1. **Version Control Using Git in Browser**

Lab Steps:

* 1. Sign Up at [GitHub](https://github.com/) or [GitLab.](https://gitlab.com/)
  2. Create a New Repository:
     + Click New Repository.
     + Provide a name and description.
     + Add a README.md file.
  3. Upload Files Manually:
     + Click Add File > Upload File.
     + Select files and commit changes.
  4. Modify Files Online:
     + Open a file and click Edit.
     + Make changes and commit.
  5. Check File History:
     + Open a file and click History.

**Expected Output**: A version-controlled project with a history of changes.

1. Version Control Using Git with Push and Pull Commands Objective: Use Git for version control from the terminal.

Lab Steps:

* 1. **Install Git (if not installed):**

🡪sudo apt update

🡪sudo apt install git -y

* 1. **Configure Git:**

🡪git config --global user.name "Your Name"

🡪git config --global user.email "you@example.com"

* 1. **Clone an Existing Repository:**

🡪git clone <repository\_url>

🡪cd <repo\_name>

* 1. **Modify a File & Commit Changes:**

🡪echo "New changes" > file.txt

🡪git add .

🡪git commit -m "Updated file"

* 1. **Push to Remote Repository:**

🡪git push origin main

* 1. **Pull Latest Changes:**

🡪git pull origin main

**Expected Output**: Changes should be reflected on GitHub/GitLab.

**2. Install Docker, explore its containerization commands, create Docker container using various operating system images and deploy containerized applications using Docker and Docker hub.**

STEP 1: Install Docker on Windows

1. Go to the official Docker website:  
   👉 <https://www.docker.com/products/docker-desktop>
2. Download Docker Desktop for Windows.
3. Install it and restart your system if prompted.
4. Open Docker Desktop and wait until it says: "Docker is running."

STEP 2: Explore Docker Commands

🡪O**pen Command Prompt or PowerShell or Git Bash, and run:**

docker --version # Check Docker version

docker info # System-wide info

docker ps # List running containers

docker ps -a # List all containers (even stopped)

docker images # List downloaded Docker images

🡪**Pull and run OS containers:**

docker pull ubuntu # Pull Ubuntu image

docker run -it ubuntu # Run Ubuntu container interactively

exit # Exit the container

docker pull alpine # Lightweight OS

docker run -it alpine

exit

docker pull centos # Pull CentOS

docker pull debian # Pull Debian

docker pull fedora # Pull Fedora

🡪**Stopping and cleaning up:**

docker ps -a # Find container ID

docker stop <container\_id> # Stop container

docker rm <container\_id> # Remove container

docker rmi <image\_id> # Remove image

STEP 3: Create Docker Containers Using OS Images

You’ve already done most of this in Step 2. Just remember:

* Use docker run -it <image> to explore that OS.
* Exit using exit.

You can try Ubuntu, Alpine, Debian, CentOS, Fedora as mentioned.

**STEP 4: Deploy a Containerised Web Application (Python Flask App)**

1. **Create a folder and go into it:**

mkdir flask-docker-app

cd flask-docker-app

2. **Create a file called app.py with this content:**

**Inside nano app.py**

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Hello, Docker!'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=5000)

3. **Create a file called Dockerfile with this content:**

FROM python:3.8-slim

WORKDIR /app

COPY app.py .

RUN pip install flask

EXPOSE 5000

CMD ["python", "app.py"]

**4. Build the Docker image:**

docker build -t python-flask-app .

**5. Run the container:**

docker run -p 5000:5000 python-flask-app

Now go to [http://localhost:5000](http://localhost:5000/) in your browser.  
You should see: "Hello, Docker!"

**STEP 5: Push Docker Image to Docker Hub**

1. Create a Docker Hub account (if not already):

👉 [https://hub.docker.com](https://hub.docker.com/)

2. Log in from terminal:

docker login

Enter your Docker Hub username and password.

3. Tag your image:

docker tag python-flask-app <your\_username>/python-flask-app

Example:

docker tag python-flask-app nstudent/python-flask-app

4. Push to Docker Hub:

docker push <your\_username>/python-flask-app

5. To test pulling:

docker pull <your\_username>/python-flask-app

**3. Design, Deploy and manage a micro services architecture on your local machine using Docker and Docker-compose.**

**Step 0: Prerequisites**

**Install Docker and Docker Compose**

docker --version

Install Docker Compose (comes with Docker now, but just in case):

sudo apt install docker-compose -y

Verify:

docker-compose --version

**Step 1: Create the Folder Structure**

In your home directory or any location:

mkdir -p microservices-lab/backend

mkdir -p microservices-lab/frontend

cd microservices-lab

touch docker-compose.yml

**Step 2: Backend Setup**

Navigate to backend folder:

cd backend

**🔹 nano Dockerfile**

FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt /app/

RUN pip install --no-cache-dir -r requirements.txt

COPY . /app/

EXPOSE 5000

CMD ["python", "app.py"]

**🔹 requirements.txt**

Flask==2.1.1

Werkzeug==2.0.3

**🔹 app.py**

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Hello, World! From Backend!'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True, host='0.0.0.0', port=5000)

**Step 3: Frontend Setup**

Navigate to frontend folder:

cd ../frontend

**🔹 Dockerfile**

FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt /app/

RUN pip install --no-cache-dir -r requirements.txt

COPY . /app/

EXPOSE 5001

CMD ["python", "app.py"]

**🔹 requirements.txt**

Flask==2.1.1

requests==2.26.0

**🔹 app.py**

Create app.py:

from flask import Flask

import requests

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

response = requests.get('http://backend:5000') # Note: Use backend service name

return f"Frontend says: {response.text}"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True, host='0.0.0.0', port=5001)

**Step 4: docker-compose.yml**

Go back to the root folder:

cd ..

Create docker-compose.yml:

version: '3.8'

services:

backend:

build:

context: ./backend

ports:

- "5000:5000"

networks:

- app-network

frontend:

build:

context: ./frontend

ports:

- "5001:5001"

networks:

- app-network

depends\_on:

- backend

networks:

app-network:

driver: bridge

**Step 5: Build and Run**

Inside microservices-lab folder:

docker-compose up --build

This will:

* Build both services.
* Run them.
* Connect them via the app-network.

**Step 6: Test the Services**

1. Open browser and go to: [http://localhost:5001](http://localhost:5001/)  
   → You should see:  
   Frontend says: Hello, World! From Backend!
2. Go to: [http://localhost:5000](http://localhost:5000/)  
   → You should see:  
   Hello, World! From Backend!

**Step 7: Stop the Services**

To stop and clean:

docker-compose down