Docker Masterclass for Machine Learning and Data Science

Thanks again for joining me in this course! Please feel free to use this resource as a recap of the information that we covered in our intuition section. In addition, please keep an eye out for any future course announcements!

Overview

This document is a free source to help recap key information from the section. It's intended to help provide useful resources, links and summaries to the related areas so that you have all of the tools necessary to learn and succeed with the course, and your career.

Introduction

What is docker?

Docker is a tool that can package and application or algorithm along with its dependencies in a virtual container that can run on a linux server. This has tremendous benefits including replication, being able to run it in the cloud, and more. In addition, Docker containers are seen as lightweight with the ability of a single server or virtual machine to run multiple containers.

Docker History

Docker Inc. was founded by Solomon Hykes and Sebastien Pahl during the Y Combinator Summer 2010 startup incubator group and launched in 2011. Hykes started the **Docker** project in France as an internal project within dotCloud, a platform-as-a-service company
Docker debuted to the public in Santa Clara at PyCon in 2013.[12] It was released as open-source in March 2013.[13] At the time, it used LXC as its default execution environment. One year later, with the release of version 0.9, Docker replaced LXC with its own component, which was written in the Go programming language.[14][15]

https://en.wikipedia.org/wiki/Docker_(software)

Related Domains

One of the reasons that Docker is such a useful tool is that it can be applied across many domains and disciplines. This can range from Data Science to DevOps to Software Engineering, System

Administration and Cybersecurity. Containerization is beneficial to be able to run and replicate algorithms across different systems and locations.

Key Terms

CI/CD

This is a term that you may hear in the DevOps space frequently. We may come across it and I want to have it included since it's a very crucial part of DevOps, and highly useful when examining Docker and why it's used. The CI/CD Pipeline refers to the Continuous Integration/Continuous Deployment cycle.

Containers

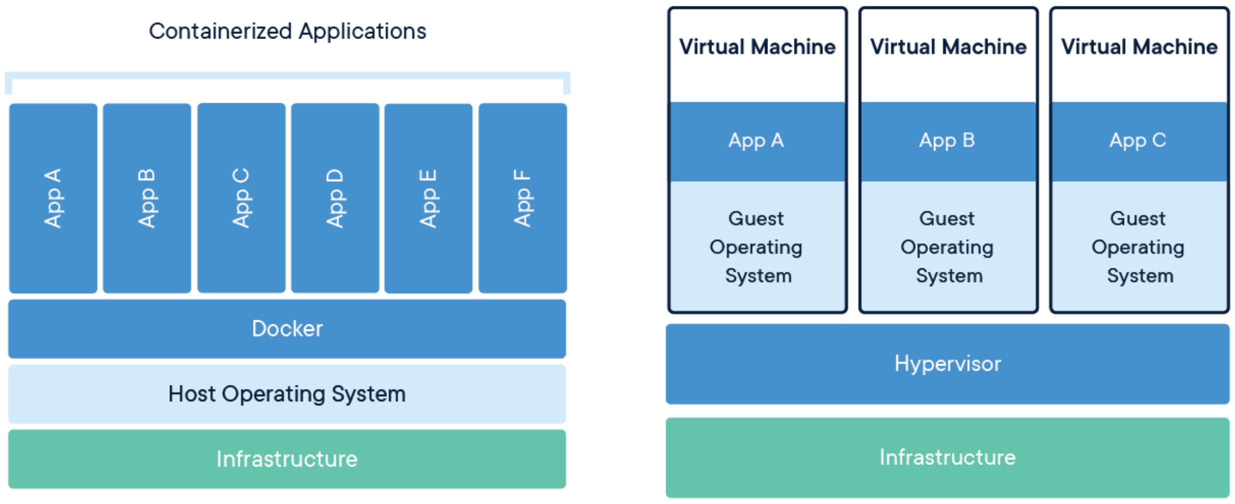
"Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems"

https://www.docker.com/resources/what-container

Virtual Machine

Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, the application, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

Example of a Containerized Application vs a VM:



https://www.docker.com/resources/what-container#/package_software

Images

A Docker image is immutable and a read only template containing the instructions required for our application. It contains the source code along with the libraries, dependencies and files needed for our application. If we want to examine how a container and image are different, a container requires an image to run.

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

To view the Overview mentioned, obtain further information and useful resources please also see: https://docs.docker.com/get-started/overview/