DEVOPS FOUNDATIONS

CERTIFICATE

- Zainuddin Saiyed -

SESSION - 3

SHIFTKEY ACADEMY

DOCKER COMPOSE

SHIFTKEY ACADEMY

What is Docker Compose?

- It is a tool for defining and running multi-container Docker applications.
- The benefit is docker compose simplifies the process of managing multiple containers.

Key Features of Docker Compose [1]:

- **Simplified control**: Orchestrate multi-container apps using single YAML file. This makes it easier to manage and replicate our application environment.
- **Rapid application development**: Docker compose caches the configuration used to create a container. Hence, helping in only recreating containers which have changed.
- **Portability across environment**: Customize the variables and moving compositions between environments.
- **Preserve volume data** when containers are orchestrated.

Read More: [1] https://docs.docker.com/get-started/workshop/08_using_compose

Docker Compose File Structure

- `version`: Specifies the Compose file format version.
- `services`: Defines the containers to be run.
- `networks`: Specifies the networks to be created.
- `volumes`: Defines the volumes to be used.

Basic Docker Compose Commands:

- `docker compose up`: Create and start containers.
- `docker compose down`: Stop and remove containers, networks.
- `docker compose ps`: List containers.
- `docker compose logs`:View output from containers.

DOCKER NETWORKS

What is Docker Networks?

Docker Networks is a means to:

- Allow containers to communicate with each other and the outside world.
- Create/orchestrate isolation between containers.
- (*provides ability to*) Link containers across multiple Docker hosts.

Types of Docker Networks:

- **Bridge**: Default network driver of docker. It creates an isolated network on the host, allowing containers to communicate with each other while remaining isolated from the host network.
- **Host**: This removes network isolation between container and Docker host.
- **Overlay**: This enables communication between containers across multiple Docker hosts. These are primarily used for connecting containers on separate Docker Engine instances.

Read More: https://spacelift.io/blog/docker-networking

Which Network Type is Best?

Bridge Networks

- Best for common scenarios (*because it provides good balance between isolation and performance*).
- Allows containers to communicate using IP addresses and DNS names.
- Offers network isolation between containers and host.

Host Networks

- Fastest Performance and is suitable when network isolation is not a concern.
- Containers share the host's network stack directly.
- Ideal when you need to bind ports directly to host interfaces (*Port bindings publish directly to the host's network interface*).
- No IP address allocation for containers.
- Improved performance due to lack of network address translation (*Simulates services running directly on the host*).

Overlay Networks

- Essential for communication between containers on different Docker hosts (Docker Swarm clusters).
- Enables creation of distributed environments (*which need container-to-container communication across hosts*).
- Useful for setting up high-availability configurations.

Factors to consider for choosing a Network

Consider factors such as:

- Level of network isolation requirements.
- Performance requirements.
- Multi-host communication requirements.

Tip: Depending on your application scenario can start with a simple setup and then adjust based on the complexity, requirements, and need of the application.

Key Docker Network Commands:

- `docker network create`: Create a network.
- `docker network connect`: Connect a container to a network.
- `docker network disconnect`: Disconnect a container from a network.
- `docker network inspect`: Display detailed information on networks.

Read More: https://spacelift.io/blog/docker-networking

DOCKER VOLUMES

SHIFTKEY ACADEMY

What is Docker Volumes?

Docker volumes are a filesystem mechanism in Docker used to **persist data independently** given a containers life-cycle. These volumes are:

- **Created** & **managed by Docker.**
- Stored in a part of the host filesystem managed by Docker (typically `/var/lib/docker/volumes` on Linux).
- These volumes can be **named** or **anonymous**.

What is the use of Docker Volumes?

- Volumes allow for storing & sharing files or directories which can be shared across multiple containers.
- Any data written into a volume remains available even if the container stops or is removed.
- These are useful for backing up, restoring, or migrating data between Docker hosts.

Benefits of Docker Volumes

- Data persistence across container restarts and removals.
- Easier sharing of data between containers.
- Improved performance for I/O intensive operations.
- Volumes which are unmounted are not automatically removed.

Types of Docker Volumes:

1. **Named volumes:**
   - Easier to manage and backup.
   - Can be used with multiple containers.
   - Create volumes using: `docker volume <volume_name>`
   - Run container with named volume:
     `docker run -v <volume_name>:<Path> <image_name>`
2. **Bind mounts:**
   - Link to a specific path on the host machine which is editable and accessible (*Direct access to files on the host*).
   - Useful for development environments.
   - Can be used to inject configuration files
   - Create volumes using: `docker volume create --name <volume_name>`
   - List all volumes: `docker volume ls`

Read More: https://semaphoreci.com/blog/docker-volumes

SHIFTKEY ACADEMY

Types of Docker Volumes:

Feature	Named Volumes	Bind Mounts
Ease of use	Simpler (as managed by Docker)	Requires specifying exact host path
Portability	More portable (managed by Docker and not tied to a specific host directory)	Less portable (as they depend on the host's file system structure)
Backup	Easier (as Docker provides commands to manage them)	Manual backup process (as these are just directories on the host)
Performance	Better performance as its optimized by Docker	Performance depends (varies) on host filesystem
Use case	Production data	Development, configuration files

MICROSERVICES ARCHITECTURE

SHIFTKEY ACADEMY

What are Microservices?

A microservices architecture is a type of application architecture where the application is developed as a collection of services. It provides the framework to develop, deploy, and maintain microservices architecture diagrams and services independently [2].

In short: Small, autonomous services that work together in contrast with monolithic architecture.

Characteristics of Microservices:

- **Independence** (Each service focuses on a specific business capability).
- **Scalability** (Services can be developed, deployed, and scaled independently).
- **Faster ship time** (Enables faster development and easier maintenance).
- **Diverse niche technology adoption** (Facilitates use of different technologies for different services).
- **Decentralized data management.**

Read More: [2] https://cloud.google.com/learn/what-is-microservices-architecture

CLOUD COMPUTING

FUNDAMENTALS

SHIFTKEY ACADEMY

Cloud Computing Fundamentals

What is Cloud Computing?

- **Definition**: Delivery of computing services over the internet.
- **Key characteristics**: On-demand, scalable, pay-as-you-go.

Cloud Service Models

- Infrastructure as a Service (IaaS).
- Platform as a Service (PaaS).
- Software as a Service (SaaS).

Read More:
https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing

Cloud Deployment Models

- Public Cloud.
- Private Cloud.
- Hybrid Cloud.
- Multi-Cloud.

Cloud Computing Fundamentals

Benefits of Cloud Computing

- Cost efficiency
- Scalability and flexibility
- Reliability and disaster recovery
- Global reach and performance

Major Cloud Providers

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform (GCP)

ANY QUESTIONS?

THANK YOU!

SHIFTKEY ACADEMY