Course 3 – Sprint 3: Database Dockerization

At the end of this course, we have now understood what problem can docker solve.

- Docker solves the problem of having identical environments across various stages of development and having isolated environments for your individual applications.
- Environment setup and management is a tedious task in every project. Docker provides a solution to this problem with containerization.
- Earlier, we used to create virtual machines, and each VM had an OS which took a lot of space and made it heavy. Now in docker container's case, you have a single OS, and the resources are shared between the containers.

Virtualization:

- It's an act of creating a virtual (rather than actual) version of something including virtual computer hardware platforms, storage devices, and computer network resources.
- It's a technique of importing a guest OS on the top of host OS.

Advantages:

1. Multiple OS in the same machine
2. Easy maintenance and recovery
3. Lower total cost of ownership

Disadvantages:

1. Multiple VMs lead to unstable performance.
2. Hypervisors are not as efficient as a host OS.
3. Long boot up process (Approximate 1 minute).
4. This is using the host system resources.
5. This is overly critical in case of real time applications where fast processing is required.
6. Scaling the no of VM's is tedious and costly affair.

What is containerization?

- Containerization means, that your application runs in an isolated container, that is an explicitly defined, reproduceable and portable environment.
- With Docker you ship the operating environment along with your application.

- One of the main benefits of using Docker, and container technology, is the portability of applications. It is possible to spin up an application on-premises or in a public cloud environment in a matter of minutes.

Docker:

- Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers.
- Docker is a container management service. The whole idea of Docker is to develop, ship & run anywhere.

Docker Containers:

- Containers are isolated from one another and bundle their own software, libraries and configuration files. They can communicate with each other through well-defined channels.

Download and Install Docker:

Pls find the below link for downloading Docker and install it.

https://desktop.docker.com/win/stable/amd64/Docker%20Desktop%20Installer.exe?utm_source=docker&utm_medium=webreferral&utm_campaign=docs-driven-download-win-amd64

For Linux:

https://docs.docker.com/engine/install/

For MacOS

https://desktop.docker.com/mac/stable/arm64/Docker.dmg?utm_source=docker&utm_medium=webreferral&utm_campaign=docs-driven-download-mac-arm64

To run any application, open a command prompt or bash window, and run the command:

```
docker run -d -p 80:80 docker/app or docker run -dp 80:80 docker/app
```

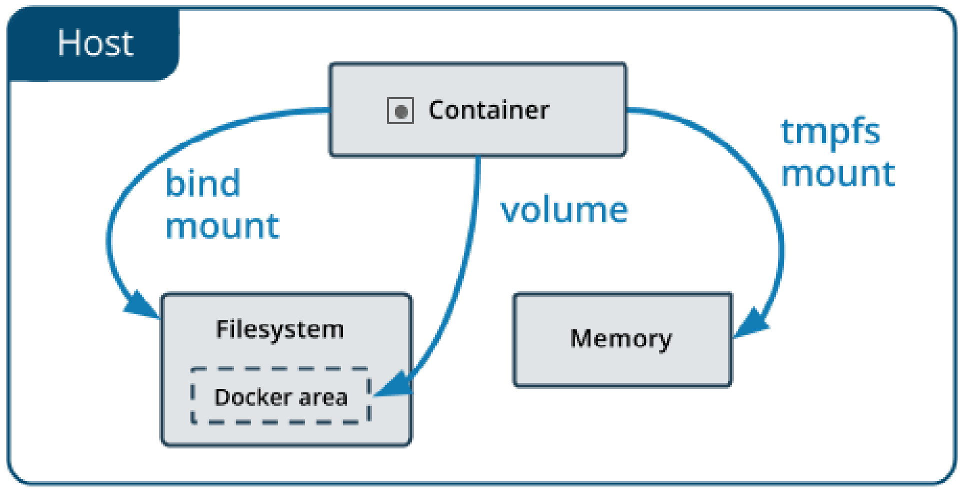
You'll notice a few flags being used. Here's some more info on them:

- -d - run the container in detached mode (in the background)

- -p 80:80 - map port 80 of the host to port 80 in the container
- docker/getting-started - the image to use

Manage Data in Docker:

- By default, all files created inside a container are stored on a writable container layer.
- Docker has two options for containers to store files in the host machine, so that the files are persisted even after the container stops: volumes and bind mounts.
- Volumes are stored in a part of the host filesystem which is managed by Docker (/var/lib/docker/volumes/ on Linux).
- Non-Docker processes should not modify this part of the filesystem. Volumes are the best way to persist data in Docker.



Start a container with a volume:

If you start a container with a volume that does not yet exist, Docker creates the volume for you. The following example mounts the volume myvol2 into /app/ in the container.

```
$ docker run -d   --name devtest  -v myvol2:/app  nginx: latest
```

Use docker inspect devtest to verify that the volume was created and mounted correctly. Look for the Mounts section:

```
"Mounts": [    {        "Type": "volume",        "Name": "myvol2",        "Source":
"/var/lib/docker/volumes/myvol2/_data",        "Destination": "/app",        "Driver": "local",
"Mode": "",        "RW": true,        "Propagation": ""    }],
```

This shows that the mount is a volume, it shows the correct source and destination, and that the mount is read-write.

Stop the container and remove the volume. Note volume removal is a separate step.

```
$ docker container stop devtest
$ docker container rm devtest
$ docker volume rm myvol2
```

Docker Basic Commands:

```
docker version / docker –v / docker --version
docker info
docker --help
docker login
```

Images

docker images	To list the images
docker images -q	To list the images -ids
docker pull	To pull an image
docker rmi	To remove one or more images
docker rmi $(docker images -q)	To remove all images

Containers

docker ps	shows only running containers
docker ps -a	shows all containers
docker rm <container-name>	remove one or more containers
docker run <image-name>	create a container from image
docker start -i <container-name>	To start the stopped container
docker stop <container-name>	To stop running container
docker logs <container-name>	To access logs of container
docker exec -it <container-name> bash	To interact with container
docker run --name a-centos -it centos bash	container will not be removed on exit
docker run --name b-centos --rm -it centos bash	container will be removed on exit

For all the other detailed command please refer to following link :
https://docs.docker.com/engine/reference/commandline/cli/

Sample applications can be found in below link:
https://docs.docker.com/get-started/02_our_app/

Using the MongoDB image

```
$ docker run -p 27017:27017 --name mongo_instance_001 -d my/repo
$ docker run -p 27017:27017 --name mongo_instance_001 -d my/repo – smallfiles
$ docker logs mongo_instance_001
$ mongo --port 27017
$ mongodb –port 27017 host 192.168.59.103
```

For dockerizing Spring application:

https://www.baeldung.com/dockerizing-spring-boot-application