Jenkins:-

Jenkins is a powerful automation server that facilitates continuous integration and continuous delivery (CI/CD) in software development. Its advantages include streamlining the build process, enhancing collaboration among teams, and improving code quality through automated testing and deployment.

“Jenkins is free & open source software is used to automate build deployment process, using jenkins we can implement CI/CD”

Build & Deployment Process:-

- **Take latest source code from git hub**
- **Compile project source code**
- **Execute code Review using SonarQube**
- **Package the application(JAR/WAR)**
- **Build Docker image**
- **Create docker container**

Note:- Using Jenkins we can do automated, just we need to write jenkins pipeline

Continuous Integration : When code changes happen it should be ready to test
Continuous Delivery : Keep it ready to release
Continuous Deployment : Deploy the project to production

Docker:-

- **Docker is an open source platform for developing, shipping and running applications in containers.**
- **Containers are lightweight, isolated environments that package applications and their dependencies**

What is Docker?

- **Docker is an open-source platform that automates the deployment, scaling, and management of applications using containerization.**

- **It allows developers to package applications with all their dependencies into standardized units called containers.**

Why Use Docker?

- **Portability: Docker containers can run on any system that supports Docker, ensuring consistent environments across development, testing, and production.**
- **Efficiency: Containers share the host OS kernel, making them lightweight and faster to start compared to traditional virtual machines.**
- **Isolation: Each container runs in its own environment, ensuring that applications do not interfere with each other, which enhances security and stability.**
- **Scalability: Docker makes it easy to scale applications up or down by adding or removing containers as needed.**
- **Simplified Deployment: With Docker, applications can be deployed quickly and reliably, reducing the time from development to production.**
- **Microservices Architecture: Docker supports microservices, allowing developers to build applications as a collection of loosely coupled services, which can be developed, deployed, and scaled independently.**

Note:- Container is an virtual machine

Dockerfile : which is used to create the image

Docker image : it is a package code + libraries + dependencies will be available

Docker Hub(registry) : The place where we are going to store the docker images

Container : container is a place where our application is going to execute

Docker Commands:

1. docker --version

   Description: **Displays the installed version of Docker.**

   Use Case: **To check if Docker is installed and to verify the version.**

2. docker pull <image>

   Description: **Downloads a Docker image from Docker Hub or a specified registry.**

   Use Case: **To obtain a pre-built image for use in your local environment, e.g.,** `docker pull ubuntu`**.**

3. docker build -t <image_name>:<tag> <path>

   Description: **Builds a Docker image from a Dockerfile located at the specified path.**

   Use Case: **To create a custom image for your application, e.g.,** `docker build -t myapp:latest ...`

4. docker images

   Description: **Lists all Docker images on the local machine.**

   Use Case: **To view available images and their sizes.**

5. docker rmi <image>

   Description: **Removes a specified Docker image from the local machine.**

   Use Case: **To free up space by deleting unused images, e.g.,** `docker rmi myapp:latest`**.**

6. docker run <options> <image>

   Description: **Creates and starts a container from a specified image.**

   Use Case: **To run an application in a container, e.g.,** `docker run -d -p 80:80 nginx`**.**

7. docker stop <container_id>

   Description: **Stops a running container.**

   Use Case: **To gracefully stop an application running in a container, e.g.,** `docker stop my_container`**.**

8. docker start <container_id>

   Description: **Starts a stopped container.**

   Use Case: **To restart a previously stopped container, e.g.,** `docker start my_container`**.**

9. docker exec -it <container_id> <command>

   Description: **Executes a command inside a running container.**

   Use Case: **To access a shell in a running container for debugging, e.g.,** `docker exec -it my_container /bin/bash`**.**

10. docker logs <container_id>

    Description: **Displays the logs of a specified container.**

    Use Case: **To view the output and error logs of an application running in a container.**

11. docker network ls

    Description: **Lists all Docker networks.**

    Use Case: **To view the available networks for container communication.**

12. docker volume ls

    Description: **Lists all Docker volumes.**

    Use Case: **To check the available volumes for persistent data storage.**

13. docker-compose up

    Description: **Starts up all services defined in a** `docker-compose.yml` **file.**

    Use Case: **To run multi-container applications defined in a single configuration file.**

14. docker-compose down

    Description: **Stops and removes all containers defined in a** `docker-compose.yml` **file.**

    Use Case: **To clean up resources after running a multi-container application.**

15. docker inspect <container_id>

    Description: **Displays detailed information about a container or image.**

    Use Case: **To retrieve configuration and state information for debugging purposes.**

16. docker commit <container_id> <new_image_name>

    Description: **Creates a new image from a container's changes.**

    Use Case: **To save the current state of a container as a new image.**

17. docker prune

    Description: **Removes unused data, including stopped containers, unused networks, and dangling images.**

    Use Case: **To clean up your Docker environment and free up space.**

18. docker rm -f <container_id>

    Description: **Forcefully removes a specified container, regardless of its state.**

    Use Case: **To delete a container that is not needed anymore, even if it is running.**

19. docker compose up -d

    Description: **Starts all services defined in a `docker-compose.yml` file in detached mode.**

    Use Case: **To launch a multi-container application in the background without blocking the terminal.**

20. docker container prune -y

    Description: **Removes all stopped containers without prompting for confirmation.**

    Use Case: **To free up space by deleting containers that are no longer running.**

21. docker system prune -y

    Description: **Removes all unused data including stopped containers, unused networks, dangling images, and build cache without prompting.**

    Use Case: **To clean up the Docker environment and reclaim disk space.**

22. docker rm -f $(docker ps -aq)

    Description: **Forcefully removes all containers, both running and stopped.**

    Use Case: **To clean up all containers in one command, useful in development environments.**

23. docker rmi -f $(docker images -aq)

    Description: **Forcefully removes all Docker images from the local machine.**

    Use Case: **To clean up all images to reclaim disk space or reset the environment.**

24. docker volume prune -f

    Description: **Removes all unused Docker volumes without prompting for confirmation.**

    Use Case: **To free up space by deleting volumes not associated with any containers.**

25. docker network prune -f

    Description: **Removes all unused Docker networks without prompting.**

    Use Case: **To clean up unused networks and maintain a tidy Docker environment.**

MySQL and System Commands

1. `sudo lsof -i :3306`
    - Description: **Lists all open files and network connections associated with port 3306 (the default MySQL port).**
    - Use Case: **To identify which process is using port 3306, which can help troubleshoot issues related to MySQL or to free up the port.**
2. `sudo kill -9 <PID>`
    - Description: **Forcefully terminates a process with the specified Process ID (PID).**
    - Use Case: **To stop a process that is not responding or is blocking a port, such as MySQL, after identifying it with `lsof`.**
3. `sudo systemctl stop mysql`
    - Description: **Stops the MySQL service on the system.**
    - Use Case: **To gracefully stop the MySQL server, which is useful for maintenance or troubleshooting.**
4. `sudo fuser -k 3306/tcp`

- Description: **Kills all processes using TCP port 3306.**
- Use Case: **To quickly free up port 3306 by terminating any processes that are currently using it.**

```
docker rm -f $(docker ps -aq)          # Remove all containers (running + stopped)
docker rmi -f $(docker images -aq)     # Remove all images
docker volume prune -f                 # Remove all volumes
docker network prune -f                # Remove all unused networks
```