Assignment - 1

1) git init - Initialized Empty git repository.
2) touch index.html - Create file in your folder
3) git config --global user.name " " - set the global user name for Git, which is used to identify the author of commits.
4) git config --global user.email " " - set the global user email for Git.
5) touch index.html
6) git status - displays the status of working directory including: a) Untracked files → Files not yet added to Git
   b) Modified files → Files change since last commit
   c) Staged files → Files added to the index.
7) git add index.html - stages file for the next commit (go to index.html file - write basic <html> code.
8) git status - check modify changes in file
9) git commit -m "first change" - commits the changes with a meaningful message.
   Hash code → Revert to back version of file used.
   (After this, add other line in same code)
10) git diff - show changes betⁿ working dir. & index also index & last commit.
    (Again git add index.html - commit - git log)
11) git log - displays log of commits or history including commit hash, author, date, message, etc.

Create branch → new branch code create on child b.
master branch → don't create disturbance (run code successfully but same requirement to change init)

1) git branch demo (branchname) - create branch
2) git checkout demo - switched to branch then ls
3) touch style.css - Add css file.

4) git status
5) git add . - It stages all changes in the current dir & subdirectories → git status.
6) git commit -m "new change"
7) git diff.
8) git log
9) git checkout master/main.

combine & meege

10) git meege demo - meege changes.
(master file) ← (for its child file)

Assign - 2

1) git clone "link" (https://---web.git)
2) git push origin branchname
3) Create token :- Collaborating with Remote Repo.
github - setting - Developer setting - Personal access tokens - Tokens (classic) - Generate new token - Generate new token (classic) - Note (demo) - Generate token - copy link (ghp-HFI---) - pwd "paste"
4) check master :- git checkout master
compare & pull req - click - Add title - Add description - create pull req.
5) git pull origin branchname.

git clone - github acc. - <copy> btn click - copy path - paste --- git clone "paste"

⇒ - Open git clone - open terminal (website)
- git status - touch mco.txt - git status - git add mco.txt - git status - git commit -m "vs" - git branch fe - git checkout fe - git status - touch fefile.txt - git add fefile.txt - git commit -m "fes" - git push origin fe - Username " " - Create token.
git status - git checkout master - git pull origin branchname

Rough

Gitlab :- Assign - 3

Gitlab → Login → + → project → new p → Proj. name → Public → create proj.
code → copy → on terminal → git clone "link"
Username → pwd : (Edit Profile - Access token - name - tick all box → create token → Your Token - copy it - paste in "pwd" Enter
Go on proj folder - open terminal -
⇒ touch index.html - open - add content/code save - terminal - git add . - git commit -m "v" - git push origin main - username - pwd (token pwd) - refresh proj - git branch new - git branch checkout - touch style.css - add content - save - git add. - git commit -m "css v" - git push origin new - username/pwd - (new → new - check in proj (new) - git checkout main - create meege req - decscript - create it - after add style - git pull origin main

git checkout new

Assign - 4

Bitbucket :-
Create repo - clone link paste on terminal - then pwd → settings - Personal Bitbucket sett - App pwd's - create app pwd - give label - tick all boxes - create - Copy 'pwd' - & paste on terminal. same step as Git lab.

23/08/25

Assign-5

Jenkins:-

- Open Teeminal
- give cmd 'ngeok http 8080'
- click on foewaeding link - open link (eight click)
- then "Visit Site"
- Go to the Jenkins
- Click on 'New Item'
- Enteer name 'demon'
- Click on Freestyle peoject
- ok
- Then in 'Geneeal' click on 'Soueee code mang.'
- choose 'Git'
- Put Repo. URL (youe git peoj.)
- Beanch Specifiee check 'main' oe 'mastee'
- Build Teiggee
- Github hook teiggee foe GIT scam peoling.
- Apply then Save.
- Go to Github account
- select option 'Settings'
- then select 'Hlebhooks'
- Add webhook
- Pwd (put github pwd).
- Payload URL put. (foewaeding URL copy)
  ⇒ ----/github-webhook/
- Content Json select. - application.
- 'Send Eveey thing' select.
- Add Hlebhook.
- ~~Ge~~ Refeesh page & check 'geeen tick'
- Then 'Jenkins'
- Build now (geeen)
- Woekspace
- code (edit code: 'indeee.html) - Commit Change.
  ---(Jts CF)

- Now CD.
- Othee location → Ubuntu → vae → www → html
- Open teeminal feom theee.
- Add cmd 'sudo mkdie new'
- pwd (acs@18)
- Jenkins
- Configuee
- Add build step → Execute shell
- Weite cmd → "cp -e * /vae/www/html/new
- Apply & Save.
- Build Now → click.
- Open teeminal feom foldee "new"
- Add cmd → "sudo chown -R jenkins:jenki
  vae/www/html/new"
- Pwd (acs@18)
- Build now click.
- Go to the beowsee
- Type → 0.0.0.0/new/indeee.html/ → Entee

Then do steps of pull & push code.....

Assign-3

Gitlab :-

Create a Gitlab account :-

- Go to Gitlab.com & click on "Sign Up" or "Register"
- Create a new project - click on "login"
- Click on the "+" icon in the top-right corner & select "New Project".
- Enter a project name, description and set Visibility "Public".
- Click "Create project".
- Go to github account click on "code" copy http URL & open terminal.
- Type command on Terminal → git clone "URL"
- Add Username
- Add Password Using → Go to Edit profile - Access token name - tick all box - create Token - Your Token - Copy it - paste it in pwd & then Enter.
- Go to Project folder & open terminal from there
- Type command → touch index.html (open file & then add content or code & save)
- Then next command → git add .
- git commit -m "v"
- git push origin main
- Then add username & pwd. (token pwd)
- Refresh project.
- git branch new
- git branch checkout
- git checkout new
- touch style.css (add content & save it)
- git add .
- git commit -m "css v"
- git push origin new (then Add Username & then new → new - check in proj. (new))
- git checkout main
- Create merge request - add description - create it
- After add style in main then used command git pull origin main

Assign-4

BitBucket :-

- Go to Bitbucket.org & click on "Sign Up" or "Get started for free."
- Login to your Bitbucket account.
- Click on the "+" icon in top-right corner & select "Repository".
- Choose Git as repository type.
- Enter name & description for your repo.
- Set the repository's privacy settings as public.
- Click "Create repository".
- Open terminal & paste "clone link" on it.
- Then for 'password' → go to 'settings' - then 'Personal Bitbucket setting' - 'App pwd' - Create app pwd - give label & tick all boxes - create - then copy 'pwd' & paste it on Terminal.
- Then type command :-
- touch index.html
- git add .
- git commit -m "v"
- git push origin main
- git branch new
- git branch checkout
- git checkout new
- touch style.css
- git add .
- git commit -m "css v"
- git push origin new
- git checkout main
- git pull origin main.

Assign-6

archetype : generate

- Uses the Archetype Plugin.
- archetype → Maven's project template system
- generate → goal that tells Maven : "create a new project from a template.

-DgroupId = com.example

- -D means you are passing a parameter (property) to Maven.
- groupId → Defines the unique ID of your project's organization or company.
- Example : com.example, org.mycollege, in.eutz
- Used for project namespace and appears in pom.xml.

-DartifactId = MyApp

- artifactId → The actual project (module) name.
- Here : MyApp.
- It becomes the folder name and is used to create JAR/WAR names (e.g., MyApp-1.0-SNAPSHOT.jar).

-DinteractiveMode = false

- Normally, mvn archetype : generate asks many questions interactively (like template number, version, etc.)
- Setting interactiveMode = false means :
  - Maven will not ask questions.
  - It will just use default value.

```
project-name
├── src
│   ├── main
│   │   ├── java        # Application source code
│   │   ├── resources   # Coding files, property files
│   │   └── webapp      (for web apps, contains HTML, JSP, etc.)
│   └── test
│       ├── java        # Test cases (JUnit / TestNG)
│       └── resources   # Test config files
├── target              # Auto generated folder (compile by-product, JAR/WAR files)
└── pom.xml             # Main Maven Configuration file
```

What is a POM file?

POM = Project Object Model

- It's an XML file (pom.xml) at the root of every Maven project.
- It tells Maven:
  - What your project is
  - What dependencies it needs
  - How to build it.

Key Tags in pom.xml:

- <groupId> → Unique Id of your project's organization (e.g., com.example)
- <artifactId> → Name of project (e.g., MyApp)
- <version> → Version of your app (e.g., 1.0-SNAPSHOT)
- <dependencies> → List of required libraries. Maven will auto-download them.
- <build> → Contain plugins & build settings.

i) sudo apt update
ii) sudo apt install maven
iii) mvn -version

* mvn archetype:generate
  -DgroupId=com.example -DartifactId=MyApp
  -DinteractiveMode=false

mvn
- Runs Maven.
- Every Maven command starts with this.

* Open Terminal & Add command:-
  i) Command (mvn archetype:generate.....→
  ii) cd MyApp
  iii) tree
  iv) mvn compile
  v) Open pom.xml file & add code after </url>
  → <properties>
  <maven.compiler.source>8</maven.compiler.source>
  <maven.compiler.target>8</maven.compiler.target>
  </properties>.
  vi) Check 'target' folder create in your folder
  vii) mvn exec:java
  viii) Open pom.xml file & add code after </dependencies>
  Add <build> tag.

```
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo
        </groupId>
      <artifactId>exec-maven-plugin
        </artifactId>
      <version>3.0.0</version>
      <executions>
        <execution>
          <goals>
            <goal>java</goal>
          </goals>
        </execution>
      </executions>
      <configuration>
        <mainClass>com.example.App
          </mainClass>
      </configuration>
    </plugin></plugins>
</build>
```

xx) mvn package

Assignment-7

Dockee:- It is a tool which provide container, means provides ability to package & run an application in a loose isolated environment.

i) Create folder 'Dockee' & open Tee
ii) touch index.html
iii) touch Dockerfile
iv) Open Dockerfile & add :-
~~v) dockee build -t project7~~
```
FROM httpd:2.4
COPY index.html /usr/local/apache2/
                htdocs/
```
v) Also add html code in index.html
vi) Open Terminal & add command:-
⇒ dockee build -t project7
instead ↳ add command:-
⇒ sudo dockee build -t project7 .

vii) ⇒ sudo dockee run -p 8092:80 -d projec
viii) Open browser & check o/p.
Type:- localhost:8092 & check.

where, -p = port
-d = detached mode.