EX1 GIT

```
# Step 1 – Initialize Git
git init

# Step 2 – Configure (first time only)
git config --global user.name "Your Name"
git config --global user.email "youremail@example.com"

# Step 3 – Add file
git add index.html

# Step 4 – Commit
git commit -m "Initial commit with index.html"

# Step 5 – Link to GitHub
git remote add origin https://github.com/yourusername/GitLabExam.git
git branch -M main

# Step 6 – Push to GitHub
git push -u origin main

# Step 7 – Edit file, then update
git add .
git commit -m "Updated index.html with success message"
git push origin main

# Step 8 – Create new branch
git checkout -b feature1
```

Step 9 – Edit file, then push branch

git add .

git commit -m "Added feature1 branch content"

git push origin feature1

Step 10 – Merge branch to main

git checkout main

git merge feature1

git push origin main

ex2 docker

Step 1 – Create project folder

mkdir docker-demo

cd docker-demo

Step 2 – Create index.html

<h1>Hello from Docker!</h1>

Step 3 – Create Dockerfile

FROM nginx

COPY index.html /usr/share/nginx/html/index.html

Step 4 – Build image

docker build -t myweb:v1 .

Step 5 – Run container

docker run -d -p 8080:80 myweb:v1

Step 6 – Check running containers

docker ps

Step 7 – Test in browser

http://localhost:8080

ex3.FLASK WITH DOCKER

app.py

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello from Flask running in Docker!"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

requirements.txt

```
Flask
```

Dockerfile

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
EXPOSE 5000
```

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

docker build -t flask-docker-app .

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

✅ Open in browser:
http://localhost:5000

Push to docker

docker login

docker tag flask-docker-app yourname/flask-docker-app:latest

docker push yourname/flask-docker-app:latest

test

docker pull yourname/flask-docker-app:latest

docker run -p 5000:5000 yourname/flask-docker-app:latest

ex4 Jenkins deployment Using Docker

Create docker-compose.yml

```
version: '3.8'
services:
  jenkins:
    image: jenkins/jenkins:lts
    ports:
      - "8080:8080"
      - "50000:50000"
    volumes:
      - jenkins_home:/var/jenkins_home
volumes:
  jenkins_home:
```

docker compose up -d

Get the admin password

docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword(copy password)

Then open Jenkins: 👉 http://localhost:8080

1. **Complete Jenkins setup**
    - Install **Suggested Plugins**
    - Create **admin user**
2. **Install "HTML Publisher" plugin**
    - Go to: *Manage Jenkins → Plugins → Available Plugins → Search "HTML Publisher" → Install*
3. **Create a Freestyle Job (Simpler than Pipeline)**
    - Click **New Item → Freestyle project**
    - Name: simple-html-site
    - In **Build Steps**:
        - Add "Execute shell" step
        - Paste this simple command:
        - mkdir site
        - echo "<h1>Welcome to Jenkins Static Site</h1>" > site/index.html
    - In **Post-build Actions**:
        - Add "Publish HTML reports"
        - Directory: site
        - Index page: index.html
        - Report title: Simple HTML Site
4. **Click Build Now**
    - After it finishes, open build → click **Simple HTML Site** on left side.

✅ Your web page appears — "Welcome to Jenkins Static Site" 🎉

EX5: CI Pipeline: Python App → GitHub → Docker Hub

app.py

print("Hello CI Pipeline from Beereddy!")

Dockerfile

```
FROM python:3.11

WORKDIR /app

COPY . .

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

Step 4 — Build and test locally

```
docker build -t beereddy2004/demo:local .

docker run beereddy2004/demo:local
```

✅ You should see:
Hello CI Pipeline from Beereddy!

Step 5 — Login and push manually (optional)

```
docker login

docker push beereddy2004/demo:local

mkdir .github/workflows/main.yml
```

```
name: Docker Build & Push


on: [push]


jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      - uses: docker/build-push-action@v6
```

with:

push: true

tags: beereddy2004/demo:latest

Initialize git and commit

git init

git add .

git commit -m "initial commit"

git branch -M main

git remote add origin https://github.com/beereddy2004/demo.git

git push -u origin main

Step 9 — Add GitHub Secrets

Go to:
GitHub → demo repo → Settings → Secrets → Actions

Add two secrets:

Secret Name	Value
DOCKERHUB_USERNAME	beereddy2004
DOCKERHUB_TOKEN	*(create token in Docker Hub → Account Settings → Security → New Access Token)*

🪜 **Step 10 — Done** ✅

Push any code change:

git add .

git commit -m "update"

git push

Then go to your GitHub repo → **Actions tab** → see the workflow run.
After success, check **Docker Hub** →

docker build -t beereddy2004/demo:local .

docker push beereddy2004/demo:local

ex6 Manage Kubernetes Resources Using CLI

download minikube

minikube start --driver=docker

kubectl get nodes

nginx-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx
    image: nginx:latest
    ports:
    - containerPort: 80
```

kubectl apply -f nginx-pod.yaml

kubectl get pods

kubectl describe pod nginx-pod

kubectl port-forward pod/nginx-pod 8080:80

Meaning: You can open **http://localhost:8080** in browser.

Create deployment

kubectl create deployment my-nginx --image=nginx

Scale deployment

kubectl scale deployment my-nginx --replicas=3

kubectl get pods

kubectl set image deployment/my-nginx nginx=nginx:1.25

kubectl expose deployment my-nginx --type=NodePort --port=80

kubectl get svc

minikube service my-nginx –url

ex7 Kubernetes Deployment and Service for Python App from Docker Hub

app.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return '<p>hello flask</p>'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

requirements.txt

```
flask
```

Dockerfile

```
FROM python:3.9-slim
WORKDIR /app
COPY . /app
RUN pip install -r requirements.txt
EXPOSE 5000
CMD ["python", "app.py"]
```

docker build -t beereddy2004/hello-flask .

docker run -p 5000:5000 beereddy2004/hello-flask

Open browser → http://localhost:5000

Login and push (only once):

docker login

docker push beereddy2004/hello-flask

hello-flask.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-flask
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flask
  template:
    metadata:
      labels:
        app: flask
    spec:
      containers:
      - name: flask
        image: beereddy2004/hello-flask
        ports:
        - containerPort: 5000
---
apiVersion: v1
kind: Service
metadata:
  name: flask-svc
spec:
  type: NodePort
  selector:
```

```
    app: flask
  ports:
  - port: 5000
    targetPort: 5000
```

Deploy on Minikube

```
minikube start --driver=docker
kubectl apply -f hello-flask.yaml
kubectl get all
```

Access the App

```
minikube service flask-svc --url
```