Docker Concepts and Commands: A Summary

1. Introduction to Docker

Docker is an open platform for developing, shipping, and running applications. It allows you to package your applications and dependencies into containers, ensuring they run consistently across different environments.

- **Docker Image**: A lightweight, standalone, and executable package that includes everything needed to run a piece of software.
- **Docker Container**: A runtime instance of a Docker image, providing isolated environments to run applications.
- **Docker Registry**: A centralized place to store and distribute Docker images (e.g., Docker Hub).

2. Key Docker Concepts

- **Images**: Docker images are the blueprint for containers. They are built using a Dockerfile and can be shared via registries.
- **Containers**: Containers are instances of images. They can be created, started, stopped, and deleted.
- **Volumes**: Volumes are used to persist data generated by Docker containers, ensuring data is not lost when containers stop.
- **Networks**: Docker networking allows communication between containers, either within a single host or across multiple hosts.

---

3. Important Docker Commands

Building Docker Images

```
docker build -t <image_name>:<tag> .
```

This command builds a Docker image from a Dockerfile located in the current directory (`.`) and tags it with the given name.

Listing Docker Images

```
docker images
```

This lists all Docker images available locally.

Running Docker Containers

```
docker run -d -p <host_port>:<container_port> <image_name>
```

Runs a Docker container from an image. The `-p` option maps a host port to a container port, and `-d` runs the container in detached mode.

Viewing Running Containers

```
docker ps
```

This lists all running containers. Use `docker ps -a` to list all containers (running and stopped).

Stopping a Running Container

```
docker stop <container_id>
```

Stops a running container based on its container ID or name.

Removing Containers

```
docker rm <container_id>
```

Removes a stopped container. The `-f` option can be used to forcefully remove a running container.

Removing Images

```
docker rmi <image_id>
```

Removes a Docker image by its image ID or name.

Copying Files from Containers

```
docker cp <container_id>:<source_path> <destination_path>
```

Copies files from a running container to the host machine.

Inspecting Docker Objects

```
docker inspect <object_id>
```

Returns detailed information about containers, images, networks, and volumes.

4. Docker Compose Concepts

Docker Compose is a tool for defining and running multi-container Docker applications. It uses a YAML file (`docker-compose.yml`) to configure the services.

- **Services**: A service is a container that is part of your application (e.g., web server, database).
- **Networks**: Docker Compose automatically sets up networks for services to communicate with each other.
- **Volumes**: Compose can create volumes to share data between containers and the host.

---

5. Docker Compose Commands

Starting Multi-Container Application

```
docker-compose up
```

Starts all the services defined in the `docker-compose.yml` file. The `-d` option can be used to run in detached mode.

Stopping Containers

```
docker-compose down
```

Stops and removes all containers, networks, and volumes created by `docker-compose up`.

Building Images with Docker Compose

```
docker-compose build
```

Builds the images defined in your Docker Compose file.

Viewing Running Services

```
docker-compose ps
```

Lists all running services in your Docker Compose setup.