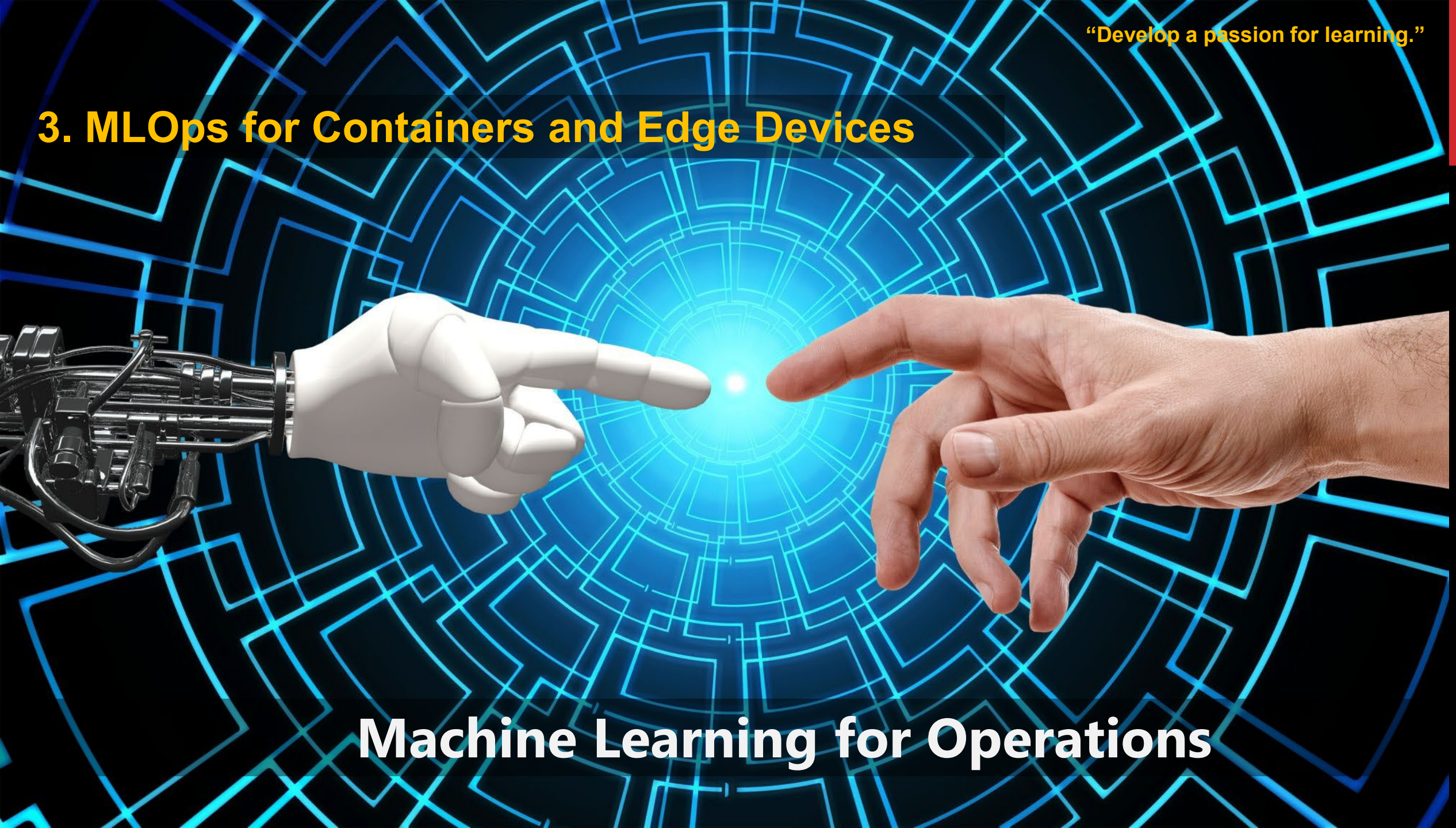
“Develop a passion for learning.”

3. MLOps for Containers and Edge Devices



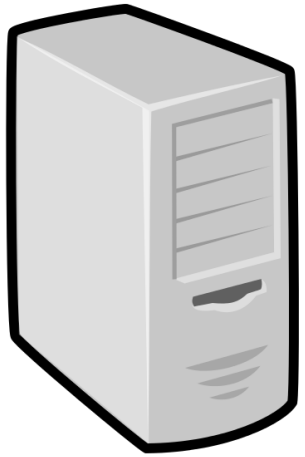
Machine Learning for Operations

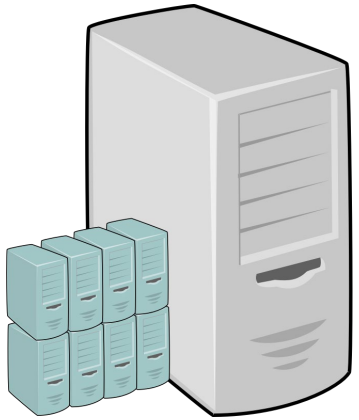
Module Topics

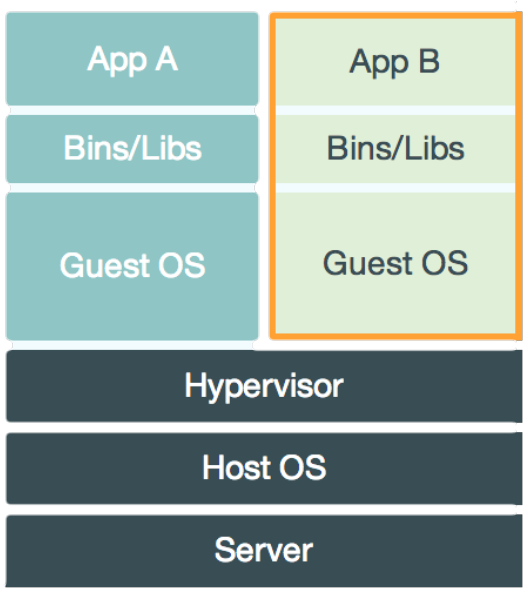
- Containers
- Serving a trained model over HTTP
- Edge Devices
- Coral
- AWS edge computing
- Tensor Flow Hub

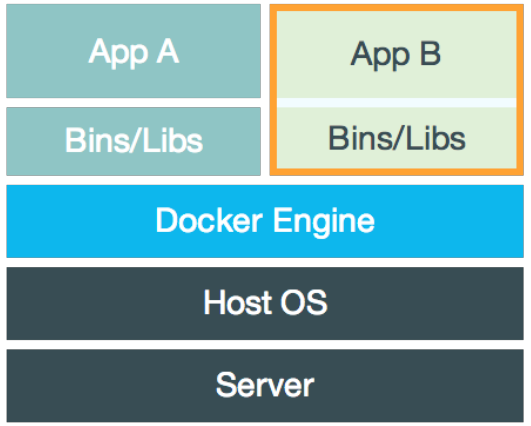
Containers

Evolution of the Data Center



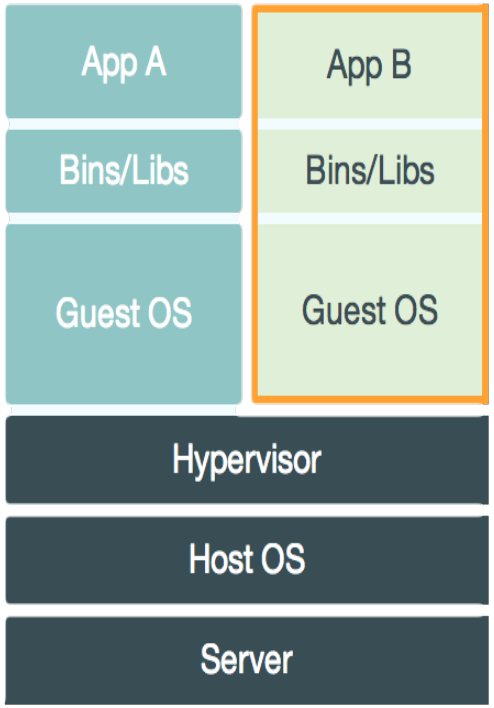






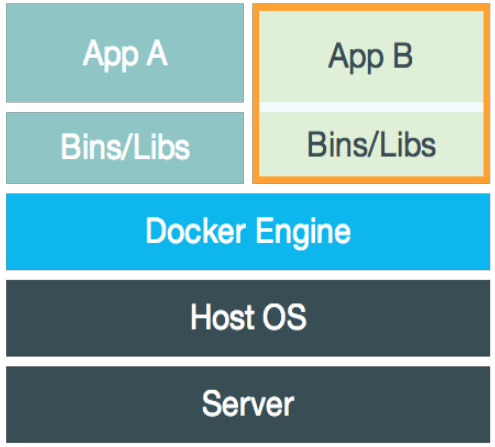
Virtual Machines

- Works in layers:
  - Hypervisor : that allows virtualization
  - Virtual hardware: simulates hardware
  - Guest OS: Linux, Windows etc.
- Performance penalty for keeping a full copy of the OS
  - Does provide complete isolation for each VM
  - Can mix OSes on the same physical host



Container

- Virtualize the OS
  - Like how hypervisors virtualizes hardware
  - Shares the kernel with the Host OS
- Lightweight package that runs as a process on the host
  - Self-contained
  - Has everything needed to run
- Docker can run Linux containers on Windows
  - Uses a minimal Linux VM (LinuxKit)
  - Now uses WLS (Windows Linux Subsystem)

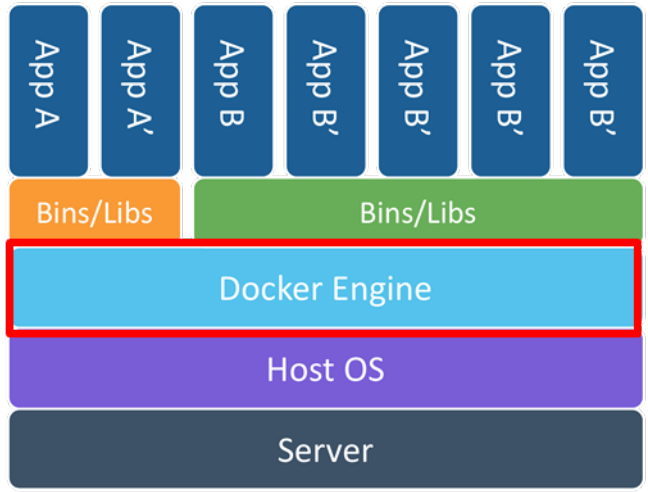


Docker

- Docker was introduced in 2015
- Built on Linux containers
- But had an ecosystem that allowed easy use and deployment
- Has become the de facto container standard

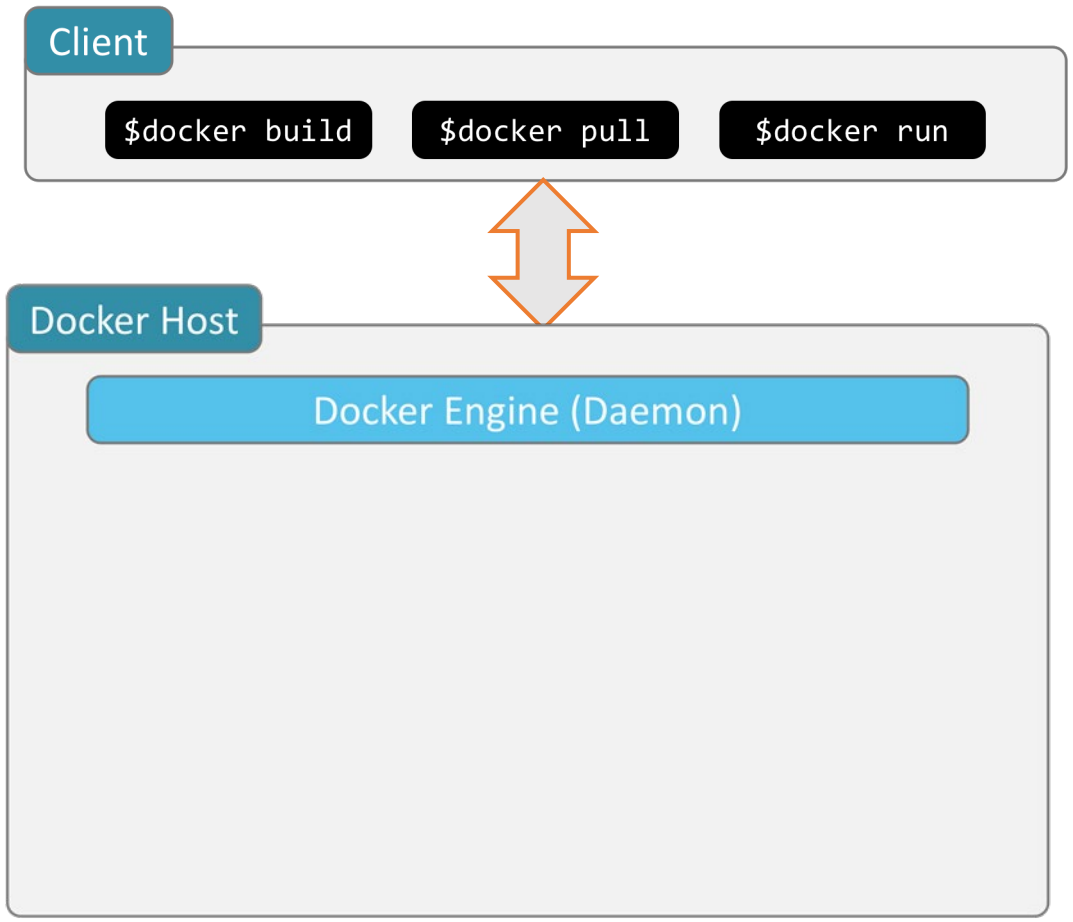
Containers: Docker Engine

- Lightweight runtime program to build, ship, and run Docker containers
  - Also known as Docker Daemon
  - Uses Linux Kernel namespaces and control groups
  - Linux Kernel (>= 3.10)
  - Namespaces provide an isolated workspace



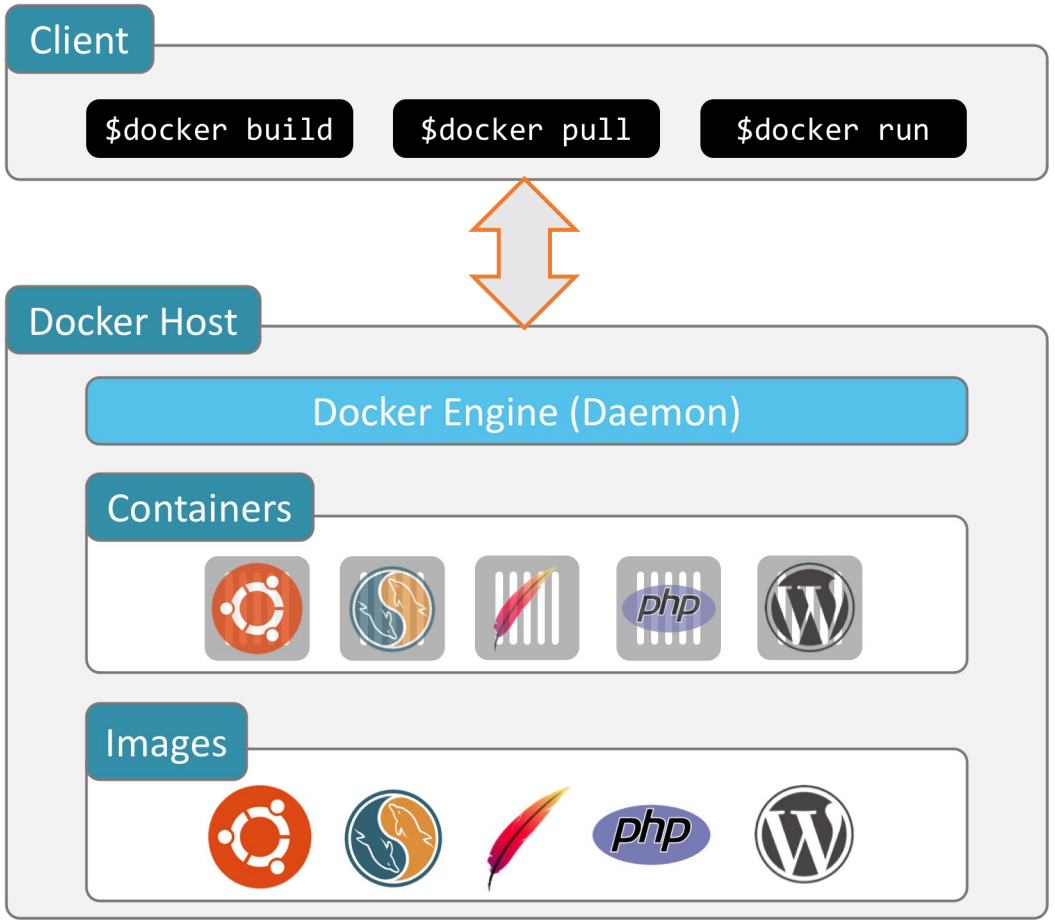
Docker Client and Engine

- The Docker Client is the docker binary
  - Primary interface to the Docker Host
  - Accepts commands and communicates with the Docker Engine (Daemon)



Containers: Docker Client

- Lives on a Docker host
- Creates and manages containers on the host

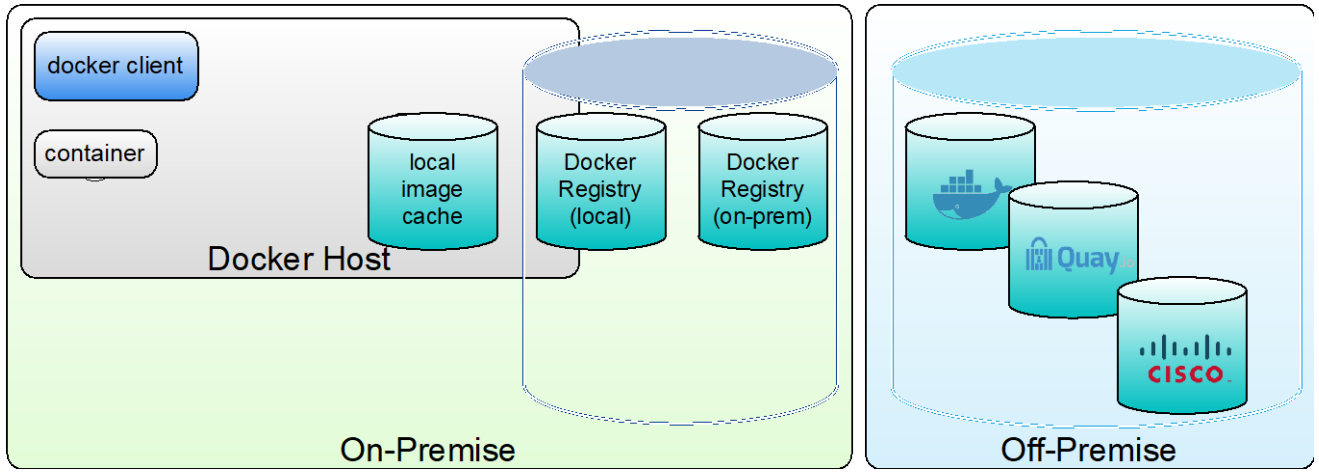


Docker Registries

- Image Storage & Retrieval System
- Docker Engine Pushes Images to a Registry
- Version Controlled
- Docker Engine Pulls Images to Run

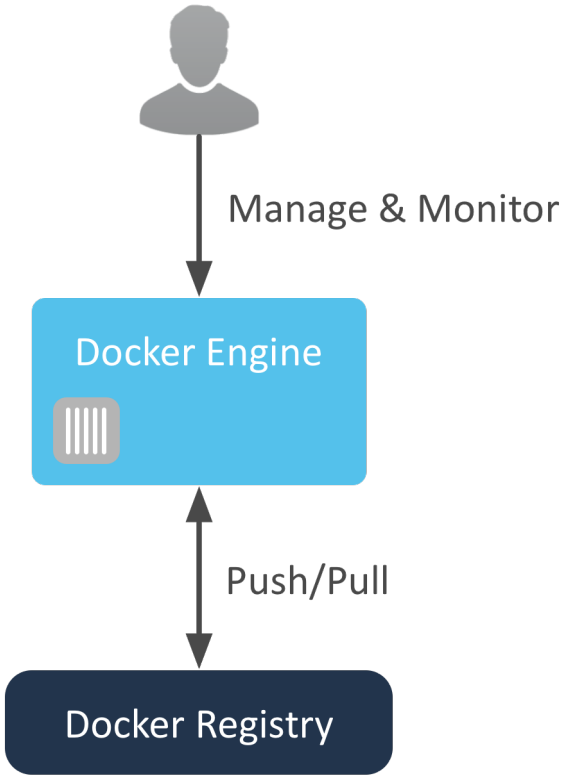
Docker Registries

- Types of Docker Registries
  - Local Docker Registry
    - On Docker Host
  - Remote Docker Registry
    - On-Premise/Off-Premise
  - Docker Hub
    - Off-Premise



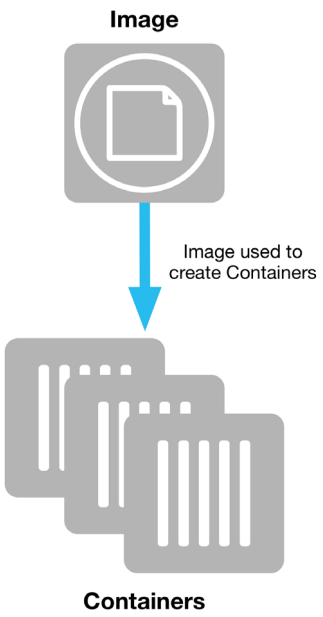
Pulling and Pushing

- Push
  - Upload an image to a registry
- Pull
  - Download an image from a registry
- Images are versioned in the registry



Docker Image Terminology

- Hierarchy of files, with meta-data for how to create/run a container
  - Read-only template used to create containers
  - Can be exported or modified to be used in new images
  - Created manually or through automated processes
  - Stored in a Registry (Docker Hub, Docker Trusted Registry, etc.)

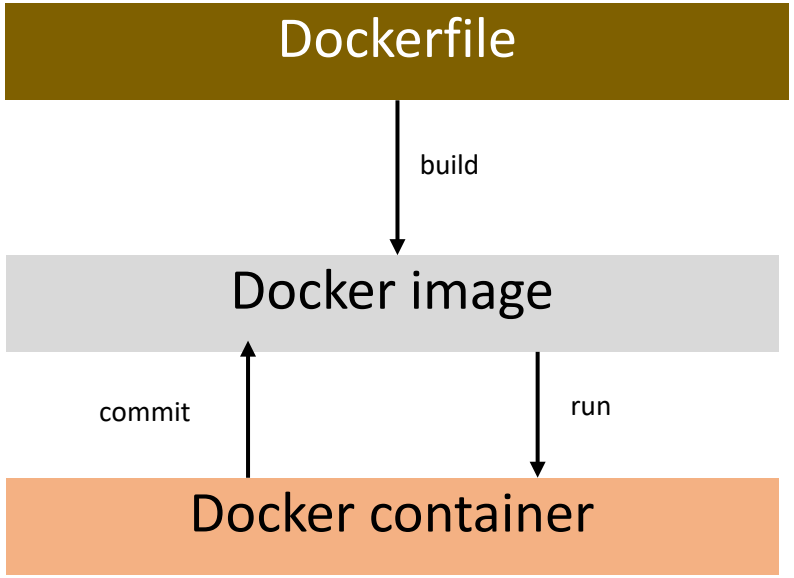


Inside a Docker Image/Container

- Docker images are built up in layers of file systems
  - Each layer is immutable
  - Each layer has a unique ID
  - The final image also has a unique ID
  - Immutable layers are shared/reused across images
- When a container is created from an image
  - An additional writeable layer is created
  - This maintains changes to the file system of a running container
  - All the immutable layers are shared across containers
- This architecture keeps containers light and small

Dockerfile

- Configuration File (script) for creating images. Defines:
  - Existing image to be the starting point
  - Set of instructions to augment that image (each of which results in a new layer of the file system)
  - Meta-data such as ports exposed
  - The command to execute when the image is run

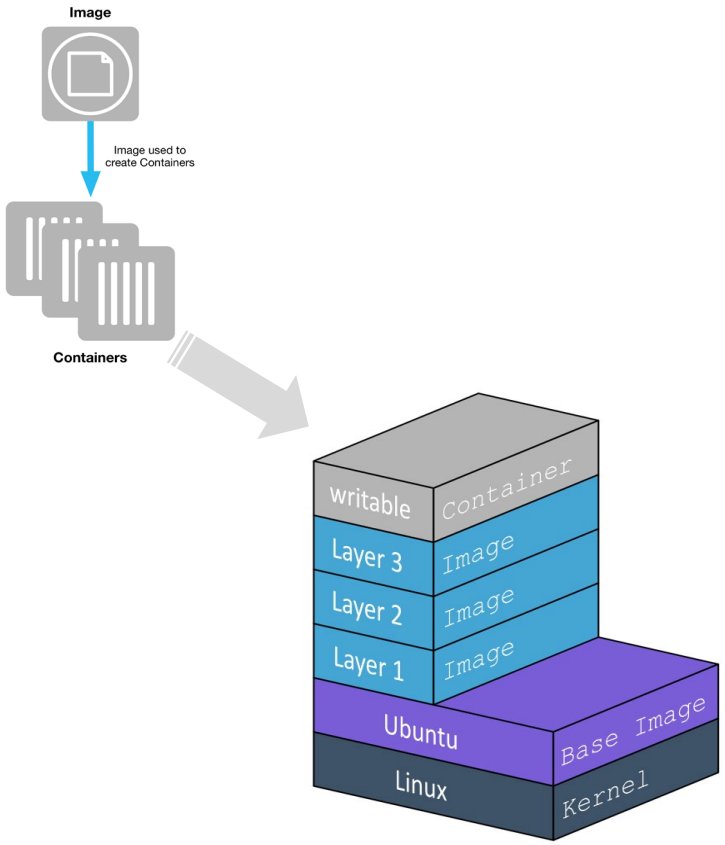


Deploying a Docker Application

Lab 4

Container

- Runtime instance of an image plus a read/write layer

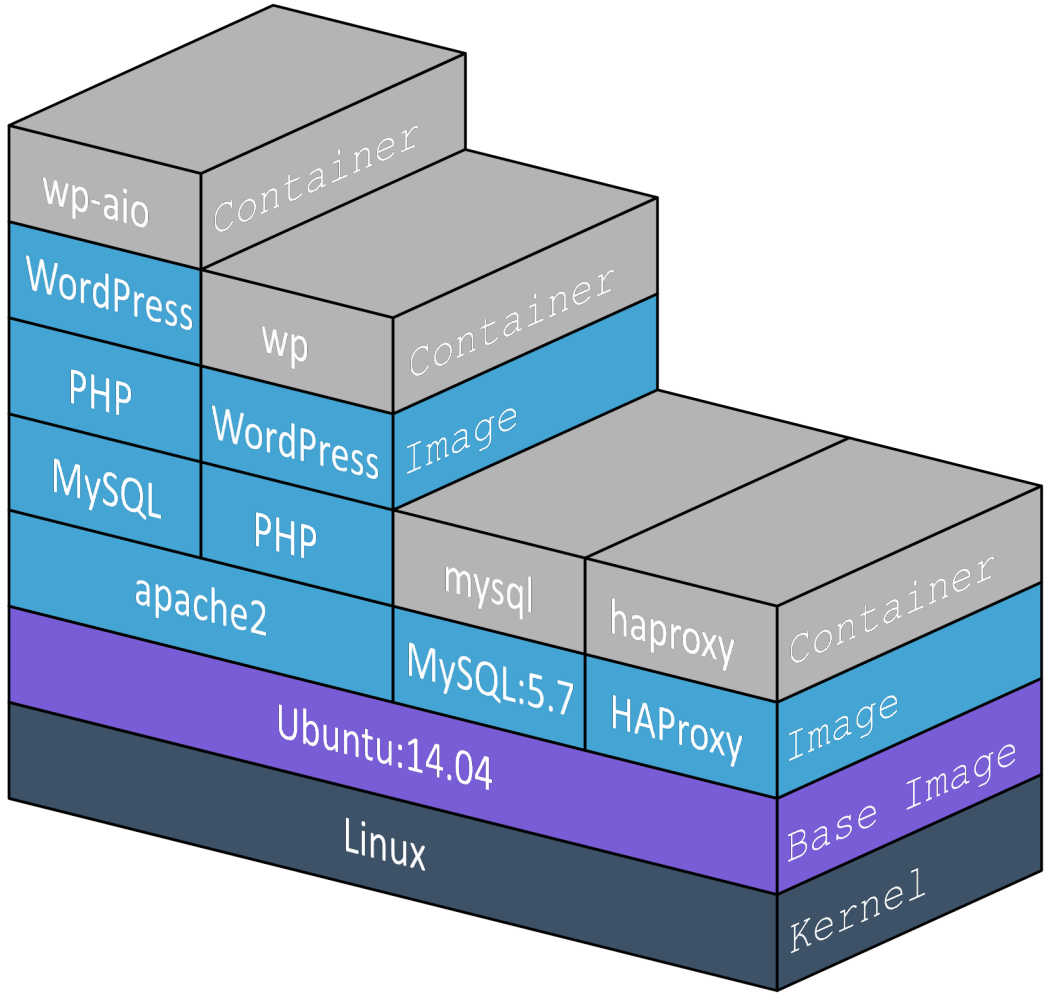


Inside a Docker Image/Container

Applications

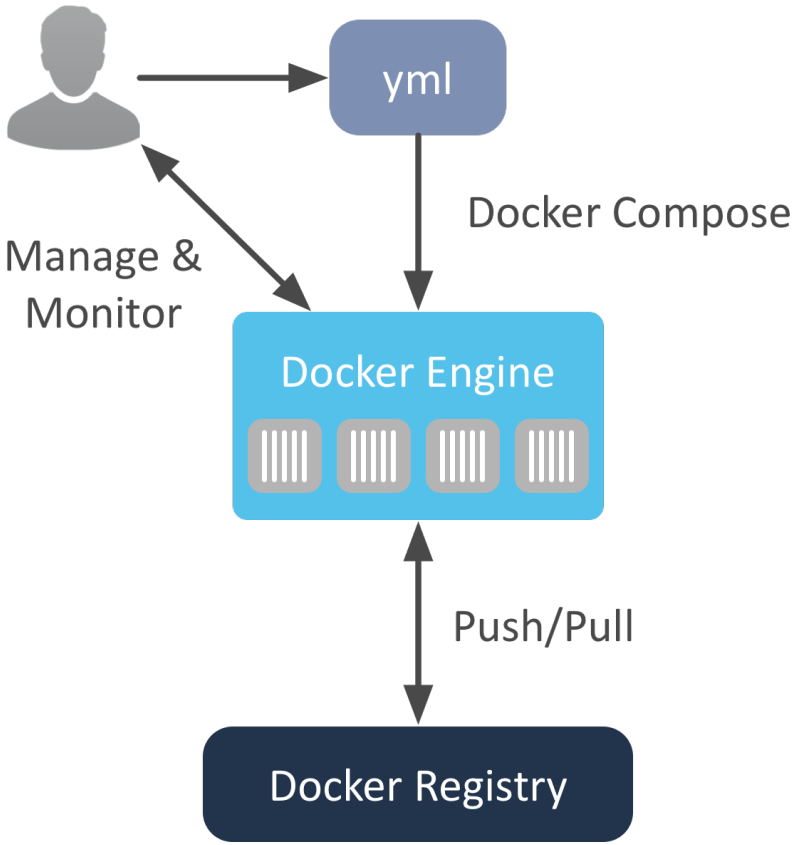
Containers

Images and/or layers used in the apps that get built



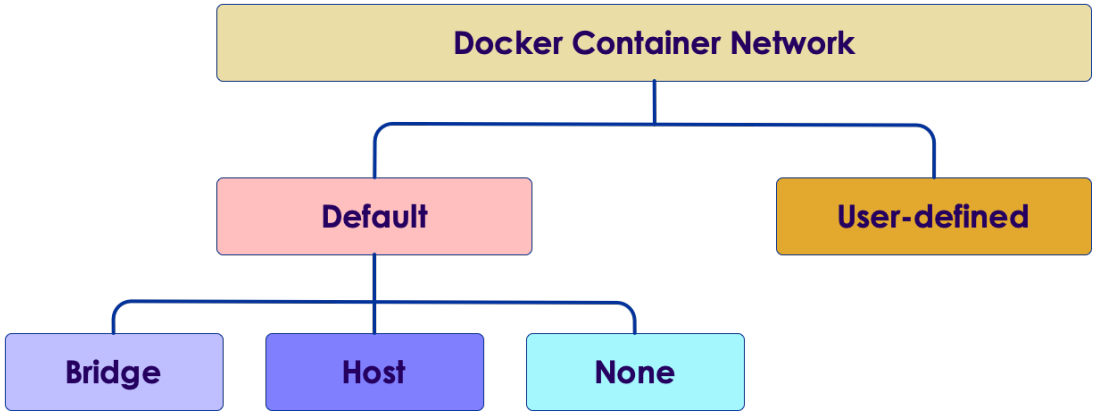
Docker Compose

- Tool to create and manage multi-container applications
- Applications defined in a single file: *docker-compose.yml*
- Transforms applications into individual containers that are linked together
- Compose will start all containers in a single command



Docker Networking

- Three default networks
- Bridge
  - Uses IPs in the range of 172.17.x.x
  - Local network for inter-container communication
- Host and None are not widely used
  - Use same network as host or none
- Additional networks can be defined



Port Mapping

- A container's functionality is accessed through a port
- Container ports on the bridge network are inaccessible
- Port mapping associates a container port with a host port
- This is used when a container is external interface
  - The following code maps the container port 80 to the host port 8998
  - Now the app can be accessed at the host port

```
docker run -d  --name app  -p 8998:80   app

curl localhost:8998
```

Docker Storage

- Containers have a writeable layer
  - This is ephemeral – it is lost when the container ends
- In order to have data persist, there are several options
  - Bind, where host directory is mounted in the container, like a NFS mount
  - The preferred mechanism is a Docker Volume
- Volumes are managed by Docker
  - Can use remote storage like AWS S3
  - Default location is /var/lib/Docker
  - Allows persistence and sharing of data between containers

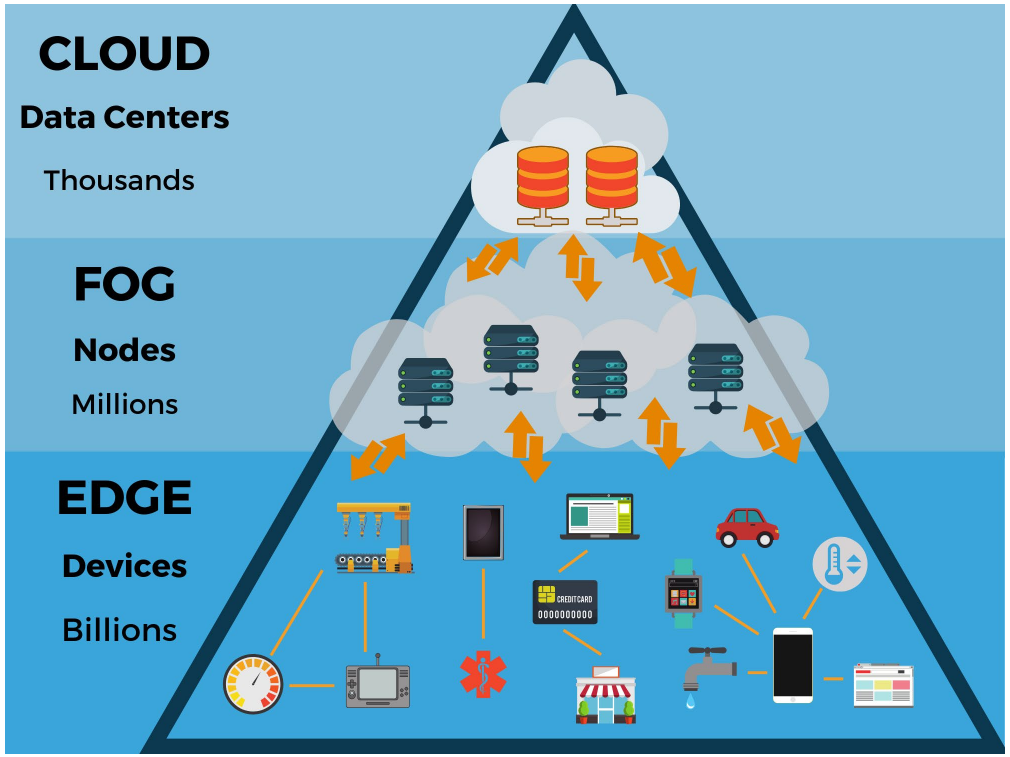
Building a Docker Application

Lab 5

Edge Devices

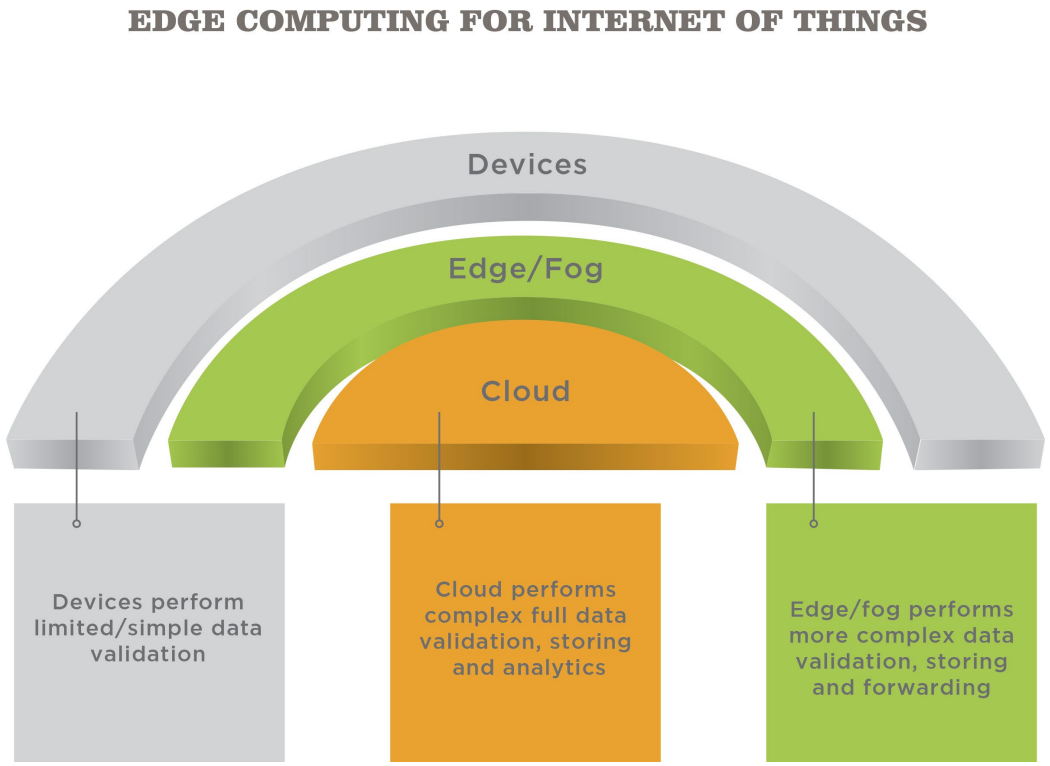
Edge Devices

- Devices with computing capability
  - Smart phones
  - Embedded processors
  - Sensors
  - Raspberry Pi
- Operate on the “edge” of cyberspace
  - Interface to the physical world
  - Cyber-physical connections
  - Internet of Things



Edge Computing

- Move the application closer to the user
  - Pushing the ML model out to edge devices
- Advantages
  - Reduces latency
  - Improves throughput
  - Parallel operations
  - Removes a point of failure
- Disadvantages
  - Relies on edge device capabilities and health
  - Updates and deployment can be problematic



The Coral Project

- Located at https://coral.ai/
- Focused on enabling privacy-preserving Edge ML with low-power, high performance products
- USB Accelerator
  - *"The Coral USB Accelerator adds an Edge TPU coprocessor to your system, enabling high-speed machine learning inferencing on a wide range of systems, simply by connecting it to a USB port."*

The Coral Project

Coral beta

Products Projects Docs Resources

Dev Board

A single-board computer with a removable system-on-module (SoM) featuring the Edge TPU.

- Supported OS: Mendel Linux (derivative of Debian)
- Supported Framework: TensorFlow Lite
- Languages: Python and C++
- Works with AutoML Vision Edge
- Datasheet

→ View product

USB Accelerator

A USB accessory featuring the Edge TPU that brings ML inferencing to existing systems.

- Supported host OS: Debian Linux
- Compatible with Raspberry Pi boards
- Supported Framework: TensorFlow Lite
- Works with AutoML Vision Edge
- Datasheet

→ View product

Camera

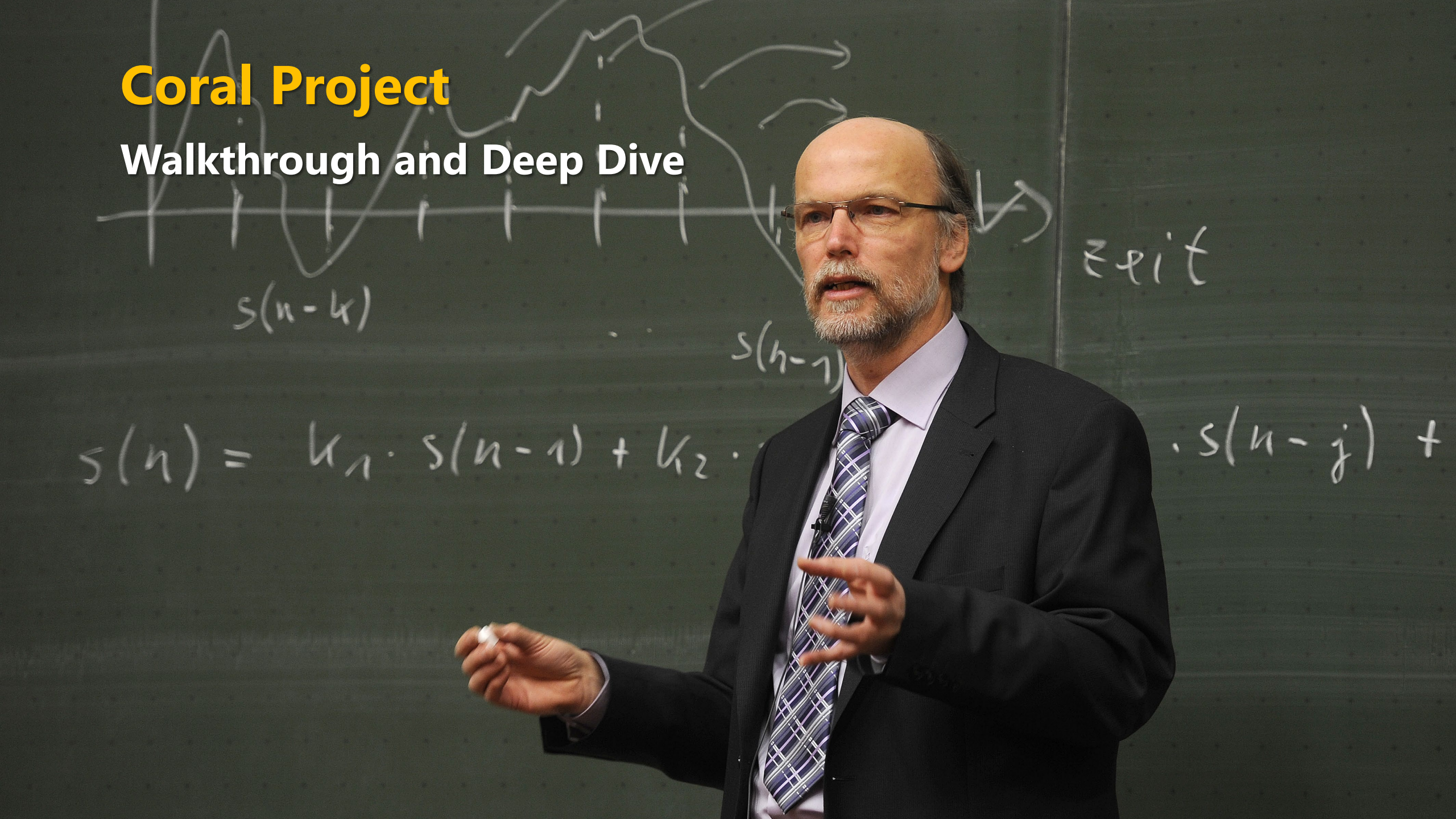
5-megapixel camera module that's compatible with the Coral Dev Board.

- Omnivision Sensor
- Autofocus
- Auto exposure control
- 25 mm x 25 mm
- Datasheet

→ View product

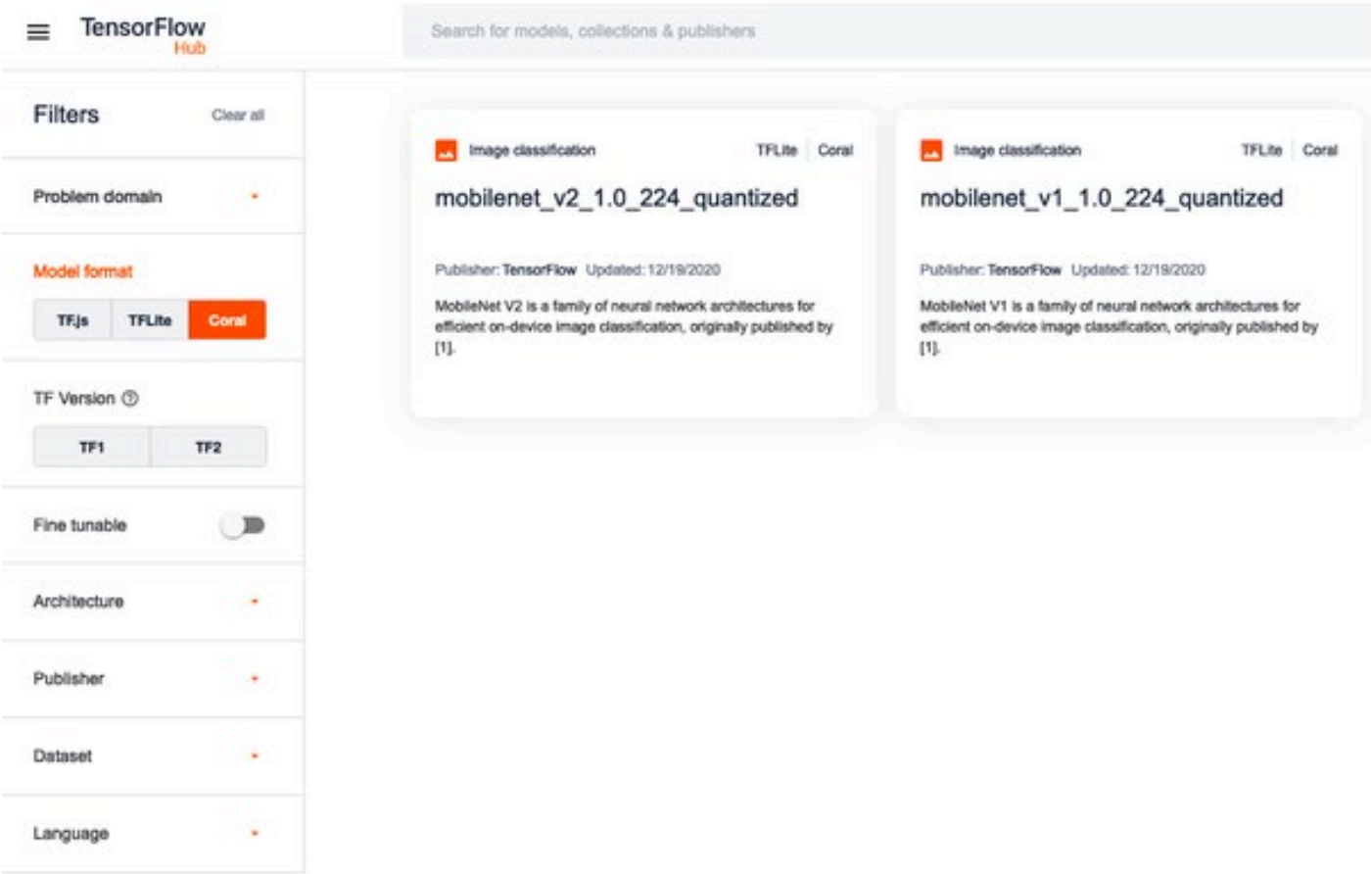
Coral Project

Walkthrough and Deep Dive



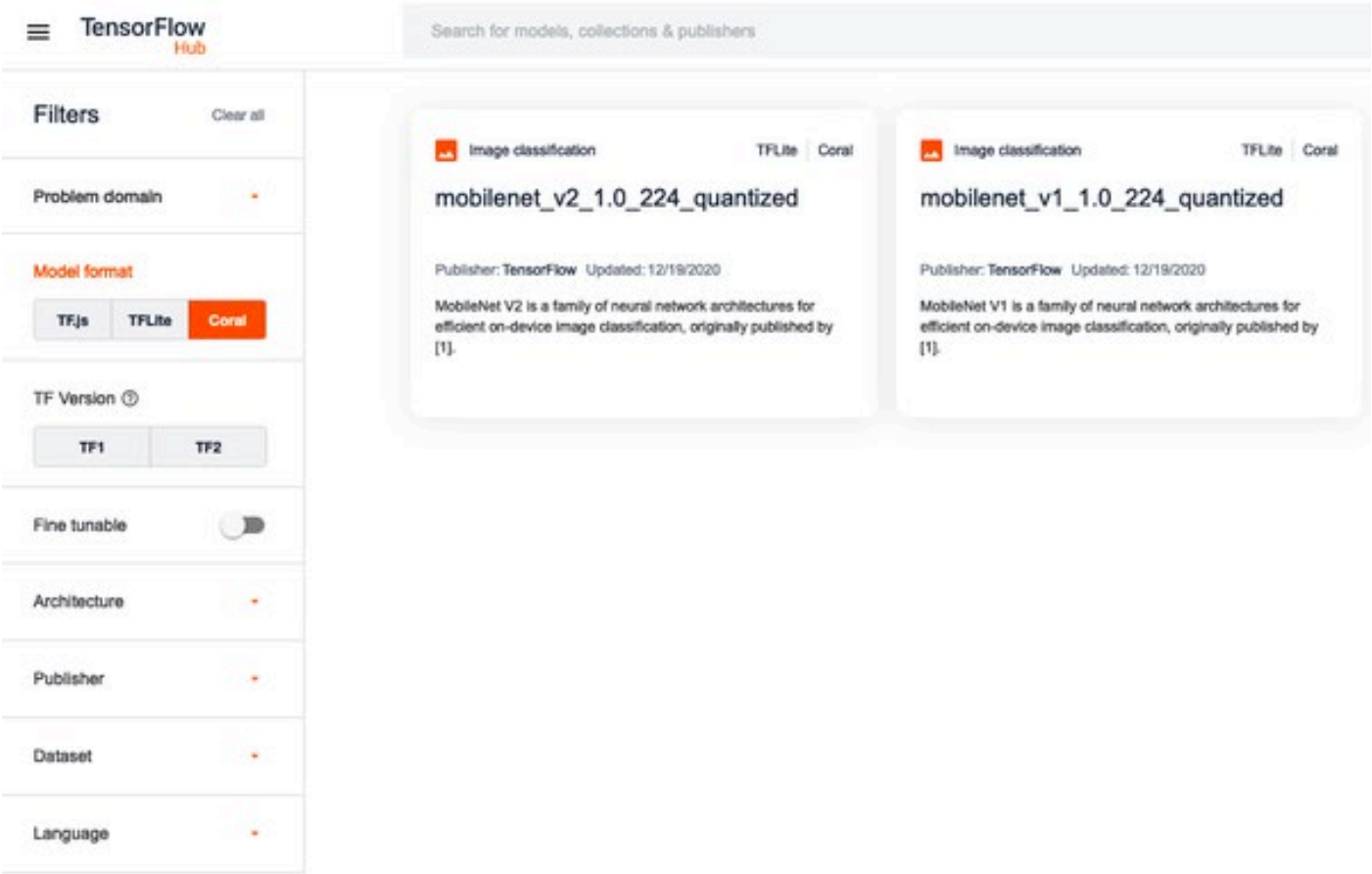
TFHub – TensorFlow Hub

- Located at https://tfhub.dev/
- Repository of thousands of pretrained models ready to be used.
  - For the Coral Edge TPU, not all models will work, though.

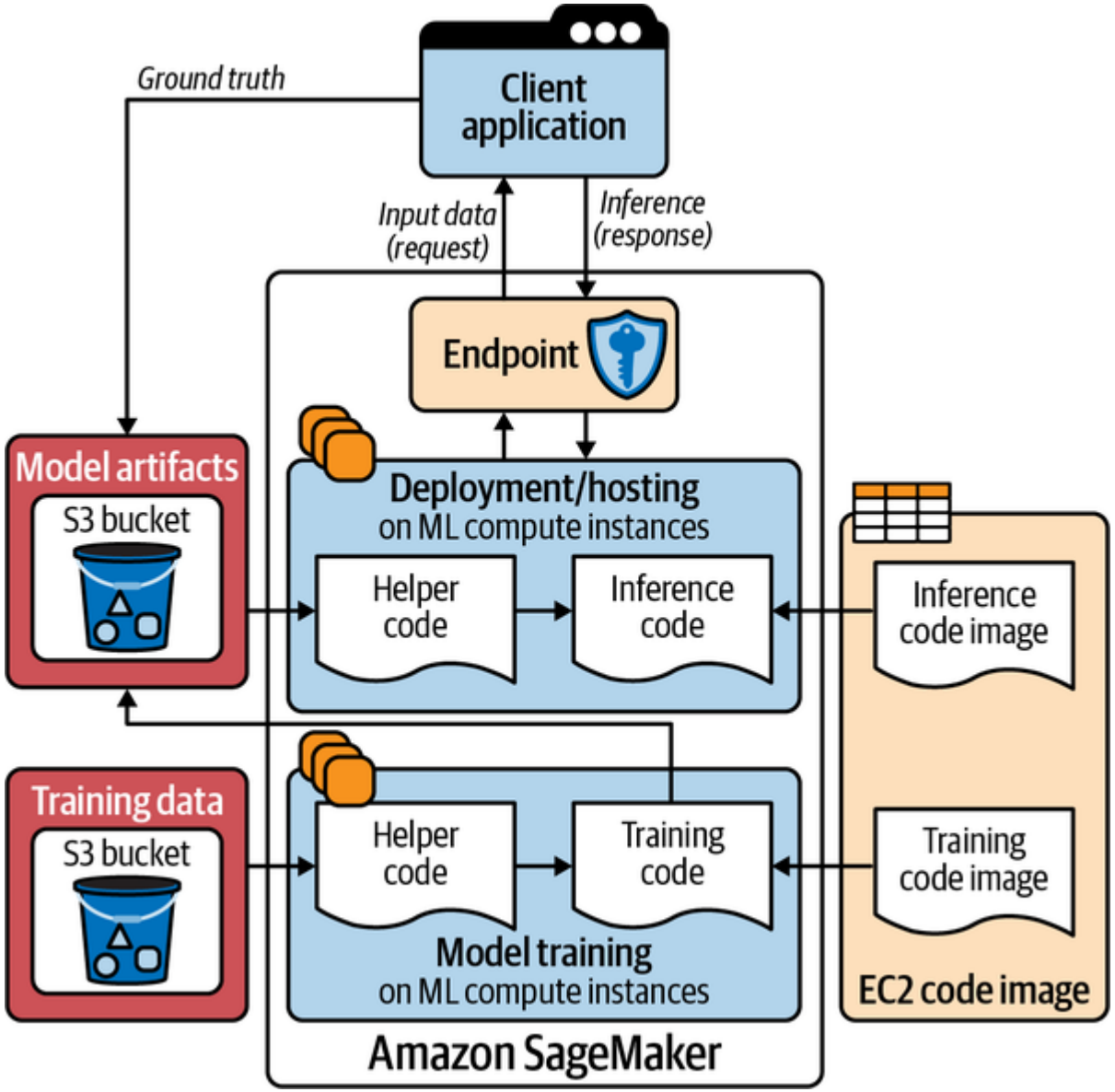


TFHub – TensorFlow Hub

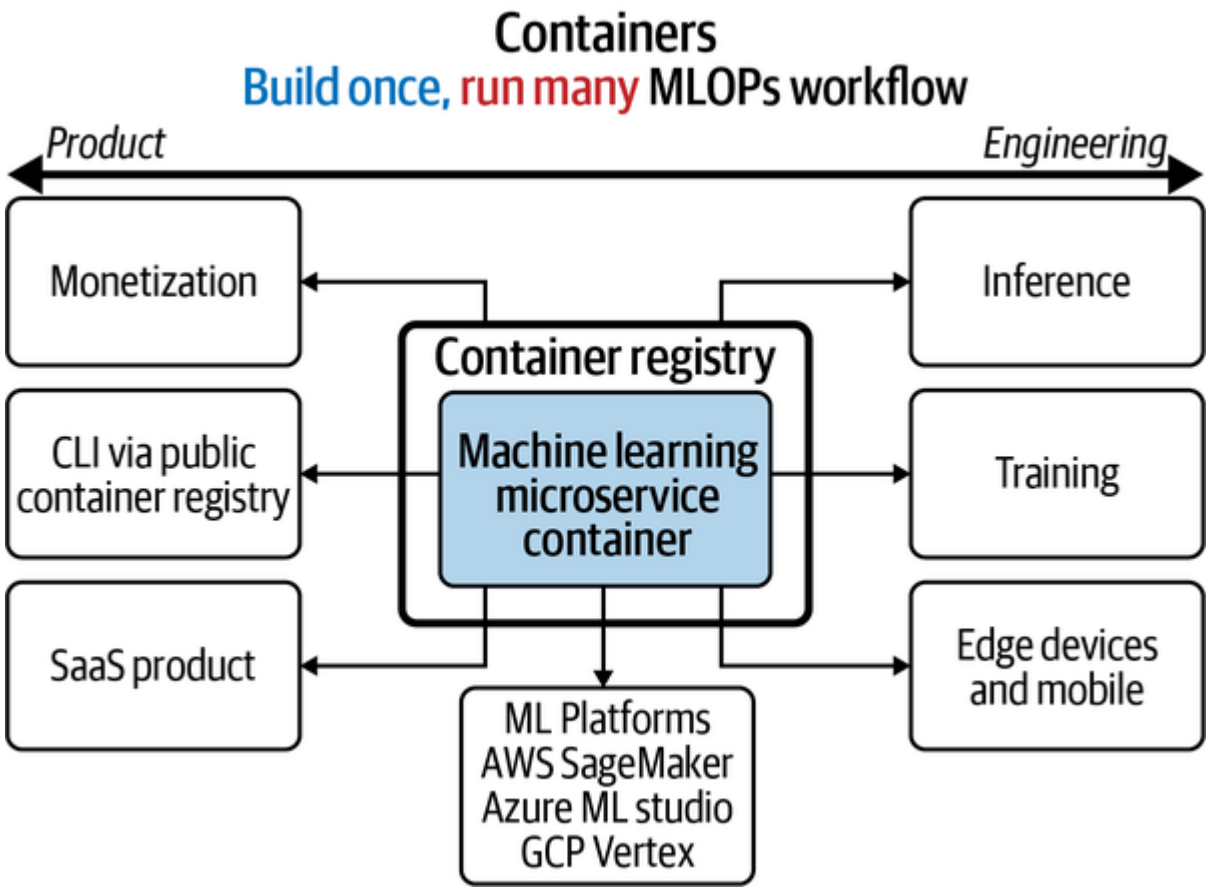
- Located at https://tfhub.dev/
- Repository of thousands of pretrained models ready to be used.
  - For the Coral Edge TPU, not all models will work, though.

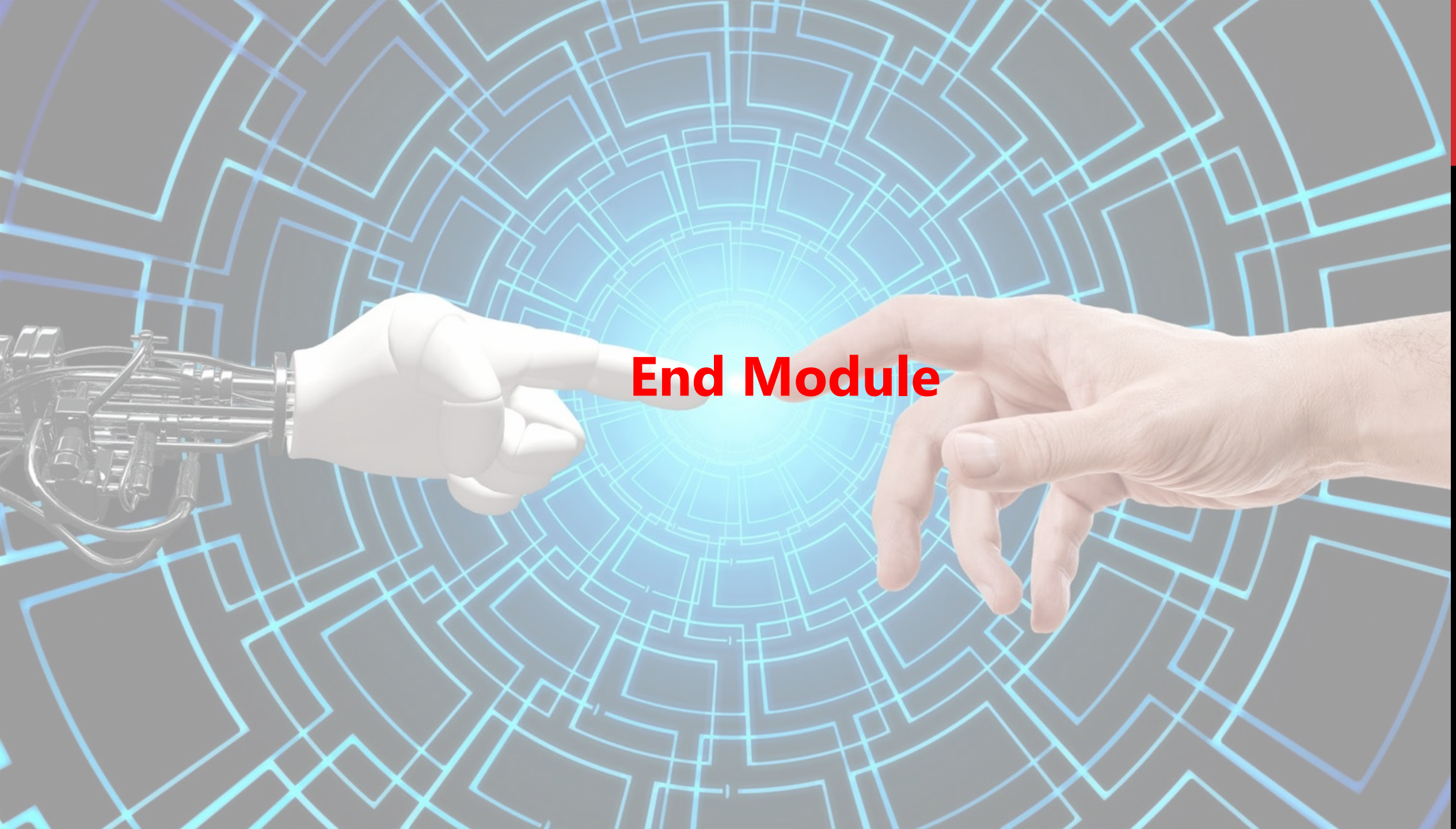


Containers for Managed ML Systems



Build Once, Run Many MLOps Workflow





End Module