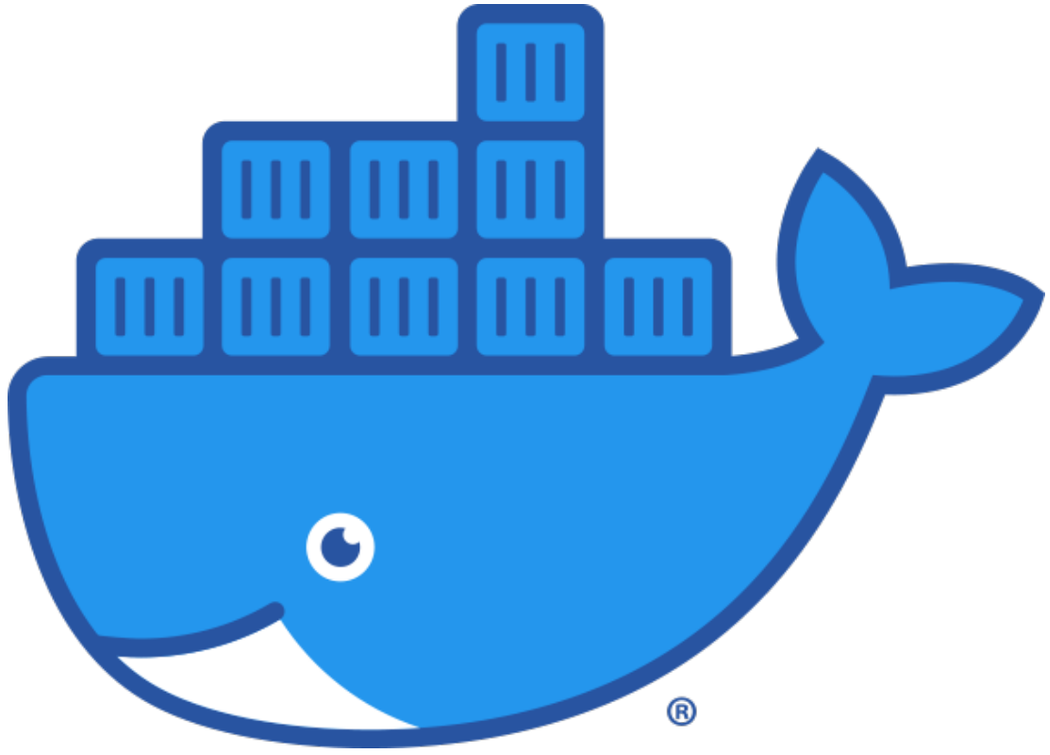
Hands-on Dockers for networking

Pietro Manzoni

Universitat Politecnica de Valencia (UPV)

Valencia - SPAIN

pmanzoni@disca.upv.es



About...

GRC GRUPO DE REDES DE COMPUTADORES

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Home | Members | Papers | Projects | Teaching | Software

The **Networking Research Group** (GRC - *Grupo de Redes de Computadores*) of the Universitat Politècnica de València (UPV) was founded in 2000 and it is mainly composed of researchers of the Computer Engineering Department (DISCA). It keeps strong bonds and collaborations with other researchers in the same area in Spain and in the rest of the world.

The group research efforts are focused on offering **Data Communication Solutions for Mobile Systems**. The main areas of application are:

- AIoT infrastructures for environmental sustainability
- Drone-based networks
- Efficient IoT infrastructures development
- Intelligent Transport Systems
- LPWAN-based networks
- Mobile edge computing
- Pub/Sub systems
- Social sensing

Infos and News:

- Overview of GRC research [Feb. 2020]
- GRC YouTube channel
- **COVIDsensing**: a tool to analize COVID spreading using AI
- Geotab University R&D Program.
- A milestone in MANETs history: the First Working Mobile Ad Hoc Network in 1999

Events and CfPs:

Conferences:

- **GoodIT**, International Conference on Information Technology for Social Good, September 9-11, 2021, Rome, Italy.
- **IEEE/ACM DS-RT 2021**, The 25th International Symposium on Distributed Simulation and Real Time Applications, September 27-29, 2021, Valencia, Spain.

Journals Special Issues:

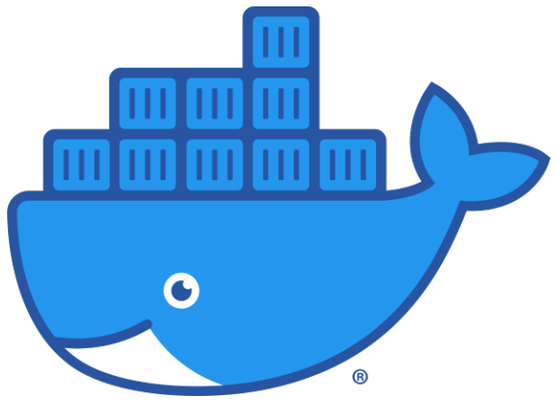
- Sensors, MDPI AG. Special Issue on "**AI for IoT**". Manuscript Submission Deadline: 31 March 2021.

http://www.grc.upv.es/

Course outline

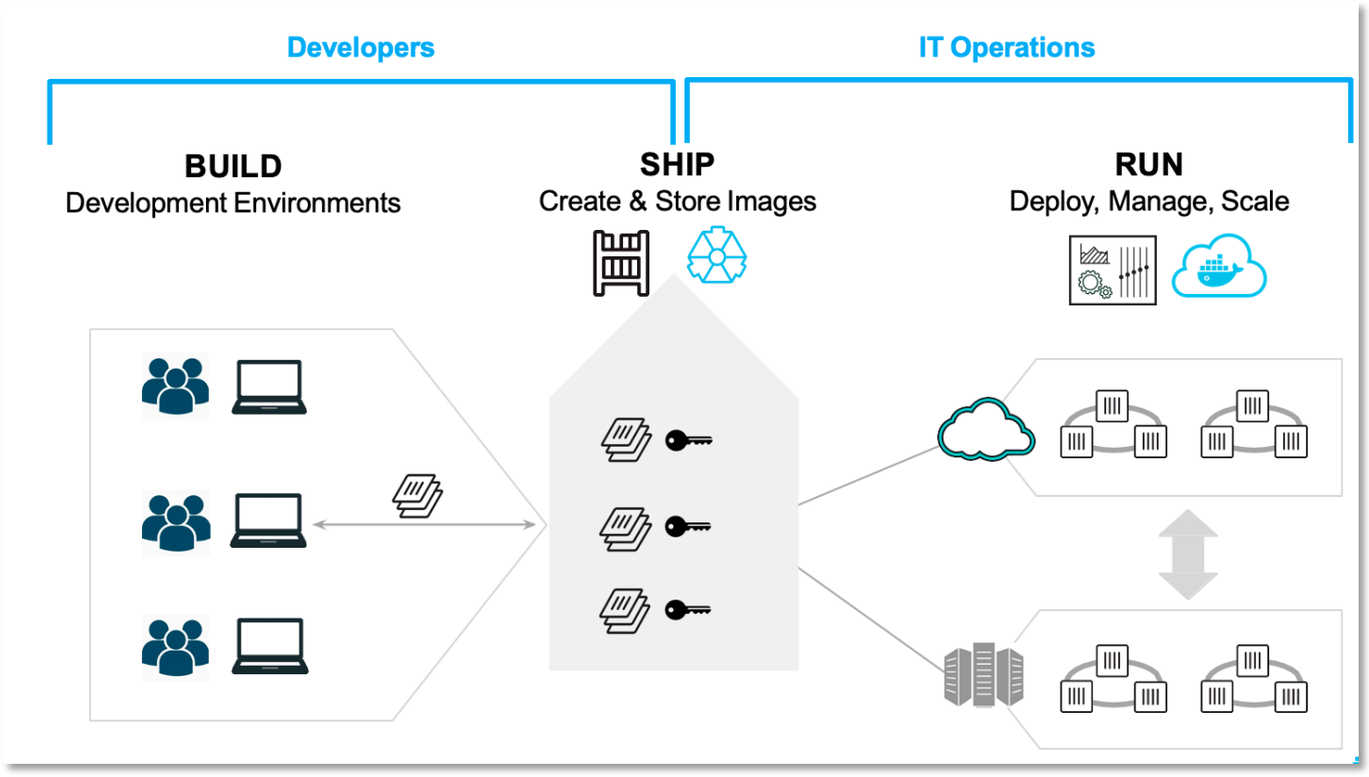
- This course will focus on the use of containers for the modelling and evaluation of applications, with a strong focus on networking.
- The course is organized as a succession of theoretical and experimental (hands-on) parts.
- A container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another.
- A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.
- As a final case-study of a networking application, the course will present OpenThread. OpenThread, released by Google, is an open-source implementation of the Thread networking protocol.

https://www.docker.com/get-started



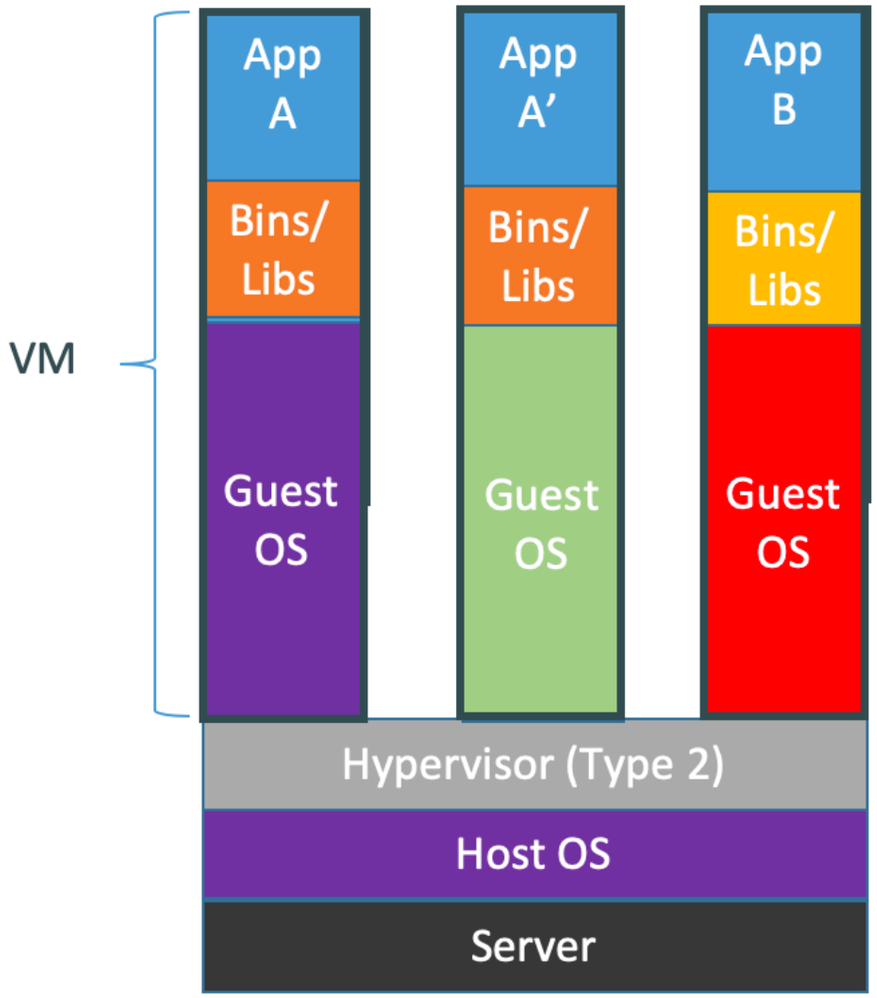
Docker overview

- Docker is an open platform for developing, shipping, and running applications.



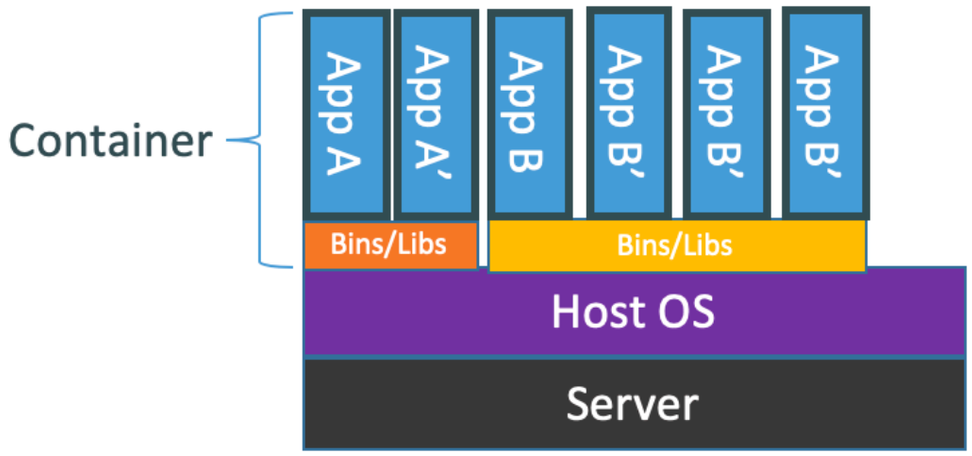
- Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
- https://docs.docker.com/get-started/overview/

Docker vs. Virtual Machine



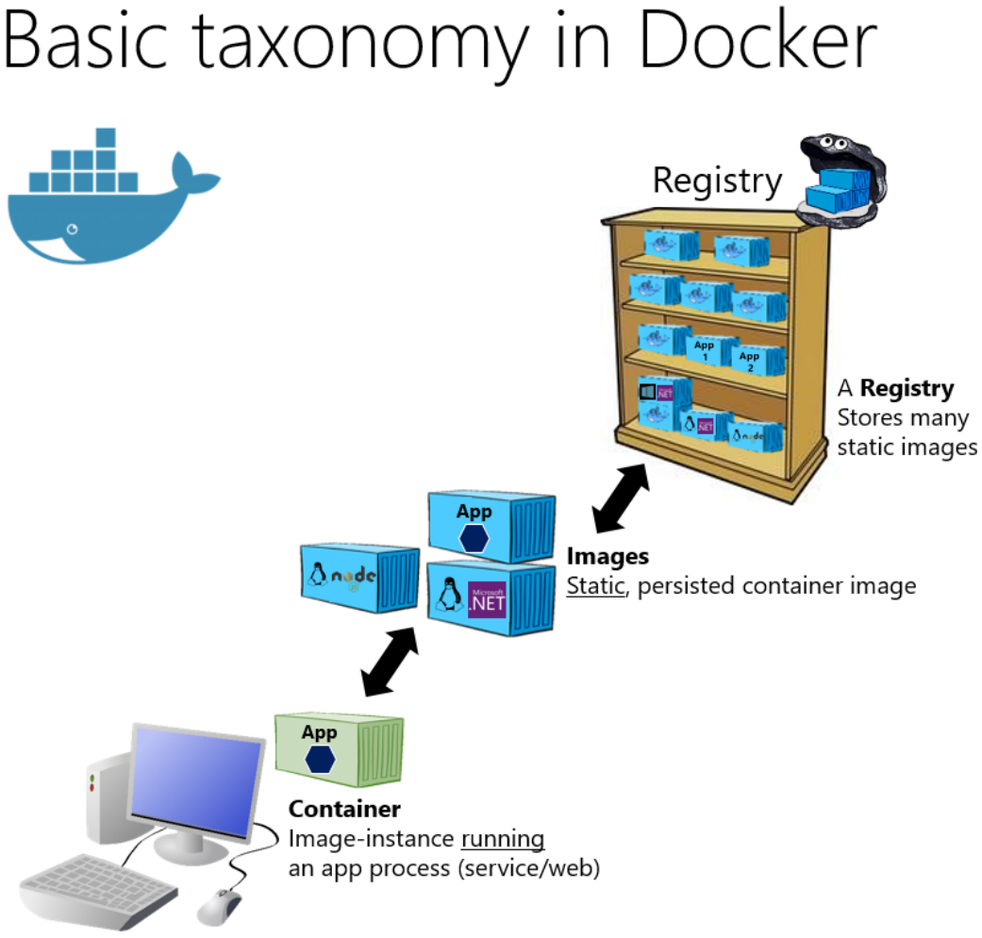
Containers are isolated, but share OS kernel and, where appropriate, bins/libraries

…result is significantly faster deployment, much less overhead, easier migration, faster restart

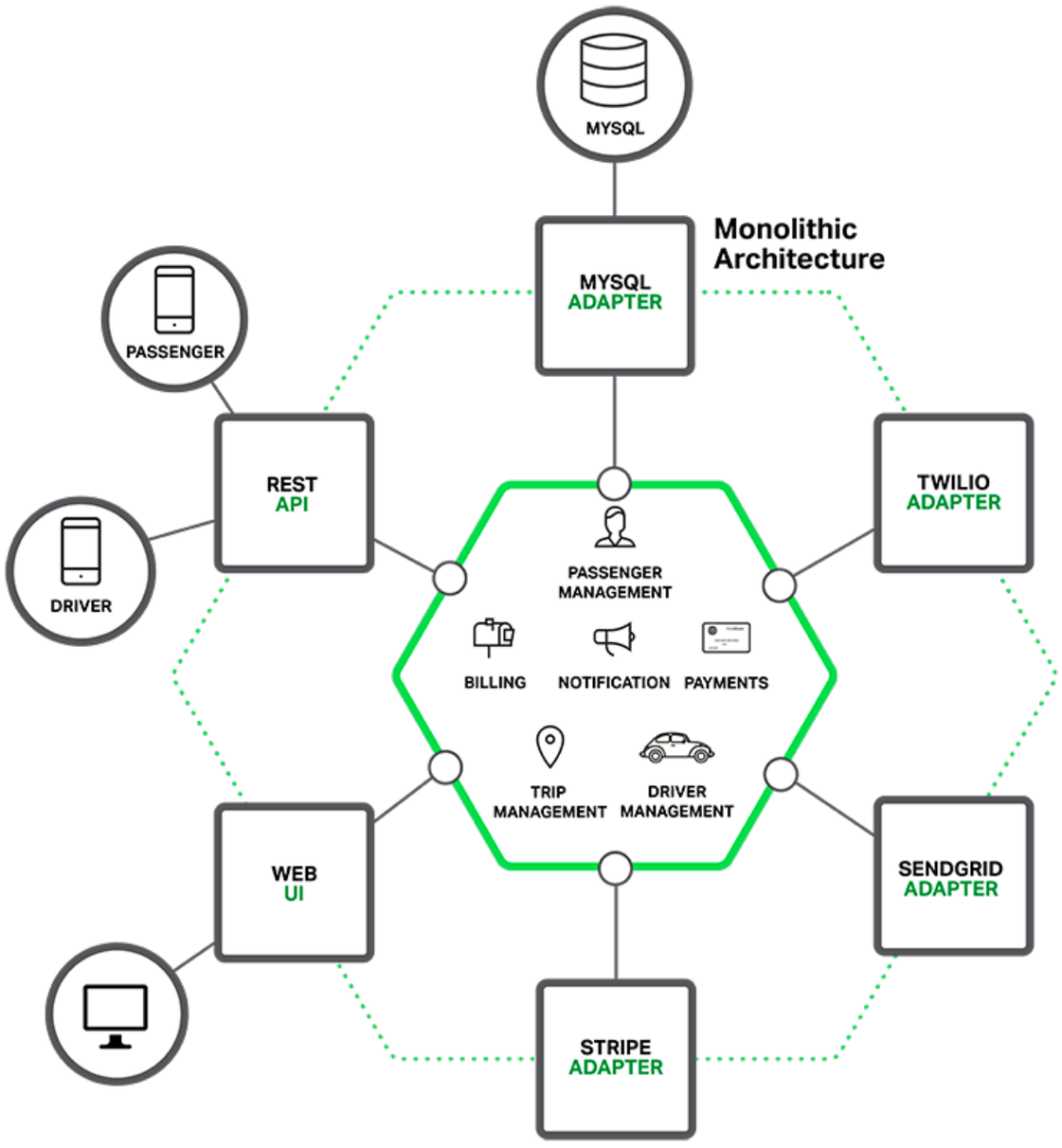


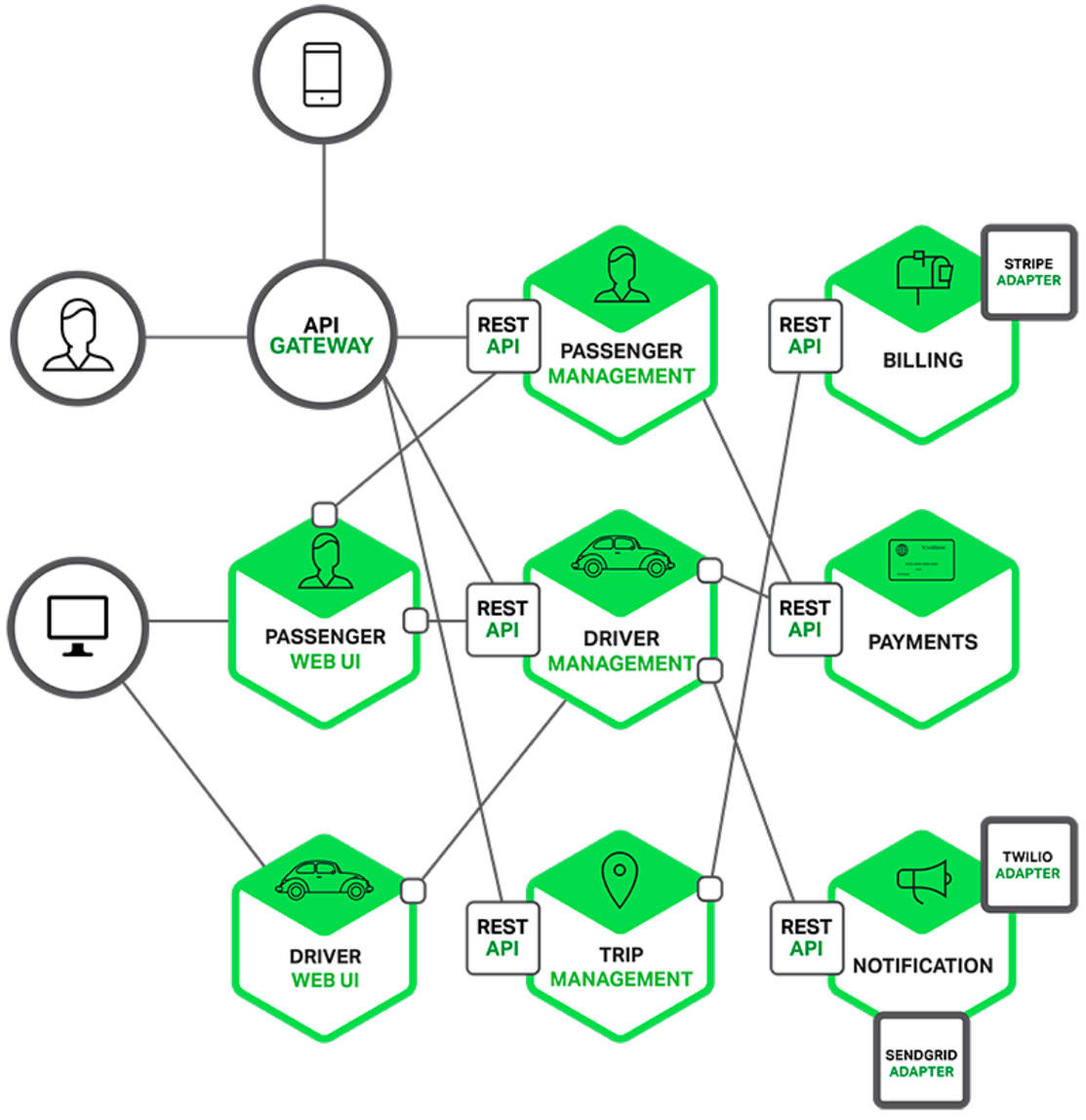
Deployment and scaling

- Docker containers can run on a developer's local laptop, on physical or virtual machines in a data center, on cloud providers, or in a mixture of environments.
- Docker's portability and lightweight nature also make it easy to **dynamically manage workloads**, scaling up or tearing down applications and services as business needs dictate, in near real time.

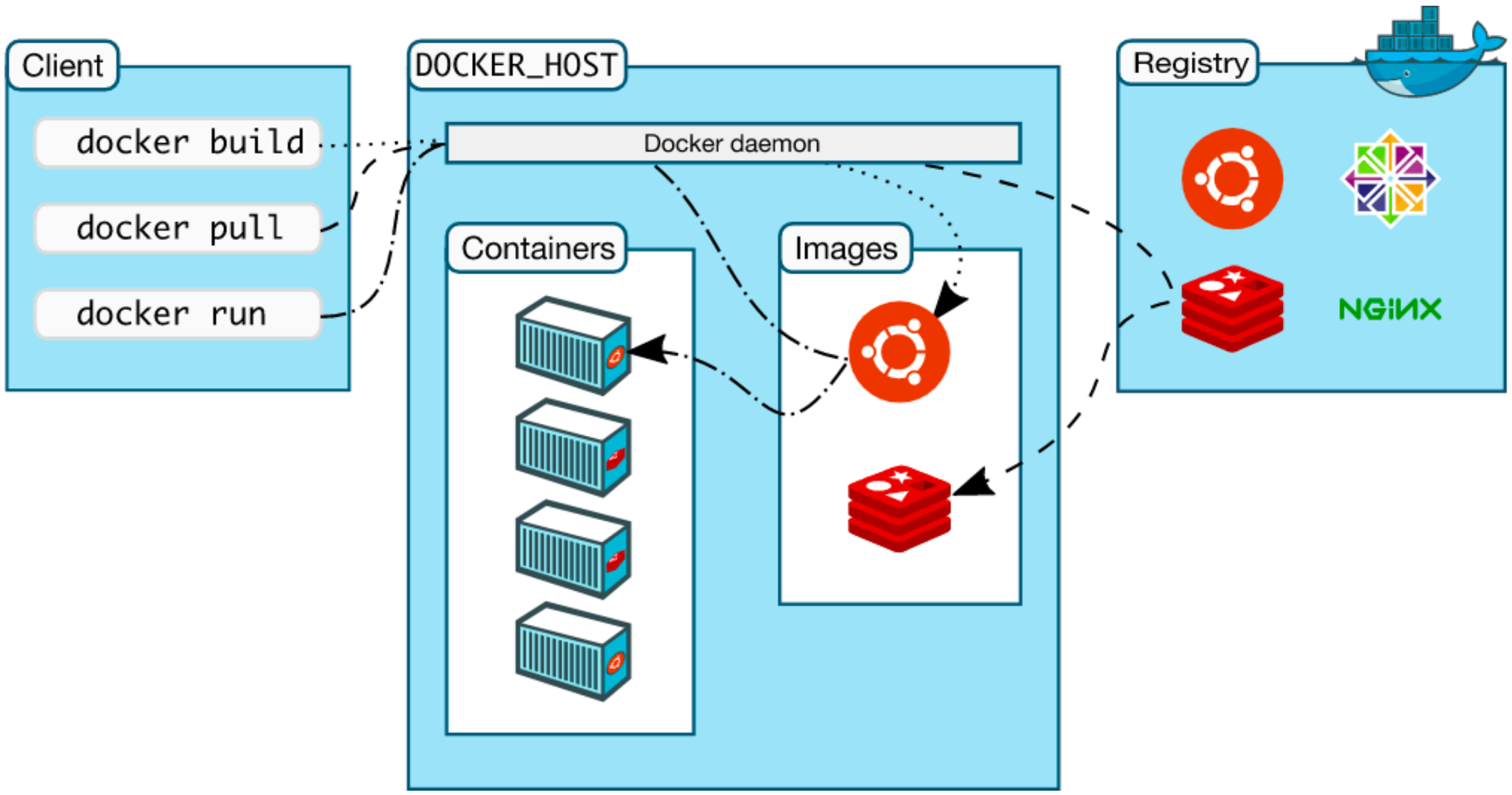


Monolithic vs. Microservices



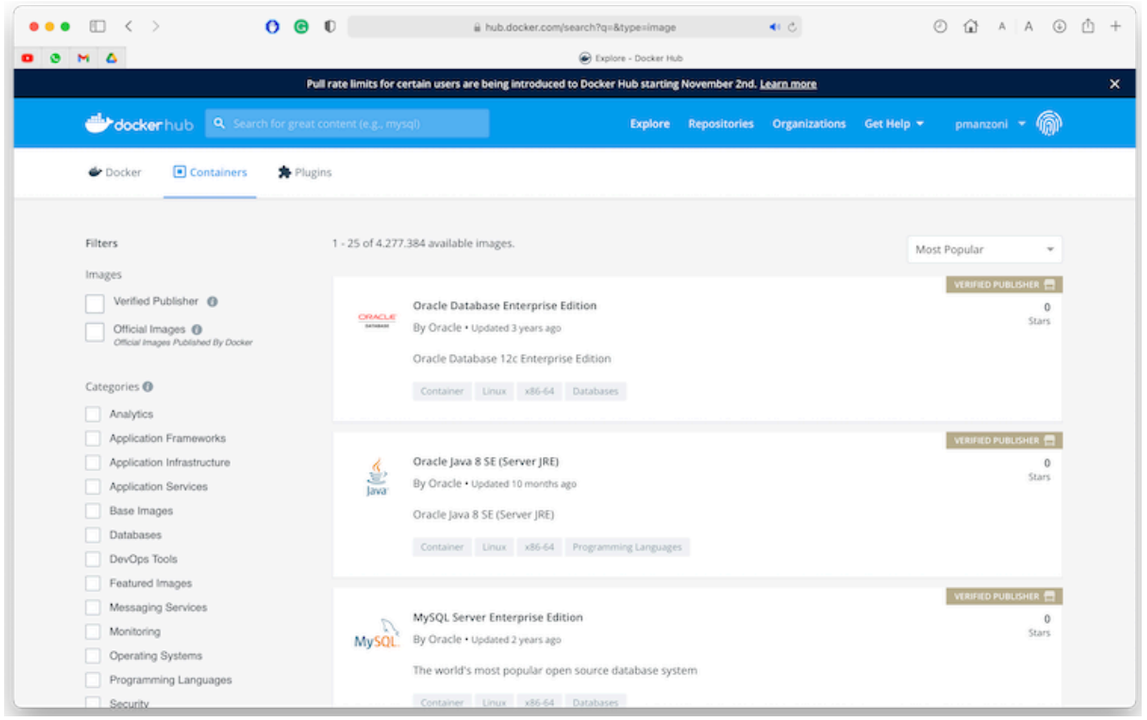


Docker architecture



Docker registries

- A Docker registry stores Docker images. Docker Hub (https://hub.docker.com/) is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default.
  - You can run your own private registry, too
  - E.g., https://docs.github.com/en/packages/guides/about-github-container-registry



Images

- An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization.
  - For example, you may build an image which is based on the ubuntu image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run.
- You might create your own images or you might only use those created by others and published in a registry.
- To build your own image, you create a Dockerfile with a simple syntax for defining the steps needed to create the image and run it.
  - Each instruction in a Dockerfile creates a **layer** in the image.
  - When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt.

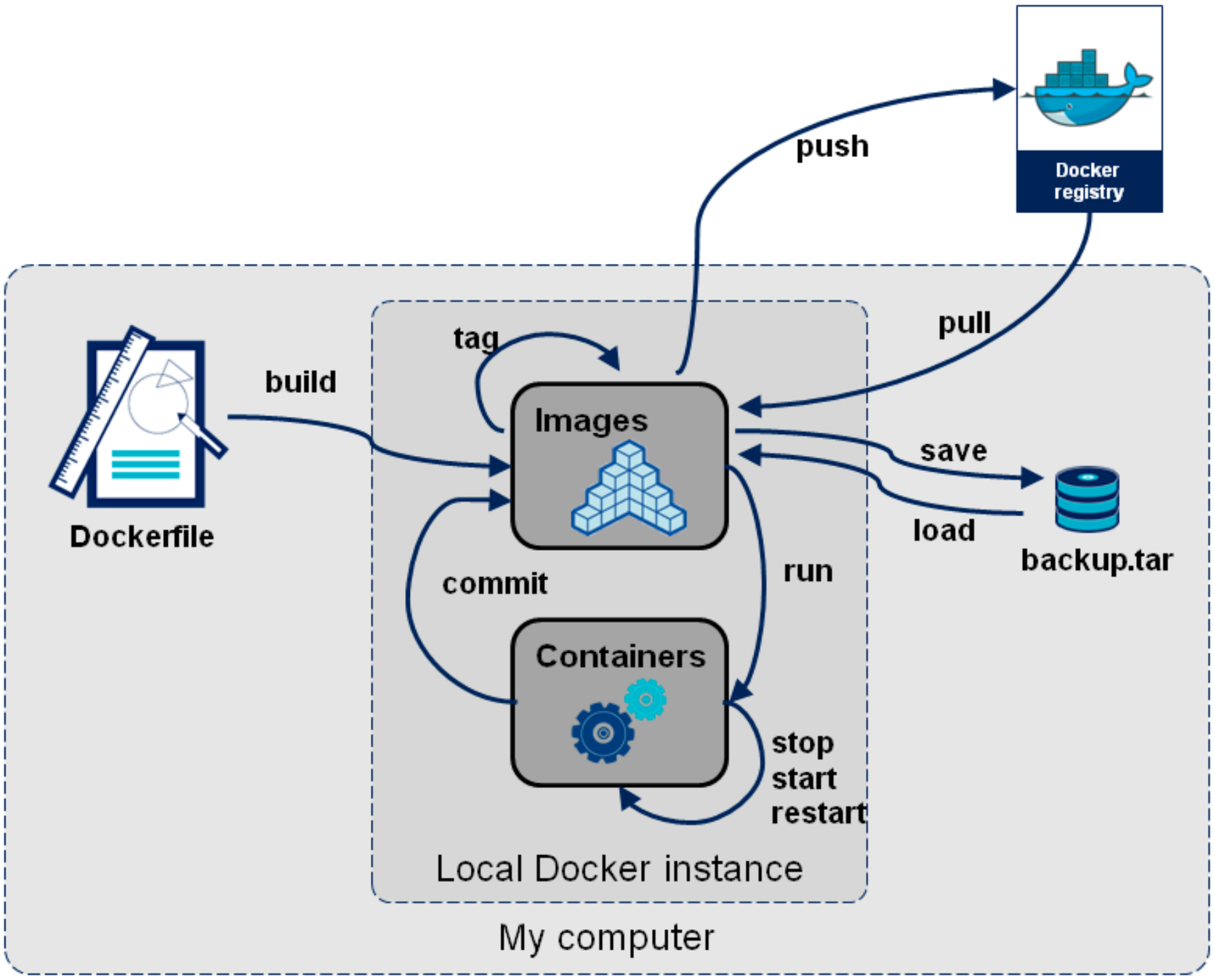
Dockerfile example

```
FROM alpine:3.5

RUN apk add --update py2-pip

RUN pip install paho-mqtt

COPY sisub.py /home/

CMD ["python", "/home/sisub.py"]
```

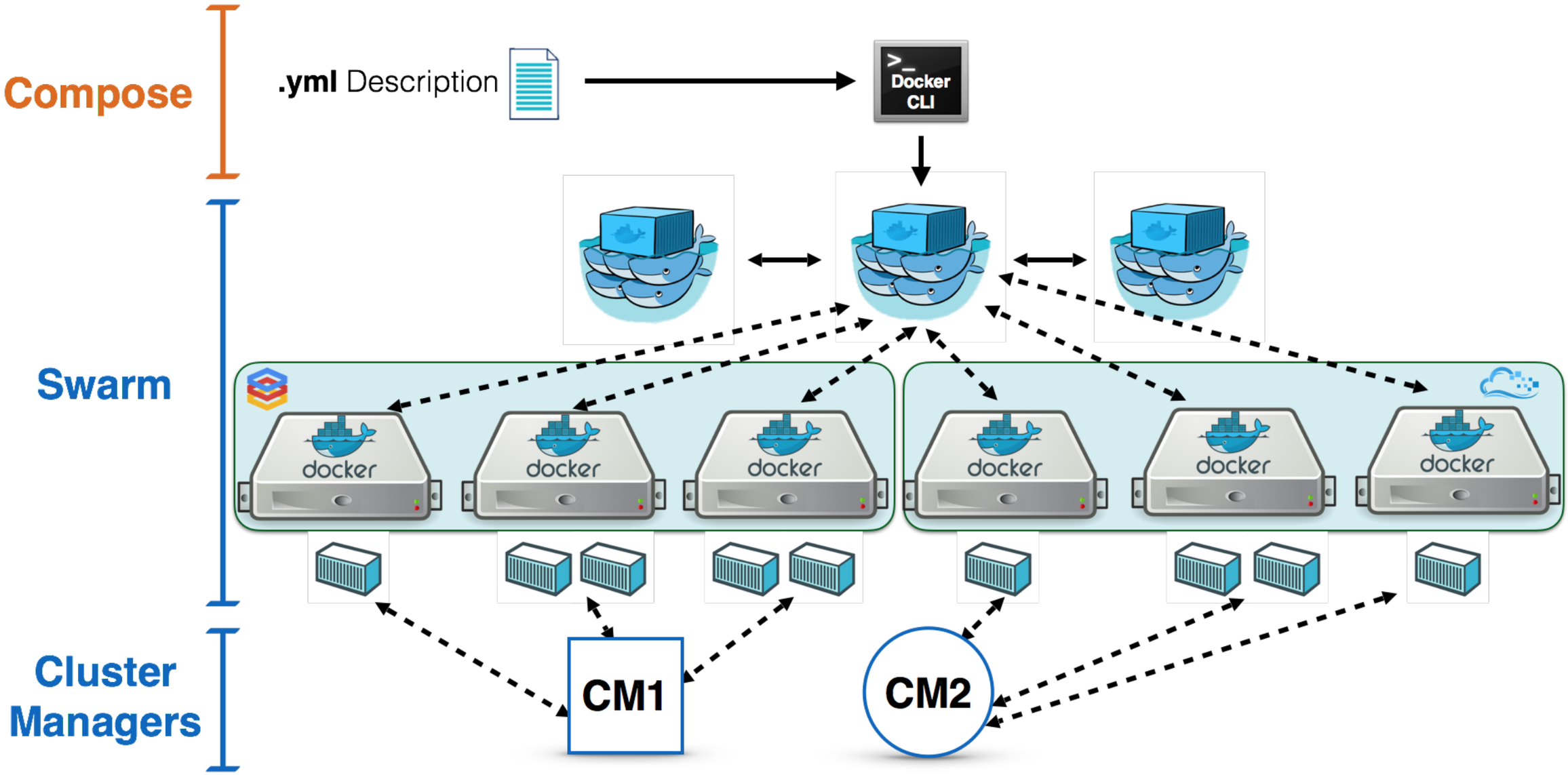
Containers

- A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.
- By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine.
- A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that are not stored in persistent storage disappear.

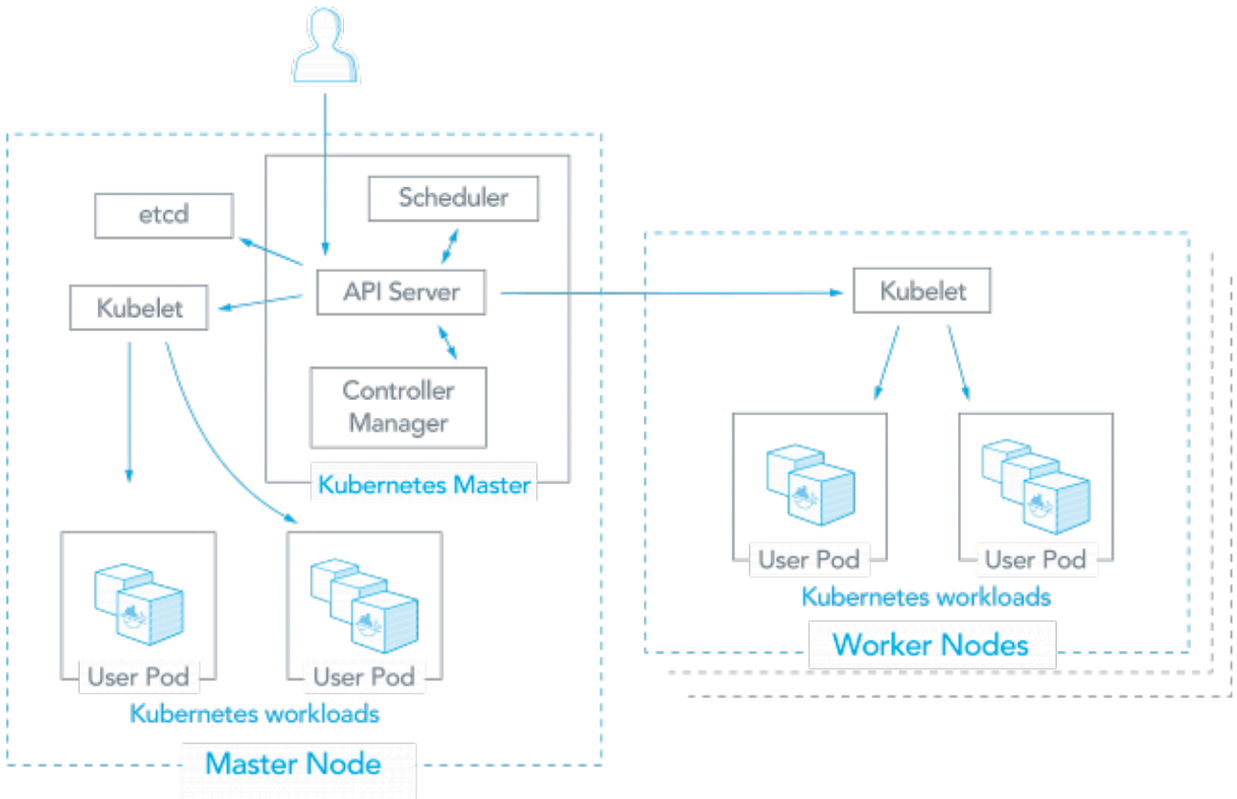
Manage the lifecycle of your containers



Docker Swarm



Kubernetes



API Server: management hub for Kubernetes
Scheduler: places a workload on the appropriate Node
Controller Manager: scales workloads up/down
etcd: stores configuration data which can be accessed by API Server

Kubelet: Receives pod specifications from API Server, updates Nodes
Master Node: places workloads on Nodes
Worker Nodes: receives requests from Master Nodes and dispatches them
User Pod: a group of containers with shared resources

- Developed by Google and introduced in the year 2014.
- An IT management tool that has been specifically designed to simplify the scalability of workloads using containers.
- It has the ability to automate deployment, scaling, and operating application containers.