Docker Concepts and Commands

Introduction

Docker is a platform for building, shipping, and running applications in containers. Containers are lightweight, portable, and self-contained units that package up an application and its dependencies. This ensures that the application runs consistently across different environments.

Key Docker Concepts:

- **Images:** Read-only templates that define the environment for a container.
- **Containers:** Instances of an image, providing a runnable environment for an application.
- **Dockerfiles:** Text files that define the steps to build a Docker image.
- **Registries:** Centralized repositories for storing and sharing Docker images.

Docker Commands

Image Commands

- **docker pull <image>:<tag>:** Pulls an image from a registry.

docker pull hello-world

- **docker images:** Lists all images on your system.

docker images

- **docker build -t <image>:<tag>:** Builds an image from a Dockerfile in the current directory.

docker build -t my-app:latest .

- **docker push <image>:<tag>:** Pushes an image to a registry.

docker push my-username/my-image

Container Commands

- **docker run <image>:<tag>:** Creates and starts a container from an image.

docker run -it hello-world

- **docker ps:** Lists all running containers.

docker ps

- **docker stop <container_id>:** Stops a container.

docker stop my-container

- **docker start <container_id>:** Starts a stopped container.

docker start my-container

- **docker rm <container_id>:** Removes a container.
- docker rm my-container

Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications. It uses a YAML file to define the services and their dependencies.

Example Docker Compose file:

YAML

```
version: '3.7'

services:
  web:
    build: .
    ports:
      - "8082:8080"
    depends_on:
      - redis
  redis:
    image: redis:latest
    ports:
      - "6379:6379"
```

```
docker-compose up
```

Best Practices

- **Use multi-stage builds:** Optimize image size by building in multiple stages.
- **Leverage Docker volumes:** Persist data outside the container for easier management.
- **Use environment variables:** Make configurations flexible and easy to change.
- **Scan images for vulnerabilities:** Ensure security best practices.

Additional Topics

- **Docker Hub:** A popular registry for sharing Docker images.

Diagram: Docker Architecture

Docker architecture diagram, showing Docker Engine, Docker Daemon, Docker Client, Images, Containers, and Registries