

## PRACTICAL

### Experiment : 06

Here are some basic Docker commands that are commonly used for content management:

1. **docker pull [image]**: Download a Docker image from a registry.
  - Example: docker pull nginx
2. **docker build -t [name] [path]**: Create a Docker image from a Dockerfile.
  - Example: docker build -t myapp .
3. **docker run [options] [image]**: Run a command in a new container.
  - Example: docker run -d -p 80:80 nginx
4. **docker ps**: List running containers.
  - Use docker ps -a to see all containers, including stopped ones.
5. **docker stop [container]**: Stop a running container.
  - Example: docker stop mycontainer
6. **docker rm [container]**: Remove a stopped container.
  - Example: docker rm mycontainer
7. **docker rmi [image]**: Remove an image from the local registry.
  - Example: docker rmi myapp
8. **docker exec [options] [container] [command]**: Run a command in a running container.
  - Example: docker exec -it mycontainer /bin/bash
9. **docker images**: List all Docker images on the local system.
10. **docker volume ls**: List all Docker volumes.

### Experiment -07 :

#### **Docker file : nano Dockerfile**

- # Use the official Nginx image from the Docker Hub
- FROM nginx
- # Copy your HTML files into the container at the default Nginx location
- COPY . /usr/share/nginx/html
- # Expose port 80 (default for Nginx)
- EXPOSE 80
- # Start Nginx
- CMD ["nginx", "-g", "daemon off;"]

#### **Docker build -t nginx .**

#### **Docker run -d -p 8082:80 nginx**

#### **Out put :**

```

vaishnavi_peddi@Vaishnavi:~/lab1$ nano Dockerfile
vaishnavi_peddi@Vaishnavi:~/lab1$ nano index.html
vaishnavi_peddi@Vaishnavi:~/lab1$ nano Dockerfile
vaishnavi_peddi@Vaishnavi:~/lab1$ docker build -t nginx .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
             Install the buildx component to build images with BuildKit:
             https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.072kB
Step 1/4 : FROM nginx
--> 9bb6f1c40fb8
Step 2/4 : COPY index.html /usr/share/nginx/html
--> d78c20e8a748
Step 3/4 : EXPOSE 80
--> Running in e9fdf6b4cda1
Removing intermediate container e9fdf6b4cda1
--> 675ab50c16e8
Step 4/4 : CMD ["nginx", "-g", "daemon off;"]
--> Running in 6024854bb0d3
Removing intermediate container 6024854bb0d3
--> 6e5967913f70
Successfully built 6e5967913f70
Successfully tagged nginx:latest

```

### **Experiment-05 :**

To demonstrate continuous integration and development using Jenkins, follow these steps:

1. **Install Jenkins:** Download Jenkins from the official site and install it on your server or computer.
2. **Start Jenkins:** Launch Jenkins by starting the Jenkins service. Usually, it runs on <http://localhost:8080> by default.
3. **Install Required Plugins:** In Jenkins, go to Manage Jenkins -> Manage Plugins and install plugins like Git, GitHub, and Maven if needed.
4. **Create a New Job:** Go to Jenkins Dashboard, click on "New Item," enter an item name, select "Freestyle project," and click OK.
5. **Configure Job:** In the job configuration:
  - Source Code Management: Select Git and provide the repository URL.
  - Build Triggers: Enable "Poll SCM" or "Build when a change is pushed to GitHub" to automate builds.
  - Build Environment: Set up the environment if needed.
  - Build: Add build steps, such as executing a shell script or running Maven goals.
6. **Add Post-build Actions:** Set up actions like sending build notifications or deploying artifacts.
7. **Save and Build:** Save the job configuration and click "Build Now" to start the build process.
8. **Monitor Build Process:** View the build status and console output in Jenkins to ensure everything runs smoothly.

By following these steps, you integrate Jenkins into your development workflow, automating testing and deployment processes, thus streamlining continuous integration and development.

1.