Outline

Planned topics for this lesson:

- Introduction to Containerisation
  HOW TO DEPLOY?
- Why use Docker?
  A BETTER WAY TO PACK EVERYTHING TOGETHER
- Introduction to Docker
  BUILDING AND RUNNING APPS IN DOCKER CONTAINER

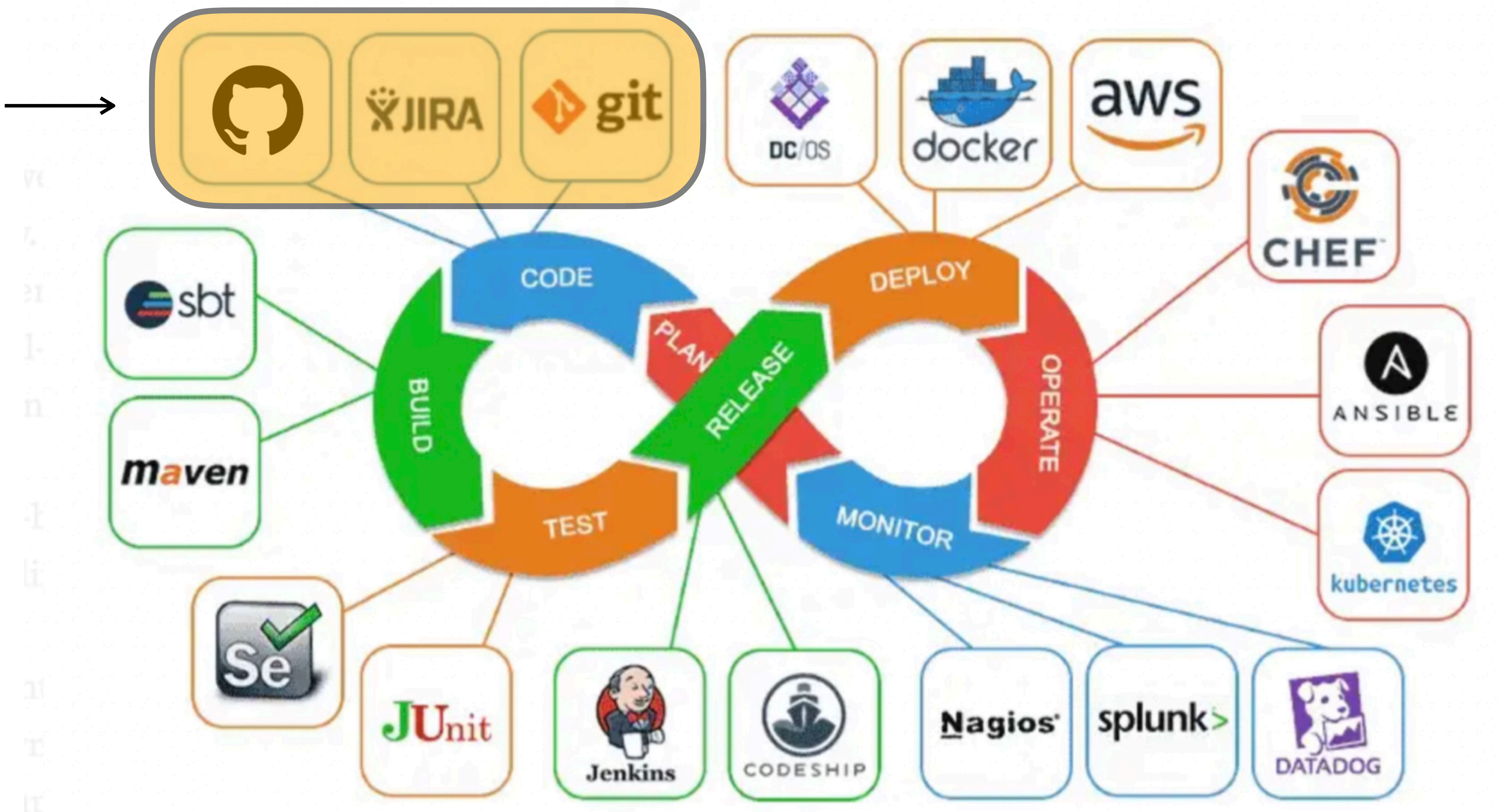
docker

Azure DevOps

CI/CD Pipeline

Example of a continuous software development system:

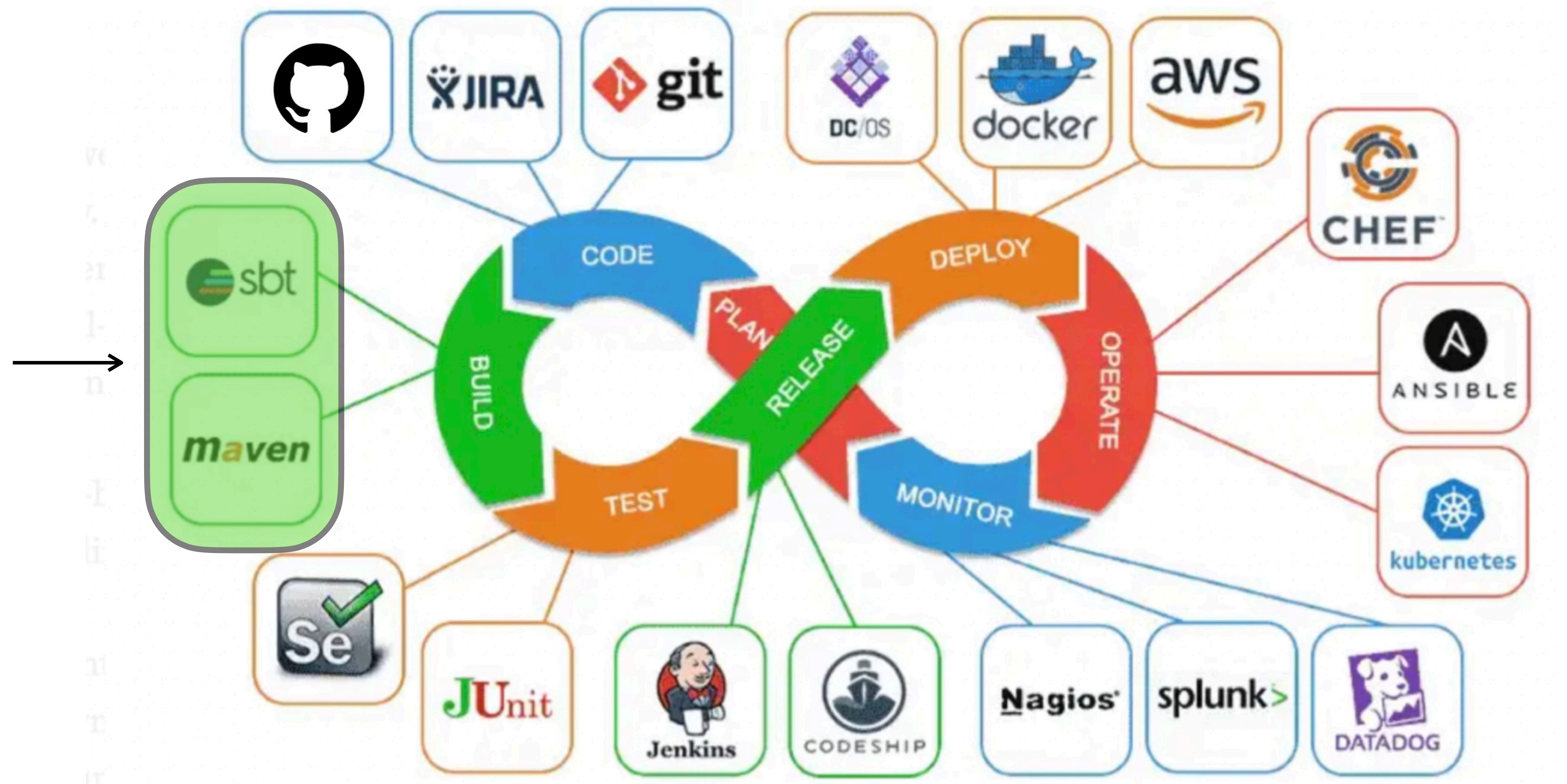
Week 1 : Version control →



CI/CD Pipeline

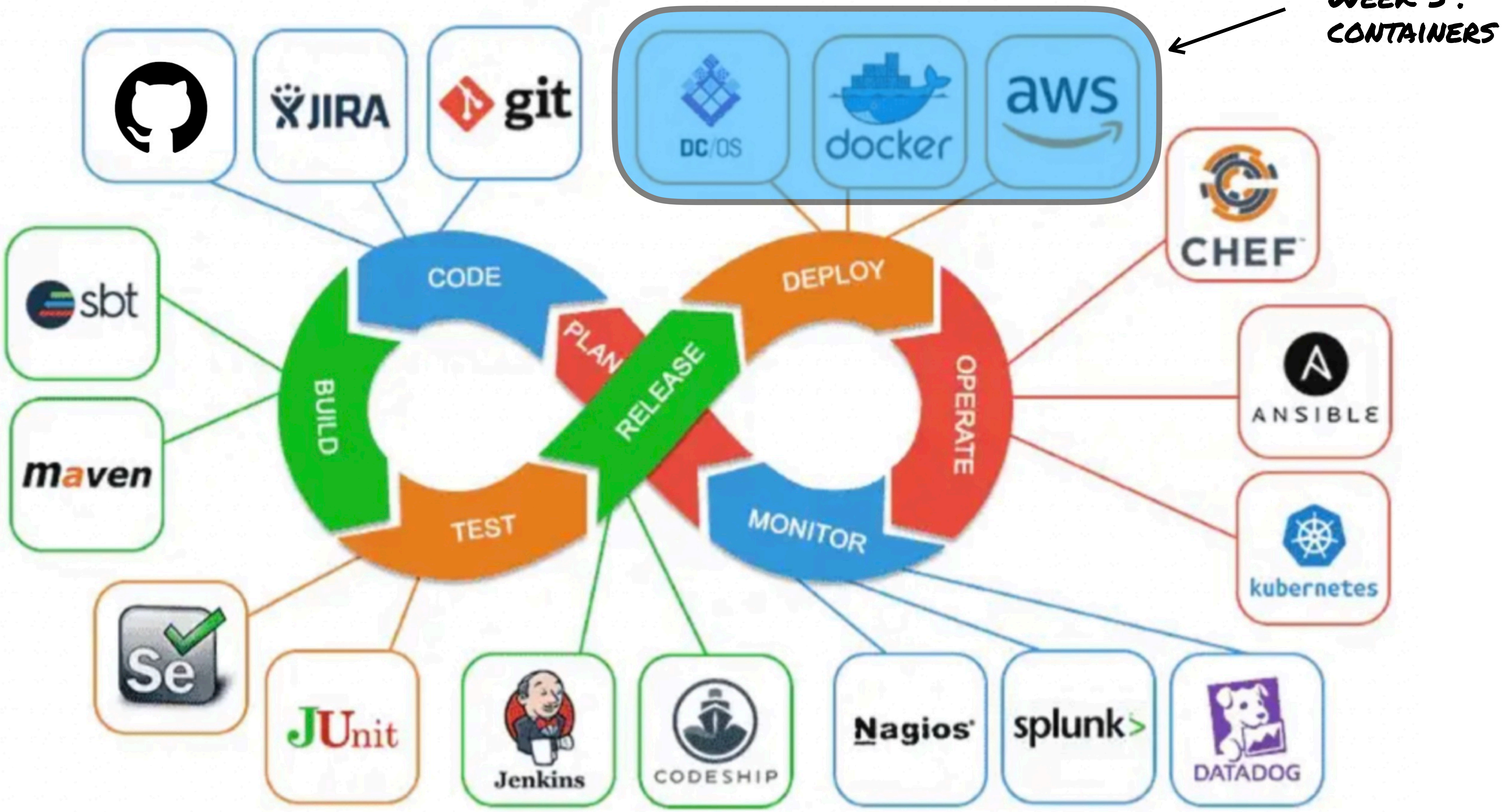
Example of a continuous software development system:

WEEK 2 :
BUILD TOOLS



CI/CD Pipeline

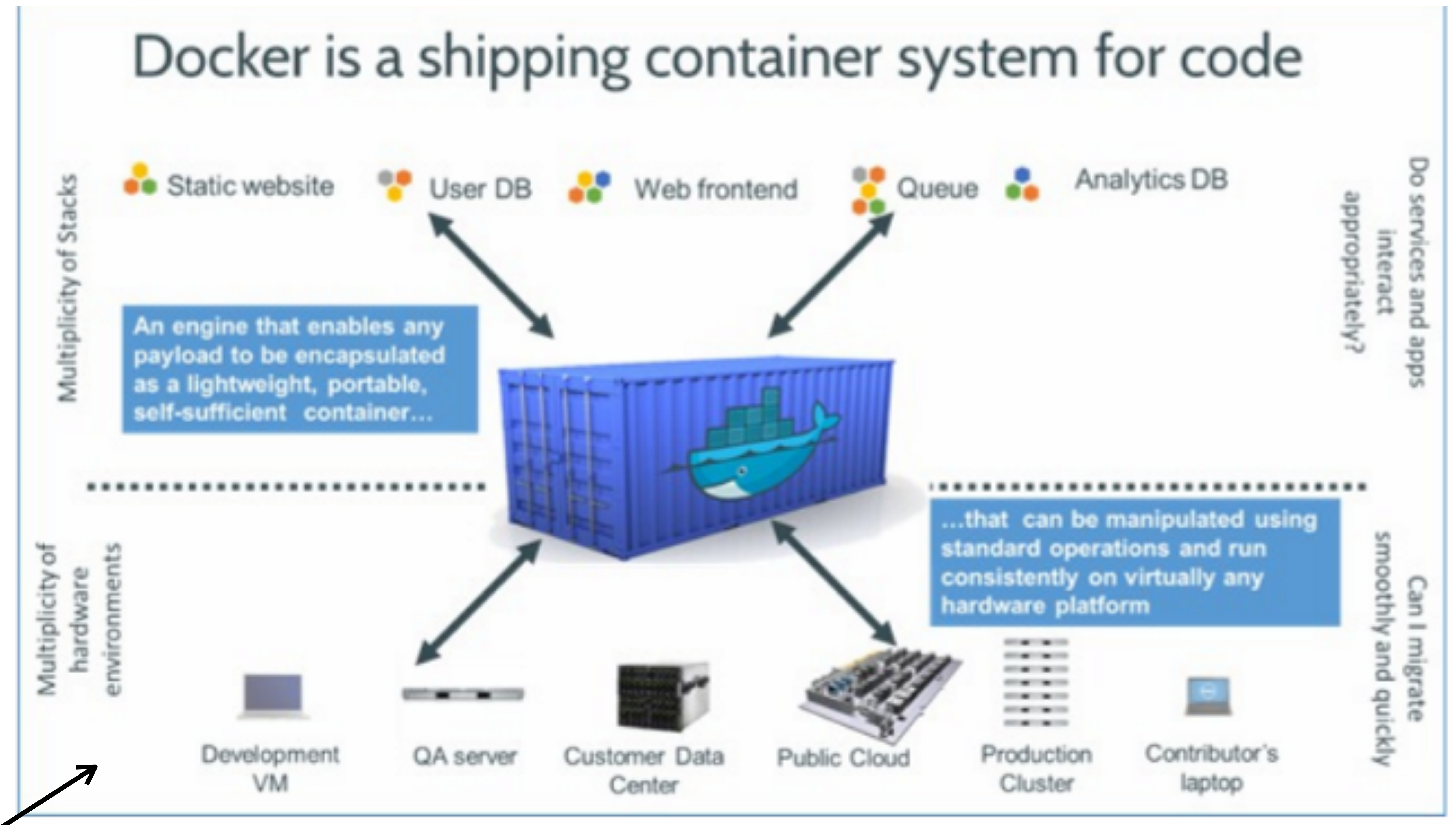
Example of a continuous software development system:



Containerisation

What is it?

- Containerisation is a method of packaging software code and all its dependencies so that it can run uniformly and consistently across any infrastructure.
- Containers are isolated environments in which applications run, ensuring consistent behaviour across different environments (e.g., development, testing, production).

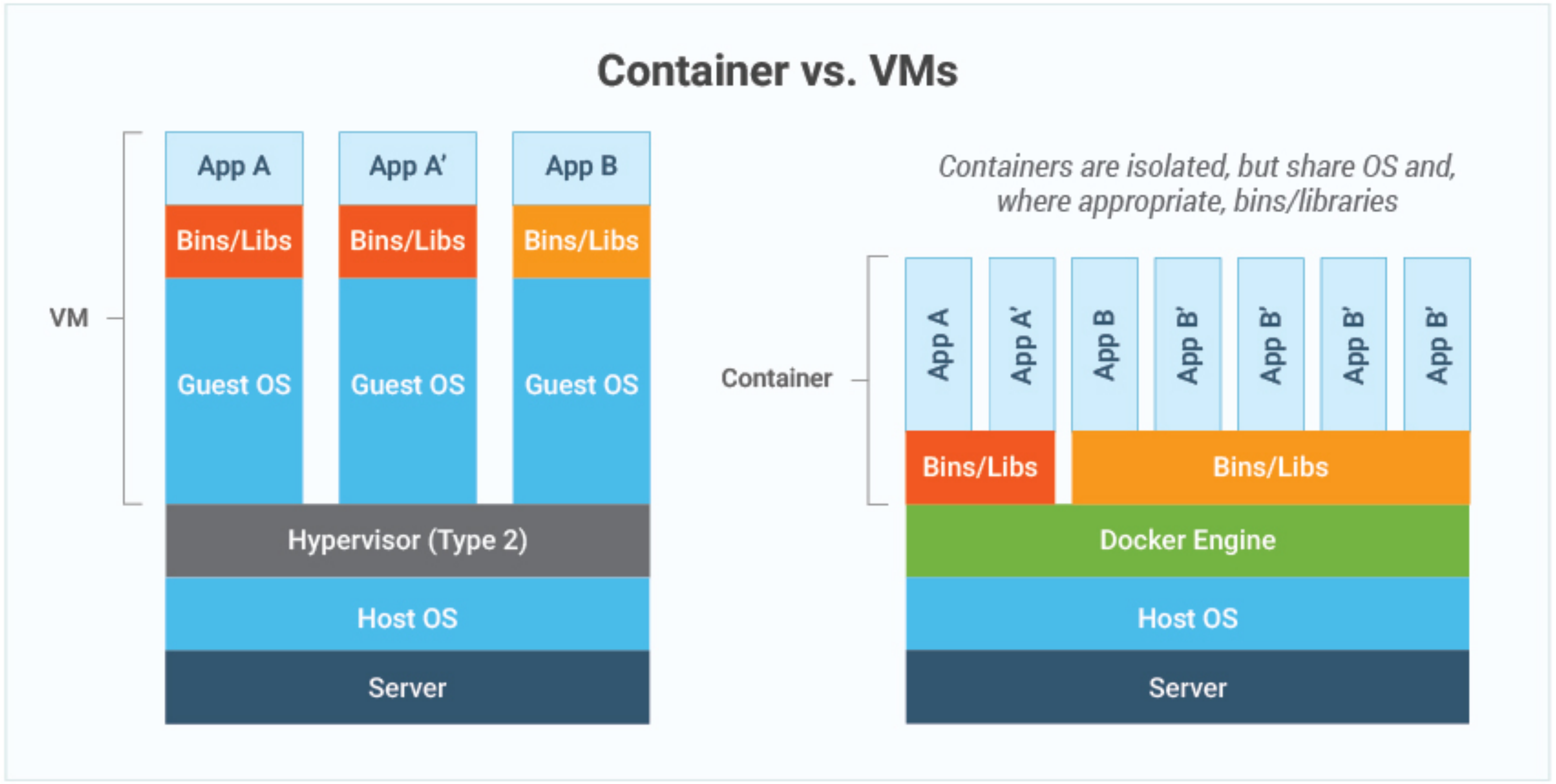

Docker is a shipping container system for code
Static website
User DB
Web frontend
Queue
Analytics DB
Multiplicity of Stacks
Do services and apps interact appropriately?
An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container...
...that can be manipulated using standard operations and run consistently on virtually any hardware platform
Multiplicity of hardware environments
Can I migrate smoothly and quickly
Development VM
QA server
Customer Data Center
Public Cloud
Production Cluster
Contributor's laptop

A Docker Container

CONTAINERS ARE LIKE SHIPPING CONTAINERS IN LOGISTICS—THEY ENCAPSULATE EVERYTHING NEEDED TO RUN A SERVICE, MAKING IT EASY TO TRANSPORT ACROSS VARIOUS PLATFORMS.

Containerisation

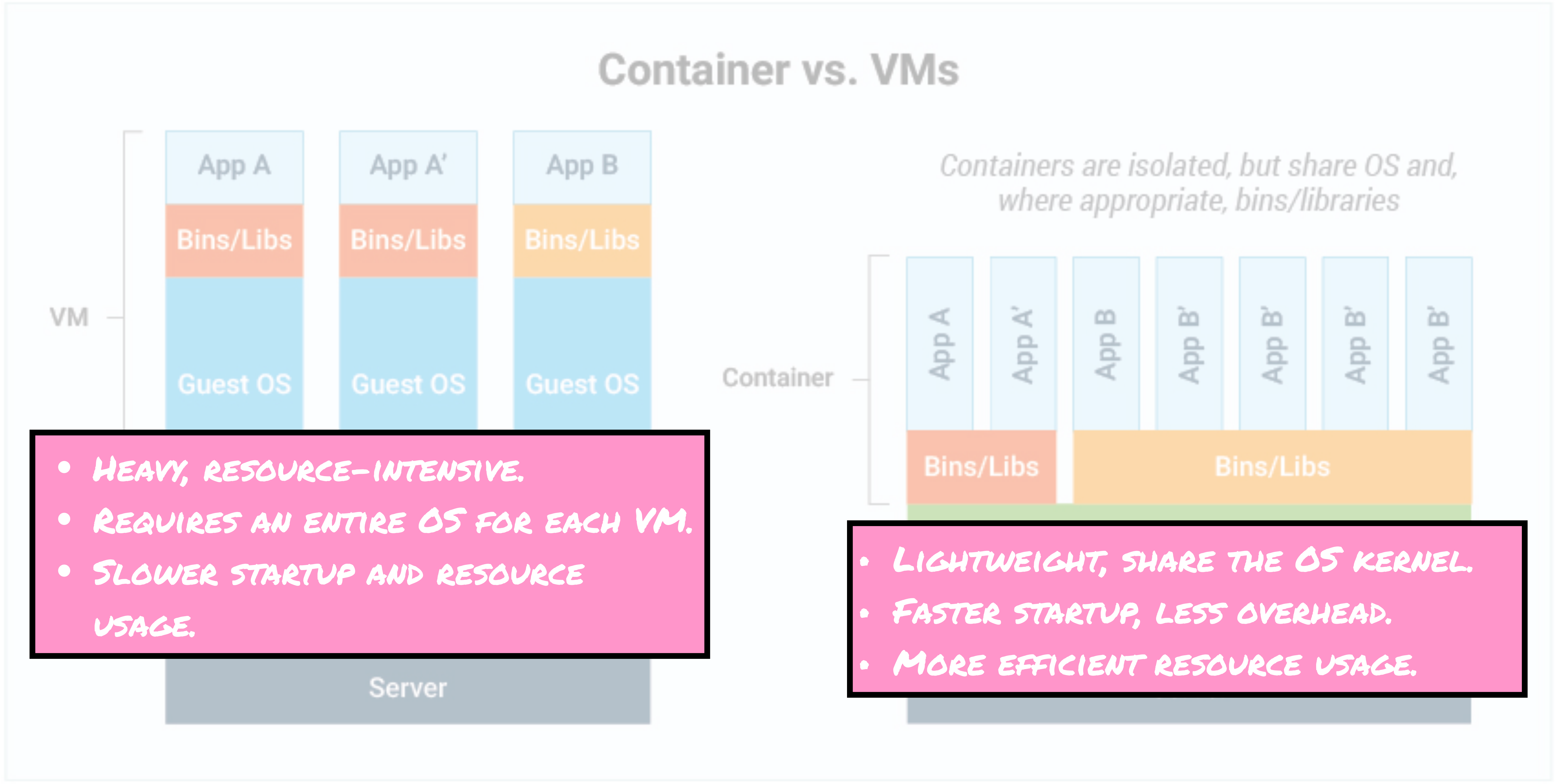
How do Containers Work?



- Containers virtualise the operating system (OS), unlike virtual machines (VMs), which virtualise hardware.
- Containers share the OS kernel but isolate the application and its dependencies.

Containerisation

How do Containers Work?



- Heavy, resource-intensive.
- Requires an entire OS for each VM.
- Slower startup and resource usage.

- Lightweight, share the OS kernel.
- Faster startup, less overhead.
- More efficient resource usage.

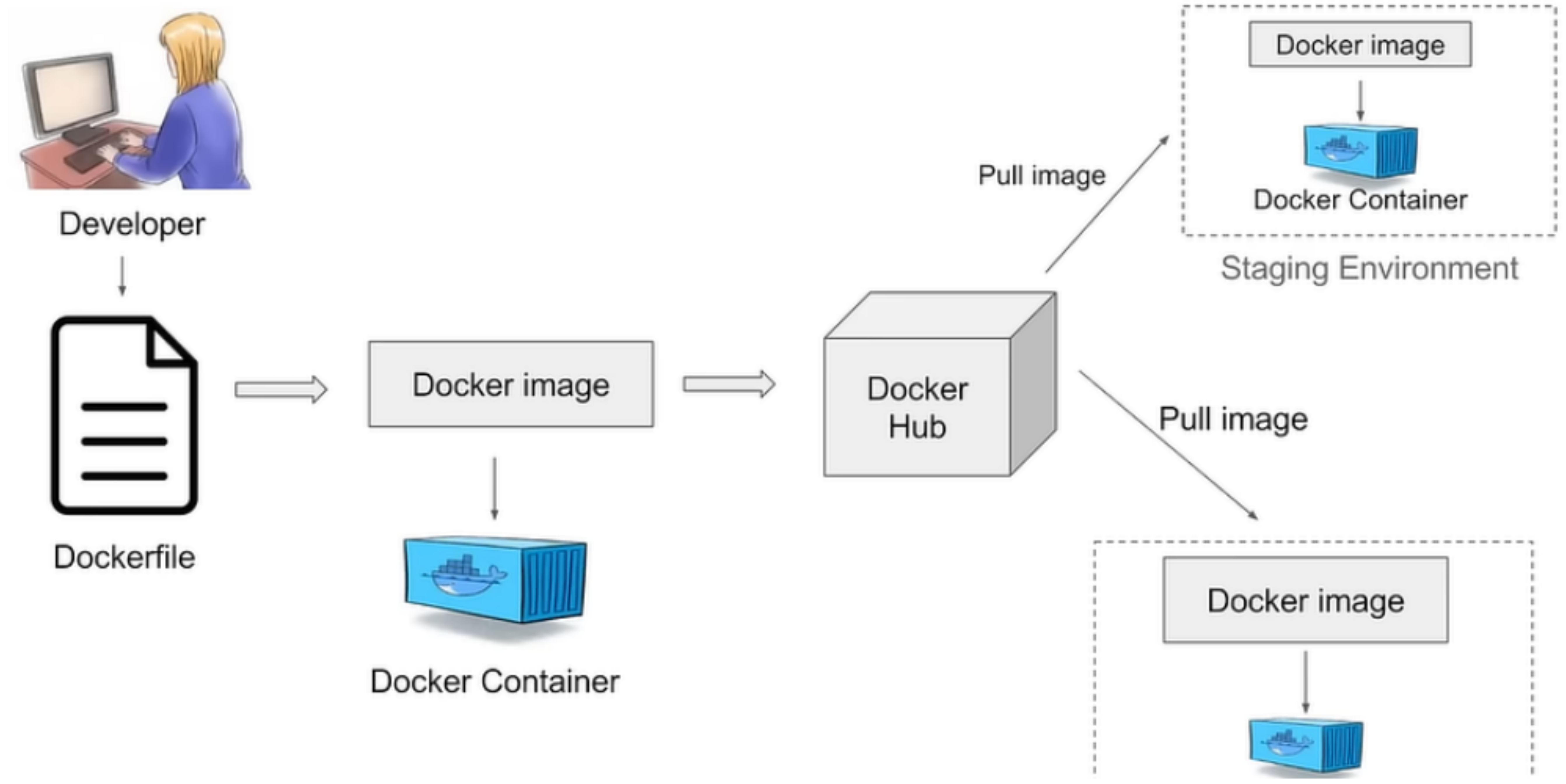
Containerisation

Why use Containerisation?

- **Portability**: Containers ensure that applications run the same regardless of the environment (`dev`, `test`, `prod`).
- **Scalability**: Containers can be scaled easily, making them ideal for microservices architecture.
- **Efficiency**: Containers are lightweight and use fewer resources than traditional VMs.
- **Isolation**: Each container is isolated, meaning multiple containers can run on the same host without interference.
- **Faster Deployment**: Containers can be started in seconds, enabling fast deployments and rollbacks.

Containerisation

What is a Docker?


Developer
Dockerfile
Docker image
Docker Container
Docker Hub
Pull image
Docker image
Docker Container
Staging Environment
Pull image
Docker image