Welcome to the CoGrammar Containerisation with Docker

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

Software Engineering Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Software Engineering Session Housekeeping cont.

- "Please check your spam folders for any important communication from us. If you have accidentally unsubscribed, please reach out to your support team.“
- Rationale here: Career Services, Support, etc will send emails that contain NB information as we gear up towards the end of the programme. Students may miss job interview opportunities, etc.

Software Engineering Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Skills Bootcamp
8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✅ Criterion 1: Initial Requirements

- ***Timeframe:*** First 2 Weeks
- ***Guided Learning Hours (GLH):*** Minimum of 15 hours
- ***Task Completion:*** First four tasks

✅ Criterion 2: Mid-Course Progress

- ***Guided Learning Hours (GLH): 60***
- ***Task Completion:*** 13 tasks

CoGrammar

Skills Bootcamp Progression Overview

✅ Criterion 3: Course Progress

- ***Completion:*** All mandatory tasks, including Build Your Brand and resubmissions by study period end
- ***Interview Invitation:*** Within 4 weeks post-course
- ***Guided Learning Hours:*** Minimum of 112 hours by support end date (10.5 hours average, each week)

✅ Criterion 4: Demonstrating Employability

- ***Final Job or Apprenticeship Outcome:*** Document within 12 weeks post-graduation
- ***Relevance:*** Progression to employment or related opportunity

CoGrammar


SKILLS FOR LIFE
SKILLS BOOTCAMPS
Department for Education
CoGrammar
Containerisation with Docker
June 2024

Learning Objectives

- Describe the concept of **containerisation** and its **benefits**.
- Define Key **Docker Components**
- Execute Basic **Docker Commands**
- Create and Use a **Dockerfile**
- **Build** and **Run** a Docker **Container**

Poll

1. What is a common use of APIs?

a. To create physical connections between hardware devices

b. To enable communication between different software systems or applications

c. To design user interfaces for applications

Poll

2. What does the term "endpoint" refer to in an API?

a. The final version of the API documentation

b. A programming function used to terminate an API session

c. A specific URL where an API service is accessible

Poll

3. What is a "payload" in the context of APIs?

a. The amount of time an API request takes to complete

b. The size limit for API requests:

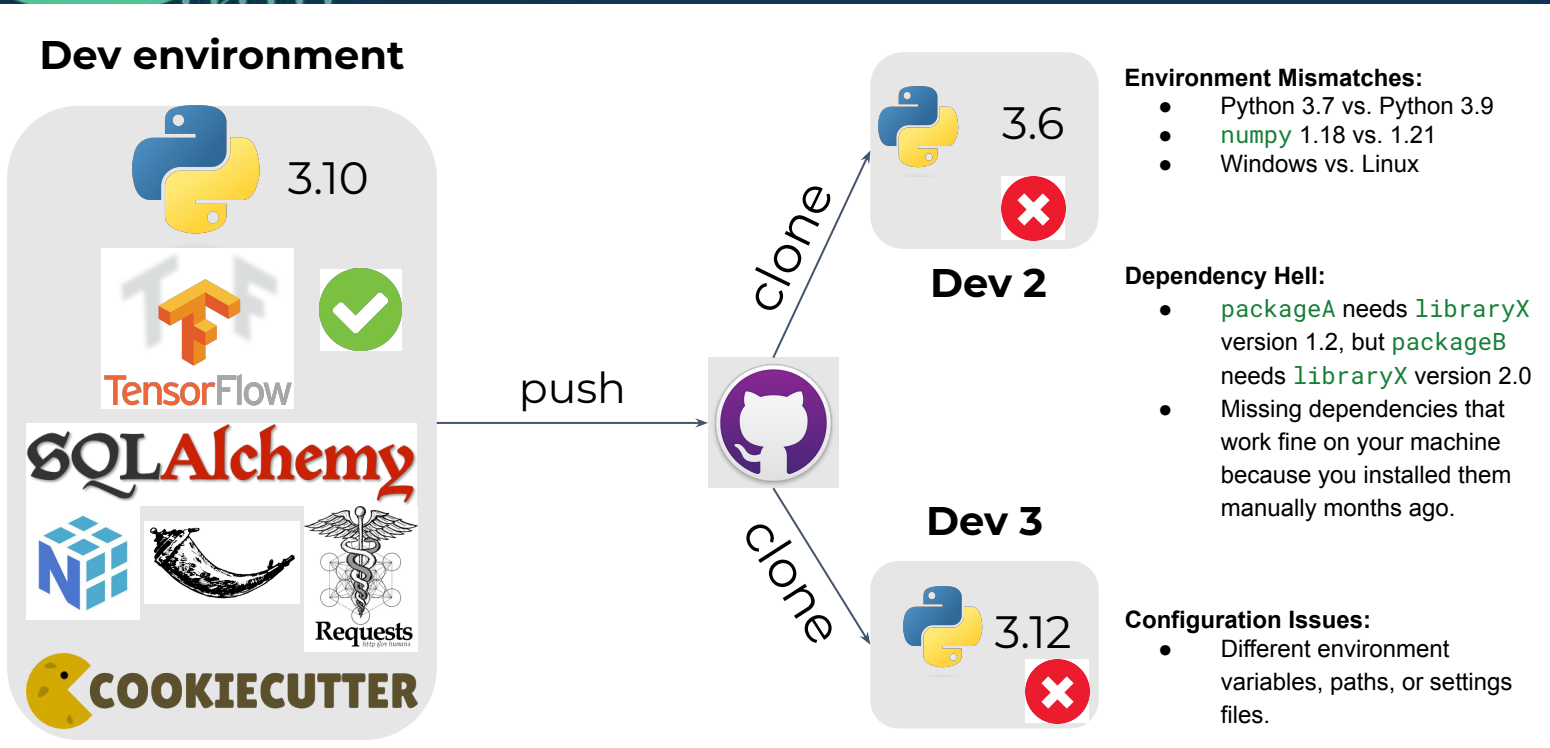
c. The data sent by the client to the server in an API request

Introduction

The "Works on My Machine" Problem

CoGrammar

The "Works on My Machine" Problem



Environment Mismatches:

- Python 3.7 vs. Python 3.9
- `numpy` 1.18 vs. 1.21
- Windows vs. Linux

Dependency Hell:

- `packageA` needs `libraryX` version 1.2, but `packageB` needs `libraryX` version 2.0
- Missing dependencies that work fine on your machine because you installed them manually months ago.

Configuration Issues:

- Different environment variables, paths, or settings files.

Traditional Solutions

1 - **Detailed setup instructions**

Which are often long and error-prone.

- Specify requirements.
- Step-by-step installation.
- Configure environment.
- Platform-specific notes.
- Troubleshooting guide.

2 - **Virtual machines**

Heavy, slow to start, and resource-intensive.

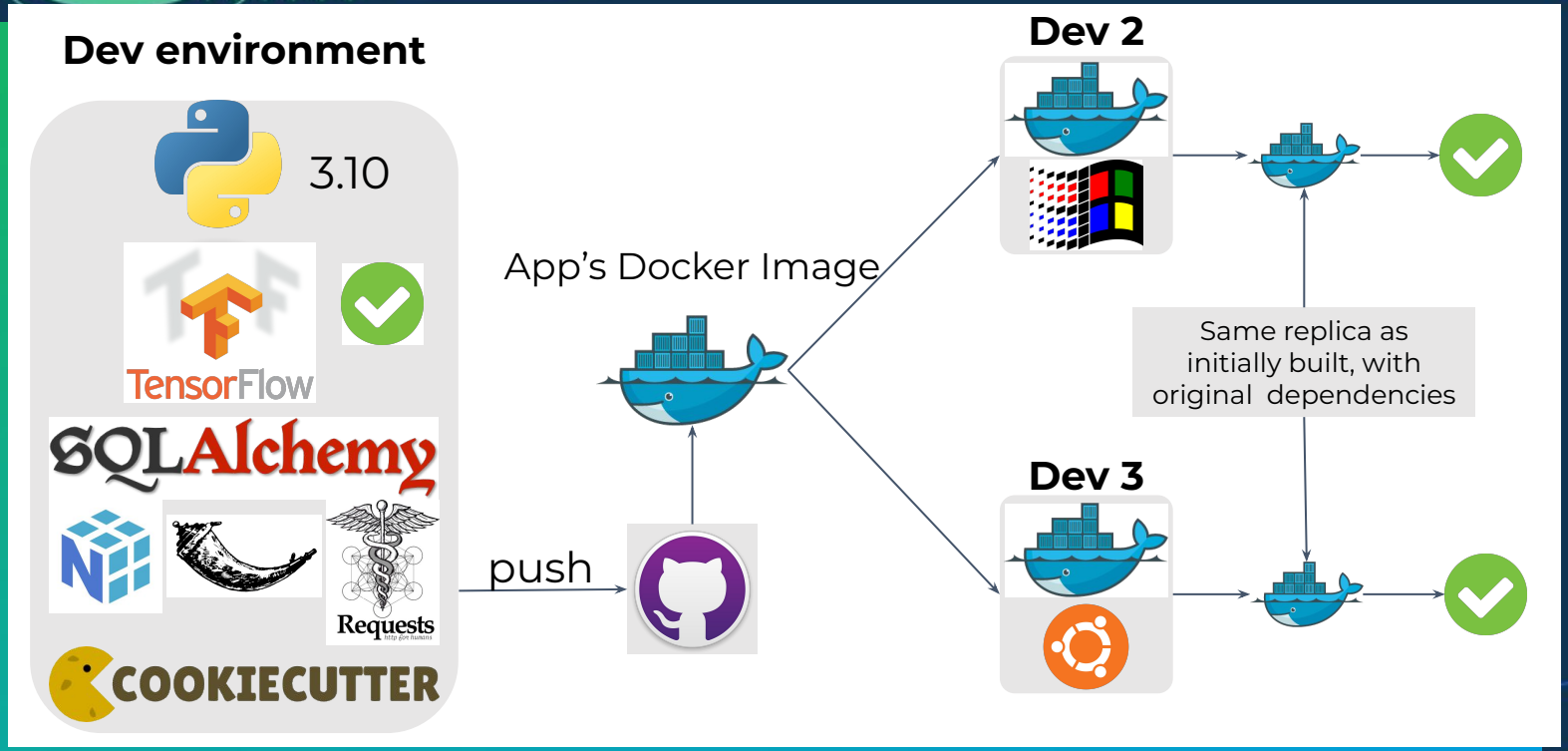
- Select VM software.
- Configure VM.
- Create snapshot/export.
- Share VM image.
- Run and test in VM.

3 - **Conda or virtual environments**

Good for Python, but not always for other tools or languages.

- Install Conda/Virtualenv.
- Create isolated environment.
- List dependencies.
- Activate environment.
- Sync and verify.

Docker


Dev environment
3.10
TensorFlow
SQLAlchemy
Requests
COOKIECUTTER
App's Docker Image
push
Dev 2
Dev 3
Same replica as initially built, with original dependencies

Docker

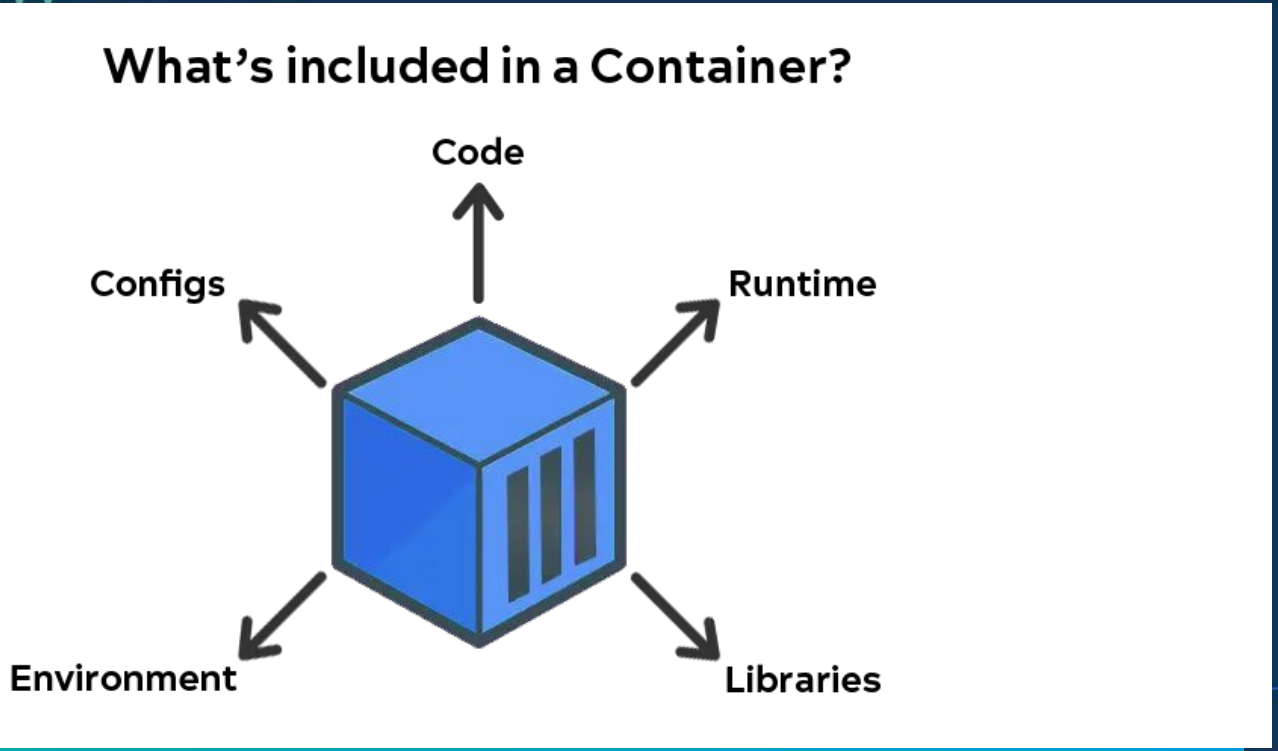
CoGrammar

Introduction to Modern Application Deployment

What is containerisation?

Containerisation 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**. It is also referred as **lightweight virtualization.**

Introduction to Modern Application Deployment

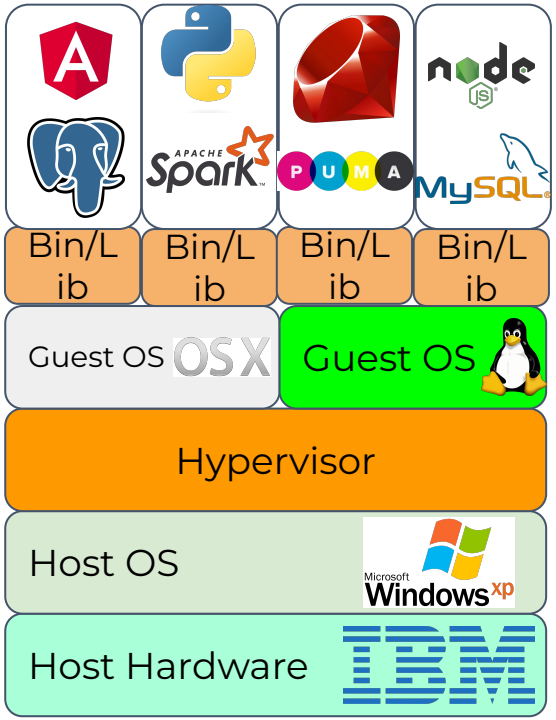


https://blog.back4app.com/what-are-containers-in-cloud-computing/

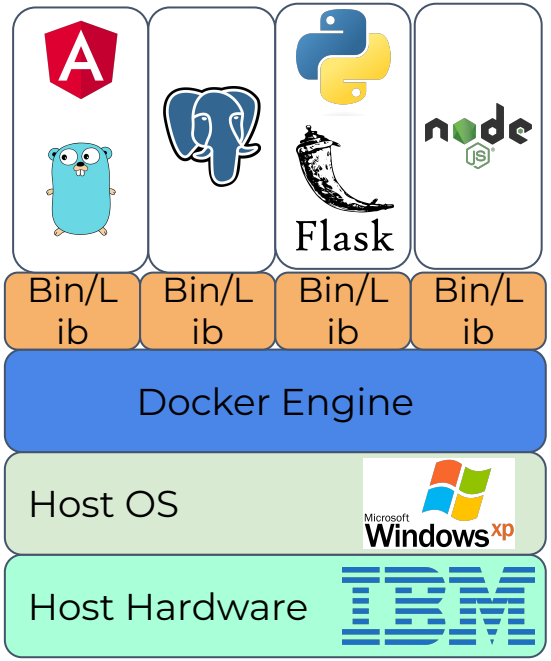
Introduction to Modern Application Deployment

Aspect	Containers	Virtual Machines
Resource Usage	Lightweight, efficient	Heavier, more resource usage
Startup Time	Quick start	Slower start
Isolation	Process-level separation	Full OS isolation
Portability	Highly portable	Compatibility concerns
Resource Overhead	Minimal overhead	Higher overhead
Isolation Level	Lighter isolation	Stronger isolation

Introduction to Modern Application Deployment



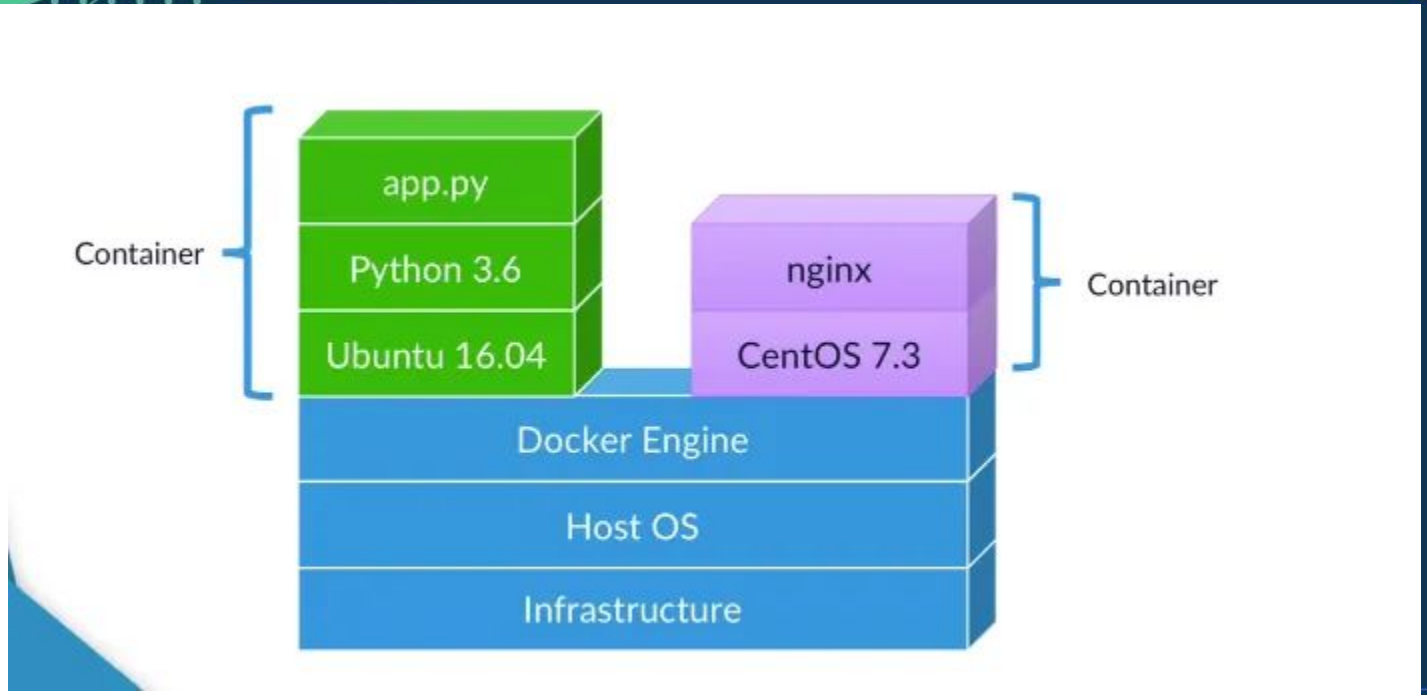
Virtual Machine



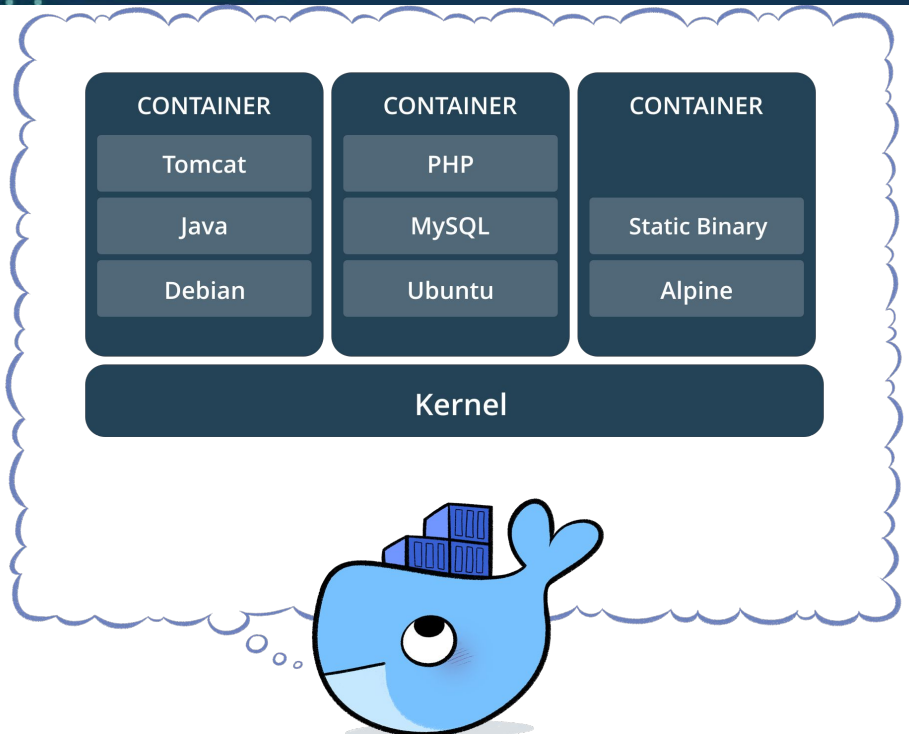
Docker Container

CoGrammar

Introduction to Modern Application Deployment



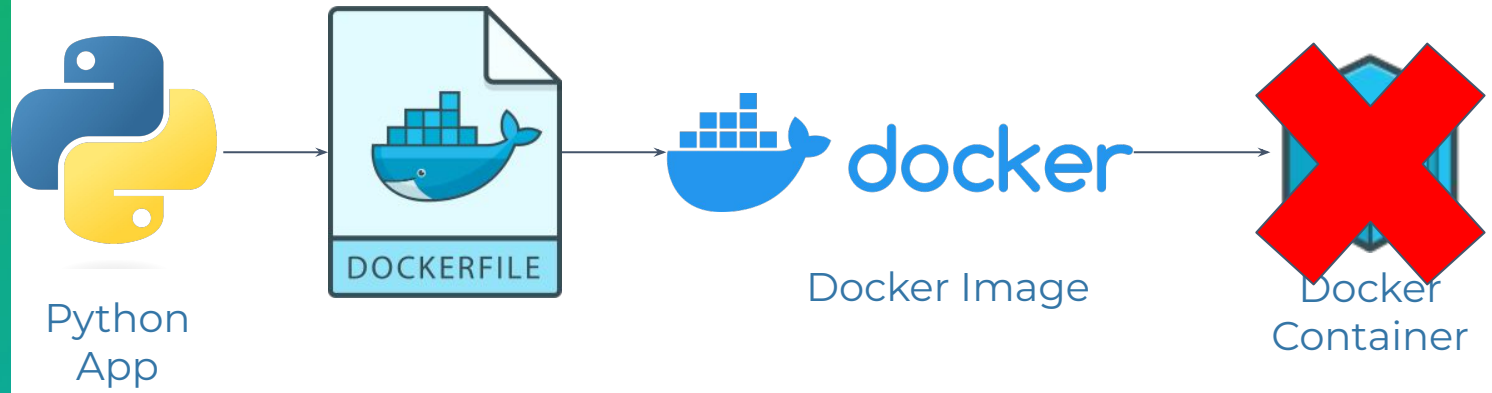
Introduction to Modern Application Deployment



CoGrammar

https://i.sstatic.net/vIBgw.png

Docker Container Lifecycle



Docker Benefits

- **Consistency**: Ensures applications run the same way everywhere.
- **Isolation**: Containers are isolated from each other and the host system.
- **Efficiency**: Containers are lightweight and share the host kernel, making them resource-efficient.

Docker Core Concepts

CoGrammar

Docker Core Concepts

- **Images**: Blueprints for containers, containing application code, libraries, and dependencies.
- **Containers**: Instances of images, isolated environments running applications.
- **Dockerfile**: A text file with instructions to build images (**FROM**, **RUN**, **EXPOSE**, **CMD**). It specifies:
  - **Base image**: The starting point for the image.
  - **Instructions**: Commands to install dependencies, copy files, and configure the environment.

Docker Core Concepts

```
1   FROM python:3.9-slim
2
3   WORKDIR /app
4
5   COPY . /app
6
7   RUN if [ -f requirements.txt ]; then pip install --no-cache-dir -r requirements.txt; fi
8
9   EXPOSE 80
10
11  CMD ["python", "app.py"]
```

Docker Core Concepts

- **Docker daemon**: A background service that manages Docker objects (images, containers, networks, volumes).
- **Docker Hub**: A public registry for sharing and discovering Docker images.

Basic Docker Commands

CoGrammar

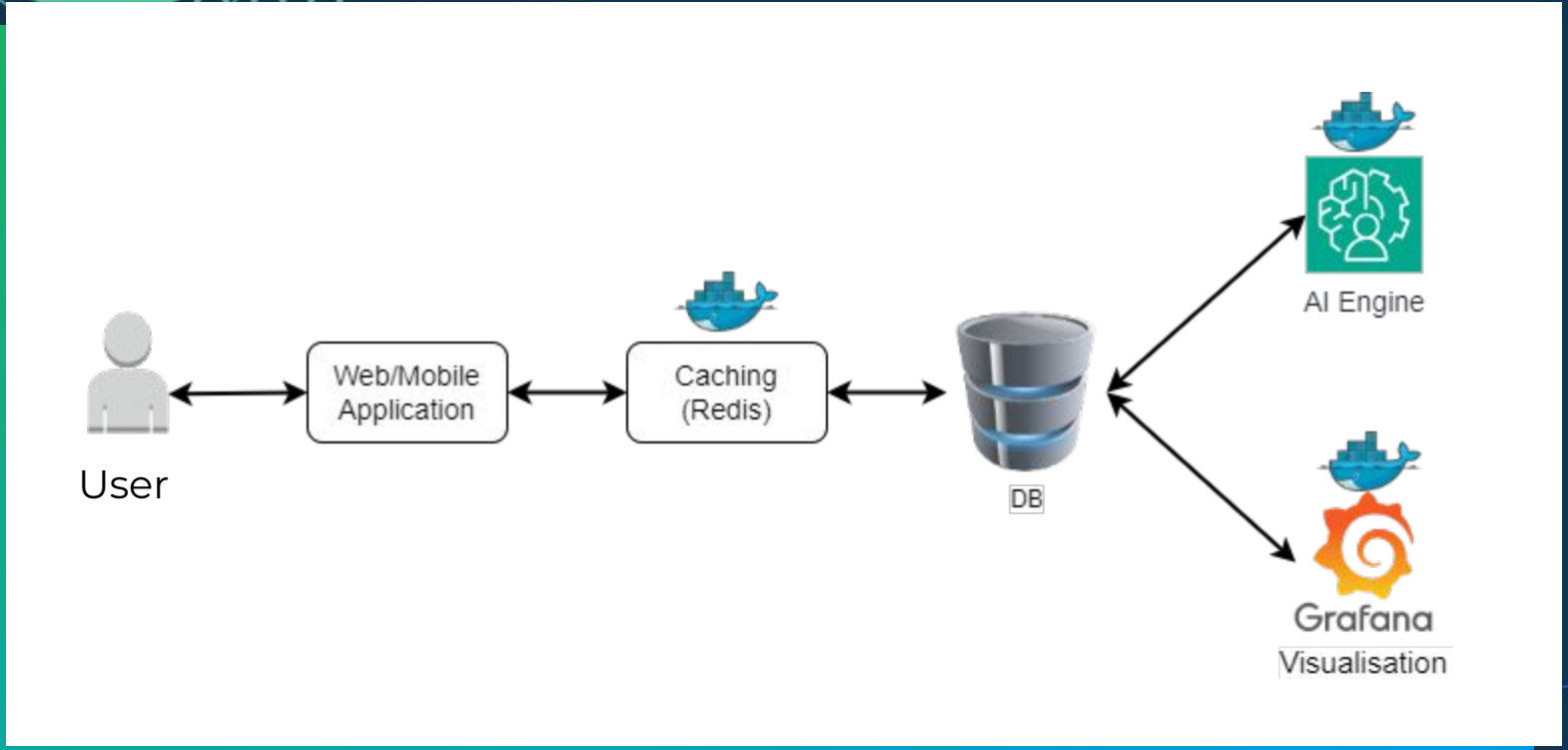
Docker Core Concepts

- **docker images**: List docker images
- **docker ps**: Lists running containers
- **docker build**: Builds an image from a Dockerfile
- **docker pull**: Downloads images from public registries (e.g., Docker Hub)
- **docker run**: Creates and starts a container from an image.

Docker Core Concepts

- **docker run**: Creates and starts a container from an image.
- **docker stop**: Stops a running container
- **docker exec**: Run commands inside a container
- **docker rm**: Removes stopped containers.
- **docker logs**: View logs generated by a container.

Docker in Real Life Scenarios



Docker in Real Life Scenarios

- **Docker in Real Life Scenarios**
  - Simplifying development and testing environments
  - Running microservices.
  - Deployment in continuous integration/continuous deployment (CI/CD) pipelines.
- **Example Applications**
  - Python web applications (e.g., Flask, Django).
  - Database management.
  - CI/CD

Let’s take a break

CoGrammar

Let’s get coding

CoGrammar

Poll

1. What is Docker primarily used for?

a. Running virtual machines on physical servers

b. Managing hardware resources directly

c. Packaging applications and their dependencies into containers

Poll

2. How does Docker ensure that an application runs the same way across different environments?

a. By requiring that all systems have identical hardware
b. By packaging the application and its dependencies into a container
c. By changing the underlying operating system of the host

Poll

3. What is a Docker image?

a. A snapshot of a running Docker container

b. A read-only template used to create Docker containers

c. A graphical representation of Docker commands

Summary

- **Containerization:** We learned about **containerization** as a way to **package applications** with their **dependencies** into lightweight, **portable** units.
- **Docker Concepts:** We explored core Docker concepts like **Images** (blueprints), **Containers** (running instances), and **Dockerfiles** (build instructions).
- **Basic Docker Commands:** We practised using essential commands like **docker pull**, **docker run**, **docker ps**, and **docker stop** to manage containers.
- **Dockerizing Python Apps:** We experienced creating a Dockerfile to build an image and run a simple Python application in a container.
- **Benefits of Docker:** We discussed the advantages of using Docker for Python applications, including **consistency**, **isolation**, and **easier deployment**.

References

- Getting started with Docker
- Docker 101
- Docker Manual
- Docker CLI Commands Cheat Sheet

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar