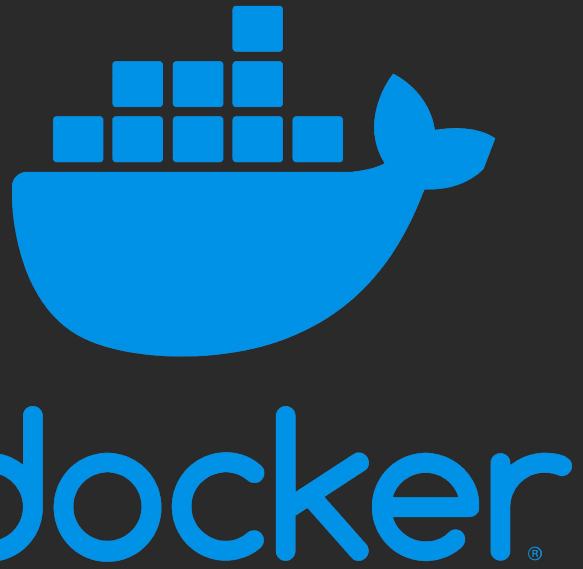
Containerization with Docker

Run your code (finally) any where…..

UULM 2024
HACKATHON X GAMEJAM

What is Docker?

- Open source platform
- Automates deployment, scaling
  And management of applications
- Lightweight and portable containers



Allows developers to package their
Applications into a consistently running
Container

3 Reasons for Docker

- Portability
  - Containers package application along with all libraries and dependencies.
  - Ensures that applications run the same across different environments
  - “ IT WORKED ON MY MACHINE “ is no more!
  - Works in Development, Testing and Production scenarios

3 Reasons for Docker

- Isolation
  - Each container operates in its isolated environment
  - Container provide a way to run multiple applications simultaneously
  - Simplifies debugging and troubleshooting

3 Reasons for Docker

- Efficiency
  - Containers are lightweight, and start up quickly
  - Use fewer resources than traditional VMs
  - Beneficial for microservices and cloud computing (More Later :))

Key Concepts

- Dockerfiles
- Images
- Container
- Registries

Dockerfiles

- “Blueprint” for the Container
- Contains Sequences of instructions
  - Environment
  - Dependencies
  - Copying Files
  - Defining Entry Points
- Commands:
  - dockerd : start the docker daemon
  - docker build -t <image-name> <path-to-dockerfile> : build the docker container
  - docker images : show all local images

Images

- Read-only template
- Used to create Docker Containers
- Can be pulled from a registry or built locally
- Commands:
  - docker run -d -p <host-port>:<container-port> --name <container-name> <image-name>
  - docker pull <image-name>:<tag> : Pull from Docker Hub
  - docker inspect <image-name>:<tag> : Inspect Image
  - docker push <image-name>:<tag> : Add Image to registry
  - docker image prune -a : Remove unused images

Container

- Containers “execute Images”
- Contain the environment and Code
- Commands:
  - docker ps (-a) : Show (all) running containers
  - docker start <container-id> : Start a stopped Container
  - docker stop <container-id> : Stop a running Container
  - docker restart <container-id> : Restart a Container
  - docker logs <container-id> : Show logs of Container
  - docker exec -it <container-id> <command> : Execute Command in running Container
  - docker inspect <container-id> : Attach to main process of Container

Registry

- Storage and distribution system for docker images
- Most popular: Docker Hub
- Also: GitHub Docker Registry
  - Integration with GitHub action
- Commands:
  - docker login <registry-url> : Log into a Registry
  - docker search <image-name> : search for an image in Docker Hub

Networks

- What are Docker Networks?
  - Enable Communication between containers
  - Manage isolation and connectivity
- Why use Docker Networks?
  - Isolate Services for security
  - Enables communication for microservices
- Commands:
  - docker network ls : Show all networks
  - docker network inspect <network-name> : Inspect Details
  - docker network create <name> : Create a network
  - docker network connect <network-name> <container> : Connect a Container

Volumes

- Mechanism to persist data beyond container lifecycle
- Simplify data sharing between containers
- Commands:
  - docker volume create <volume-name> : Create a Volume
  - docker run -v <host-path>:<container-path> <container> : Directly mount volume to container
  -

Docker Compose

- Tool to define and manage multi-container applications
- Simplifies deployment of multi-service apps
- Configuration in docker-compose.yml
- Commands:
  - docker-compose build
  - docker-compose up (-d)
  - docker-compose down (--volumes)
  - docker-compose ps
  - docker-compose logs (-f) (<service-name>)

Docker Compose - Services

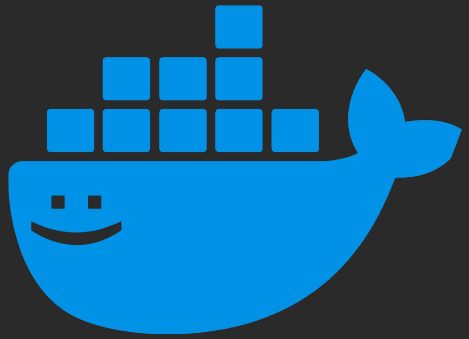
- Comprised of Services
- Parts of the a Service:
  - Image: <image-name> / build: <path-to-dockerfile>
  - container_name: <container-name>
  - environment:
    - ENV1:VAL1……
  - ports:
  - volumes:
  - network:

Docker Swarm

- Native container orchestration tool in Docker
- Manages a cluster of Docker engines
- Increases Scalability
- Key Concepts:
    - Swarm Mode : initialize with “docker swarm init”
    - Nodes : manager and worker nodes
    - Services : Define tasks and replicas
    - Tasks: : Individual unit of work

Demo Time!

Thank you!



docker

UULM 2024
HACKATHON X GAMEJAM