1a) Git Repository Initialization

Aim:
To create a new Git repository named `LabProject` and add three files (`index.html`, `style.css`, and `script.js`), and make an initial commit.

Algorithm:

1. Create a folder named `LabProject`.
2. Initialize Git.
3. Add files.
4. Commit the files.

Code:

bash
Copy code
```
mkdir LabProject
cd LabProject
git init
touch index.html style.css script.js
git add .
git commit -m "Initial commit with HTML, CSS, and JS files"
```

Output:

sql
Copy code
```
Initialized empty Git repository in LabProject/.git/
[master (root-commit) abc1234] Initial commit with HTML, CSS, and JS
files
```

Result:
A Git repository named `LabProject` is created and initialized with three files.

1b) Difference Between `git fetch`, `git pull`, and `git push`

Command	Description	Example

`git fetch`	Downloads changes from remote but does not apply them.	`git fetch origin`
`git pull`	Fetches and merges changes from the remote branch.	`git pull origin main`
`git push`	Uploads local commits to the remote repository.	`git push origin main`

2a) Clone and Push Changes

Aim:
Clone a remote repository, modify a file, commit, and push changes.

Algorithm:

1. Clone the repo.
2. Modify or create a file.
3. Commit and push.

Code:

bash

Copy code

```
git clone https://github.com/username/repo.git

cd repo

echo "Updated content" >> index.html

git add index.html

git commit -m "Updated index.html content"

git push origin main
```

Output:

csharp

Copy code

```
Cloning into 'repo'...

[main abcdefg] Updated index.html content
```

Result:
Changes pushed successfully to the remote repository.

2b) Bare vs Non-Bare Git Repository

Feature	Bare Repository	Non-Bare Repository
Working Directory	No	Yes
Use Case	Central/shared repository for collaboration	Developer’s local working copy
Example	`repo.git`	`repo/`

3a) Docker Nginx Container

Aim:
Install Docker, pull the `nginx` image, and run a container.

Algorithm:

1. Install Docker.
2. Pull `nginx` image.
3. Run the container.

Code:

bash

Copy code

```
sudo apt install docker.io  # or use brew/choco based on OS

sudo docker pull nginx

sudo docker run --name nginx-container -d -p 8080:80 nginx
```

Output:

arduino

Copy code

```
Status: Downloaded newer image for nginx:latest
```

Result:
Nginx container is running and accessible at `http://localhost:8080`.

3b) Virtualization vs Containerization

Aspect	Virtualization	Containerization
Technology	Hypervisor (VMWare, VirtualBox)	Docker, Podman
OS Overhead	High (includes guest OS)	Low (shares host OS kernel)
Speed	Slower	Faster and lightweight

4a) Deploy Java App on Docker

Aim:
Deploy a Java Servlet ("Hello World") using Docker and make it accessible via browser.

Algorithm:

1. Create servlet using Java + Tomcat.
2. Write Dockerfile to build image.
3. Run the Docker container.

Code: (`Dockerfile`)

Dockerfile

Copy code

```
FROM tomcat:9.0

COPY HelloWorld.war /usr/local/tomcat/webapps/
```

bash

Copy code

```
docker build -t java-hello-app .

docker run -d -p 9090:8080 java-hello-app
```

Output:
Java servlet accessible at `http://localhost:9090/HelloWorld`.

Result:
Java web app deployed successfully in Docker and reachable via browser.

4b) Docker Secrets

Concept:
Docker Secrets provide a secure way to store and manage sensitive information (like passwords, API keys).

Security Benefits:

- Secrets are encrypted during transit and at rest.
- Only accessible by services explicitly granted access.
- Not exposed in environment variables or Dockerfiles.

5a) Idempotent Chef Recipe

Aim:
Ensure that rerunning a Chef recipe does not repeat tasks unnecessarily.

Algorithm:

1. Use conditionals (`not_if`, `only_if`) in recipe.
2. Create resources that check for the desired state.

Code:

ruby

Copy code

```
file '/tmp/hello.txt' do

  content 'Hello, Chef!'

  mode '0755'

  action :create

end
```

Output:

bash

Copy code

```
Created file /tmp/hello.txt (only once unless changed)
```

Result:
Re-running does not change file unless its state differs.

5b) Chef Server vs Chef Client

Role	Chef Server	Chef Client

Function	Central repository for cookbooks	Node that pulls and applies recipes
Use Case	Stores policies, roles, environments	Applies configuration on machines
Communication	Receives pull request from clients	Pulls data from Chef Server