Containerization

Agenda

- Drawbacks of Virtualization
- Containerization
- Base Image & Container Image
- Dockerfile
- REST API

Great Learning
POWER AHEAD

Let's begin the discussion by answering a few questions on Containerization and REST APIs.

Drawbacks of Virtualization

Why is virtualization considered more resource-intensive than containerization?

A. Virtualization requires internet access

B. Virtualization requires storing data in the cloud

C. Virtualization runs a full operating system in each instance

D. Virtualization uses containers inside them

Drawbacks of Virtualization

Why is virtualization considered more resource-intensive than containerization?

A. Virtualization requires internet access

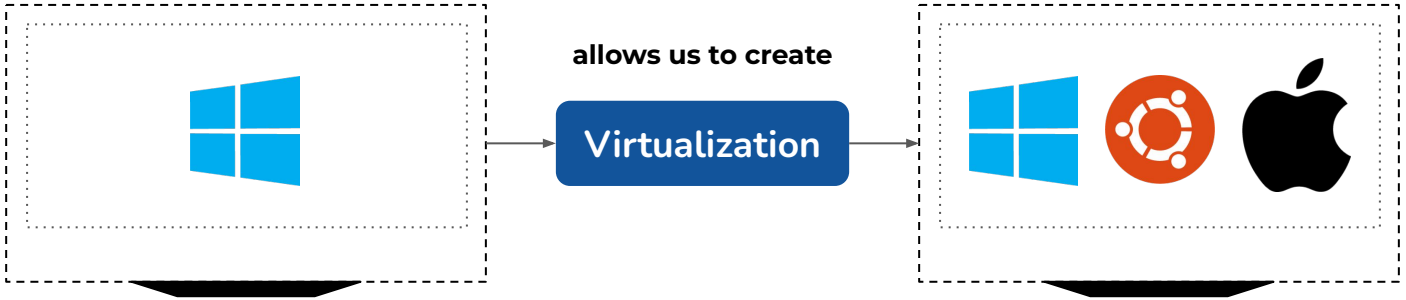
B. Virtualization requires storing data in the cloud

C. Virtualization runs a full operating system in each instance

D. Virtualization uses containers inside them

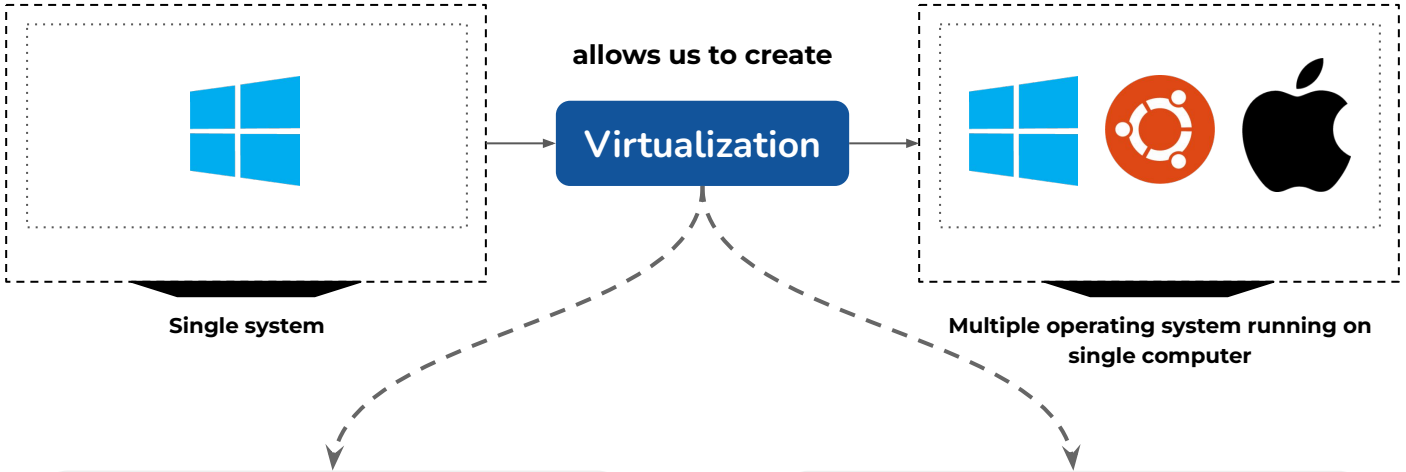
Virtualization

Virtualization is **technology** that one can use to **create virtual representations** of servers, storage, networks, and other physical machines.



Virtualization

Virtualization is **technology** that one can use to **create virtual representations** of servers, storage, networks, and other physical machines.



Requires a lot of memory and processing power because they run full operating system(s)

More users increase workload, slowing down inference if resources are not scaled accordingly

Containerization

Which of the following scenarios BEST highlights a key benefit of containerization over virtualization?

A. A data science model is tested in an isolated environment with no user interface

B. A developer runs the same container image on Windows and Linux systems with identical performance

C. Virtualization fails due to low disk space

D. A model is retrained every day with new data

Containerization

Which of the following scenarios BEST highlights a key benefit of containerization over virtualization?

A. A data science model is tested in an isolated environment with no user interface

B. A developer runs the same container image on Windows and Linux systems with identical performance

C. Virtualization fails due to low disk space

D. A model is retrained every day with new data

Containerization

Containerization is the **packaging of software** code with just the **operating system (OS) libraries and dependencies** required to run the code to create a **single lightweight executable** - called a **container** - that **runs constantly on any infrastructure**.

No need to install everything manually - **just run the container**!

Works the same way on every OS - whether on Windows, macOS, or Linux

Everything is **packaged into a single unit**, making sharing and deploying models effortless

Unlike virtualization, **containers don't need to run full operating system(s)** - they use only the necessary files to create a consistent environment.

Base Image & Container Image

If you add only your application code into a container without a base image, what will happen?

A. The container will work because the code will automatically find its dependencies

B. The container will be unable to function as it lacks an OS and necessary dependencies

C. The code will automatically install the necessary operating system

D. The container will use the cloud server as a base for the app

Base Image & Container Image

If you add only your application code into a container without a base image, what will happen?

A. The container will work because the code will automatically find its dependencies

B. The container will be unable to function as it lacks an OS and necessary dependencies

C. The code will automatically install the necessary operating system

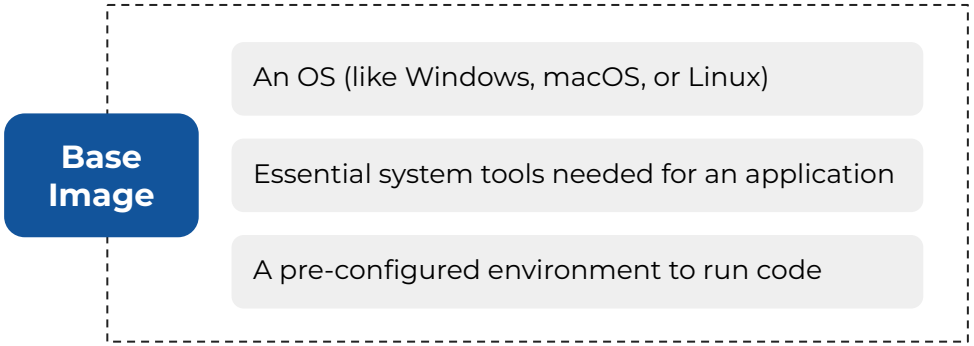
D. The container will use the cloud server as a base for the app

Base Image & Container Image

A **container image** represents binary data that **encapsulates an application and all its software dependencies**.

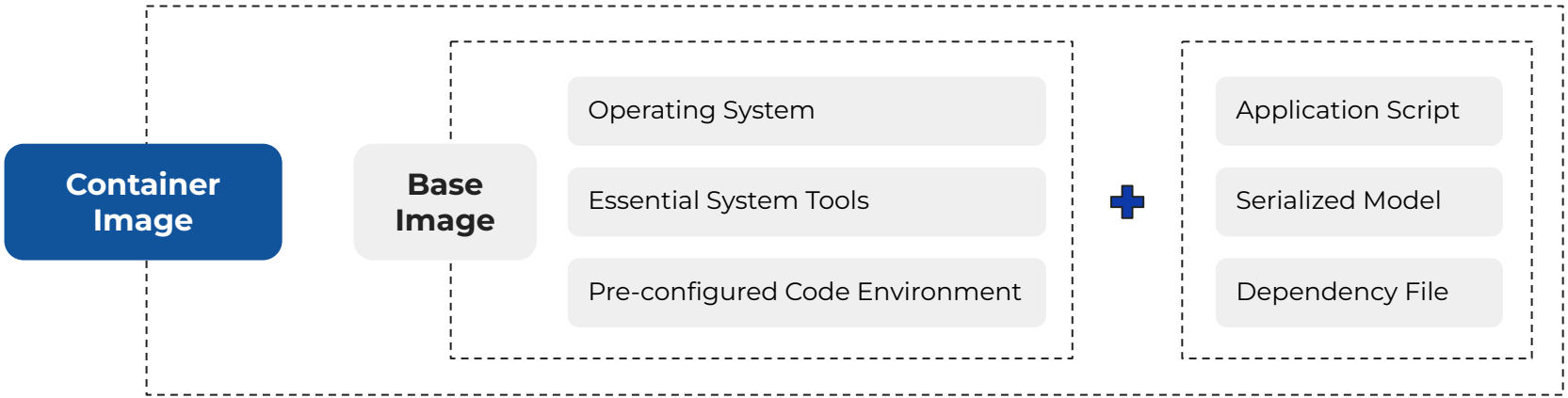
But they still need a foundational environment to run, they don't come with built-in operating system or tools by default.

A **base image** is the **starting point** for most container-based development workflows



Base Image & Container Image

A container image is built on top of a base image.



Removing the base layer (OS and tools) breaks this structure, rendering the container unusable.

Dockerfile

What problem does the Dockerfile solve compared to manually building containers?

A. Reduces container size to zero

B. Automates and standardizes container builds

C. Ensures container runs without internet

D. Provides built-in user interfaces for models

Dockerfile

What problem does the Dockerfile solve compared to manually building containers?

A. Reduces container size to zero

B. Automates and standardizes container builds

C. Ensures container runs without internet

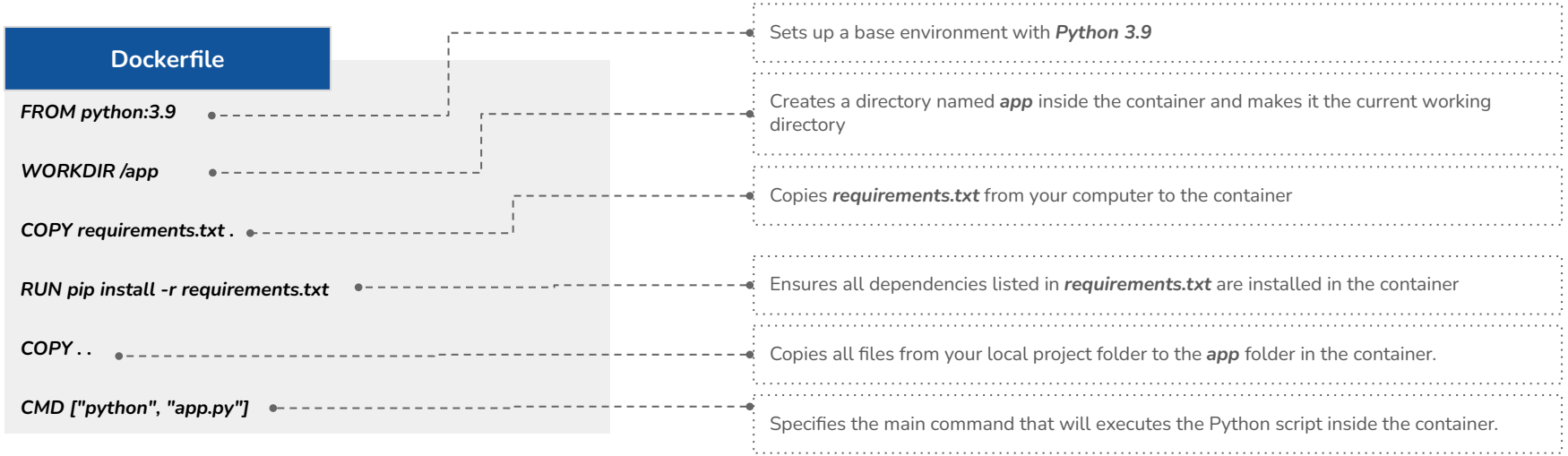
D. Provides built-in user interfaces for models

Dockerfile

A **Dockerfile** is a text document that **contains all the commands** a user requires to assemble a container image.

It **provides instructions** on the commands to run, files to copy, when to startup, and more

One can **build consistent images automatically** by reading the instructions from a Dockerfile



REST API

What does decoupling the frontend from the backend using a REST API accomplish?

A. It allows developers to update the frontend without affecting the backend

B. It increases the overall size of the application code

C. It makes it difficult to reuse model predictions

D. It complicates the API unnecessarily

REST API

What does decoupling the frontend from the backend using a REST API accomplish?

A. It allows developers to update the frontend without affecting the backend

B. It increases the overall size of the application code

C. It makes it difficult to reuse model predictions

D. It complicates the API unnecessarily

REST API

Common model deployment frameworks allows us to build interactive web applications - frontend plus backend

Frontend and backend coupled together => developers can't access model outputs

Changes in the model or its deployment **may require significant updates** to the web application

May **limit the application's potential** by **not allowing for easy integration** with other services or systems

REST APIs follow a standard architecture (REST) that enables fast, structured, and scalable communication between frontend and backend.

REST follows universal rules that simplify software integration for developers.

Decoupling enables frontend and backend to be developed and enhanced independently

Great Learning

Power Ahead!