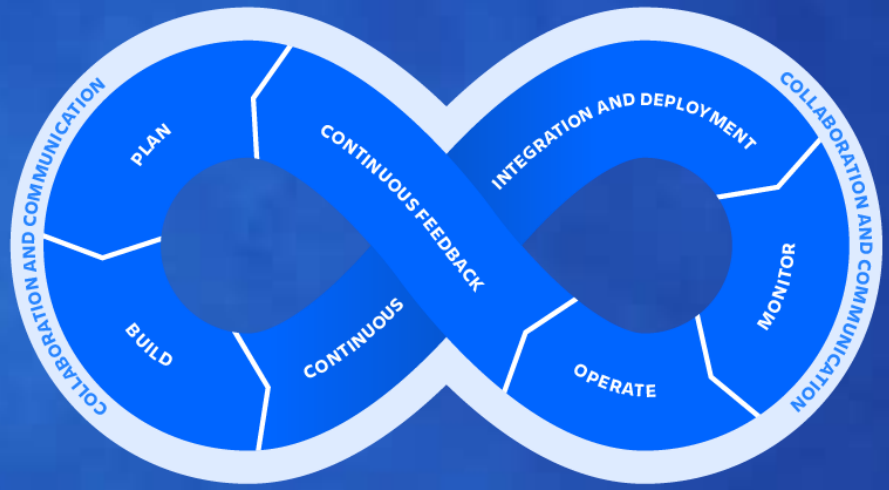

Session 11. Docker Installation

Ram N Sangwan

Agenda

- Installing Docker on Linux.
- Understanding Installation of Docker on windows.
- Some Docker commands.

Installing Docker

- The Docker installation packages are available in the official Linux repository. # yum install docker-engine
- After installation has completed, start the Docker daemon:

  # systemctl start docker
- Verify that it’s running:

  # systemctl status docker

Docker Desktop on Windows

Docker Desktop for Windows is the Community version of Docker for Windows.

System Requirements

- Hyper-V backend and Windows containers
  - Windows 10 64-bit: Pro, Enterprise, or Education (Build 17134 or higher).
  - Hyper-V and Containers Windows features must be enabled.
- WSL 2 backend
  - Windows 10 64-bit: Home, Pro, Enterprise, or Education, version 1903 (Build 18362 or higher).
  - Enable the WSL 2 feature on Windows.
- Common H/W Requirements
  - 64-bit processor with Second Level Address Translation (SLAT)
  - 4GB system RAM
  - BIOS-level hardware virtualization support must be enabled in the BIOS settings.

Installation Steps for Windows

1. If you haven't already downloaded the installer (Docker Desktop Installer.exe), you can get it from Docker Hub.
2. Run Docker Desktop Installer.
3. When prompted, ensure the Enable Hyper-V Windows Features or the Install required Windows components for WSL 2 option is selected on the Configuration page.
4. Follow the instructions on the installation wizard to authorize the installer and proceed with the install.
5. When the installation is successful, click Close.
6. If your admin account is different to your user account, you must add the user to the docker-users group. Run Computer Management as an administrator and navigate to Local Users and Groups > Groups > docker-users. Right-click to add the user to the group. Log out and log back in for the changes to take effect.

Using the Docker Command

- Using docker consists of passing it a chain of options and subcommands followed by arguments:

  # docker [option] [command] [arguments]

- To view all available subcommands, type:

  # docker

```
[root@localhost ~]# docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/root/
  -c, --context string     Name of the context to use to connect to the daem
                           (overrides DOCKER_HOST env var and default conte
                           "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"er
                           (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "/root/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/root/.do
      --tlskey string      Path to TLS key file (default "/root/.docker/key
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  builder     Manage builds
```

System Information

- To view system-wide information, use:

  # docker info

```
[root@localhost ~]# docker info
Client:
 Debug Mode: false

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 19.03.9
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
```

Working with Docker Containers

- Docker containers are run from Docker images.
- By default, it pulls these images from Docker Hub.
- Anybody can build and host their Docker images on Docker Hub.
- To check whether you can access and download images from Docker Hub:

  # docker run hello-world

```
[root@localhost ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:6a65f928fb91fcfbc963f7aa6d57c8eeb426ad9a20c7ee045538ef34847f44f1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Running a Docker Container

- Containers can be interactive.
- As an example, let's run a container using the latest image of mysql-server. The combination of the -i and -t switches gives you interactive shell access into the container:

  # docker run --name=mysql1 -d mysql/mysql-server:8.0

```
[root@localhost ~]# docker run --name=mysql1 -d mysql/mysql-server:8.0
Unable to find image 'mysql/mysql-server:8.0' locally
8.0: Pulling from mysql/mysql-server
0e690826fc6e: Pull complete
0e6c49086d52: Pull complete
862ba7a26325: Pull complete
7731c802ed08: Pull complete
Digest: sha256:a82ff720911b2fd40a425fd7141f75d7c68fb9815ec3e5a5a881a8eb49677087
Status: Downloaded newer image for mysql/mysql-server:8.0
3a5f12589cacc581744ebfb81255a45ef71e52f6261cd895001f710493de0d18
```

Verify Running Container

- Initialization for the container begins, and the container appears in the list of running containers when you run the **docker ps** command. For example:

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE                     COMMAND                  CREATED              STATUS                          PORTS
         NAMES
3a5f12589cac        mysql/mysql-server:8.0    "/entrypoint.sh mysq…"   About a minute ago   Up About a minute (healthy)     3306/tc
60/tcp   mysql1
[root@localhost ~]#
```

- The -d option used in the docker run command above makes the container run in the background. Use this command to monitor the output from the container:

docker logs mysql1

```
2020-05-27T11:45:34.428930Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self sign
2020-05-27T11:45:34.502356Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connect
lib/mysql/mysql.sock'  port: 0  MySQL Community Server - GPL.
Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/leapseconds' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/tzdata.zi' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.
[Entrypoint] GENERATED ROOT PASSWORD: Cac;ecIHLIs80sKOn1Ow#ic4sEm

[Entrypoint] ignoring /docker-entrypoint-initdb.d/*

ystem] [MY-013172] [Server] Received SHUTDOWN from user root.
stem] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete
```

```
[root@localhost ~]# docker logs mysql1 2>&1 | grep 'GENERATED ROOT PASSWORD'
[Entrypoint] GENERATED ROOT PASSWORD: Cac;ecIHLIs80sKOn1Ow#ic4sEm
[root@localhost ~]#
```

Connecting to MySQL Server from within Container

- Use the docker exec -it command to start a mysql client inside the Docker container you have started, like the following:

  # docker exec -it mysql1 mysql -uroot -p

```
[root@localhost ~]# docker exec -it mysql1 mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 24
Server version: 8.0.20
```

```
mysql> alter user root@localhost identified by 'Sangwan#12';
Query OK, 0 rows affected (0.02 sec)
```

Container Shell Access

- To have shell access to your MySQL Server container, use the docker exec -it command to start a bash shell inside the container:

  # docker exec -it mysql1 bash

  bash-4.2#

Stopping and Deleting a Container

- To stop the MySQL Server container, use:

  # docker stop mysql1

- Notice that when the main process of a container is stopped, the Docker container stops automatically.
- To start the MySQL Server container again:

  # docker start mysql1

- To stop and start again the MySQL Server container with a single command:

  # docker restart mysql1

- To delete the MySQL container, stop it first, and then use the ***docker rm*** :

  # docker stop mysql1

  # docker rm mysql1

Listing Docker Containers

- To view all containers — active and inactive, pass it the -a switch:

  # docker ps -a
- To view the latest container you created, pass it the -l switch:

  # docker ps -l
- Stopping a running or active container is as simple as typing:

  # docker stop container-id

Clean UP

- Check all Containers with
  # docker ps -a
- Stop Running Containers with
  # docker stop <Container ID>
- Delete Containers with
  # docker rm <Container ID>
- List all Images
  # docker images
- Remove all Images
  # docker rmi -f $(docker images -a -q)

Create and manage volumes

- Unlike a bind mount, you can create and manage volumes outside the scope of any container.
- **Create a volume:**

$ docker volume create my-vol

List volumes:

$ docker volume ls

local my-vol

Inspect a volume:

```
$ docker volume inspect my-vol
[
    {
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/my-vol/_data",
        "Name": "my-vol",
        "Options": {},
        "Scope": "local"
    }
]
```

Remove a volume:

$ docker volume rm my-vol

Thank You