# Summary

## Session No – 4

- WordPress is a product for managing content like blogs
- Deploying WordPress  in a container
    - Pulling WordPress image

```
[root@ip-172-31-40-68 ~]# docker pull  wordpress:latest
latest: Pulling from library/wordpress
bd159e379b3b: Already exists
```

- WordPress store its data in the MY-SQL database
- Almost in all the applications if a user connects the application and creates a post it gets stored in the database
- If a user connects to a web app and stores the data in a database this kind of architecture is known as Three-tier architecture
  USER → WebApp → Database
- WordPress needs a database to store the data of blogs
- Pulling MY-SQL image

```
[root@ip-172-31-40-68 ~]# docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
295ca2342728: Pulling fs layer
79af4312a7e0: Pulling fs layer
48d3d73d1704: Pulling fs layer
```

- First, we have to launch My-SQL then WordPress
- Launching MY-SQL database in a container

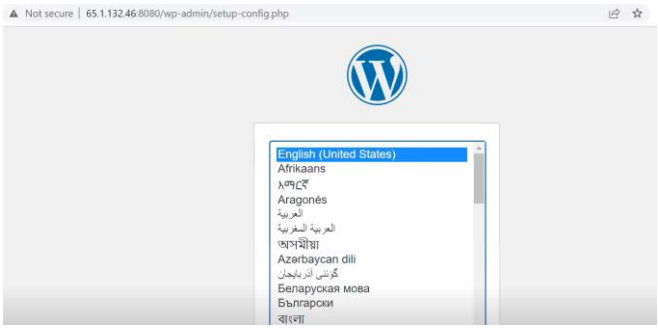
```
[root@ip-172-31-40-68 ~]# docker run -dit --name db  -e MYSQL_ROOT_PASSWORD=redhat -e   \
> MYSQL_DATABASE=mydb  -e MYSQL_USER=vimal  -e MYSQL_PASSWORD=redhat  mysql:latest
6829c0255c475483f98c654f22369586c0b4efea46533291ba499b3c4a254334
[root@ip-172-31-40-68 ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                 NAMES
6829c0255c47   mysql:latest   "docker-entrypoint.s…"   6 seconds ago   Up 5 seconds   3306/tcp, 33060/tcp   db
[root@ip-172-31-40-68 ~]#
```

- A container is by default isolated i.e. it does not have connectivity with the outside world
- Patting is done to expose the container to the outside world
- Word Press is written in the PHP language

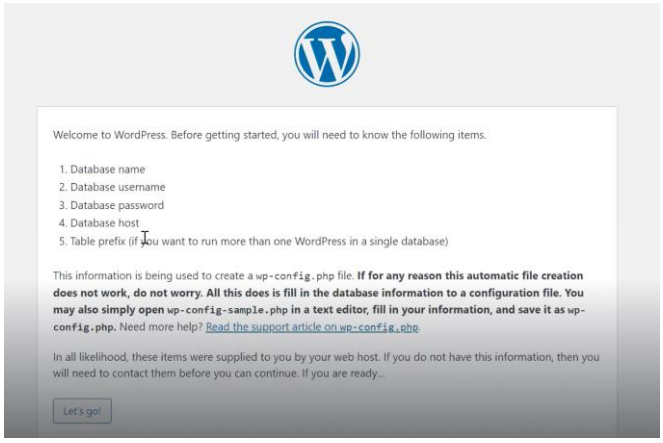
- Launching WordPress in a container

```
[root@ip-172-31-40-68 ~]# docker run -dit  --name mywp  -p  8080:80   wordpress:latest
f63c5e30f6e77e63cb760455b86820079dfa9e5b172c3705ec920286f5b7024c
[root@ip-172-31-40-68 ~]#
```

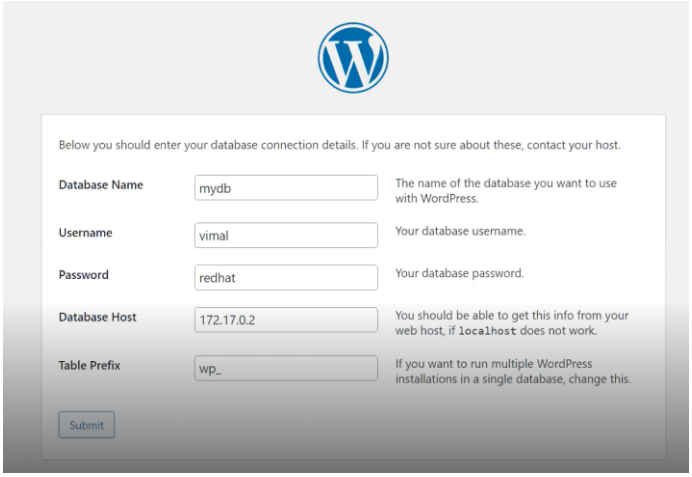
- Connecting to Word Press server or application




- Word press works only with database

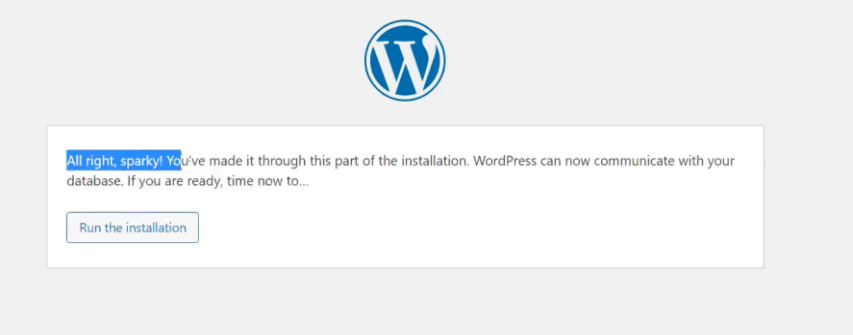



- Connecting Word press with MY-SQL  database

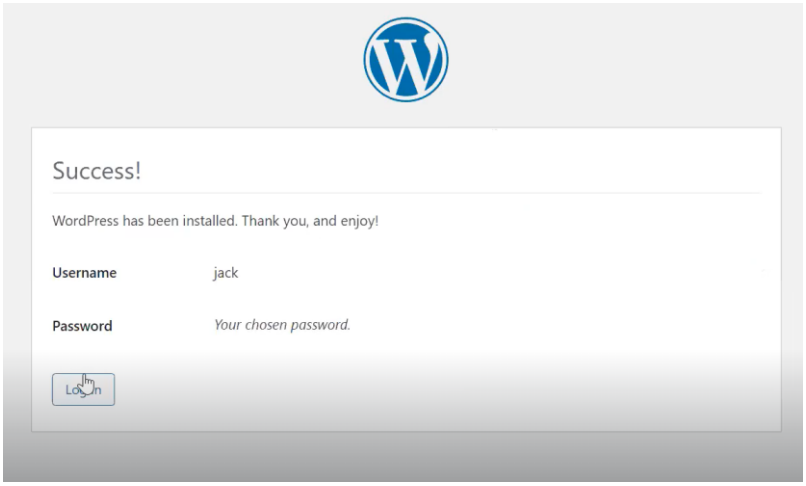



  - Database name, Username, and password → Given while launching MY-SQL container
  - Database host → Ip address of container running MySql

- Installing Word Press




- Login Word press




- All the look-end fields of the website are coming from the word press container and the data is coming from the MY-SQL database container

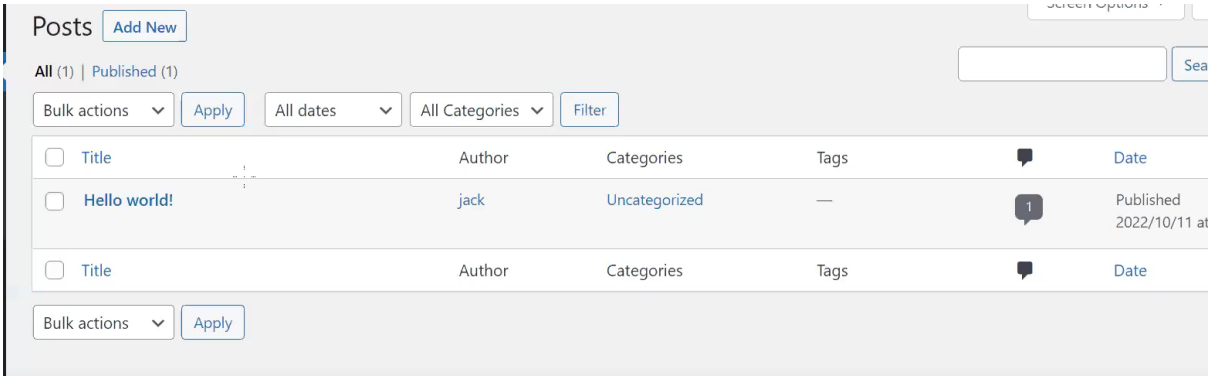

- Deleting the database container

```
[root@ip-172-31-40-68 ~]# docker rm -f db
db
[root@ip-172-31-40-68 ~]# docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS         PORTS
          NAMES
85c630e5ab7c   wordpress:latest   "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:8080->80/tcp, :::808
0->80/tcp   mywp
[root@ip-172-31-40-68 ~]#
```

- If by chance the database container goes down good thing about docker is we can launch the container within a second

```
[root@ip-172-31-40-68 ~]# docker run -dit --name db  -e MYSQL_ROOT_PASSWORD=redhat -e   MYSQL_DATABASE=mydb  -e MYSQL_
USER=vimal  -e MYSQL_PASSWORD=redhat  mysql:latest
04d00ecb02f8f1544ac528dfa58079b860b6a0399ad1bc78802fe8c1a76d1944
```

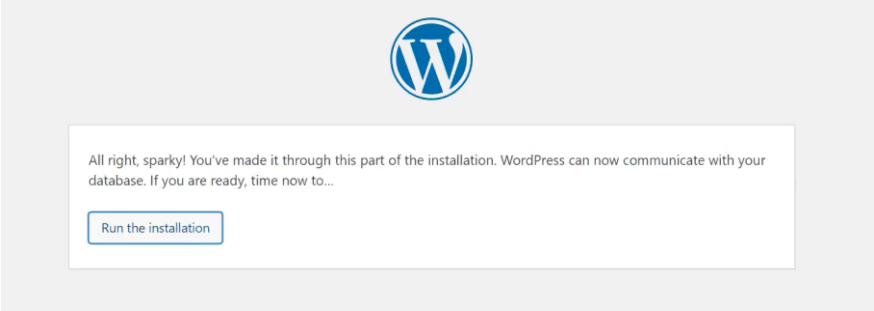
- As the database is deleted again it will ask to setup the word press and install word press
- The entire information of the data will be lost because the data in the database container was ephemeral




- Every service will store data in its directory e.g. HTTPd stores the data in /var/www/html
- My-SQL server stores data in **/var/lib/mysql**
- Launching My-SQL database container with persistence storage

```
[root@ip-172-31-40-68 ~]# mkdir /mydata
[root@ip-172-31-40-68 ~]# docker run -dit --name db  -e MYSQL_ROOT_PASSWORD=redhat -e   MYSQL_DATABASE=mydb  -e MYSQL_
USER=vimal  -e MYSQL_PASSWORD=redhat     -v   /mydata:/var/lib/mysql   mysql:latest
304855d3aef38632d30b5b91ffee46322215f11897ffe48f048ae5e465dbe15e
[root@ip-172-31-40-68 ~]#
```

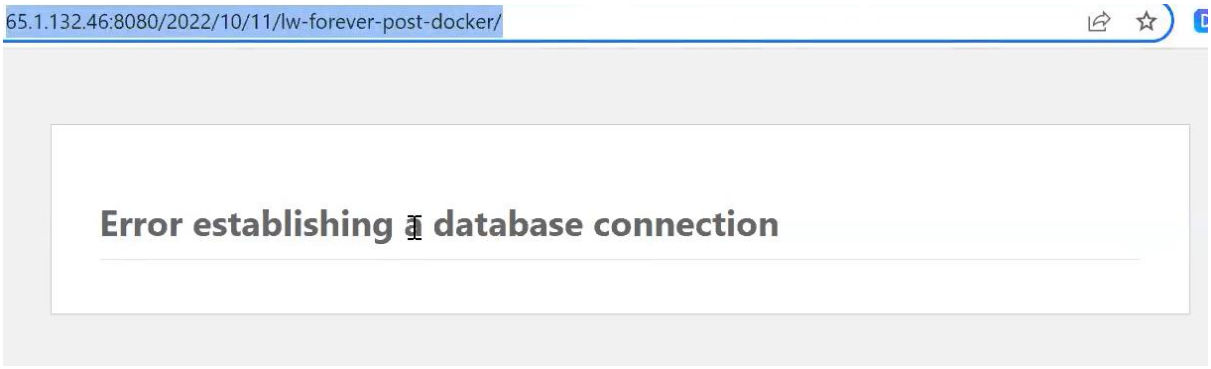
- Now the entire data stored in the database will be permanent
- Again connecting & installing the word press




- Now whatever My-SQL is storing is stored in the base system

```
[root@ip-172-31-40-68 ~]# cd /mydata/
[root@ip-172-31-40-68 mydata]# ls
304855d3aef3.err  binlog.index     client-key.pem     ibdata1       mydb          performance_schema  server-key.pem
auto.cnf          ca-key.pem       #ib_16384_0.dblwr  ibtmp1        mysql         private_key.pem     sys
binlog.000001     ca.pem           #ib_16384_1.dblwr  #innodb_redo  mysql.ibd     public_key.pem      undo_001
binlog.000002     client-cert.pem  ib_buffer_pool     #innodb_temp  mysql.sock    server-cert.pem     undo_002
[root@ip-172-31-40-68 mydata]#
```

- Even if we remove the container & launch again the data in the word press will be persistence
- If the IP Address of MySql changes after re-creating the container, word press keeps on hitting the same IP and there will be no connectivity




- Good network practice is if two operating systems want to connect with each other don't rely on IP address for the connection
- The one thing we never change in the container is the name
- Instead of using an IP address, we can use a name this concept is called container linking
- Linking container
    - --link keyword in the docker run command is used for linking

```
[root@ip-172-31-40-68 ~]# docker run -dit --name os1  ubuntu:14.04
59a615541c16214dd9f3367deb21235d45ec3fcd136f8b8e5475121fdf8b31fc
[root@ip-172-31-40-68 ~]#
[root@ip-172-31-40-68 ~]# docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED         STATUS         PORTS     NAMES
59a615541c16   ubuntu:14.04   "/bin/bash"   3 seconds ago   Up 2 seconds             os1
[root@ip-172-31-40-68 ~]# docker run -dit --name os2  --link  os1  ubuntu:14.04
ae31f1e3e64d1ad1f762fcda87540f244b3c7d124a35ce658dd4dc042196bd54
[root@ip-172-31-40-68 ~]# docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED          STATUS          PORTS     NAMES
ae31f1e3e64d   ubuntu:14.04   "/bin/bash"   9 seconds ago    Up 8 seconds              os2
59a615541c16   ubuntu:14.04   "/bin/bash"   36 seconds ago   Up 35 seconds             os1
```

- Pinging with the name of another container

```
root@ae31f1e3e64d:/# ping  os1
PING os1 (172.17.0.2) 56(84) bytes of data.
64 bytes from os1 (172.17.0.2): icmp_seq=1 ttl=64 time=0.074 ms
64 bytes from os1 (172.17.0.2): icmp_seq=2 ttl=64 time=0.056 ms
64 bytes from os1 (172.17.0.2): icmp_seq=3 ttl=64 time=0.057 ms
```

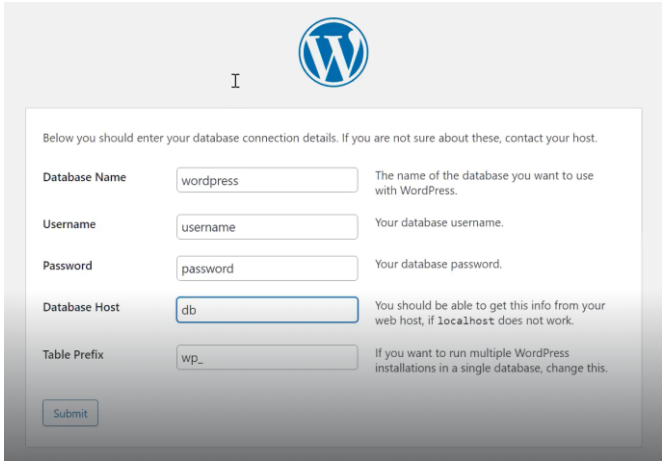
- The limitation of linking is it works in one way for example if os1 wants to connect to os2 it will not work

```
root@59a615541c16:/# ping  os2
ping: unknown host os2
root@59a615541c16:/# exit
exit
[root@ip-172-31-40-68 ~]# h
```

- Launching my-SQL and word press container and linking word press with My-SQL

```
[root@ip-172-31-40-68 ~]# docker run -dit --name db  -e MYSQL_ROOT_PASSWORD=redhat -e   MYSQL_DATABASE=mydb  -e MYSQL_
USER=vimal  -e MYSQL_PASSWORD=redhat    -v   /mydata:/var/lib/mysql   mysql:latest
d809d018c6eb8ae8d344cc1d5b4a086aa7173cada1296c577ea62faae1b9a819
[root@ip-172-31-40-68 ~]#
[root@ip-172-31-40-68 ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                    NAMES
d809d018c6eb   mysql:latest   "docker-entrypoint.s…"   3 seconds ago   Up 2 seconds   3306/tcp, 33060/tcp   db
[root@ip-172-31-40-68 ~]# docker run -dit  --name mywp  -p  8080:80  --link db  wordpress:latest
dccb48418f9a22329436c093147d84de7f65023ac5af8b3cd8c32ddc0f871a0c
```

- Now instead of the IP address, we have to give the name of the container




- Challenges in linking
  - One way
  - Internally linking converts name into IP address only so if the IP change connectivity is lost
- If the IP change two services will not be able to connect in a multi-tier architecture