1. What is Git and why is it used?

Git is a distributed version control system (DVCS) used to track changes in source code during software development. It allows multiple developers to collaborate, manage code versions, and maintain a history of changes. Git was created by Linus Torvalds in 2005.

**Version Control** – Tracks changes in files and allows reverting to previous versions if needed.  
**Collaboration** – Multiple developers can work on the same project without overwriting each other’s changes.  
**Branching & Merging** – Enables working on different features or fixes separately and then merging them into the main codebase.  
 **Backup & Recovery** – Since it's distributed, every developer has a copy of the project, reducing data loss risks.  
**Faster Development** – Efficiently handles large projects with minimal performance issues.  
**Integration** – Works seamlessly with platforms like GitHub, GitLab, and Bitbucket for code hosting and CI/CD automation.

1. Explain the difference between Git and GitHub

Git is a distributed version control system (DVCS) used to track changes in source code during software development. It allows multiple developers to collaborate, manage code versions, and maintain a history of changes.

GitHub is a web-based platform that uses Git for version control and is primarily used for hosting and managing software projects.

1. How do you install Git on your machine?

Go to the official Git website: [https://git-scm.com/downloads](https://git-scm.com/downloads" \t "_new).

Click on **Windows** to download the latest version.

Open **Git Bash** or **Command Prompt** and type:

git --version --It was install display the version

4.How do you configure your username and email in Git?

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

git config --global user.email "your\_email@example.com"

1. What is a repository in Git?

A **repository (repo)** in Git is a storage location where all project files, along with their version history, are stored. It helps track changes, collaborate with others, and manage different versions of a project efficiently.There are two types of repos.

**--> Local Repository** – Stored on your personal computer.

**--> Remote Repository** – Hosted on platforms like GitHub, GitLab, or Bitbucket for collaboration.

1. How do you create a new Git repository?

**Go to** [GitHub](https://github.com/" \t "_new) and log in.

Click on **"New Repository"**.  
Enter a **repository name**, and choose between **public or private**.  
Click **"Create Repository"**.

1. How do you clone a repository from GitHub?

**Copy the Repository URL** from GitHub (HTTPS or SSH).

**Open Git Bash or Terminal** and navigate to your desired directory:

--> cd path/to/your/folder

**Run the Clone Command**:

--> git clone https://github.com/username/repository.git

**Navigate into the Cloned Repository:**

**--> cd repository.**

1. What is the purpose of the .gitignore file?

The .gitignore **file** is used to **exclude specific files and directories** from being tracked by Git. This helps keep the repository clean by ignoring unnecessary, sensitive, or system-generated files.

1. How do you check the status of your working directory in Git?

--> git status

It will shows :

**Untracked files** (new files not added to Git).

**Modified files** (changes made but not staged).

**Staged files** (files ready to be committed).

**Branch information** (current branch and ahead/behind status)

1. How do you add files to the staging area in Git?

To **add files to the staging area** in Git, use the git add command. This prepares the files for the next commit.

git add <file> --> Stage a specific file

git add . --> Stage all files

### **Intermediate Git Questions**

1. Explain the concept of commits in Git.

A **commit** in Git is a **snapshot** of your repository at a specific point in time. It records changes made to files, allowing you to track and revert to previous versions if needed.

Key Features of a Commit:

Each commit has a **unique ID (SHA-1 hash)**.

Commits **store changes**, not whole files.

They include a **commit message** describing the changes.

Commits form the **history** of a project.

1. How do you create a new commit in Git?

**Stage the Files** (Prepare files for commit):

--> git add filename.txt

(or)

--> git add .

**Commit the Changes** (Save with a message):

--> git commit -m "Your commit message"

1. What is the purpose of the git log command?

The git log command is used to **view the commit history** of a repository. It shows details like commit IDs, authors, timestamps, and commit messages.

1. How do you view the history of commits in a repository?

To view the commit history in a Git repository, use the git log command.

--> git log --oneline

1. How do you view the changes made in a commit?

To see the changes made in a commit, use the git show or git diff commands.

--> git show commit\_id

1. What is branching in Git and why is it useful?

Branching in Git allows developers to create **separate versions** of a project without affecting the main codebase. Each branch is an independent line of development that can be worked on separately.

1. How do you switch between branches in Git?

---> git branch new-branch-name

1. What is the difference between git merge and git rebase?

Both git merge and git rebase are used to combine changes from one branch into another, but they work differently.

git merge: Combines changes from one branch into another by creating a **new merge commit**. This preserves the commit history.

--> git merge feature-branch

git rebase: Moves or replays commits from one branch onto another, **rewriting history** for a cleaner linear commit structure.

--> git rebase main

19.How do you resolve merge conflicts in Git?

git merge combines branches by creating a new merge commit.

git rebase moves or combines commits to keep a linear history.

20.How do you create a new branch in Git?

--> git branch new-branch-name

### **Git Exercises**

1. Create a new Git repository and configure your username and email.

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

git config --global user.email ["your.email@example.com"](mailto:\"your.email@example.com\")

22.Create a file, add some content to it, and commit the changes.

echo "Hello, Git!" > myfile.txt

git add myfile.txt

git commit -m "Added myfile.txt with initial content"

23.Create a .gitignore file and add rules to ignore specific files and directories.

echo "node\_modules/" > .gitignore

echo "\*.log" >> .gitignore

git add .gitignore

git commit -m "Added .gitignore file"

25.Create a new branch, make some changes, and switch back to the main branch

Create and Switch to a New Branch :

--> git checkout -b new-feature-branch

Make Some Changes :

--> echo "This is a new feature" > feature.txt

Add and Commit the Changes :

--> git add feature.txt

--> git commit -m "Added feature.txt with new content"

Switch Back to the Main Branch :

--> git checkout main