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:

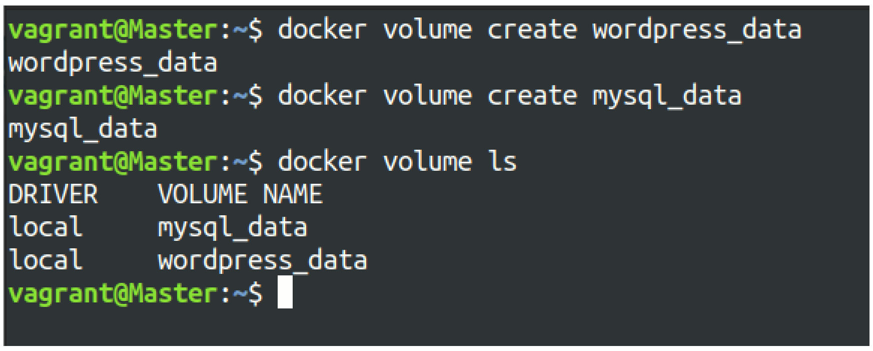
1. **Create Docker Volumes**
2. **Create a Docker Network**
3. **Write a Docker Compose File**
4. **Deploy the Application with Docker Compose**
5. **Manage Docker Logs**
6. **Deploy the Application Using Docker Swarm**

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

docker volume create mysql_data
```

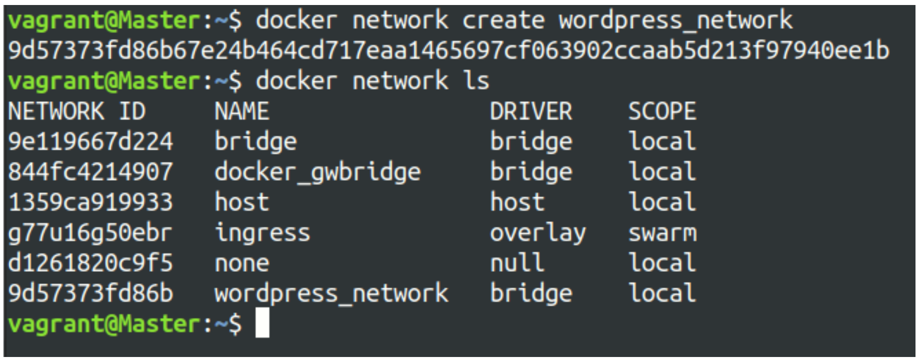


```
vagrant@Master:~$ docker volume create wordpress_data
wordpress_data
vagrant@Master:~$ docker volume create mysql_data
mysql_data
vagrant@Master:~$ docker volume ls
DRIVER    VOLUME NAME
local     mysql_data
local     wordpress_data
vagrant@Master:~$
```

2. Create a Docker Network

Create a custom network for the containers to communicate.

```
docker network create wordpress_network
```



```
vagrant@Master:~$ docker network create wordpress_network
9d57373fd86b67e24b464cd717eaa1465697cf063902ccaab5d213f97940ee1b
vagrant@Master:~$ docker network ls
NETWORK ID     NAME                DRIVER    SCOPE
9e119667d224   bridge              bridge    local
844fc4214907   docker_gwbridge     bridge    local
1359ca919933   host                host      local
g77u16g50ebr   ingress             overlay   swarm
d1261820c9f5   none                null      local
9d57373fd86b   wordpress_network   bridge    local
vagrant@Master:~$
```

3. Write a Docker Compose File

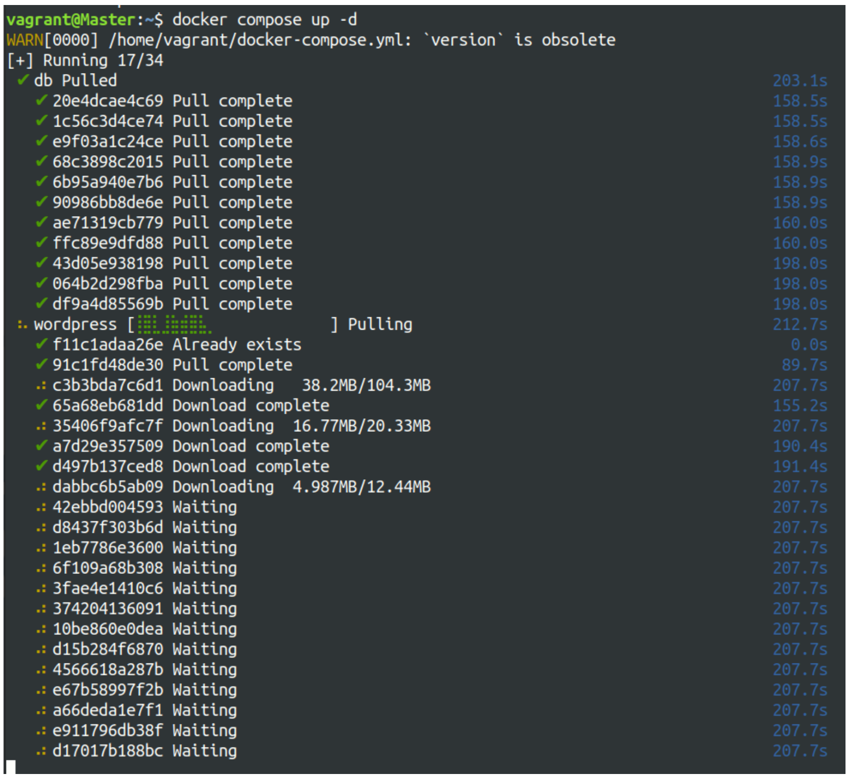
Create a `docker-compose.yml` file to define and manage the services.

```
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.

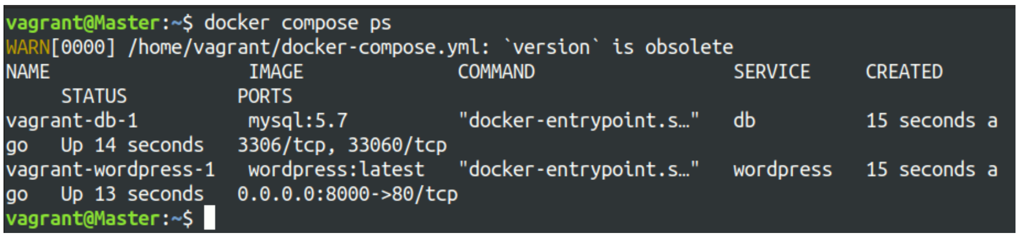
```
docker-compose up -d
```



```
vagrant@Master:~$ docker compose up -d
WARN[0000] /home/vagrant/docker-compose.yml: `version` is obsolete
[+] Running 17/34
 ✔ db Pulled                                                       203.1s
   ✔ 20e4dcae4c69 Pull complete                                    158.5s
   ✔ 1c56c3d4ce74 Pull complete                                    158.5s
   ✔ e9f03a1c24ce Pull complete                                    158.6s
   ✔ 68c3898c2015 Pull complete                                    158.9s
   ✔ 6b95a940e7b6 Pull complete                                    158.9s
   ✔ 90986bb8de6e Pull complete                                    158.9s
   ✔ ae71319cb779 Pull complete                                    160.0s
   ✔ ffc89e9dfd88 Pull complete                                    160.0s
   ✔ 43d05e938198 Pull complete                                    198.0s
   ✔ 064b2d298fba Pull complete                                    198.0s
   ✔ df9a4d85569b Pull complete                                    198.0s
 :. wordpress [⣿⣿ ⣿⣿⣿⣿⣿              ] Pulling                          212.7s
   ✔ f11c1adaa26e Already exists                                     0.0s
   ✔ 91c1fd48de30 Pull complete                                     89.7s
   .: c3b3bda7c6d1 Downloading   38.2MB/104.3MB                    207.7s
   ✔ 65a68eb681dd Download complete                                155.2s
   .: 35406f9afc7f Downloading  16.77MB/20.33MB                    207.7s
   ✔ a7d29e357509 Download complete                                190.4s
   ✔ d497b137ced8 Download complete                                191.4s
   .: dabbc6b5ab09 Downloading  4.987MB/12.44MB                    207.7s
   .: 42ebbd004593 Waiting                                         207.7s
   .: d8437f303b6d Waiting                                         207.7s
   .: 1eb7786e3600 Waiting                                         207.7s
   .: 6f109a68b308 Waiting                                         207.7s
   .: 3fae4e1410c6 Waiting                                         207.7s
   .: 374204136091 Waiting                                         207.7s
   .: 10be860e0dea Waiting                                         207.7s
   .: d15b284f6870 Waiting                                         207.7s
   .: 4566618a287b Waiting                                         207.7s
   .: e67b58997f2b Waiting                                         207.7s
   .: a66deda1e7f1 Waiting                                         207.7s
   .: e911796db38f Waiting                                         207.7s
   .: d17017b188bc Waiting                                         207.7s
```

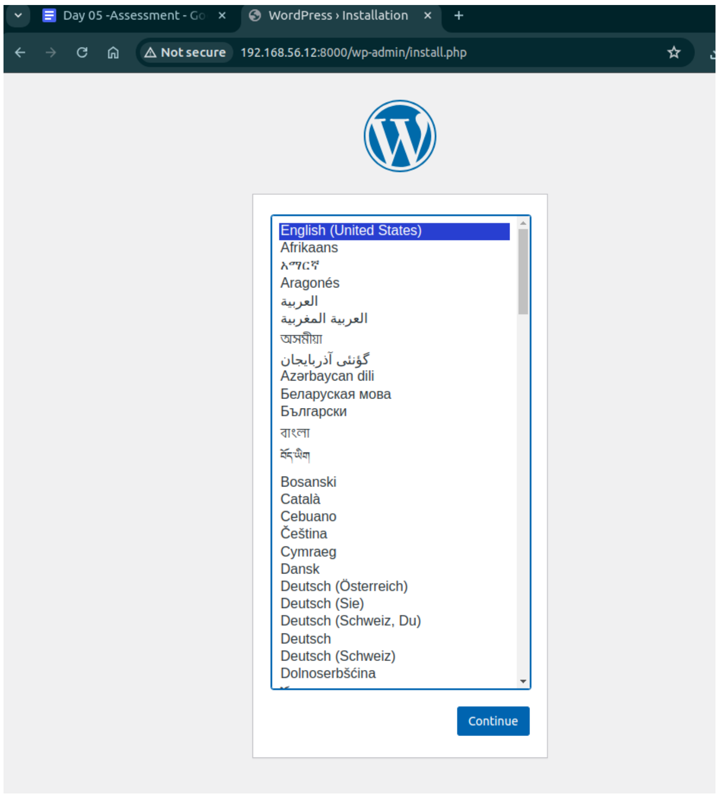
- Verify that the containers are running.

```
docker-compose ps
```



```
vagrant@Master:~$ docker compose ps
WARN[0000] /home/vagrant/docker-compose.yml: `version` is obsolete
NAME                  IMAGE              COMMAND                   SERVICE     CREATED
     STATUS          PORTS
vagrant-db-1          mysql:5.7          "docker-entrypoint.s…"   db          15 seconds a
go    Up 14 seconds   3306/tcp, 33060/tcp
vagrant-wordpress-1   wordpress:latest   "docker-entrypoint.s…"   wordpress   15 seconds a
go    Up 13 seconds   0.0.0.0:8000->80/tcp
vagrant@Master:~$
```

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



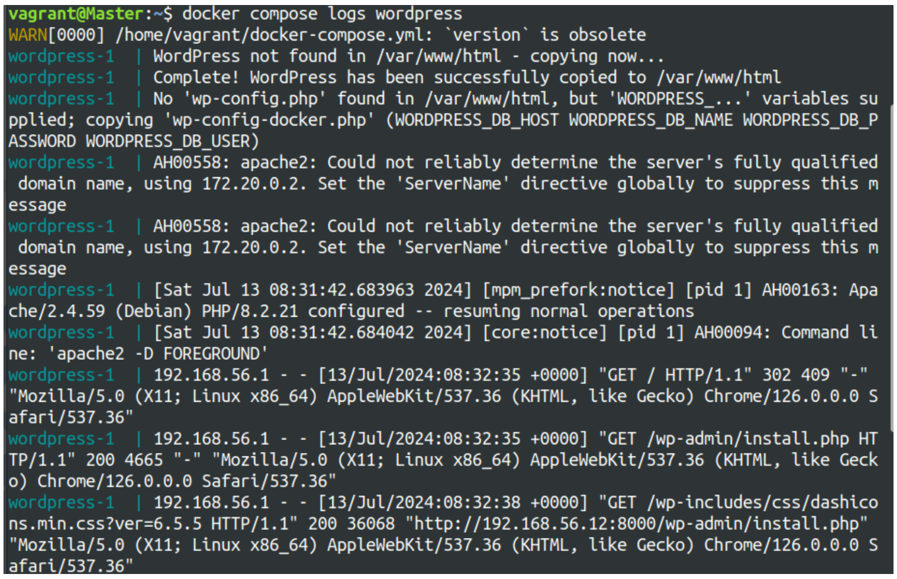
5. Manage Docker Logs

- View logs for a specific service.

```
docker-compose logs wordpress
```

- Follow logs for real-time updates.

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



```
vagrant@Master:~$ docker compose logs wordpress
WARN[0000] /home/vagrant/docker-compose.yml: `version` is obsolete
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_...' variables su
pplied; copying 'wp-config-docker.php' (WORDPRESS_DB_HOST WORDPRESS_DB_NAME WORDPRESS_DB_P
ASSWORD WORDPRESS_DB_USER)
wordpress-1  | AH00558: apache2: Could not reliably determine the server's fully qualified
 domain name, using 172.20.0.2. Set the 'ServerName' directive globally to suppress this m
essage
wordpress-1  | AH00558: apache2: Could not reliably determine the server's fully qualified
 domain name, using 172.20.0.2. Set the 'ServerName' directive globally to suppress this m
essage
wordpress-1  | [Sat Jul 13 08:31:42.683963 2024] [mpm_prefork:notice] [pid 1] AH00163: Apa
che/2.4.59 (Debian) PHP/8.2.21 configured -- resuming normal operations
wordpress-1  | [Sat Jul 13 08:31:42.684042 2024] [core:notice] [pid 1] AH00094: Command li
ne: 'apache2 -D FOREGROUND'
wordpress-1  | 192.168.56.1 - - [13/Jul/2024:08:32:35 +0000] "GET / HTTP/1.1" 302 409 "-"
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 S
afari/537.36"
wordpress-1  | 192.168.56.1 - - [13/Jul/2024:08:32:35 +0000] "GET /wp-admin/install.php HT
TP/1.1" 200 4665 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Geck
o) Chrome/126.0.0.0 Safari/537.36"
wordpress-1  | 192.168.56.1 - - [13/Jul/2024:08:32:38 +0000] "GET /wp-includes/css/dashico
ns.min.css?ver=6.5.5 HTTP/1.1" 200 36068 "http://192.168.56.12:8000/wp-admin/install.php"
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 S
afari/537.36"
```

6. Deploy the Application Using Docker Swarm

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

- Initialize Docker Swarm.

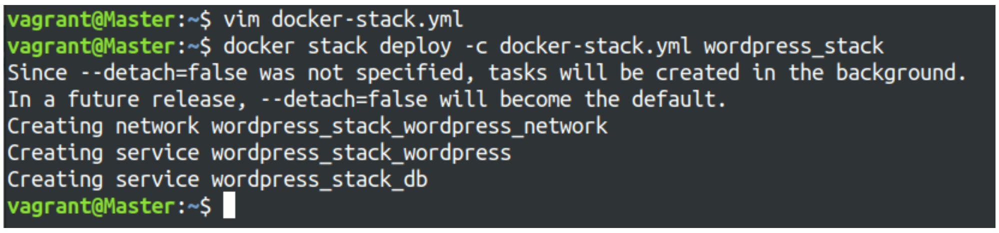
```
docker swarm init
```

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

```
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.

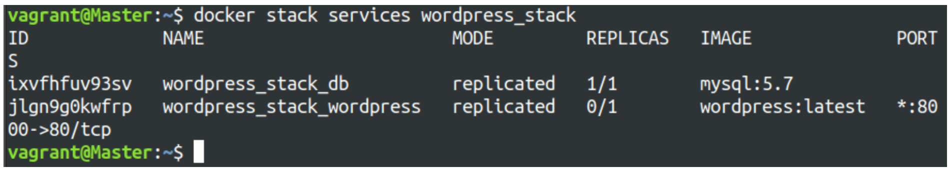
```
docker stack deploy -c docker-stack.yml wordpress_stack
```



```
vagrant@Master:~$ vim docker-stack.yml
vagrant@Master:~$ docker stack deploy -c docker-stack.yml wordpress_stack
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network wordpress_stack_wordpress_network
Creating service wordpress_stack_wordpress
Creating service wordpress_stack_db
vagrant@Master:~$
```

- Verify the stack is running.

```
docker stack services wordpress_stack
```



```
vagrant@Master:~$ docker stack services wordpress_stack
ID             NAME                        MODE         REPLICAS   IMAGE              PORT
S
ixvfhfuv93sv   wordpress_stack_db          replicated   1/1        mysql:5.7
jlgn9g0kwfrp   wordpress_stack_wordpress   replicated   0/1        wordpress:latest   *:80
00->80/tcp
vagrant@Master:~$
```

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:

1. **Initialize Docker Swarm and Join Worker Nodes**
2. **Label Nodes for Specific Component Placement**
3. **Create a Docker Stack File**
4. **Deploy the Application**
5. **Monitor and Troubleshoot Using Docker Logs**
6. **Modify and Redeploy the Application**

Step-by-Step Guide

1. Initialize Docker Swarm and Join Worker Nodes

On the manager node, initialize Docker Swarm:

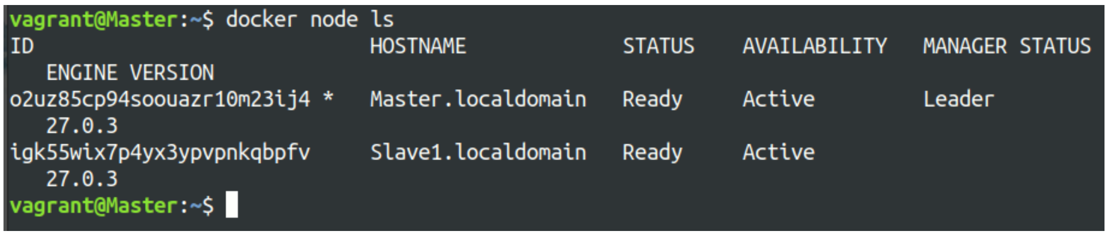
```
docker swarm init --advertise-addr <MANAGER-IP>
```

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
```



```
vagrant@Master:~$ docker node ls
ID                            HOSTNAME             STATUS    AVAILABILITY   MANAGER STATUS
   ENGINE VERSION
o2uz85cp94soouazr10m23ij4 *   Master.localdomain   Ready     Active         Leader
   27.0.3
igk55wix7p4yx3ypvpnkqbpfv     Slave1.localdomain   Ready     Active
   27.0.3
vagrant@Master:~$ 
```

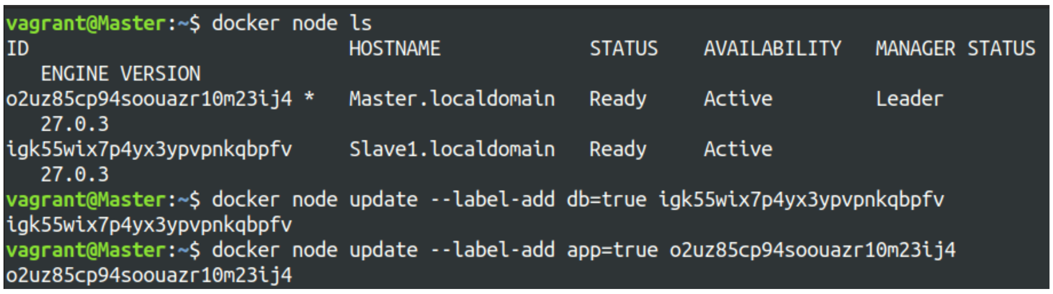
2. Label Nodes for Specific Component Placement

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:

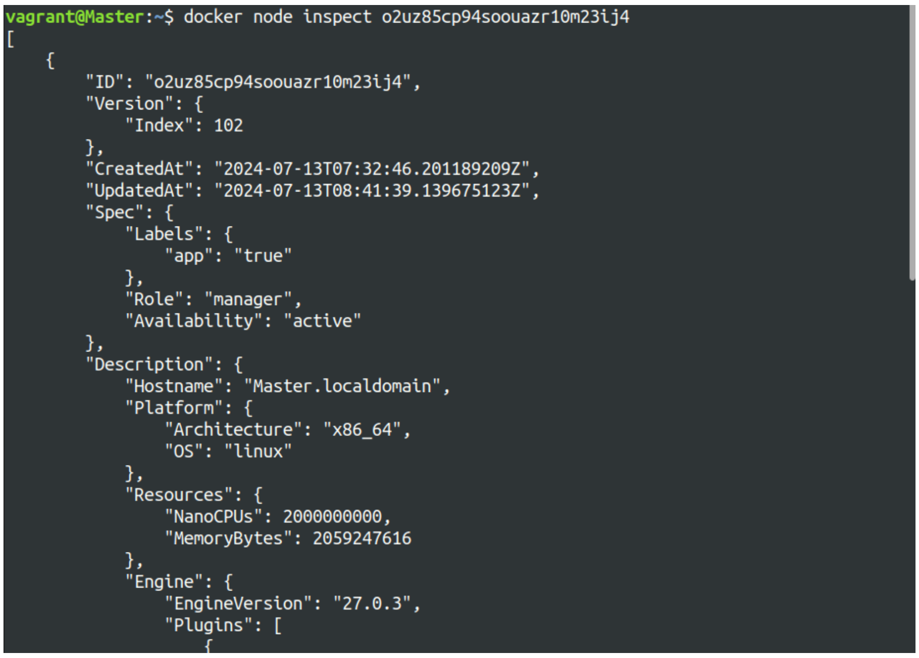
```
docker node update --label-add app=true <NODE-ID>
```



```
vagrant@Master:~$ docker node ls
ID                            HOSTNAME              STATUS    AVAILABILITY   MANAGER STATUS
   ENGINE VERSION
o2uz85cp94soouazr10m23ij4 *   Master.localdomain    Ready     Active         Leader
   27.0.3
igk55wix7p4yx3ypvpnkqbpfv     Slave1.localdomain    Ready     Active
   27.0.3
vagrant@Master:~$ docker node update --label-add db=true igk55wix7p4yx3ypvpnkqbpfv
igk55wix7p4yx3ypvpnkqbpfv
vagrant@Master:~$ docker node update --label-add app=true o2uz85cp94soouazr10m23ij4
o2uz85cp94soouazr10m23ij4
```

Verify the labels:

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



```
vagrant@Master:~$ docker node inspect o2uz85cp94soouazr10m23ij4
[
    {
        "ID": "o2uz85cp94soouazr10m23ij4",
        "Version": {
            "Index": 102
        },
        "CreatedAt": "2024-07-13T07:32:46.201189209Z",
        "UpdatedAt": "2024-07-13T08:41:39.139675123Z",
        "Spec": {
            "Labels": {
                "app": "true"
            },
            "Role": "manager",
            "Availability": "active"
        },
        "Description": {
            "Hostname": "Master.localdomain",
            "Platform": {
                "Architecture": "x86_64",
                "OS": "linux"
            },
            "Resources": {
                "NanoCPUs": 2000000000,
                "MemoryBytes": 2059247616
            },
            "Engine": {
                "EngineVersion": "27.0.3",
                "Plugins": [
```

3. Create a Docker Stack File

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

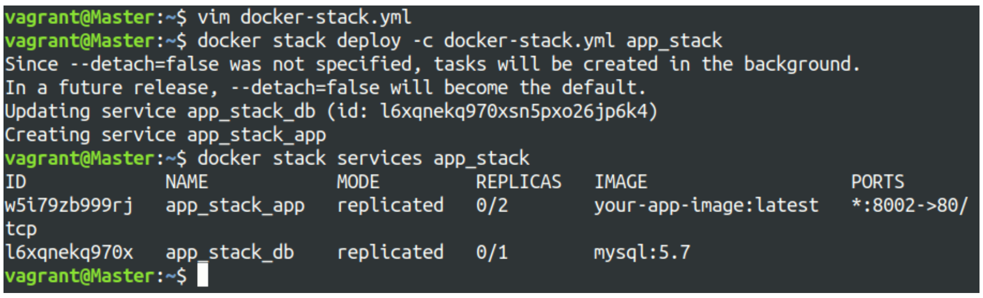
```
version: '3.8'

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
```



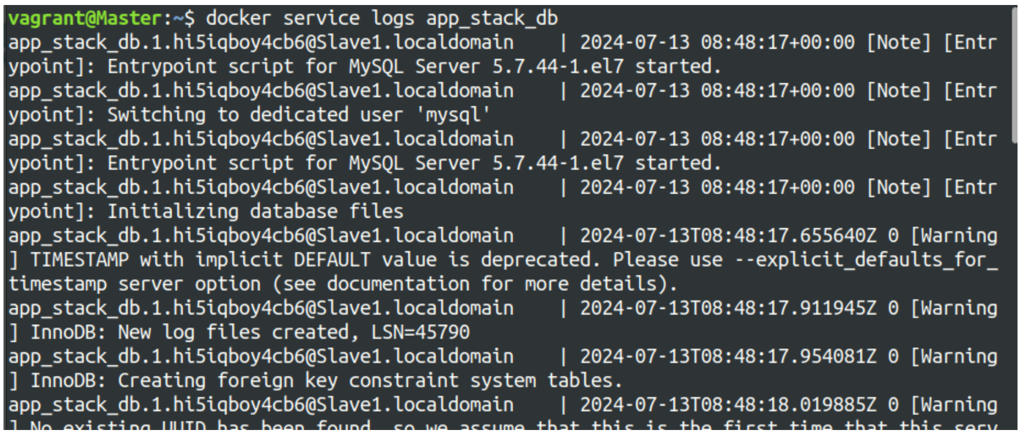
```
vagrant@Master:~$ vim docker-stack.yml
vagrant@Master:~$ docker stack deploy -c docker-stack.yml app_stack
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Updating service app_stack_db (id: l6xqnekq970xsn5pxo26jp6k4)
Creating service app_stack_app
vagrant@Master:~$ docker stack services app_stack
ID             NAME            MODE         REPLICAS   IMAGE                     PORTS
w5i79zb999rj   app_stack_app   replicated   0/2        your-app-image:latest     *:8002->80/
tcp
l6xqnekq970x   app_stack_db    replicated   0/1        mysql:5.7
vagrant@Master:~$
```

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
```



```
vagrant@Master:~$ docker service logs app_stack_db
app_stack_db.1.hi5iqboy4cb6@Slave1.localdomain    | 2024-07-13 08:48:17+00:00 [Note] [Entr
ypoint]: Entrypoint script for MySQL Server 5.7.44-1.el7 started.
app_stack_db.1.hi5iqboy4cb6@Slave1.localdomain    | 2024-07-13 08:48:17+00:00 [Note] [Entr
ypoint]: Switching to dedicated user 'mysql'
app_stack_db.1.hi5iqboy4cb6@Slave1.localdomain    | 2024-07-13 08:48:17+00:00 [Note] [Entr
ypoint]: Entrypoint script for MySQL Server 5.7.44-1.el7 started.
app_stack_db.1.hi5iqboy4cb6@Slave1.localdomain    | 2024-07-13 08:48:17+00:00 [Note] [Entr
ypoint]: Initializing database files
app_stack_db.1.hi5iqboy4cb6@Slave1.localdomain    | 2024-07-13T08:48:17.655640Z 0 [Warning
] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --explicit_defaults_for_
timestamp server option (see documentation for more details).
app_stack_db.1.hi5iqboy4cb6@Slave1.localdomain    | 2024-07-13T08:48:17.911945Z 0 [Warning
] InnoDB: New log files created, LSN=45790
app_stack_db.1.hi5iqboy4cb6@Slave1.localdomain    | 2024-07-13T08:48:17.954081Z 0 [Warning
] InnoDB: Creating foreign key constraint system tables.
app_stack_db.1.hi5iqboy4cb6@Slave1.localdomain    | 2024-07-13T08:48:18.019885Z 0 [Warning
] No existing UUID has been found, so we assume that this is the first time that this serv
```

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
```