# Project 01

## Objectives:

- Create and manage Docker volumes for data persistence.
- Set up a Docker network for container communication.
- Use Docker Compose to manage multi-container applications.
- View and manage Docker logs.
- Deploy the application using Docker Swarm.

## Project Outline:

1. **Create Docker Volumes**
2. **Create a Docker Network**
3. **Write a Docker Compose File**
4. **Deploy the Application with Docker Compose**
5. **Manage Docker Logs**
6. **Deploy the Application Using Docker Swarm**

## Step-by-Step Guide

### 1. Create Docker Volumes

Docker volumes are used to persist data generated by and used by Docker containers.

```
docker volume create wordpress_data
```

```
einfochips@AHMLPT1108:~/DevOPs_Training/Day-5$ docker volume create wordpress_data
wordpress_data
```

```
docker volume create mysql_data
```

```
einfochips@AHMLPT1108:~/DevOPs_Training/Day-5$ docker volume create mysql_data
mysql_data
```

```
einfochips@AHMLPT1108:~/DevOPs_Training/Day-5$ docker volume ls
DRIVER    VOLUME NAME
local     d589ce314cc00ca5403c72812c9f1d204be8d2739dabacf3df3ed36e1a287d45
local     dockercompose_postgresql
local     dockercompose_postgresql_data
local     dockercompose_sonarqube_data
local     dockercompose_sonarqube_extensions
local     dockercompose_sonarqube_logs
local     jenkins-docker-home
local     jenkins-home
local     jenkins_home
local     minikube
local     mysql_data
local     wordpress_data
```

### 2. Create a Docker Network

Create a custom network for the containers to communicate.

```
docker network create wordpress_network
```

```
einfochips@AHMLPT1108:~/DevOPs_Training/Day-5$ docker network create wordpress_network
289e9d4807200b044b00d28a9a85c8cf998df038660c19323eaec36239f58bc0
einfochips@AHMLPT1108:~/DevOPs_Training/Day-5$ docker network ls
NETWORK ID     NAME                     DRIVER    SCOPE
49cd2ef06b25   bridge                   bridge    local
94e72bbda352   host                     host      local
cecb7a5fada9   jenkins-config_default   bridge    local
d79dc124ee2d   minikube                 bridge    local
8823d581f4b2   network1                 bridge    local
9aca64b4098f   network2                 bridge    local
956f10e56f7d   none                     null      local
289e9d480720   wordpress_network        bridge    local
einfochips@AHMLPT1108:~/DevOPs_Training/Day-5$
```

### 3. Write a Docker Compose File

Create a `docker-compose.yml` file to define and manage the services.

```
version: '3.3'


services:
  db:
    image: mysql:5.7
    volumes:
      - mysql_data:/var/lib/mysql
    networks:
      - wordpress_network
    environment:
      MYSQL_ROOT_PASSWORD: example
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress


  wordpress:
    image: wordpress:latest
    volumes:
      - wordpress_data:/var/www/html
    networks:
      - wordpress_network
```

```
    ports:
      - "8000:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress


volumes:
  mysql_data:
  wordpress_data:


networks:
  wordpress_network:
```

**4. Deploy the Application with Docker Compose**

Run the following command to start the services defined in the `docker-compose.yml` file.

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

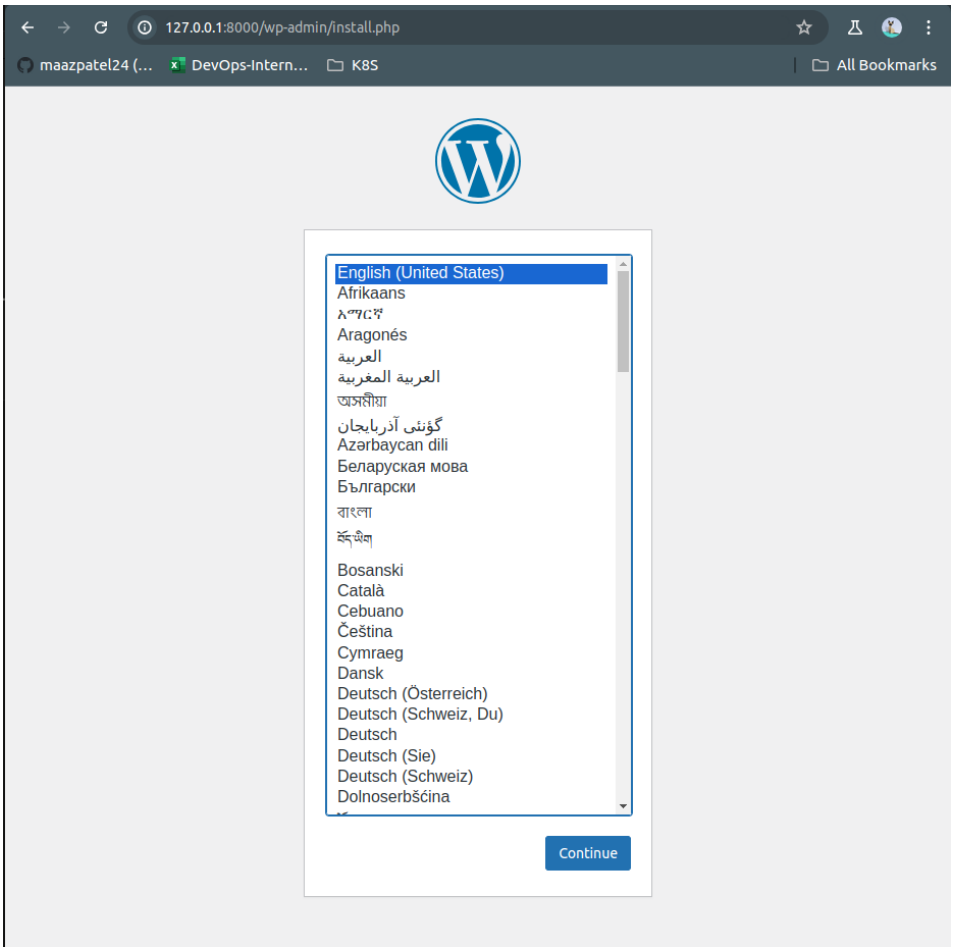
```
einfochips@AHMLPT1108:~/DevOPs_Training/Day-5$ docker-compose up -d
Creating network "day-5_wordpress_network" with the default driver
Creating volume "day-5_mysql_data" with default driver
Creating volume "day-5_wordpress_data" with default driver
Pulling db (mysql:5.7)...
5.7: Pulling from library/mysql
20e4dcae4c69: Downloading [=>                                                  ]
.026MB/50.5MB Pulling fs layer
e9f03a1c24ce: Download complete
83.5kB/983.5kBDownloading [==============>                                    ]
6b95a940e7b6: Download complete
3.08kB/3.08kB Waiting
ae71319cb779: Waiting
ffc89e9dfd88: Waiting
43d05e938198: Waiting
064b2d298fba: Waiting
df9a4d85569b: Waiting
```

- Verify that the containers are running.

```
docker-compose ps
```

```
einfochips@AHMLPT1108:~/DevOPs_Training/Day-5$ docker-compose ps
      Name                    Command               State            Ports
-------------------------------------------------------------------------------
day-5_db_1          docker-entrypoint.sh          Up      3306/tcp, 33060/tcp
                    mysqld
day-5_wordpress_1   docker-entrypoint.sh apach    Up      0.0.0.0:8000->80/tcp,:::8
                    ...                                   000->80/tcp
```

- Access the WordPress setup by navigating to `http://localhost:8000`.




### 5. Manage Docker Logs

- View logs for a specific service.

```
docker-compose logs wordpress
```

```
einfochips@AHMLPT1108:~/DevOPs_Training/Day-5$ docker-compose logs wordpress
Attaching to day-5_wordpress_1
wordpress_1  | WordPress not found in /var/www/html - copying now...
wordpress_1  | Complete! WordPress has been successfully copied to /var/www/html
wordpress_1  | No 'wp-config.php' found in /var/www/html, but 'WORDPRESS_...' vari
ables supplied; copying 'wp-config-docker.php' (WORDPRESS_DB_HOST WORDPRESS_DB_NAM
E WORDPRESS_DB_PASSWORD WORDPRESS_DB_USER)
wordpress_1  | AH00558: apache2: Could not reliably determine the server's fully q
ualified domain name, using 172.23.0.3. Set the 'ServerName' directive globally to
 suppress this message
wordpress_1  | AH00558: apache2: Could not reliably determine the server's fully q
ualified domain name, using 172.23.0.3. Set the 'ServerName' directive globally to
 suppress this message
wordpress_1  | [Fri Jul 12 06:12:06.518393 2024] [mpm_prefork:notice] [pid 1] AH00
163: Apache/2.4.59 (Debian) PHP/8.2.21 configured -- resuming normal operations
wordpress_1  | [Fri Jul 12 06:12:06.518428 2024] [core:notice] [pid 1] AH00094: Co
mmand line: 'apache2 -D FOREGROUND'
wordpress_1  | 172.23.0.1 - - [12/Jul/2024:06:23:08 +0000] "GET / HTTP/1.1" 302 40
5 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
me/126.0.0.0 Safari/537.36"
wordpress_1  | 172.23.0.1 - - [12/Jul/2024:06:23:08 +0000] "GET /wp-admin/install.
php HTTP/1.1" 200 4657 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KH
TML, like Gecko) Chrome/126.0.0.0 Safari/537.36"
wordpress_1  | 172.23.0.1 - - [12/Jul/2024:06:23:10 +0000] "GET /wp-includes/css/d
ashicons.min.css?ver=6.5.5 HTTP/1.1" 200 36068 "http://127.0.0.1:8000/wp-admin/ins
tall.php" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/126.0.0.0 Safari/537.36"
wordpress_1  | 172.23.0.1 - - [12/Jul/2024:06:23:10 +0000] "GET /wp-includes/css/b
uttons.min.css?ver=6.5.5 HTTP/1.1" 200 1808 "http://127.0.0.1:8000/wp-admin/instal
l.php" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chr
ome/126.0.0.0 Safari/537.36"
wordpress_1  | 172.23.0.1 - - [12/Jul/2024:06:23:10 +0000] "GET /wp-admin/css/l10n
```

- Follow logs for real-time updates.

```
docker-compose logs -f wordpress
```

```
einfochips@AHMLPT1108:~/DevOPs_Training/Day-5$ docker-compose logs -f wordpress
Attaching to day-5_wordpress_1
wordpress_1  | WordPress not found in /var/www/html - copying now...
wordpress_1  | Complete! WordPress has been successfully copied to /var/www/html
wordpress_1  | No 'wp-config.php' found in /var/www/html, but 'WORDPRESS_...' vari
ables supplied; copying 'wp-config-docker.php' (WORDPRESS_DB_HOST WORDPRESS_DB_NAM
E WORDPRESS_DB_PASSWORD WORDPRESS_DB_USER)
wordpress_1  | AH00558: apache2: Could not reliably determine the server's fully q
ualified domain name, using 172.23.0.3. Set the 'ServerName' directive globally to
 suppress this message
wordpress_1  | AH00558: apache2: Could not reliably determine the server's fully q
ualified domain name, using 172.23.0.3. Set the 'ServerName' directive globally to
 suppress this message
wordpress_1  | [Fri Jul 12 06:12:06.518393 2024] [mpm_prefork:notice] [pid 1] AH00
163: Apache/2.4.59 (Debian) PHP/8.2.21 configured -- resuming normal operations
wordpress_1  | [Fri Jul 12 06:12:06.518428 2024] [core:notice] [pid 1] AH00094: Co
```

**6. Deploy the Application Using Docker Swarm**

Docker Swarm is a native clustering and orchestration tool for Docker.

- Initialize Docker Swarm.

```
docker swarm init
```

- Convert the Docker Compose file to a Docker Stack file, `docker-stack.yml`.

```yaml
version: '3.3'

services:
  db:
    image: mysql:5.7
    volumes:
      - mysql_data:/var/lib/mysql
    networks:
      - wordpress_network
    environment:
      MYSQL_ROOT_PASSWORD: example
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
    deploy:
      replicas: 1

  wordpress:
    image: wordpress:latest
    volumes:
      - wordpress_data:/var/www/html
    networks:
      - wordpress_network
    ports:
      - "8000:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    deploy:
      replicas: 1

volumes:
  mysql_data:
```

```
  wordpress_data:

networks:
  wordpress_network:
```

- Deploy the stack using Docker Swarm.

```
docker stack deploy -c docker-stack.yml wordpress_stack
```

- Verify the stack is running.

```
docker stack services wordpress_stack
```

# Project 02:

## Objectives:

- Deploy an application across multiple Docker Swarm worker nodes.
- Place specific components on designated nodes.
- Monitor and troubleshoot using Docker logs.
- Modify and redeploy the application.

## Project Outline:

7. **Initialize Docker Swarm and Join Worker Nodes**
8. **Label Nodes for Specific Component Placement**
9. **Create a Docker Stack File**
10. **Deploy the Application**
11. **Monitor and Troubleshoot Using Docker Logs**
12. **Modify and Redeploy the Application**

## Step-by-Step Guide

### 1. Initialize Docker Swarm and Join Worker Nodes

On the manager node, initialize Docker Swarm:

```
docker swarm init --advertise-addr <MANAGER-IP>
```

Join the worker nodes to the swarm. On each worker node, run the command provided by the `docker swarm init` output:

```
docker swarm join --token <SWARM-TOKEN> <MANAGER-IP>:2377
```

Verify the nodes have joined:

```
docker node ls
```

### 2. Label Nodes for Specific Component Placement

Label nodes to specify where certain components should run. For example, label a node for the database service:

```
docker node update --label-add db=true <NODE-ID>
```

Label another node for the application service:

```
docker node update --label-add app=true <NODE-ID>
```

Verify the labels:

```
docker node inspect <NODE-ID>
```

**3. Create a Docker Stack File**

Create a `docker-stack.yml` file to define the services and node placement constraints:

```
version: '3.8'

services:
  db:
    image: mysql:5.7
    volumes:
      - mysql_data:/var/lib/mysql
    networks:
      - app_network
    environment:
      MYSQL_ROOT_PASSWORD: example
      MYSQL_DATABASE: appdb
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    deploy:
      placement:
        constraints:
          - node.labels.db == true

  app:
    image: your-app-image
    networks:
```

```
      - app_network
    ports:
      - "8000:80"
    environment:
      DB_HOST: db
    deploy:
      replicas: 2
      placement:
        constraints:
          - node.labels.app == true
volumes:
  mysql_data:
networks:
  app_network:
```

**4. Deploy the Application**

Deploy the stack using Docker Swarm:

```
docker stack deploy -c docker-stack.yml app_stack
```

```
docker stack services app_stack
```

**5. Monitor and Troubleshoot Using Docker Logs**

Check the logs for the services:

```
docker service logs app_stack_db
docker service logs app_stack_app
```

Follow the logs in real-time to monitor issues:

```
docker service logs -f app_stack_app
```

### 6. Modify and Redeploy the Application

Make modifications to the application or the stack file as needed. For example, change the number of replicas:

```
services:
  app:
    deploy:
      replicas: 3
```

Update the stack with the new configuration:

```
docker stack deploy -c docker-stack.yml app_stack
```

Verify the changes:

```
docker stack services app_stack
```