FullStack #WebDevelopment Bootcamp

Course Week 6

Content

Practical

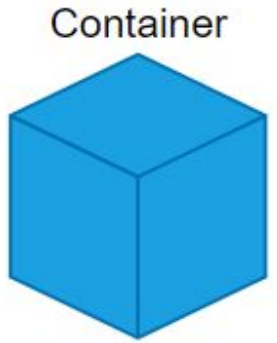
- *Implement Relationship between Answers, Questions and User*
- *Implement Role-Based Restriction for Admin User*
- *Create Dockerfile for API*

Presentation

- *What is Docker*
- *Docker Images and Containers*
- *Docker Compose*
- *AWS Short Introduction*
  - *Regions, VPC and Subnets*
  - *EC2*
  - *Security Groups*
- *Terraform: Infrastructure as Code*

Docker: An Introduction

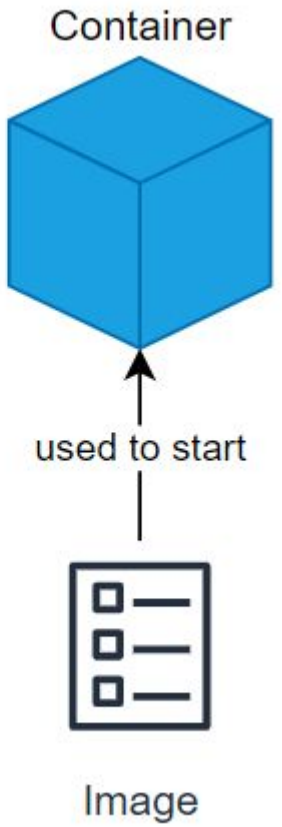
Used for developing, deploying and running application and all its dependencies into a single unit called a ***Container***



Docker: An Introduction

Used for developing, deploying and running application and all its dependencies into a single unit called a ***Container***

This ***Container*** *is created from an* ***Image****, which represents a read-only template containing all the files, dependencies and configurations necessary to start a* ***Container***

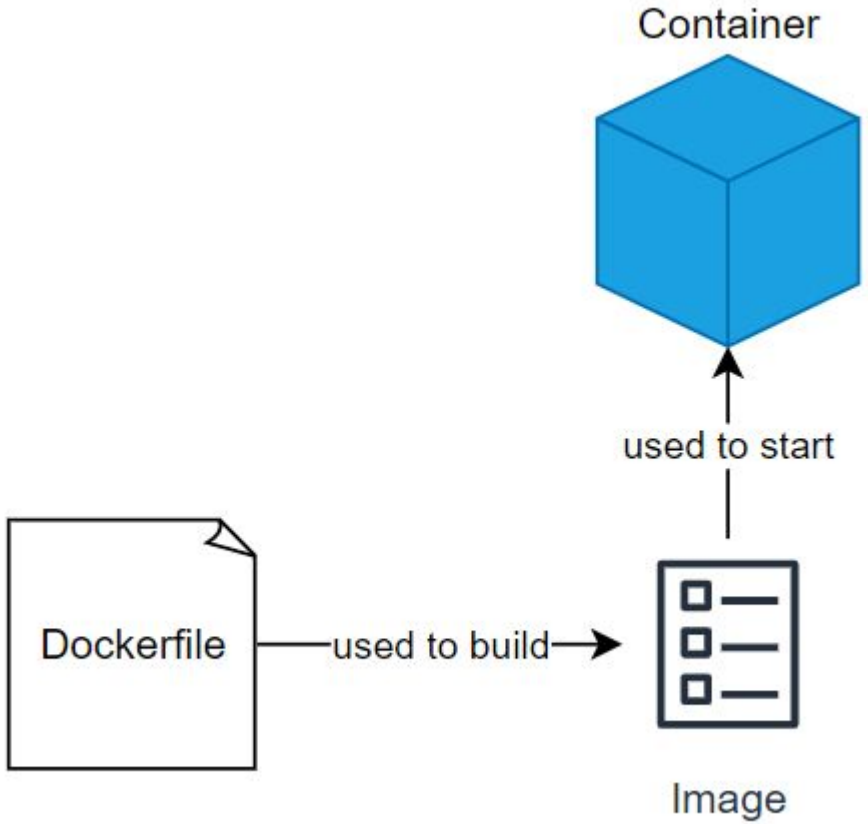


Docker: An Introduction

Used for developing, deploying and running application and all its dependencies into a single unit called a ***Container***

This ***Container*** *is created from a* ***Image****, which represents a read-only template containing all the files, dependencies and configurations necessary to start a* ***Container***

An ***Image*** *is created from a* ***Dockerfile*** *which represents a script that specifies the steps required to build the image*

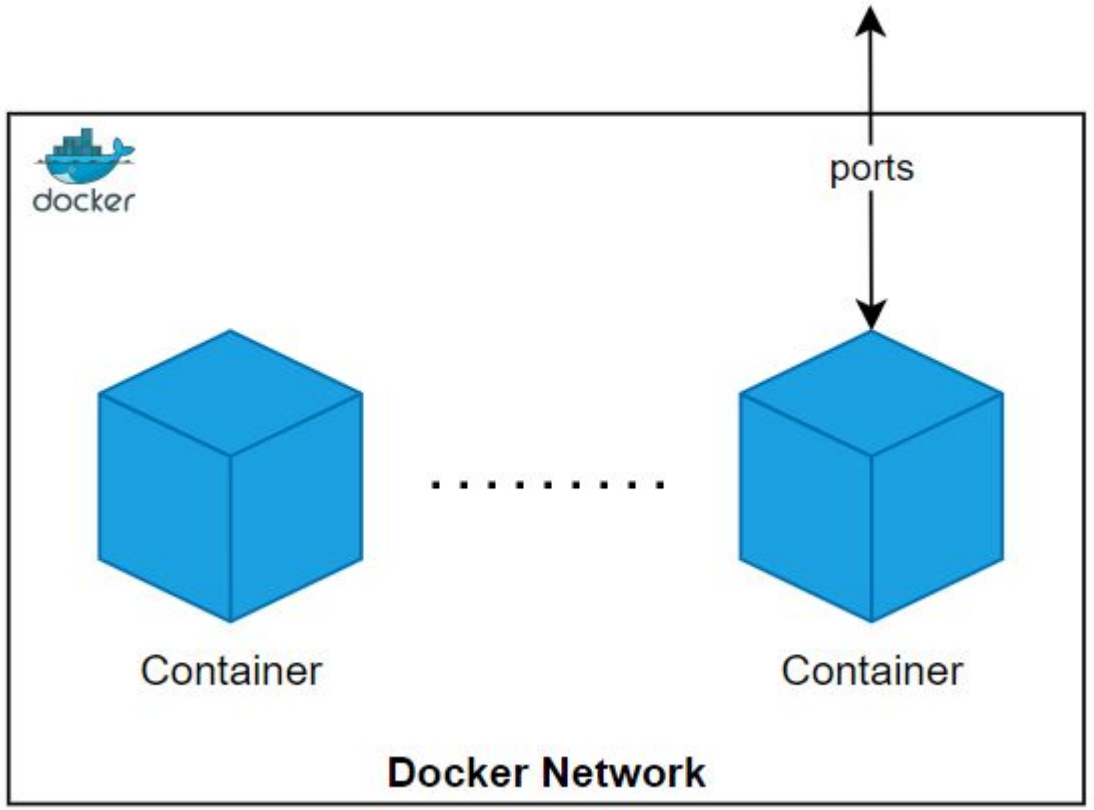


Docker: An Introduction

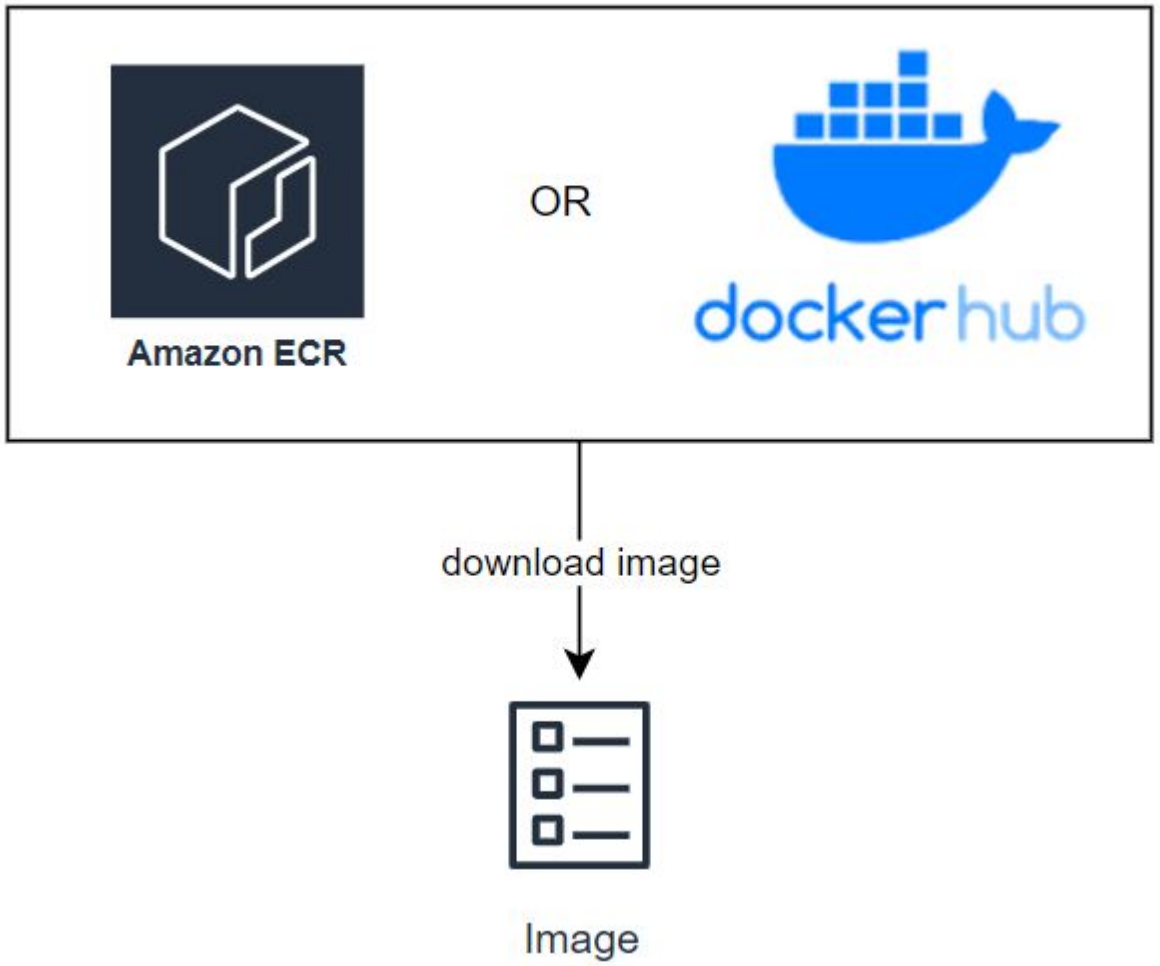
*You can start multiple **Containers** from its own Image, each having their own separate environment (file system, network, resources).*

All of these containers live inside the Docker Network in the machine.

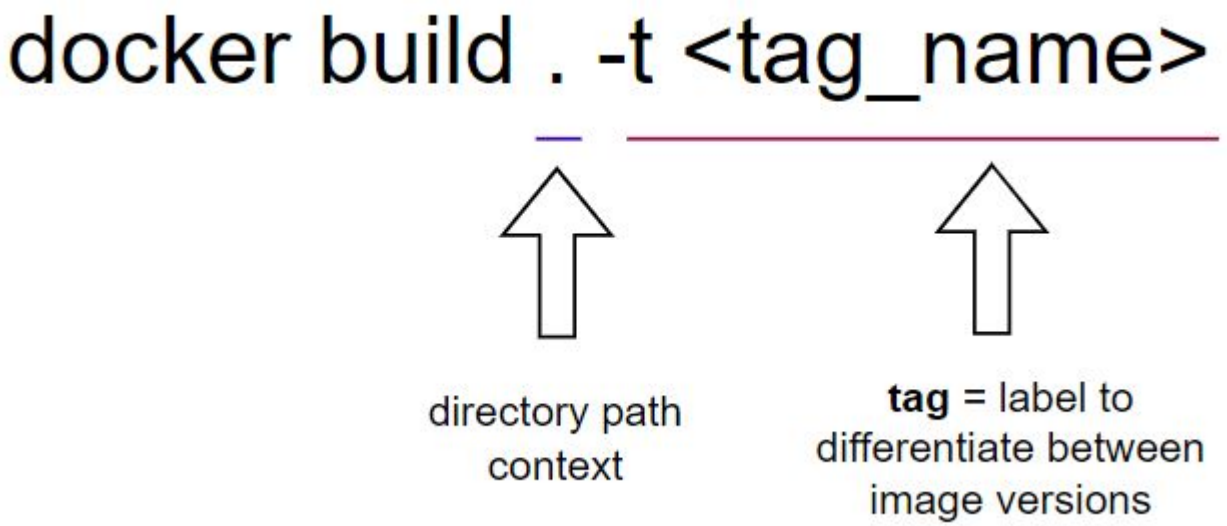
*You can expose for them **ports,** so they can be accessed from outside*



*Images can pushed and pulled (downloaded) from an **Image Repository** (e.g.: Docker Hub (default), Amazon ECR, etc)*



Docker: Generic Commands



Docker: Generic Commands

```
docker build . -t <tag_name> -f Api.Dockerfile
```

directory path context

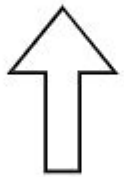
tag = label to differentiate between image versions

file = directly specify the Dockerfile

Docker: Generic Commands

Docker: Generic Commands

```
docker run -p 80:8080 myappImage
```



Ports
<Host Machine Port> : <Container Port>

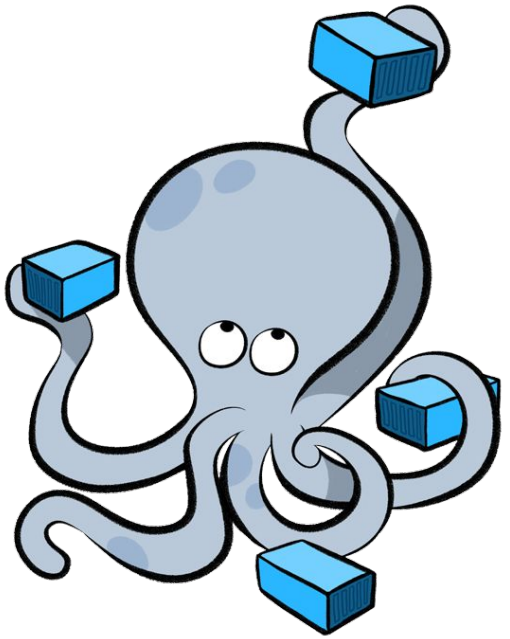
Image Name

Docker Compose

Docker Compose *is a tool that allows you to run and manage multi-container application.*

It uses a compose file (YAML) to define how:

- *the containers should be started*
- *what images are used*
- *what ports are exposed*
- *set the environment variables*
- *etc.*



Docker Compose: An Example

```yaml
version: '3.9'

services:
  utcndb:
    image: "postgres"
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=postgres
    ports:
      - "5432:5432"
    volumes:
      - utcndb-volume:/var/lib/postgresql/data
  utcnapi:
    build:
      context: '../'
      dockerfile: 'Api.Dockerfile'
    ports:
      - "3000:3000"
    environment:
      DATABASE_HOST: utcndb
      DATABASE_PORT: 5432
      DATABASE_USERNAME: postgres
      DATABASE_PASSWORD: postgresS
      DATABASE_NAME: postgres
      PRODUCTION_FLAG: false
      JWT_SECRET: 2d519e67aac090efc1940f31e31ad8fee07545753d5e18f7ea5c082819df9a7a
    depends_on:
      - utcndb
volumes:
  utcndb-volume:
```

Amazon Web Services (AWS): An Introduction

AWS *represents a cloud provider offering a wide variety of functionality:*

- *Cloud Computing (VM, etc)*
- *Cloud Storage (File Storage, etc)*
- *Network Management (IP Addresses, Exposed Ports, etc)*
- *Security (DDOS Protection)*
- *SaaS (Software as a Service) = AWS managed solutions for different use cases (Authentication, Authorization, Infrastructure Deployment, AI, etc)*

AWS: Regions

***AWS** provides cloud capabilities around the world.*

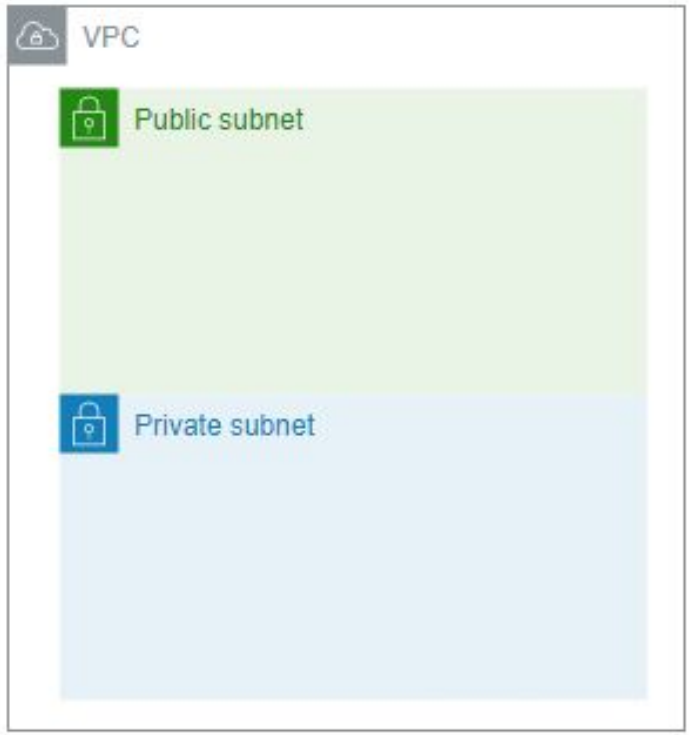
*This is possible by having multiple **Regions** where multiple data centers (server racks) are hosted*

AWS: Virtual Private Cloud (VPC)

*You usually work inside a **Region** and deploy a resource inside one or more **Availability Zones***

These resources exist in an isolated virtual network (a VPC) where you can configure:

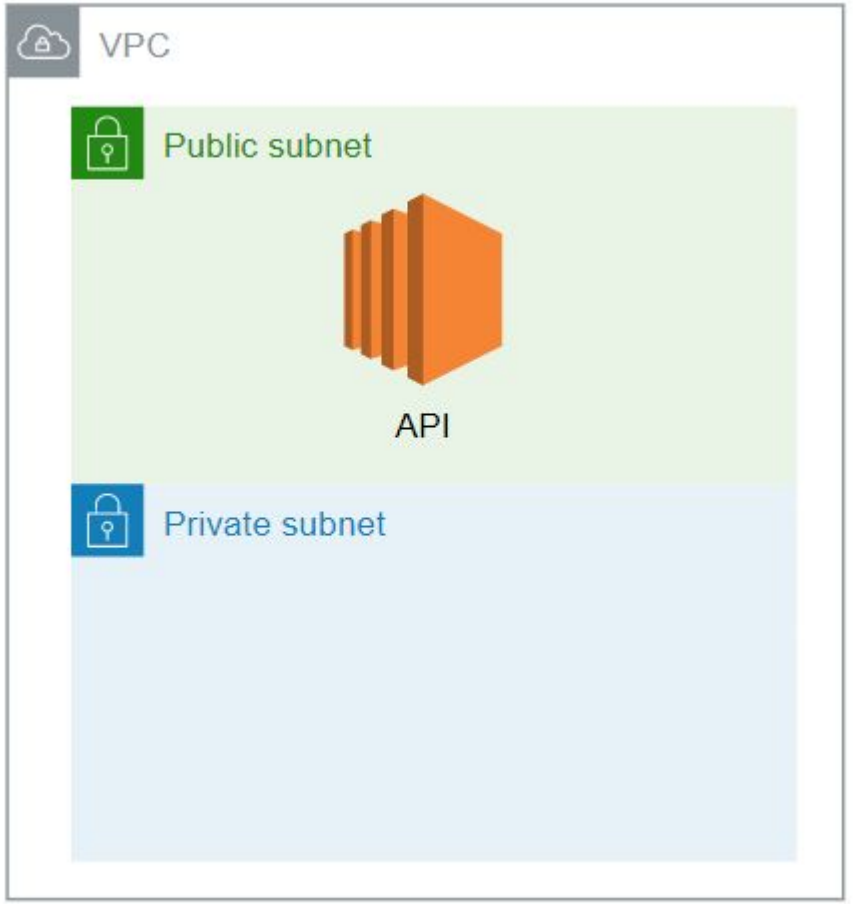
- *IP Addresses Ranges*
- *IP Addresses Routing*
- *Subnets and their access to the internet*
- *etc*



AWS: EC2

The most common offering of AWS is EC2 (Elastic Compute Cloud) which allows users to manage and configure a virtual server (usually a Linux server)

This can be used for any kind of processing (e.g.: our API)

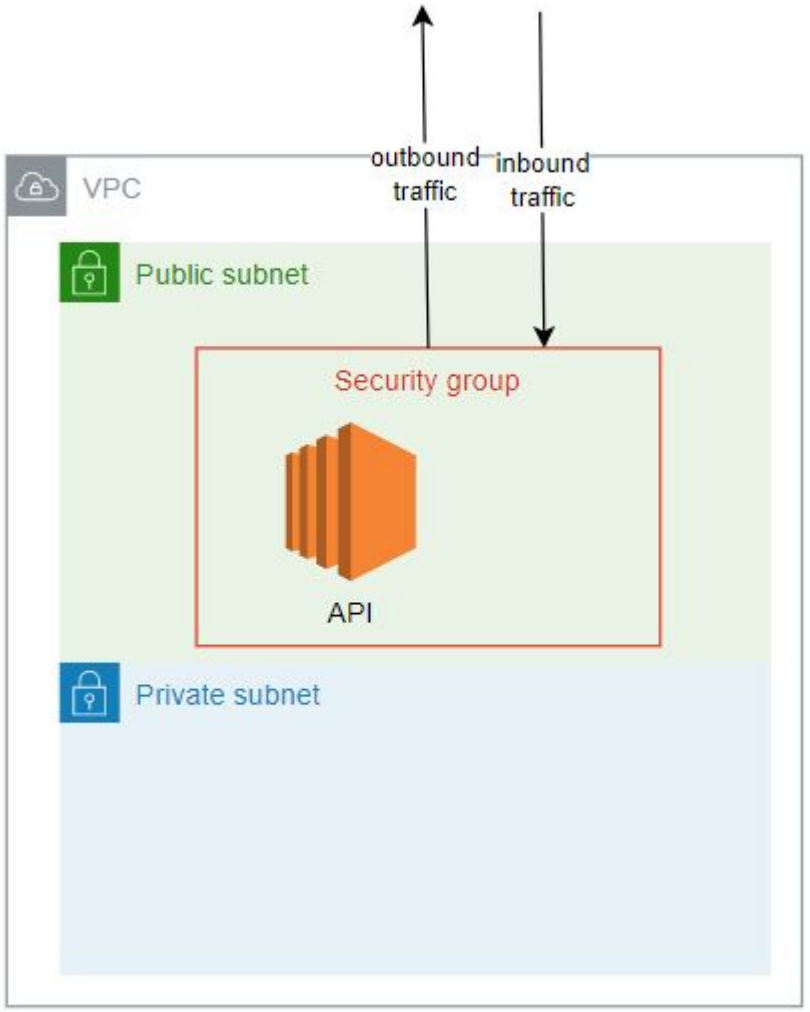


AWS: Security Groups

A security group is attached to a Cloud Resource and defines how exactly the network access looks like.

For example: You can configure for our API to be

- *only reachable over the HTTP protocol on port 3000 on any inbound traffic (traffic that comes in)*
- *and any outbound traffic is permitted*



AWS: How to deploy

There are several way to deploy

- *Manual Click Way:*
  - *You go to the AWS Console start to create your resources and attach them manually*
- *Manual Terminal Way:*
  - *AWS CLI allows you to create any resource through terminal commands*
- *Infrastructure as Code (IaC) Way*
  - *Write code to specify which resources should be created and in which order*

Infrastructure as Code

Infrastructure as Code represent a way to provision your cloud resources through code.

It can automatically manage the creation, update and destruction of said resources. Moreover, it allows for versioning and collaboration on the infrastructure