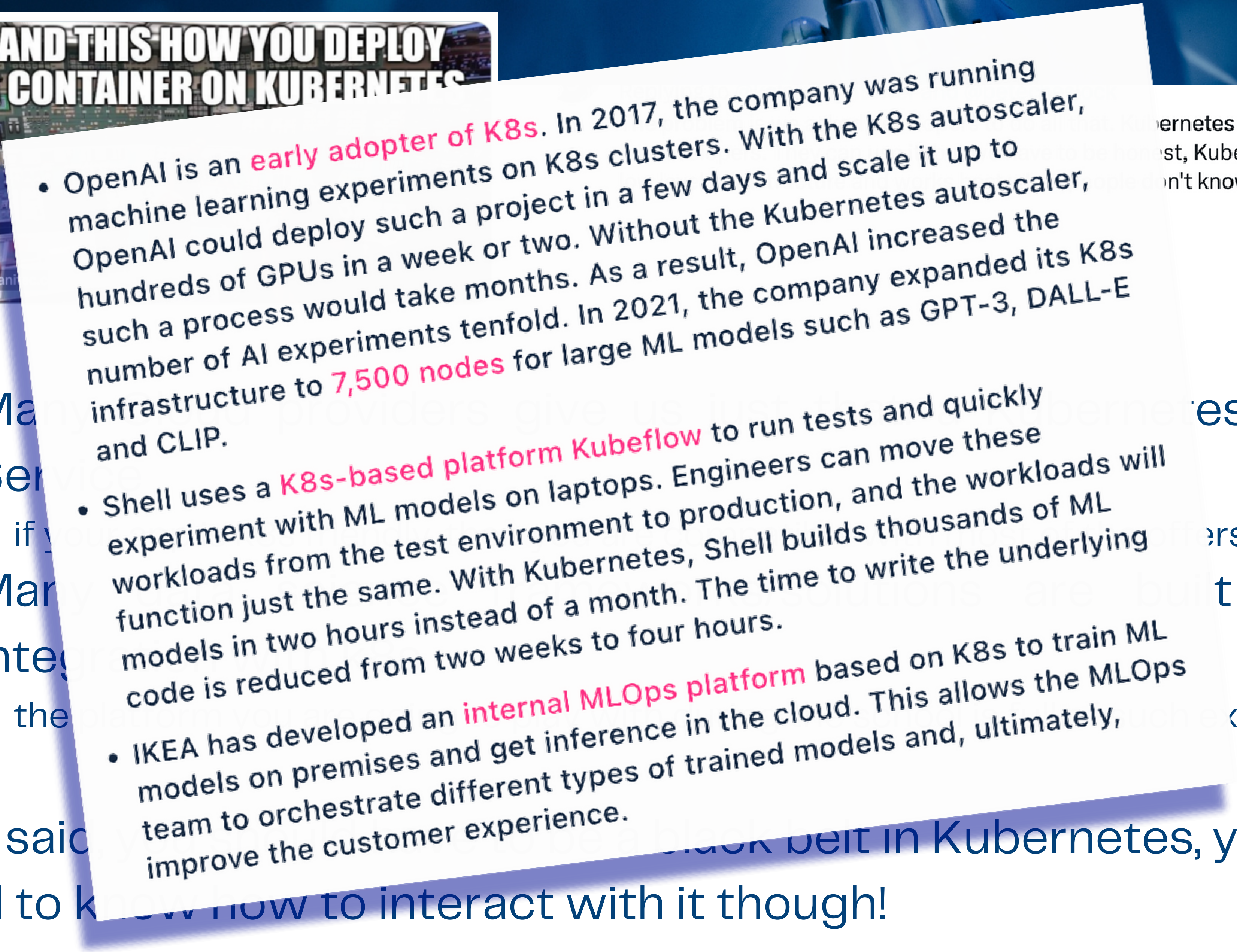
- Many **Cloud providers offers Kubernetes as a Service**
  - if your app is k8s friendly, then you are compatible with most of the offers
- Many data science **frameworks/solutions are built with integration with k8s**
  - the platform you are going to play with during the school is full of such examples

That said, you should have to be a black belt in Kubernetes, you do need to know how to interact with it though!

Why do I need to care



...ernetes is not a tool
...st, Kubernetes is
...n't know it's there.



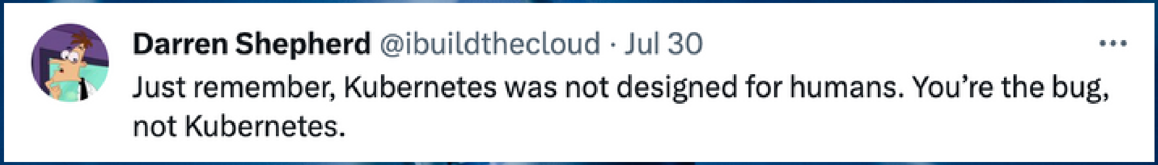
- OpenAI is an early adopter of K8s. In 2017, the company was running machine learning experiments on K8s clusters. With the K8s autoscaler, OpenAI could deploy such a project in a few days and scale it up to hundreds of GPUs in a week or two. Without the Kubernetes autoscaler, such a process would take months. As a result, OpenAI increased the number of AI experiments tenfold. In 2021, the company expanded its K8s infrastructure to 7,500 nodes for large ML models such as GPT-3, DALL-E and CLIP.
- Shell uses a K8s-based platform Kubeflow to run tests and quickly experiment with ML models on laptops. Engineers can move these workloads from the test environment to production, and the workloads will function just the same. With Kubernetes, Shell builds thousands of ML models in two hours instead of a month. The time to write the underlying code is reduced from two weeks to four hours.
- IKEA has developed an internal MLOps platform based on K8s to train ML models on premises and get inference in the cloud. This allows the MLOps team to orchestrate different types of trained models and, ultimately, improve the customer experience.

- Many cloud providers give us just the Kubernetes as a Service
  - if y... ers
- Man... t with integ...
  - the platform you are ... xamples

That said ... black belt in Kubernetes, you do need to know how to interact with it though!

#SOSC23

Homeworks


Darren Shepherd @ibuildthecloud · Jul 30

Just remember, Kubernetes was not designed for humans. You're the bug, not Kubernetes.

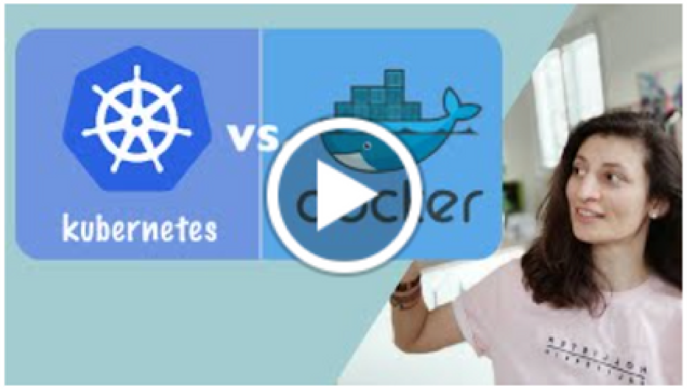
Only one way to learn --> try it with a dummy project

Just an idea

- Setup a "school platform"-like on your laptop

You can start from installing Kubernetes via KinD (Kubernetes in docker) and then deploying a jupyterLab pod on it.

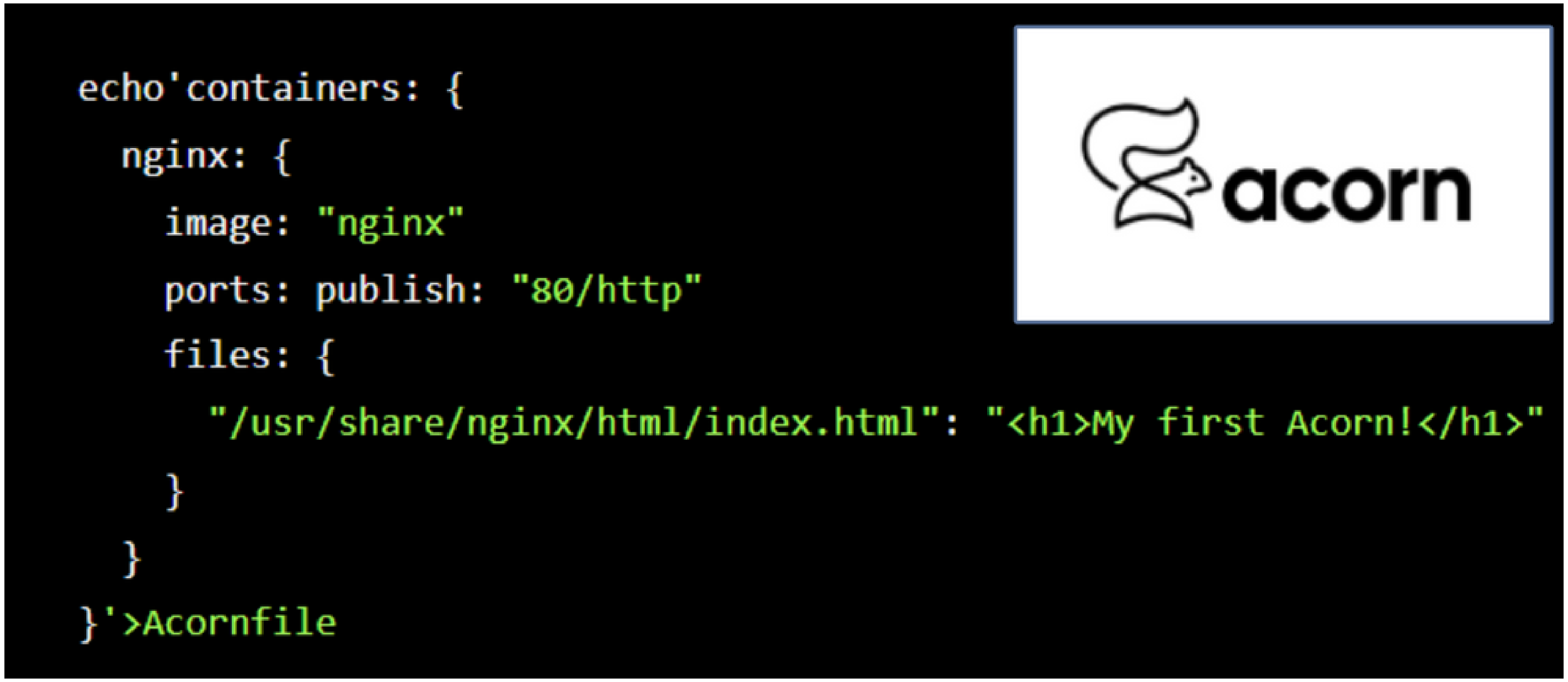
https://www.youtube.com/watch?v=9_s3h_GVzZc



News from the world: acorn.io

Can I deploy my app on a K8s as a Service instance in a way that is as easy as running a docker image??

Acorn is a new solution proposed to make building and deploying app in kubernetes as simple as doing it locally via Docker.



```
echo'containers: {
  nginx: {
    image: "nginx"
    ports: publish: "80/http"
    files: {
      "/usr/share/nginx/html/index.html": "<h1>My first Acorn!</h1>"
    }
  }
}'>Acornfile
```