Docker Compose Basics - Brief Introduction

1. What is Docker Compose?

Docker Compose is a tool used for defining and managing multi-container Docker applications. Instead of running multiple docker run commands individually, Docker Compose allows you to define all the services, networks, and volumes in a single YAML file (docker-compose.yml) and manage them together.

2. Benefits of Using Docker Compose

Simplifies multi-container management: Start/stop all services with a single command.
Reusable configurations: Define services once and reuse them across environments.
Easy scaling: Quickly scale services by specifying the number of replicas.
Environment consistency: Ensures the same environment setup for development, testing, and production.

3. Writing docker-compose.yml File

A docker-compose.yml file is a YAML configuration file where you define:

Services: The individual containers that make up your application.
Networks: Isolated networks for secure communication between containers.
Volumes: Persistent data storage for containers.

Example YAML:

yaml

```
version: '3.8'
services:
  app:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - ./app:/usr/share/nginx/html
    networks:
      - my-network

networks:
  my-network:
```

Docker Compose Commands

docker-compose up → Start services

Starts all the services defined in the docker-compose.yml file.
Adds -d flag to run in detached mode (background).

```
docker-compose up -d
```

docker-compose down → Stop and remove services

Stops and removes the containers, networks, and volumes defined in the file.

docker-compose down
docker-compose ps → List running services
Displays the status of running services in your Docker Compose project.

docker-compose ps
Environment Variables in Docker Compose
You can use environment variables to manage configuration dynamically. These variables are defined in a .env file or directly in the YAML file.

Example: .env file:

bash
DB_USER=admin
DB_PASS=secret

docker-compose.yml:

```
services:
  db:
    image: mysql:5.7
    environment:
      MYSQL_USER: ${DB_USER}
      MYSQL_PASSWORD: ${DB_PASS}
```

Scaling Services with Docker Compose

You can scale services to handle more traffic or perform parallel processing using the --scale flag.

Example:

docker-compose up --scale web=3

This creates 3 replicas of the web service.

Linking Services & Container-to-Container Communication
Docker Compose uses service discovery by creating an internal DNS, enabling services to communicate using their service names.

Example:

```
services:
  web:
    image: nginx
    networks:
      - app-network
  db:
    image: mysql
    networks:
      - app-network

networks:
  app-network:
```

The web service can connect to the db service using the name db.

Health Checks in Docker Compose
Health checks allow Docker Compose to monitor the status of a service and restart it if it becomes unhealthy.

Example:

```
services:
  app:
    image: nginx
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost"]
      interval: 30s
      retries: 3
      start_period: 10s
      timeout: 5s
```

Docker checks the container's health every 30 seconds and retries 3 times if it fails.

Conclusion
Docker Compose simplifies the management of multi-container applications by providing easy service definition, scaling, and communication. Mastering these basics will help you efficiently deploy and manage your containerized applications.