Docker

ML in Production - Recitation 6

Outline

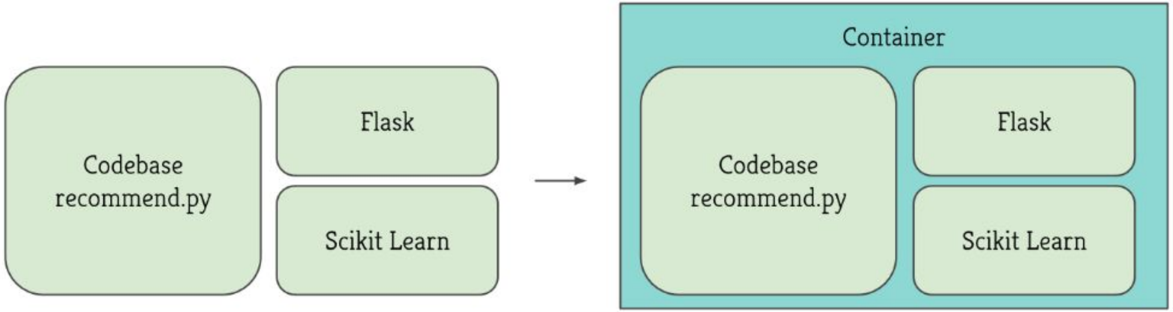
- Virtual Machines
- Containers
- Virtual Machines vs Containers
- Docker
- Demo - Docker Images, Containers, Docker Compose
- A/B Testing using Containers
- Demo - Load Balancers

Virtual Machines

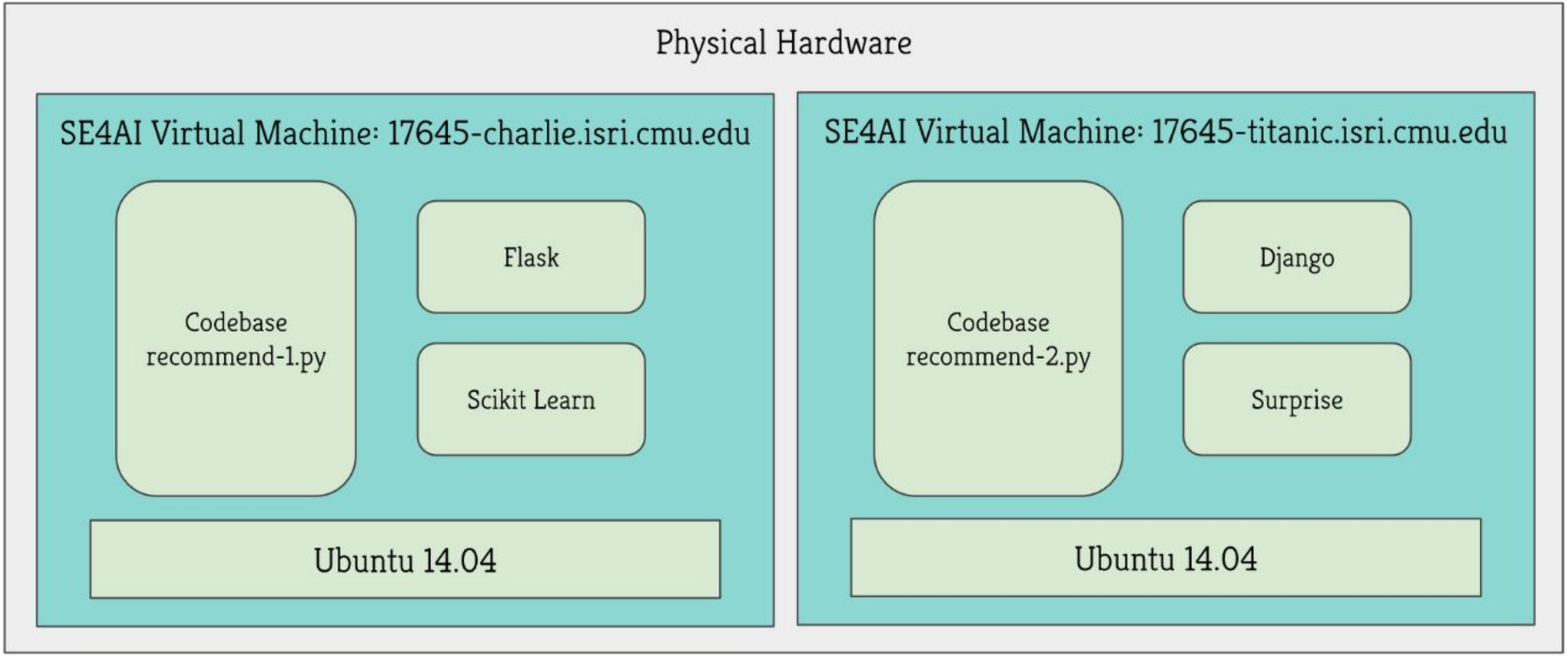
- Industry standard - Virtual Machines (VMs) to run s/w applications
- Virtual Machines
    - Applications run inside a virtual “guest” machine
    - Guest machine runs on a physical “host” machine
    - Multiple guests can run on same host machine
    - Each guest has its own OS and functions independently
- Pros:
    - Provide great isolation for applications
    - Problems in host OS rarely affect apps in guest OS and vice-versa
- Cons:
    - Isolation comes at great computational cost
    - Significant overhead virtualizing h/w for guest OS

Containers

- Containers take a different approach:
  - Leverage low-level mechanics of host OS
  - Provides most of the isolation as VMs
  - At a fraction of the cost
- Containers encapsulate or package an application such that
  - It is abstracted from environment in which they run
  - Decoupling allows ease of deployment irrespective of target environment
    - Public cloud, private data center, or a developer's personal laptop
  - It is lightweight in terms of loading and transporting
    - Granular control of resources to improve efficiency



Without Container



Physical Hardware

SE4AI Virtual Machine: 17645-charlie.isri.cmu.edu

Codebase
recommend-1.py

Flask

Scikit Learn

Ubuntu 14.04

SE4AI Virtual Machine: 17645-titanic.isri.cmu.edu

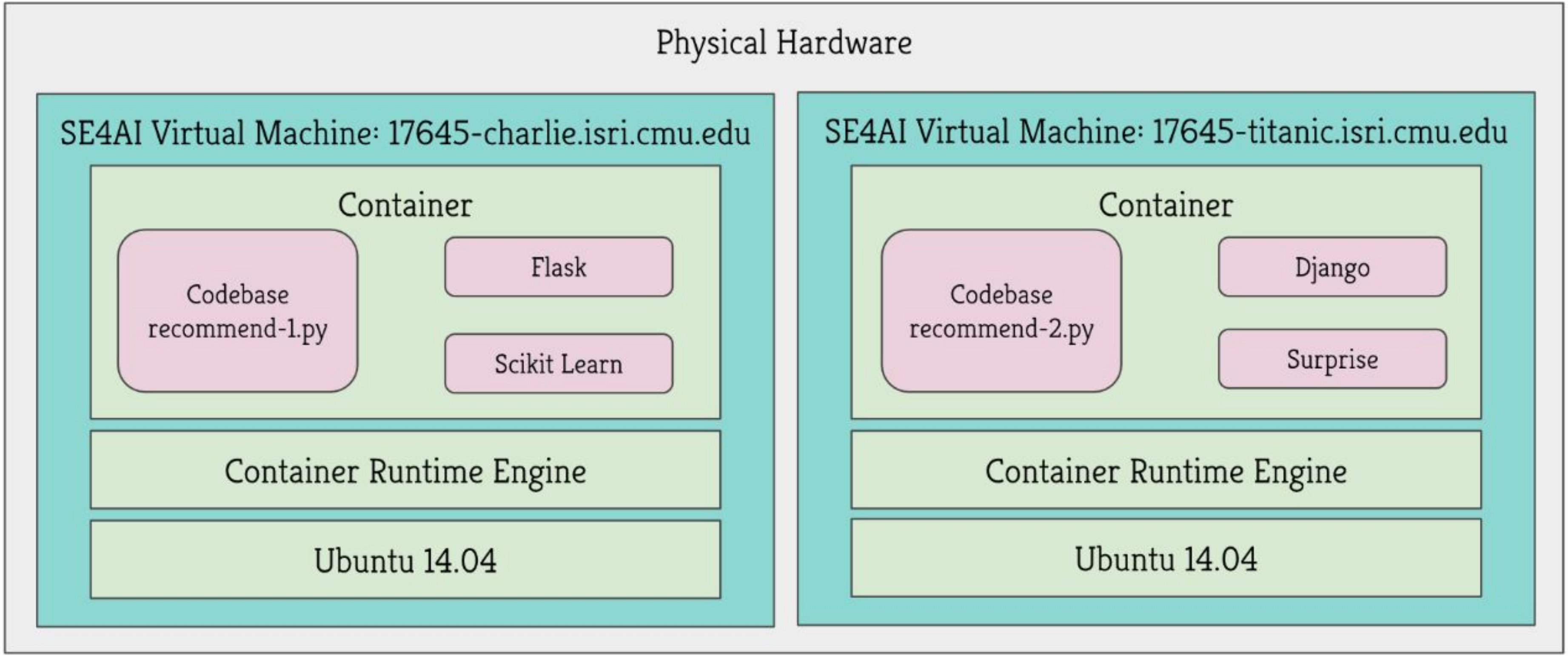
Codebase
recommend-2.py

Django

Surprise

Ubuntu 14.04

With Container



Physical Hardware

SE4AI Virtual Machine: 17645-charlie.isri.cmu.edu

Container

Codebase
recommend-1.py

Flask

Scikit Learn

Container Runtime Engine

Ubuntu 14.04

SE4AI Virtual Machine: 17645-titanic.isri.cmu.edu

Container

Codebase
recommend-2.py

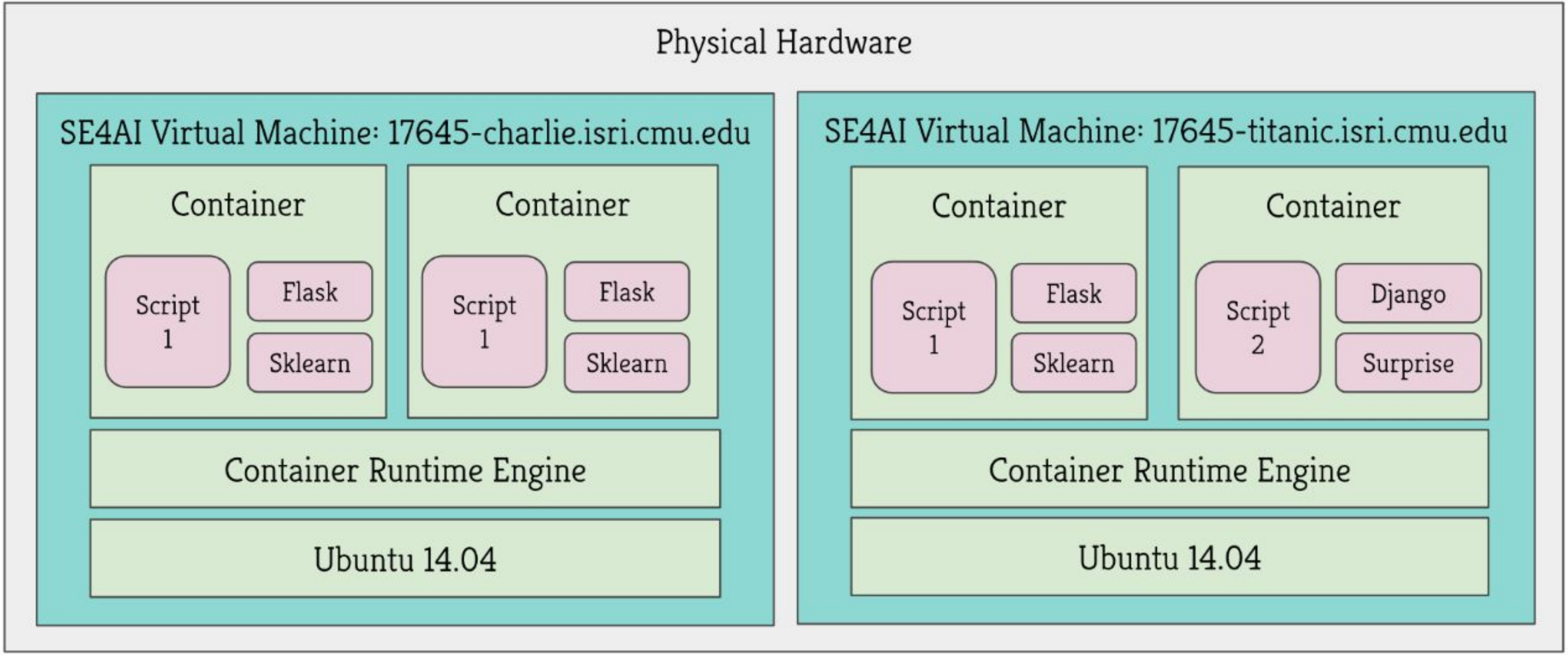
Django

Surprise

Container Runtime Engine

Ubuntu 14.04

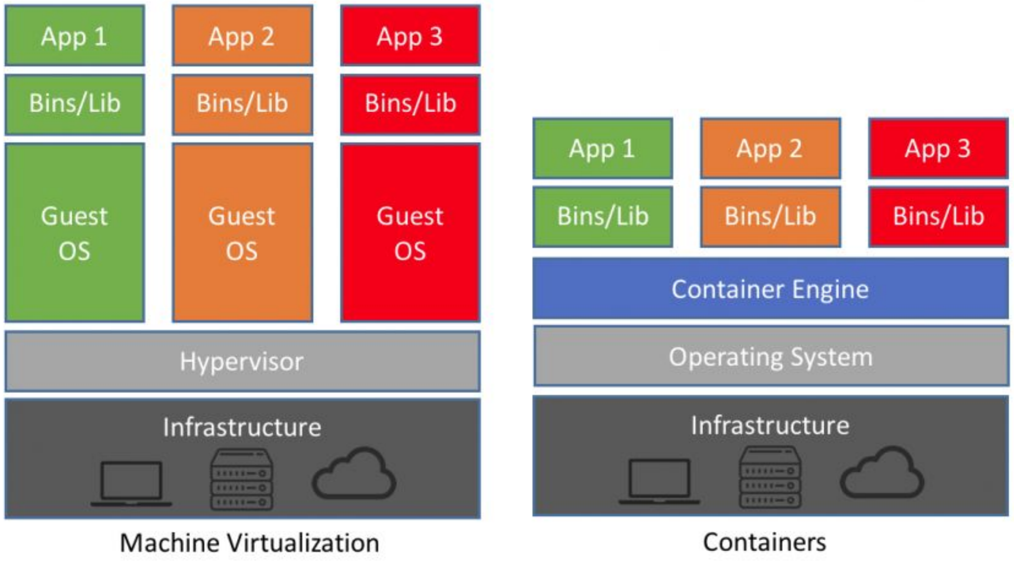
With Containers



Virtual Machines vs Containers

- Similarities
  - Both encapsulate your application
- Differences:
  - Size:
    - Containers are smaller in size as they do not contain the OS
  - Portability:
    - VMs are more portable (OS comes along with the VM)
    - Containers are portable as long as the container runtime supports the same format, ie. the same runtime engine has to be installed in the machine where it needs to run
  - OS:
    - Containers are constrained by the OS; VMs are not
    - Each VM has its own OS; Containers share OS of the host machine
  - State:
    - Containers are stateless by default (can be made stateful, although not recommended)

Virtual Machines vs Containers



Source: https://blog.netapp.com/blogs/containers-vs-vms/

Virtual Machines vs Containers

- Use containers when:
    - You care about the start times of your application (Containers are fast, VMs are slow)
    - Efficiency of resource utilization is of priority (Containers consume less RAM and CPU)
    - You have budget constraints (Docker & Kubernetes are free and open-source)
    - You want to share container images widely (Docker images can be created and shared easily, whereas VM images can be challenging)

- Use VMs when:
    - You are highly concerned with security want to isolate your environment (VMs provide a fully isolated environment by default)
    - You want portability across operating systems (Windows VMs can be deployed on Linux hosts and vice versa; Docker is not as portable)
    - You want to have a rollback feature (VMs can easily go back to a previous snapshot)

Source: https://www.weave.works/blog/a-practical-guide-to-choosing-between-docker-containers-and-vms

Docker

- Platform-as-a-service product to automate deployment of s/w applications
- Applications deployed in containers and run on host OS
  - Many docker containers can be run simultaneously on a host
- Allows packaging of application and its dependencies into a standardized unit
- Terminologies:
  - Image - Everything that is need to configure a fully operational environment
  - Container - A running instance of an image
  - Dockerfile - Definition/Spec to create an image
  - Container Registry - System to host and distribute images
  - Container Repository - Specific physical locations to store related images

Docker-compose

- Compose is a tool for defining and running multi-container Docker applications
- With Compose, you use a YAML file to configure your application’s services
- Create and start all the services from your configuration with a single command
- Using Compose is basically a three-step process:
    - Define your app’s environment with a Dockerfile so it can be reproduced anywhere
    - Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment
    - Run “docker-compose up” and the Docker compose command starts and runs your entire app. You can alternatively run docker-compose up using the docker-compose binary

Demo - Docker Images, Containers & Docker Compose

- Creating an image using a Dockerfile
- Creating a container using the image
- Using docker-compose
  - Running containers on different ports on the same machine
  - Inspecting container logs
- Files: Code Here

A/B Testing using Containers

- A/B testing is an experiment to compare two versions of a variable to find out which performs better in a controlled environment
- Compare performance of different models using this technique
- How to use containers for this?
    - Deploy different containers each having different models
    - Decide a strategy to route users to each model
    - Have a load balancer to execute this strategy
    - Collect results

Definition source: https://www.analyticsvidhya.com/blog/2020/10/ab-testing-data-science/

Demo - Load Balancers

Outcomes:

- Constructing an efficient load balancer
- Understanding a simple randomizer strategy to route traffic

Resources

- Docker documentation: https://docs.docker.com/get-started/
- Docker commands cheat sheet: https://www.edureka.co/blog/docker-commands/
- Docker tutorial for beginners: https://docker-curriculum.com/
- Load balancing: https://en.wikipedia.org/wiki/Load_balancing_(computing)
- Docker Demo Code Zip: