

Assignment

1. What is Git and why is it used?

GIT is a version control system which was created by Linus Torvalds in 2005. Git is a decentralized version control system designed to track modifications in source code throughout the software development lifecycle.

It facilitates concurrent collaboration among developers

Git enables developers to work on parallel branches, perform seamless integrations, and roll back changes whenever we need .

it maintains the history of all the codes and the changes etc.

2. Explain the difference between Git and other version control systems.

There are 2 types of vcs dvcs and cvcs

Git is a Decentralized version control system/ distributed

Every Developer Has a Copy Each person gets a full copy of the project, including its history.

Works Offline You can make changes without an internet connection and sync later.

Faster Operations Most actions (like commits) happen on your local machine.

Safer Backup Since every developer has a copy, data loss is less likely.

Examples :- Git

CVCS : centralized version control system

Single Central Server – All versions of the project are stored in one central place.

Requires Internet – Developers must connect to the server to access or update files.

Slower Operations – Since every action depends on the server, it can be slower.

Risk of Data Loss – If the central server crashes, all history may be lost.

3. How do you initialize a Git repository?

To initialize a new Git repository, navigate to the directory and execute

git init cmd

this command will generate the .git directory for the folder

to verify this type **ls** in the cmd line there we can see the .git in the folder

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

The .gitignore file specifies patterns for directories and files to be ignored by version control. It is typically used to exclude logs, temporary files, and sensitive credentials to keep a repository clean and secure. A properly defined .gitignore keeps the repository free from unnecessary files while allowing only applicable files to be tracked.

to add gitignore file to the git repository use the cmd **touch .gitignore** this will create a the .gitignore folder

in that i want to ignore the log files

ignore ALL .log files

***.log**

5. How do you stage changes in Git?

Staging changes involves using the **git add** command to prepare specific modifications for commit

`git add <filename>` this will add only one file to staging area and ready for commit

`git add .` this will add all the files to staging area and ready for commit

6. What is the difference between `git commit` and `git commit -m`?

git commit without options invokes the text editor set up in the configuration, allowing the user to input a commit message interactively.

git commit -m "message" supports inline specification of the commit message, making for efficient commits. The latter is best suited for rapid updates, while the former is convenient for more elaborate descriptions of changes.

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

To create a new branch in the git we use **git branch <branchname>** and **git checkout -b <branch name>**

ex: i want to create a branch dev1

`git branch dev1` # this will create a branch dev1

`git checkout -m dev1` # this will create the branch and change the branch to dev1

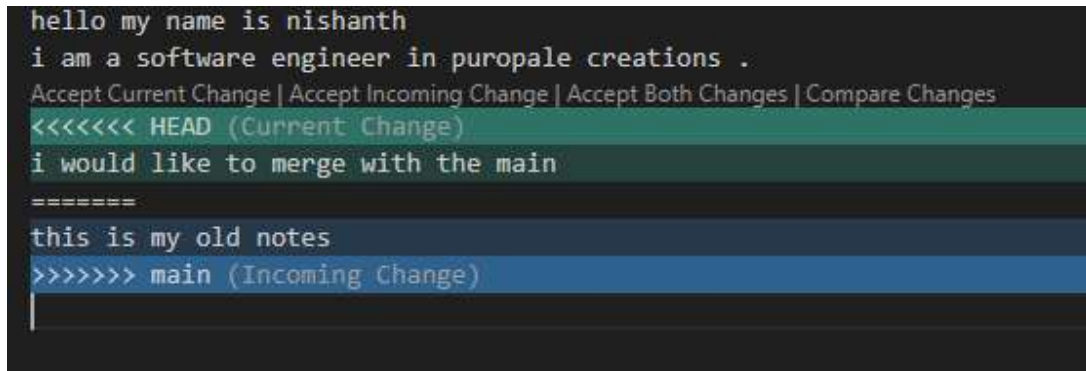
8. What is the difference between `git merge` and `git rebase`?

git merge merges the changes from a different branch while maintaining the commit record, typically causing a merge commits. it will merge the branch with the different branch .

git rebase applies commits sequentially from one branch onto the other, which results in a linear history and avoids unnecessary merge commits. Rebase is commonly applied to keeping feature branches current with the main branch without a messy commit history.

9. How do you resolve merge conflicts in Git?

the merge conflicts will occur when we use **git merge** and **git rebase** . this will occur when the changes are made in the same file or the same code . this will cause the merge conflicts .



```

hello my name is nishanth
i am a software engineer in puropale creations .
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
i would like to merge with the main
=====
this is my old notes
>>>>>>> main (Incoming Change)

```

it shows accept current change or accept incoming change or accept the both changes we can do any of the operation to resolve the merge conflicts . and then we do **git add** . and then **git commit -m <resolved the conflicts>** etc.

10. What is the purpose of git stash?

The git stash command stores uncommitted changes temporarily, allowing for switching branches without losing work. Stashed changes can be reapplied later using **git stash pop**. This is helpful for keeping work-in-progress code in check without changing the commit history.

11. Explain the use of git pull and git fetch.

git fetch retrieves remote updates (commits, branches, tags) without updating the local branch, so developers can inspect changes before merging the repository .

git pull is a combination of **git fetch** and **git merge**, which merges remote updates into the current branch immediately.

We use **git fetch** for safer updates and conflict avoidance and use **git pull** for synchronization directly.

12. How do you revert a commit in Git?

The git revert command generates a new commit reversing the changes that a given commit brought, maintaining previous commits undisturbed.

git revert <hash>

to get hash we use **git log**

git revert preserves history and is ideal for collaborative workflows.

13. What is the difference between git clone and git fork?

git clone copies a remote repository to the local system. The local copy is still linked to the original repository and can thus be updated with the help of git pull and git push.

example : **git clone [git@github.com:NishanthDhanalakoti/ppt.git](https://github.com/NishanthDhanalakoti/ppt.git)**

this will copy all the main branch folders to my local repo .

git Fork makes a copy of a repository independent on a site such as GitHub. The fork repository is independent, allowing developers to make changes at will without changing the original repository. To bring changes back, a pull request has to be made. after the changes are done in the folders etc.

14. Explain the concept of remote repositories in Git.

A remote repository is a copy of the repository on a host such as GitHub, GitLab that allows distributed teams to work together. Developers can push and pull from local to remote or remote to local repositories using commands such as:

for example :

git remote add origin <repo_url> this will connect the local repo to the remote repo .

git push origin <branch name> – this will push all the content from local to remote repo .

git pull origin <branch name> – this will pull all the data from the remote to local .

15. What are Git tags and how do you use them?

Git **tags** are used to mark specific points in a repository's history. Unlike branches, tags do not change over time; they act as fixed reference points for commits.

example

git tag -a nishanth v.1 -m "new version from nishanth"

git push origin nishanth v.1

this will create a tag and push the data to the remote repo

git tag will list all the tags in repo .

16. How do you view the commit history in Git?

To view the commit history in the git we use

git log

this will show all the commit history in the terminal

```

PS C:\Users\Administrator\Desktop\varma> git log
commit f8fd7b9dfbfe5769cbb21fcad664240dc13c7ea2 (HEAD -> feature, origin/feature)
Author: NishanthDhanalakoti <sainishanth@puropalecreations.com>
Date: Thu Mar 13 09:53:40 2025 +0530

    python files

commit 848b877049577808cd057d2eeab517563c8ada6b
Author: NishanthDhanalakoti <sainishanth@puropalecreations.com>
Date: Wed Mar 12 12:28:45 2025 +0530

    new file added

commit be05cd6856abbe973559833e028d58a563b3b9b7
Author: NishanthDhanalakoti <sainishanth@puropalecreations.com>
Date: Tue Mar 11 14:05:15 2025 +0530

```

git log --oneline it shows all the content in one line that is easy to understand

```

PS C:\Users\Administrator\Desktop\varma> git log --oneline
f8fd7b9 (HEAD -> feature, origin/feature) python files
848b877 new file added
be05cd6 this is my intro
PS C:\Users\Administrator\Desktop\varma>

```

17. What is the purpose of git diff?

it shows all the differences between the branches or the files

git diff <branch names>

git diff --staged it will compare the changes from the last commit

git diff <branch 1> <branch2> it will show the diff between the branches .

18. How do you delete a branch in Git?

to delete the branch in the git we use

git branch -d <branch name >

by using this cmd we can delete the branch in the git .

19. What are Git hooks and how are they used?

Git hooks are scripts that automatically execute whenever a specific event happens in a Git repository. They allow you to extend Git's internal behavior and invoke customizable actions at important points in the development cycle.

20. Explain the concept of a pull request in Git.

A pull request (PR) is a request to pull changes from one branch (often in a forked repository) into another (e.g., main). It is utilized in collaborative development, particularly on sites like GitHub and GitLab. generally we do the pull request from the remote repo .

Create a feature branch and commit changes.

Push the branch to the remote repository.

Then we will Create a pull request on GitHub.

The manager or the team lead will Review, approve and merge the changes in the git hub.

21. What is DevOps and why is it important?

DevOps is an integration of Development (Dev) and Operations (Ops) practices with the objective of automating and integrating software development and IT operations processes. DevOps is based on continuous integration, continuous deployment (CI/CD), automation, and teamwork to enhance the quality of the software and delivery speed.

Importance of DevOps:

Quick Software Delivery and Speeds up release cycles and enhances time-to-market.
 Better Collaboration and Reduces disputes between development and operations teams.
 Greater Reliability Provides stable releases with automated monitoring and testing.
 Scalability and Facilitates infrastructure automation and scalable deployments.
 Fewer Failures and Provides rapid rollback and improved fault tolerance.

22. Explain the key principles of DevOps.

the key principles of devops

1. the collaboration and communication between the development and the operations team
2. By using the automation tools it reduces the manual work by automating testing ,deployment and the management etc
3. continuous integration and continuous deployment it ensures the code is developed correctly and tested and deployed frequently
4. it allows the continuous monitoring and continuous feed back from the customer or client or the end user . etc
5. the devops ensure the security without breakage of any information .

23. What are the benefits of continuous integration (CI)?

Advantages of Continuous Integration (CI)

It ensures the Early Bug Detection and Automated testing guarantees that the problems are caught and corrected easily and fast

It has a Faster Development Cycle that the developers can merge code often without tension .

It ensures Better Code Quality and Code reviews and tests ensure quality.

It will Improved Collaboration between the Teams collaborate on common repositories with version control. etc

24. What is continuous delivery (CD) and how does it differ from continuous deployment?

Continuous Delivery guarantees code is built, tested, and ready for release automatically but needs manual approval before deployment to production. It is best suited for organizations that require compliance checks or approval procedures prior to pushing changes live.

Continuous Deployment, however, automates the process of releasing the entire code, pushing code into production without human intervention when it goes through automated tests. This is most appropriate for situations with high-frequency deployment need.

25. Explain the concept of Infrastructure as Code (IaC).

IaC is a DevOps technique where infrastructure is controlled by code instead of manual methods it uses automated tools . It enables teams to define, deploy, and manage infrastructure automatically through scripts or configuration files.etc

the features of iac :

through automation tools it Decreases manual effort and human mistakes.

Version Control: Allows tracking and auditing of infrastructure changes.

Most Common IaC Tools: Terraform, Ansible, Puppet, Chef.

26. What tools are commonly used in a DevOps pipeline?

Version Control: Git, GitHub, GitLab.

CI/CD: Jenkins, GitHub Actions, GitLab CI/CD.

Configuration Management: Ansible, Puppet, Chef.

Containerization: Docker, Podman.

Orchestration: Kubernetes, OpenShift.

Infrastructure as Code: Terraform, CloudFormation.

27. What is the role of configuration management in DevOps?

Configuration Management (CM) maintains infrastructure and software configurations constantly across environments. It automates:

Provisioning: Installation of systems with necessary configurations.

Enforcement: Keeping systems in a state that is wanted.

Change Management: Tracking the configuration changes for rollback in case of necessity.

Top CM Tools: Ansible, Puppet, Chef

28. How does containerization help in a DevOps environment?

Containerization bundles applications and dependencies into lightweight, portable containers that provide consistency in development, testing, and production environments.

Advantages of Containers:

Portability: Supports all operating systems and cloud platforms.

Scalability: Can scale up or down as needed.

Isolation: Independent operation of containers to avoid conflicts.

Faster Deployment: Fast startup and optimized resource use.

Major Container Tools: Docker, Kubernetes.

29. What is the purpose of monitoring in DevOps?

Monitoring facilitates real-time tracking of system health, performance, and availability. Monitoring allows teams to identify problems early and take corrective action.

Important Monitoring Metrics:

Application Performance and Response time, error rates.

Infrastructure Health and CPU, memory, disk usage.

Log Analysis and Debugging and security monitoring.

30. What are microservices and how do they relate to DevOps?

Microservices architecture is a decomposition of applications into little, autonomous services that talk through API and applications

Continuous Deployment: It allows for continuous updates with very low risk.

Scalability: Services scale independently according to demand.

Quicker Development: Different services are developed by teams independently.

Fault Tolerance: Failure in one service does not affect the entire system.

Tools for Microservices: Kubernetes, Docker.

31. Explain in detail about devops tools?

Version Control Systems (VCS):

Git: Distributed VCS for code changes tracking.

GitHub, GitLab, Bitbucket: Git repository hosting platforms.

Continuous Integration & Continuous Deployment (CI/CD):

Jenkins: Open-source automation server for CI/CD.

GitLab CI/CD, GitHub Actions, CircleCI, Travis CI: Automate build, test, deployment.

Configuration Management:

Ansible: Agentless configuration and deployment automation.

Puppet & Chef: Declarative configuration management systems.

Containerization & Orchestration:

Docker: Platform for container creation and management.

Kubernetes: Orchestrates and manages containerized applications.

Infrastructure as Code (IaC):

Terraform: Cloud provisioning declarative IaC tool.

AWS CloudFormation: Provisions AWS infrastructure automatically.