Lecture 14: Containers

Advanced Practical Data Science, MLOps

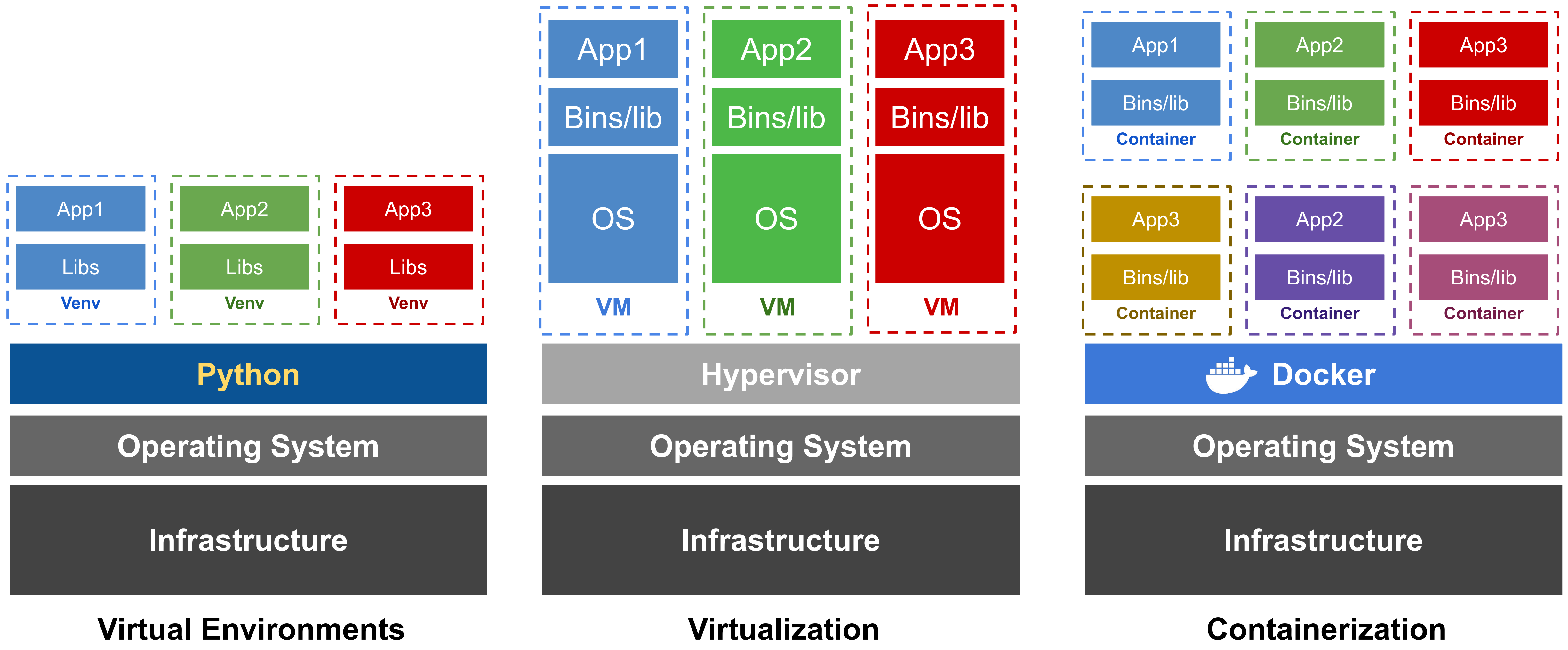
AC295

Pavlos Protopapas
Institute for Applied Computational Science, Harvard

Outline

1. Recap
2. Why use Containers - Part 2?
3. Tutorial: Building the Mega Pipeline App

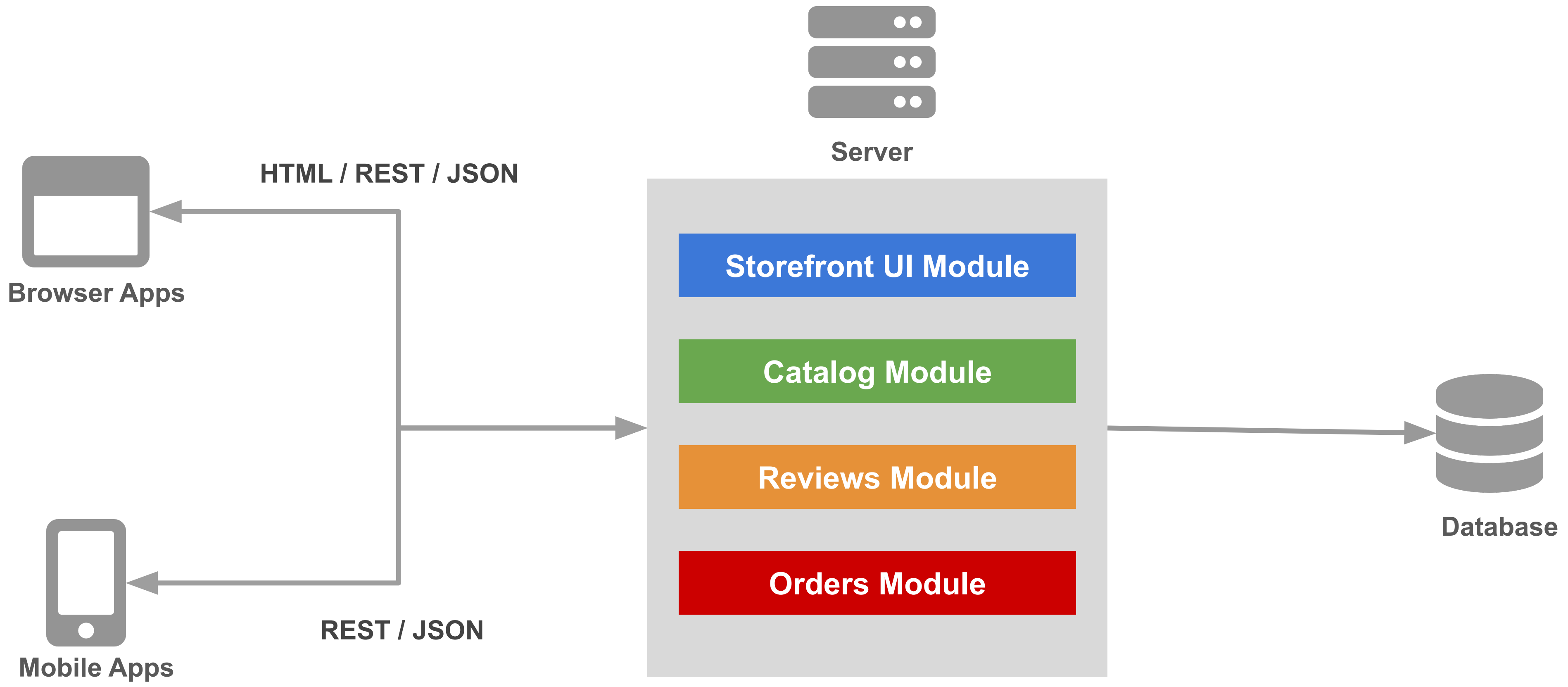
Environments vs Virtualization vs Containerization



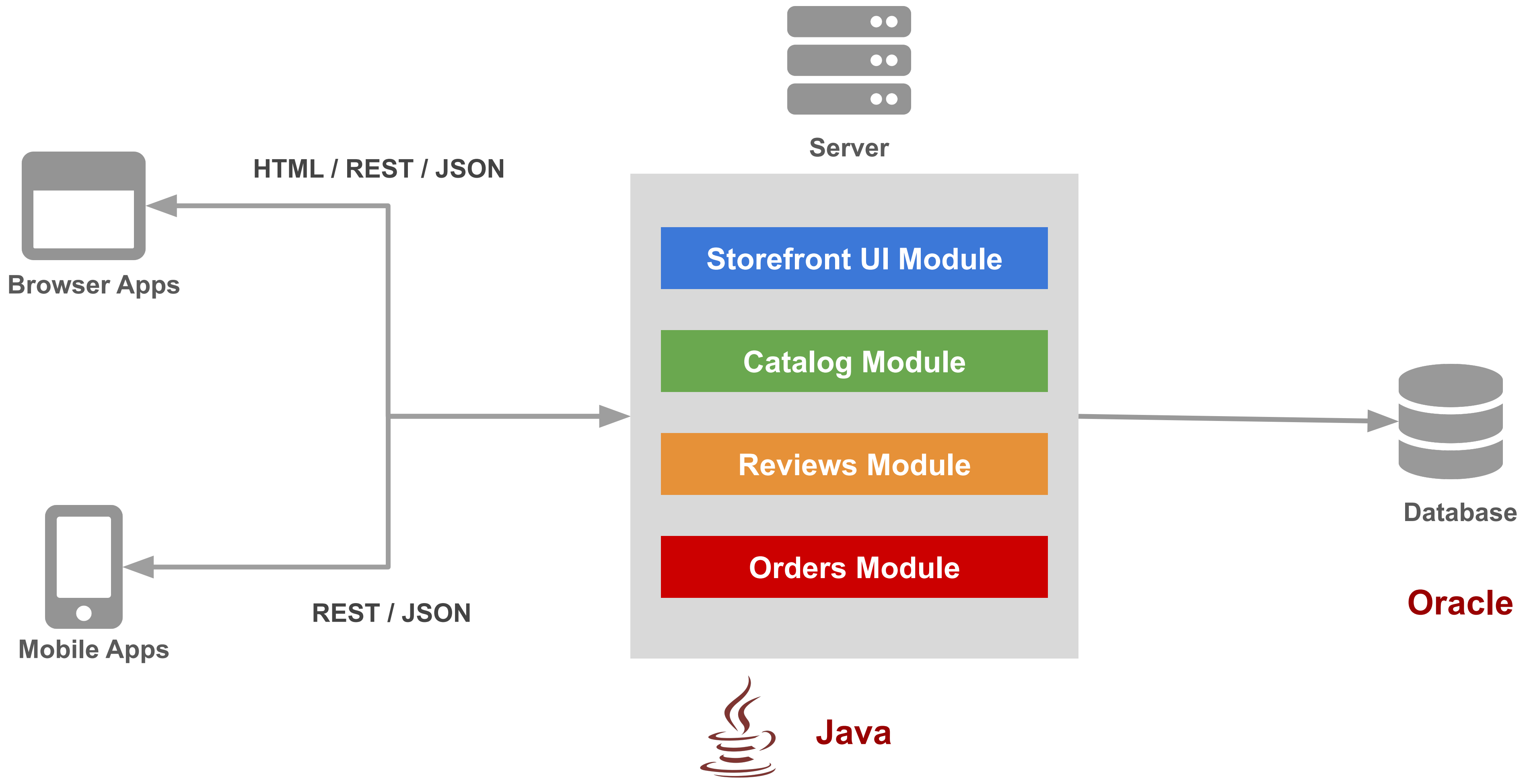
Why use Containers?

- Imagine you are building a large complex application (e.g. Online Store)
- Traditionality you would build this using a Monolithic Architecture

Monolithic Architecture



Monolithic Architecture



Monolithic Architecture - Advantages

Simple to **Develop**, **Test**, **Deploy** and **Scale**:

1. Simple to develop because all the tools and IDEs support the applications by default.
2. Easy to deploy because all components are packed into one bundle.
3. Easy to scale the whole application.

Monolithic Architecture - Disadvantages

1. Very difficult to maintain
2. One component failure will cause the whole system to fail
3. Very difficult to create the patches for monolithic architecture
4. Adapting to new technologies is challenging
5. Take a long time to startup because all the components needs to get started

Applications have changed dramatically

A decade ago

Apps were monolithic
Built on a single stack (e.e. .NET or Java)
Long lived
Deployed to a single server

Today

Apps are constantly being developed
Build from loosely coupled components
Newer version are deployed often
Deployed to a multitude of servers

Applications have changed dramatically

A decade ago

Apps were monolithic
Built on a single stack (e.e. .NET or Java)
Long lived
Deployed to a single server

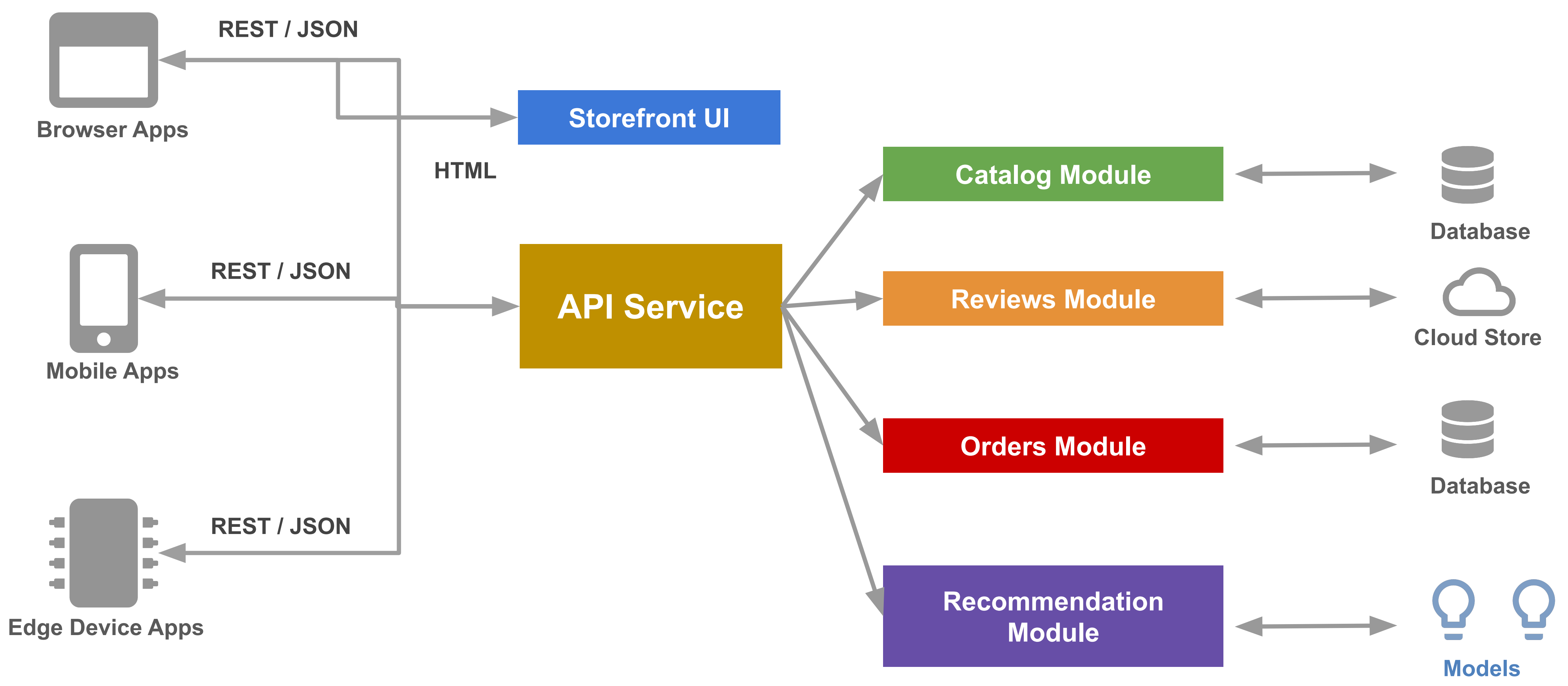
Today

Apps are constantly being developed
Build from loosely coupled components
Newer version are deployed often
Deployed to a multitude of servers

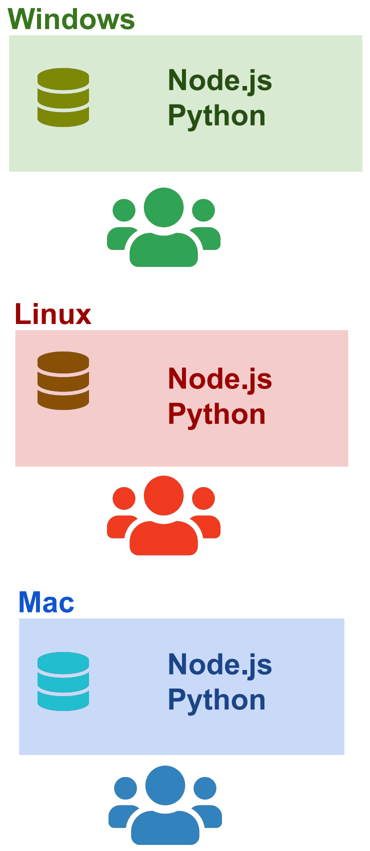
Data Science

Apps are being integrated with various data types/sources and models

Today: Microservice Architecture

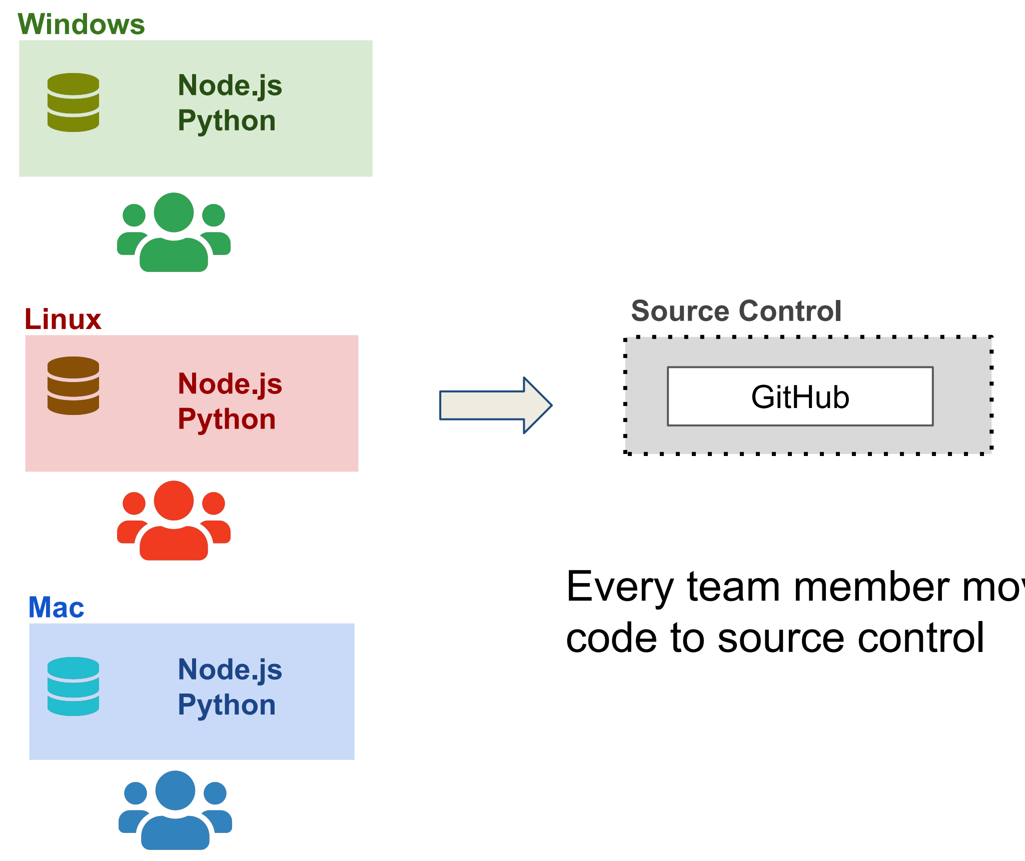


Software Development Workflow (no Docker)



OS Specific **installation** in every developer machine

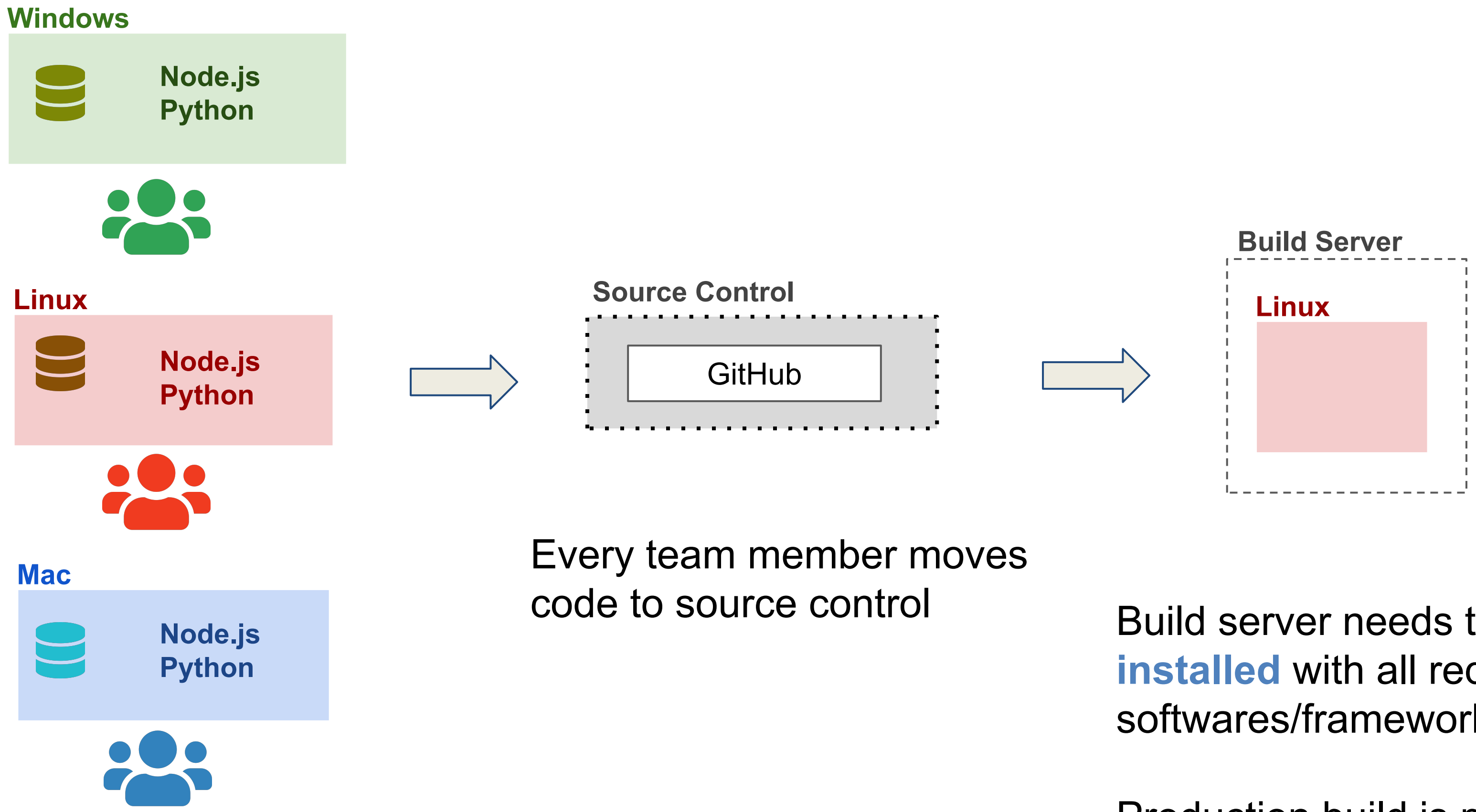
Software Development Workflow (no Docker)



Every team member moves code to source control

OS Specific **installation** in every developer machine

Software Development Workflow (no Docker)



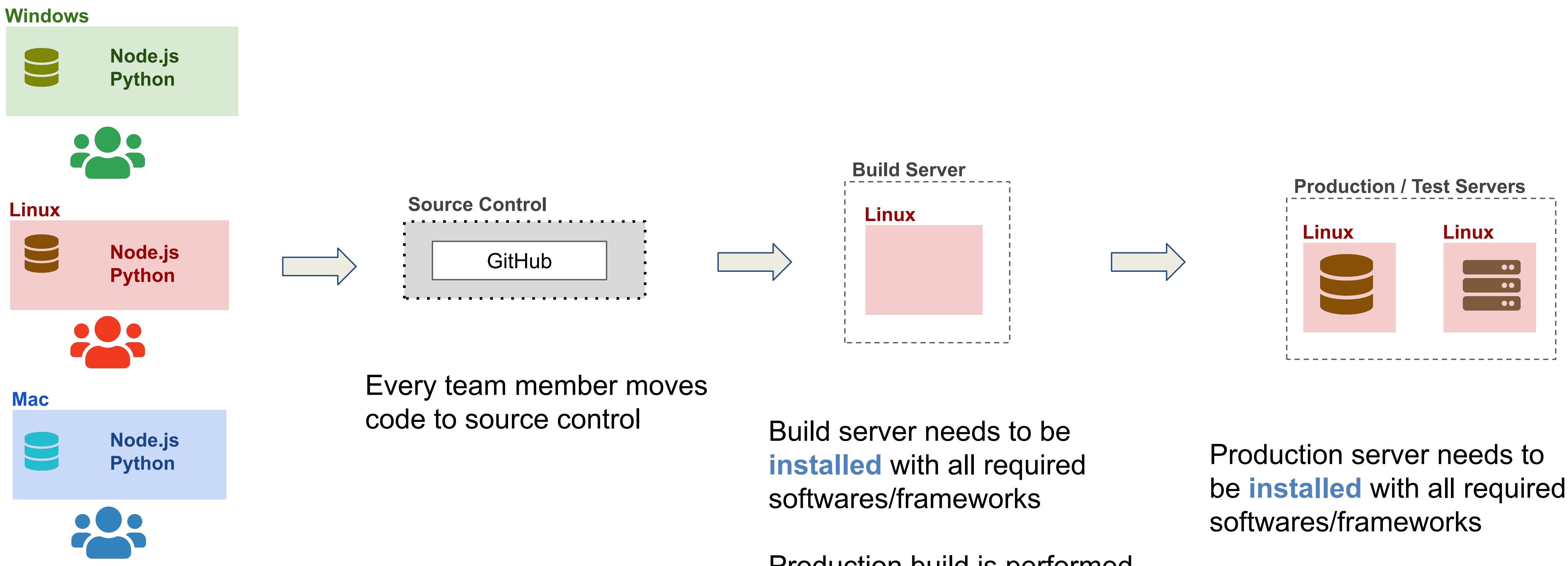
Every team member moves code to source control

Build server needs to be **installed** with all required softwares/frameworks

Production build is performed by pulling code from source control

OS Specific **installation** in every developer machine

Software Development Workflow (no Docker)



OS Specific **installation** in every developer machine

Every team member moves code to source control

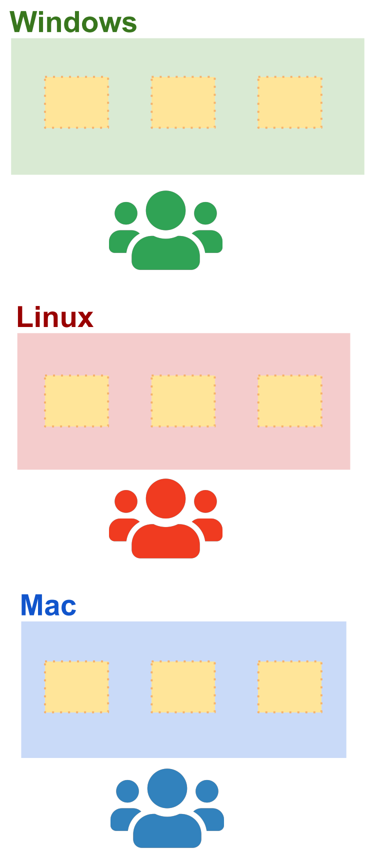
Build server needs to be **installed** with all required softwares/frameworks

Production build is performed by pulling code from source control

Production server needs to be **installed** with all required softwares/frameworks

Production server will be different OS version than development machines

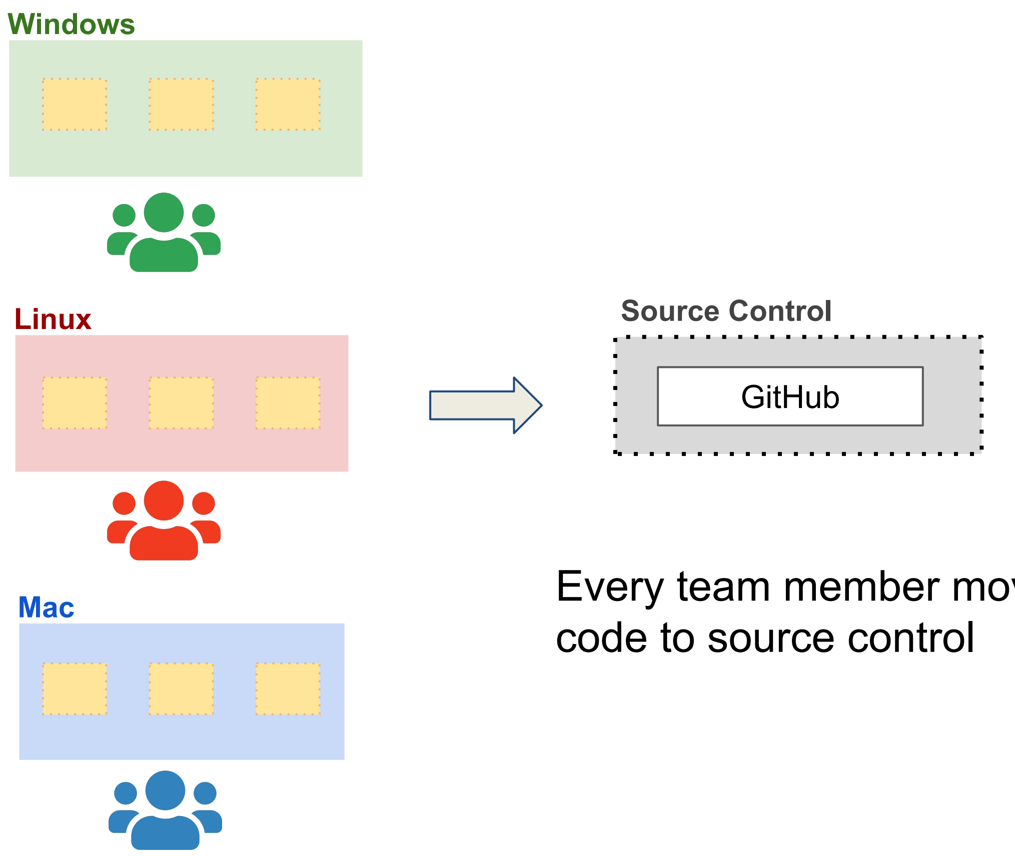
Software Development Workflow (with Docker)



Development machines only needs **Docker installed**

Containers need to be setup only once

Software Development Workflow (with Docker)

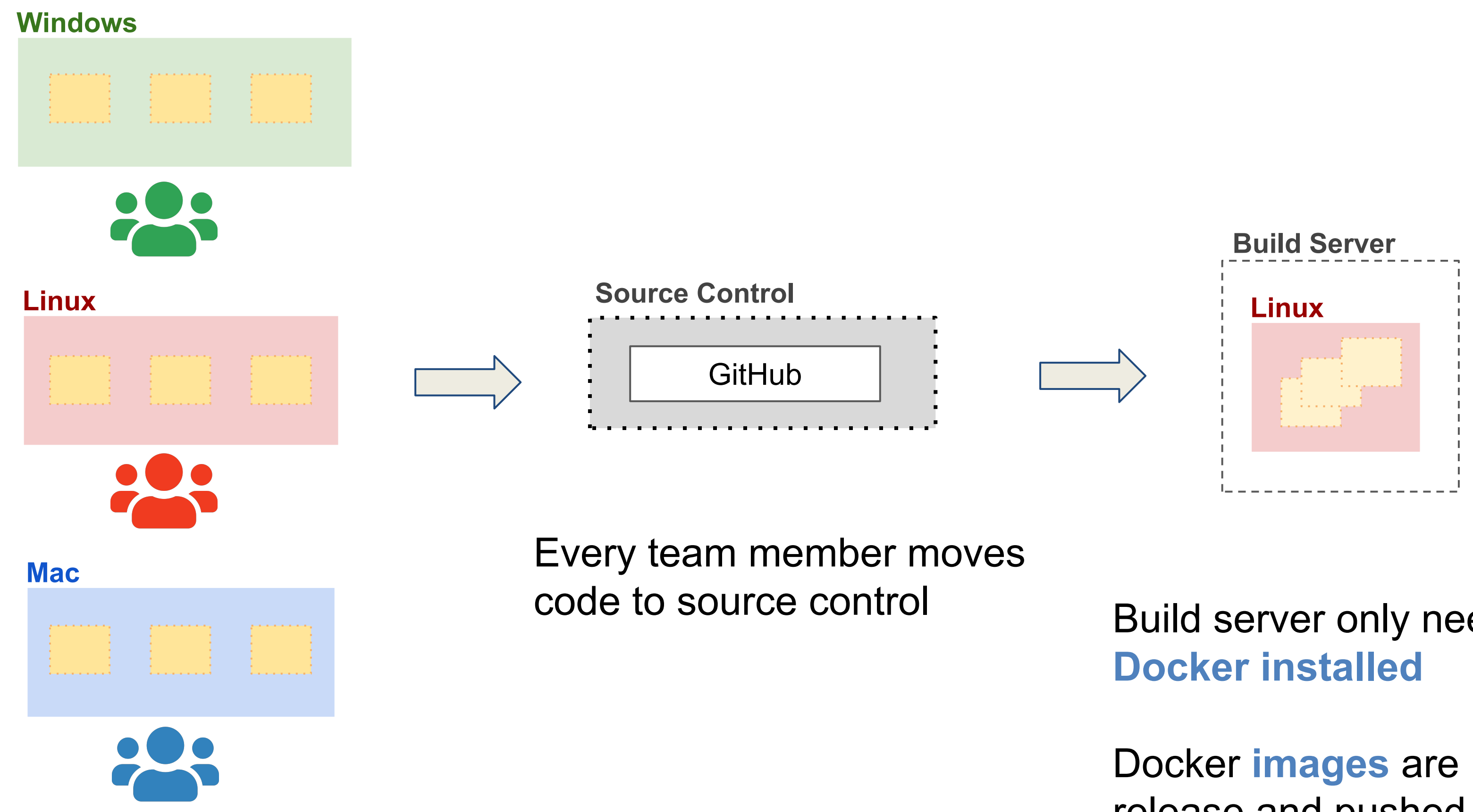


Every team member moves code to source control

Development machines only needs **Docker installed**

Containers need to be setup only once

Software Development Workflow (with Docker)



Every team member moves code to source control

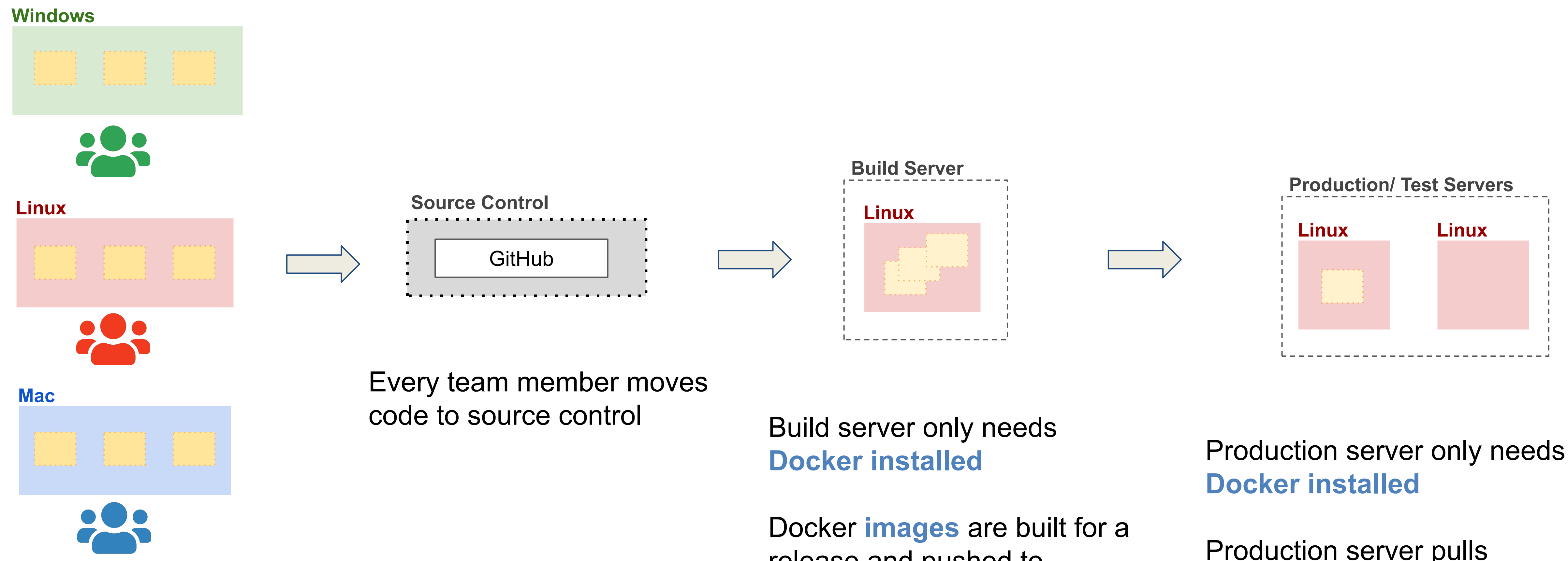
Development machines only needs **Docker installed**

Containers need to be setup only once

Build server only needs **Docker installed**

Docker **images** are built for a release and pushed to **container registry**

Software Development Workflow (with Docker)



Development machines only needs **Docker installed**

Containers need to be setup only once

Every team member moves code to source control

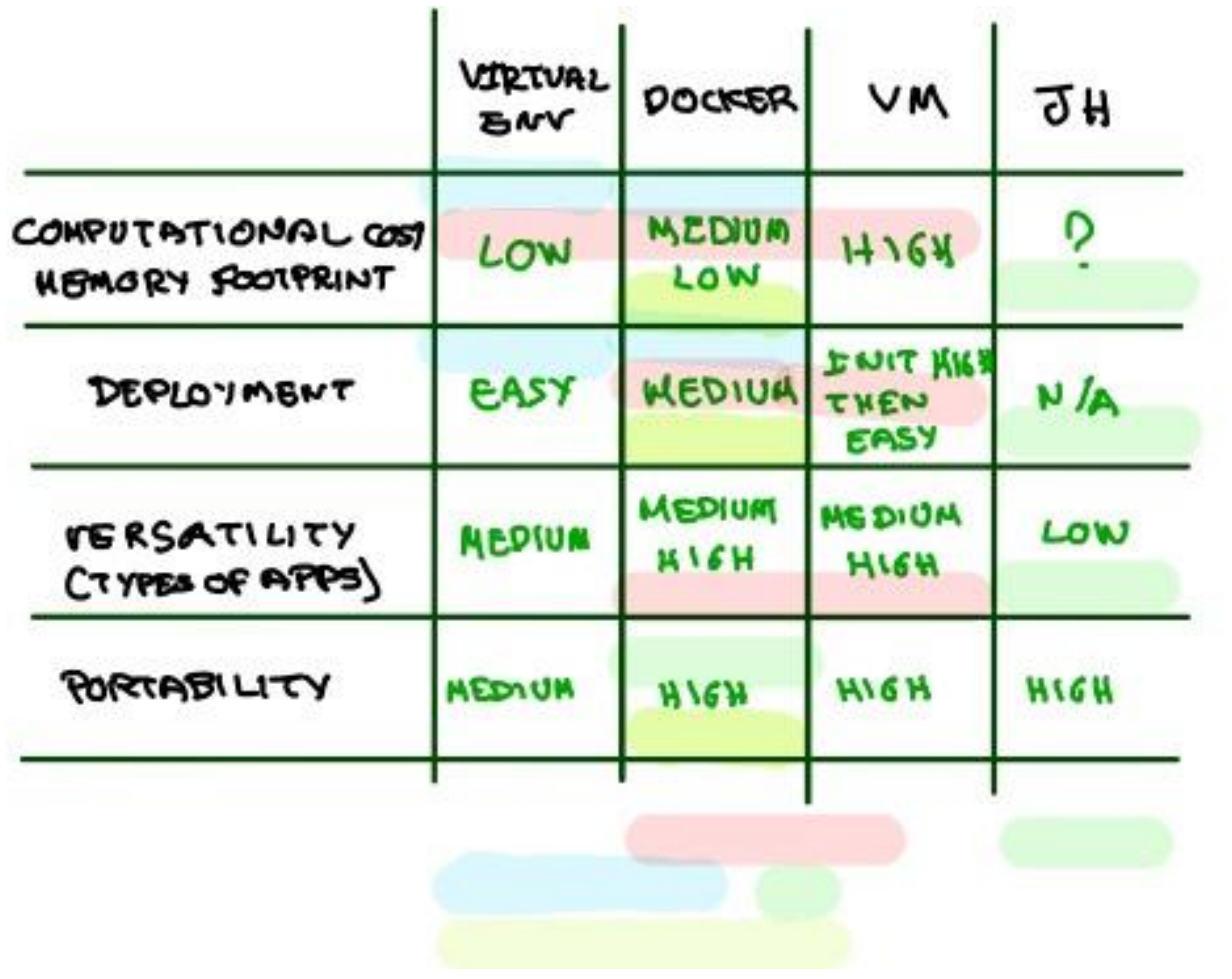
Build server only needs **Docker installed**

Docker **images** are built for a release and pushed to **container registry**

Production server only needs **Docker installed**

Production server pulls Docker **images** from **container registry** and runs them

Comparison



	VIRTUAL ENV	DOCKER	VM	JH
COMPUTATIONAL COST MEMORY FOOTPRINT	LOW	MEDIUM LOW	HIGH	?
DEPLOYMENT	EASY	MEDIUM	INIT HIGH THEN EASY	N/A
VERSATILITY (TYPES OF APPS)	MEDIUM	MEDIUM HIGH	MEDIUM HIGH	LOW
PORTABILITY	MEDIUM	HIGH	HIGH	HIGH

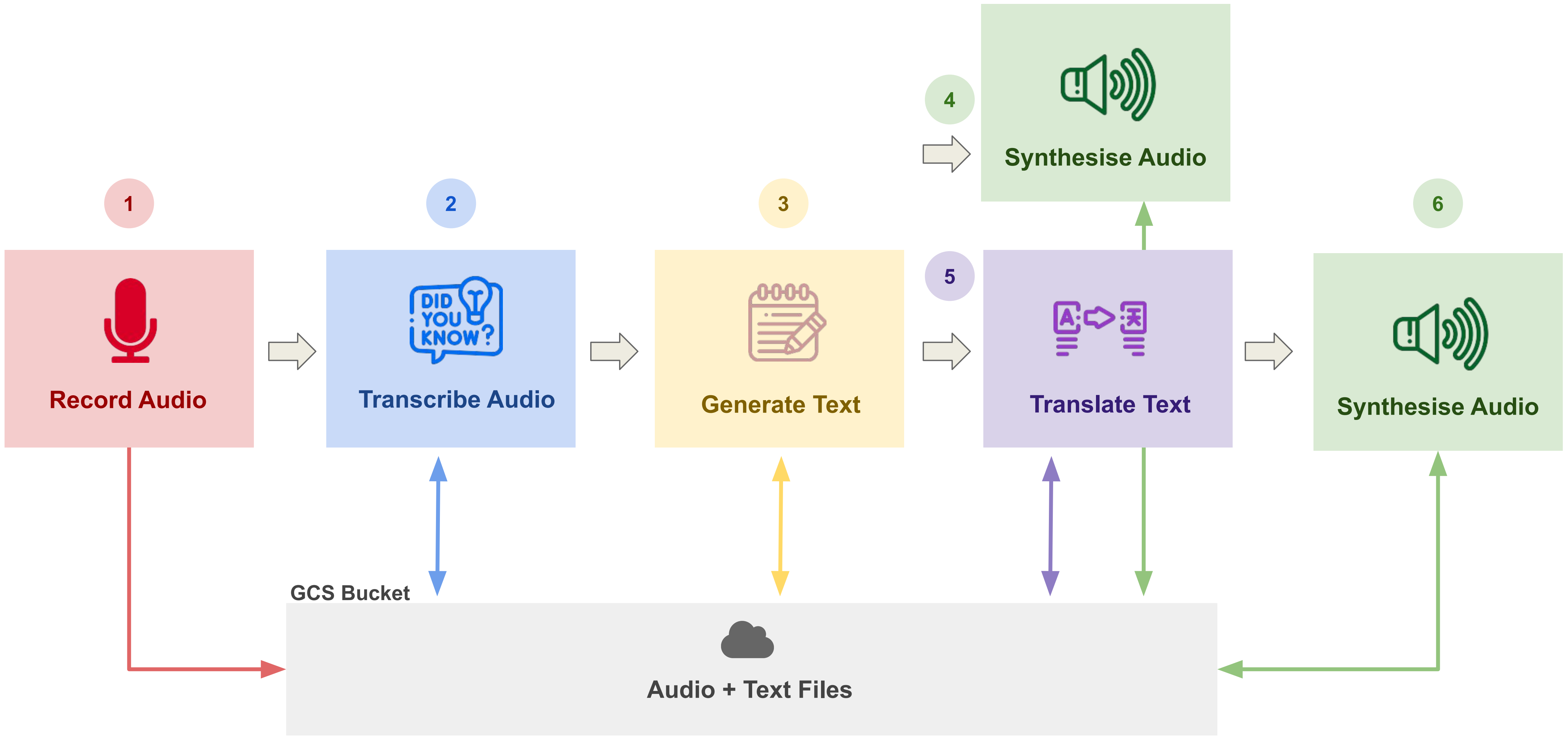
● COMPUTATIONAL SCIENCE

● DEVOPS

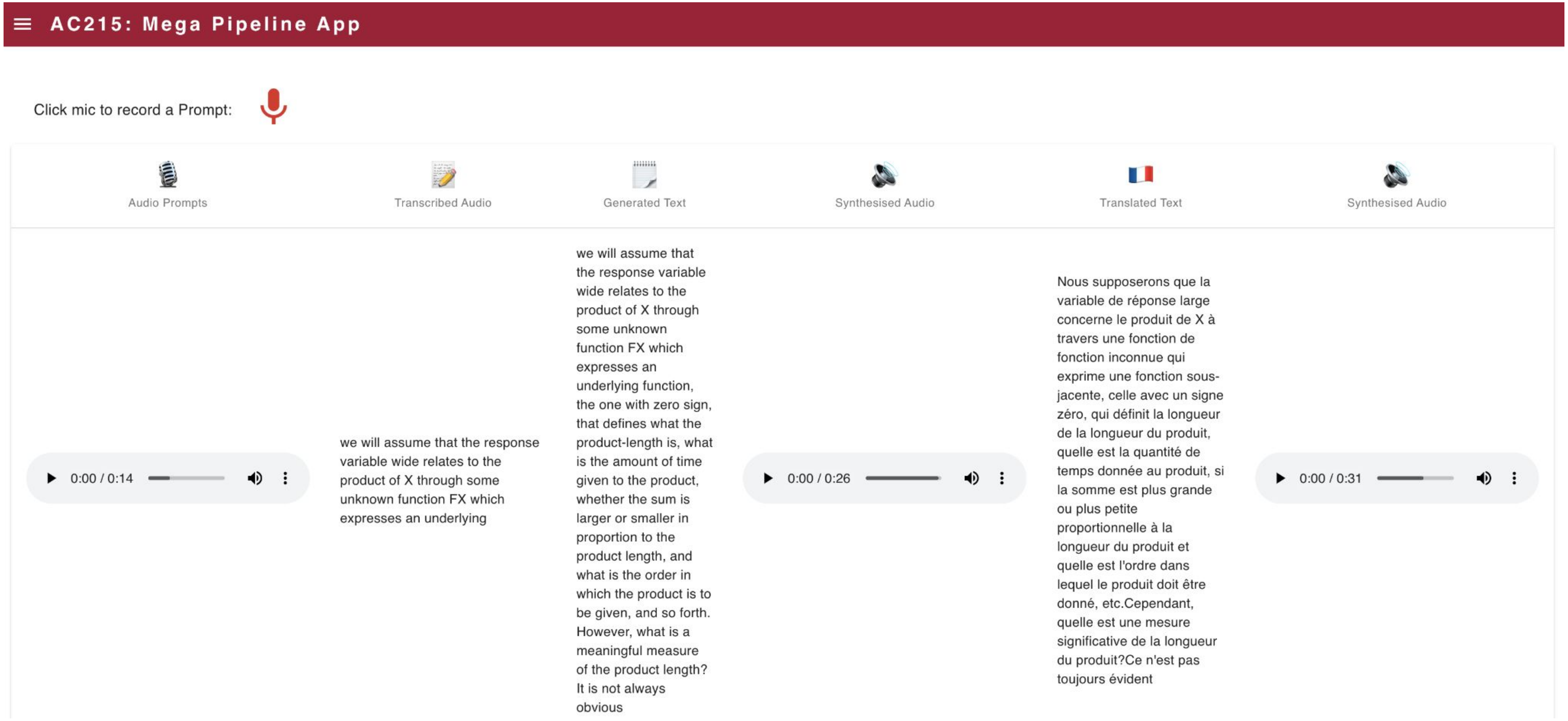
● DATA SCIENCE (NO PIPELINE)

● DATA SCIENCE (PIPELINES)

Tutorial - Building the Mega Pipeline App



Tutorial - Building the Mega Pipeline App



we will assume that the response variable wide relates to the product of X through some unknown function FX which expresses an underlying

we will assume that the response variable wide relates to the product of X through some unknown function FX which expresses an underlying function, the one with zero sign, that defines what the product-length is, what is the amount of time given to the product, whether the sum is larger or smaller in proportion to the product length, and what is the order in which the product is to be given, and so forth. However, what is a meaningful measure of the product length? It is not always obvious

Nous supposerons que la variable de réponse large concerne le produit de X à travers une fonction de fonction inconnue qui exprime une fonction sous-jacente, celle avec un signe zéro, qui définit la longueur de la longueur du produit, quelle est la quantité de temps donnée au produit, si la somme est plus grande ou plus petite proportionnelle à la longueur du produit et quelle est l'ordre dans lequel le produit doit être donné, etc.Cependant, quelle est une mesure significative de la longueur du produit?Ce n'est pas toujours évident

THANK YOU