GIT Interview Questions

1. What is Git?
Answer: Git is a distributed version control system used to track changes in files and coordinate work on those files among multiple people.

2. What is the difference between Git and SVN?
Answer: Git is a distributed version control system, while SVN (Subversion) is a centralized version control system. In Git, every user has a local copy of the repository, allowing them to work offline and commit changes locally.

3. What is a repository in Git?
Answer: A repository in Git is a data structure that stores metadata and objects representing a project. It contains all versions of the project's files and directories.

4. How do you create a new Git repository?
Answer: You can create a new Git repository using the command git init followed by the project directory.

5. What is a commit in Git?
Answer: A commit is a snapshot of changes made to a repository at a specific point in time. It includes a unique identifier, a commit message, and a pointer to the parent commit(s).

6. How do you commit changes in Git?
Answer: To commit changes in Git, you use the command git commit -m "commit message". This will record the changes in the local repository.

7. What is the purpose of the staging area (index) in Git?
Answer: The staging area is where you prepare and review changes before committing them to the repository. It allows you to selectively include or exclude specific changes from a commit.

8. How do you add files to the staging area in Git?
Answer: You can add files to the staging area using the command git add filename or git add . to add all modified files.

9. What is a branch in Git?
Answer: A branch in Git is a separate line of development that allows you to work on features or bug fixes without affecting the main codebase.

10. How do you create a new branch in Git?
Answer: You can create a new branch using the command git checkout -b branch_name.

11. How do you switch between branches in Git?
Answer: You can switch between branches using the command git checkout branch_name.

12. What is a merge in Git?
Answer: A merge in Git combines the changes from one branch into another branch. It is used to integrate feature branches back into the main branch.

13. How do you merge branches in Git?
Answer: To merge branches in Git, you first switch to the target branch (git checkout target_branch) and then use the command git merge source_branch.

14. What is a conflict in Git?
Answer: A conflict in Git occurs when two branches have made changes to the same part of a file and Git is unable to automatically resolve the differences.

15. How do you resolve a merge conflict in Git?
Answer: To resolve a merge conflict, you need to manually edit the conflicting file(s), choose which changes to keep, and then commit the changes.

16. What is a remote repository in Git?
Answer: A remote repository in Git is a version of a project hosted on a server that allows collaboration between multiple contributors.

17. How do you clone a Git repository?
Answer: You can clone a Git repository using the command git clone repository_url.

18. What is the difference between git pull and git fetch?
Answer: git pull fetches and merges changes from a remote repository to the current branch, while git fetch only downloads changes from the remote repository without merging them.

19. What is a tag in Git?
Answer: A tag in Git is a reference that points to a specific commit. It is used to mark specific points in history, such as release versions.

20. How do you create a tag in Git?
Answer: You can create a tag in Git using the command git tag tag_name.

21. How do you push a tag to a remote repository?
Answer: You can push a tag to a remote repository using the command git push origin tag_name.

22. What is a Git hook?
Answer: A Git hook is a script that Git executes before or after certain events, such as committing changes or merging branches.

23. What is Git bisect?
Answer: Git bisect is a command used for binary search of bugs. It helps you find the commit that introduced a bug.

24. What is Git rebase?
Answer: Git rebase is a command used to integrate changes from one branch into another by moving or combining commits.

25. What is the difference between a merge and a rebase in Git?
Answer: Merging creates a new commit that combines the changes from two branches, while rebasing moves or combines a sequence of commits to a new base commit.

26. How do you undo the last Git commit?
Answer: You can undo the last Git commit using the command git reset HEAD~1.

27. What is Git cherry-pick?
Answer: Git cherry-pick is a command used to apply a specific commit from one branch to another.

28. What is Git stash?
Answer: Git stash is a command used to save changes that are not ready to be committed, allowing you to switch branches without losing your work.

29. How do you list all the branches in a Git repository?
Answer: You can list all branches in a Git repository using the command git branch.

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

Answer: The .gitignore file is used to specify which files and directories Git should ignore, meaning they will not be tracked or committed to the repository.

31. How do you remove a file from Git without deleting it?
Answer: You can remove a file from Git without deleting it using the command git rm --cached filename.

32. How do you rename a file in Git?
Answer: You can rename a file in Git using the command git mv old_filename new_filename.

33. What is a Git submodule?
Answer: A Git submodule is a separate Git repository embedded within another Git repository. It allows you to include other projects as dependencies.

34. How do you update a Git submodule?
Answer: You can update a Git submodule using the command git submodule update --remote.

35. What is Git flow?
Answer: Git flow is a branching model that defines a consistent way to organize branches and merges in a Git repository.

36. What is GitLab/GitHub Actions?
Answer: GitLab/GitHub Actions are Continuous Integration/Continuous Deployment (CI/CD) services provided by GitLab/GitHub for automating tasks like testing and deploying code.

37. How do you revert a commit in Git?
Answer: You can revert a commit in Git using the command git revert commit_sha.

38. What is the difference between git rebase and git merge?
Answer: git rebase integrates changes from one branch onto another by moving or combining commits, resulting in a linear history. git merge creates a new commit that combines changes from two branches, resulting in a branch-like history.

39. How do you squash multiple commits into one?

Answer: You can squash multiple commits into one using the interactive rebase command: git rebase -i HEAD~n, where n is the number of commits you want to squash.

40. How do you amend the last commit message in Git?
Answer: You can amend the last commit message using the command git commit --amend.

41. What is Git LFS (Large File Storage)?
Answer: Git LFS is an extension to Git that allows the versioning of large files by storing them on a remote server, while keeping lightweight references in the repository.

42. How do you configure Git user information?
Answer: You can configure Git user information using the commands: git config --global user.name "Your Name" git config --global user.email "your.email@example.com"

43. How do you show the commit history in Git?
Answer: You can show the commit history using the command git log.

44. What is Git Cherry-Pick and when would you use it?
Answer: Git Cherry-Pick is a command used to apply a specific commit from one branch to another. It is useful when you want to apply a specific fix or feature to a different branch without merging the entire branch.

45. How do you create and apply patches in Git?
Answer: To create a patch, use the command git format-patch -1 <commit>. To apply a patch, use the command git am <patch>.

46. How do you view the changes between two Git commits?
Answer: You can view the changes between two Git commits using the command git diff commit1 commit2.

47. What is Git reflog?
Answer: Git reflog is a log of all reference updates in a repository. It allows you to recover lost commits or branches.

48. What is Git blame?
Answer: Git blame is a command that shows information about who last modified each line of a file, and in which commit.

49. How do you delete a branch in Git?
Answer: You can delete a branch in Git using the command git branch -d branch_name.

50. How do you force push in Git?
Answer: You can force push in Git using the command git push -f origin branch_name.

51.What is Git and how does it differ from other version control systems?
Answer: Git is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Unlike centralized version control systems, Git allows every developer to have a full copy of the repository, enabling them to work offline.

52.Explain the difference between Git and GitHub.
Answer: Git is a version control system, while GitHub is a web-based platform for hosting and collaborating on Git repositories. GitHub provides additional features such as pull requests, issue tracking, and collaborative tools.

53.What is a Git repository?
Answer: A Git repository is a collection of files and their history, stored in a .git directory at the root of the project. It contains the complete history of the project, along with metadata and configuration settings.

54.What is a commit in Git?
Answer: A commit is a snapshot of changes made to the repository at a specific point in time. It includes a unique identifier (SHA-1 hash), author information, timestamp, and a reference to the previous commit.

55.Explain the Git staging area.
Answer: The staging area (or index) is a crucial part of Git that acts as a middle-ground between the working directory and the repository. Changes are first staged using git add before being committed to the repository.

56.What is a branch in Git?
Answer: A branch in Git is a lightweight movable pointer to a commit. It allows developers to work on separate features or fixes without affecting the main codebase. Branches can be easily created, merged, or deleted.

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

Answer: Use the command git branch <branch_name> to create a new branch. To switch to the new branch, use git checkout <branch_name> or git switch <branch_name>.

58.Explain the concept of Git merging.
Answer: Merging in Git combines changes from different branches. It can be done using git merge to integrate changes from a source branch into a target branch.

59.What is a Git merge conflict, and how can it be resolved?
Answer: A merge conflict occurs when Git cannot automatically reconcile changes between branches. To resolve conflicts, manually edit the conflicting files, mark them as resolved using git add, and complete the merge with git merge --continue.

60.Describe the rebase operation in Git.
Answer: Git rebase is the process of moving or combining a sequence of commits to a new base commit. It can be used to maintain a cleaner and linear project history.

61.What is a detached HEAD state in Git?
Answer: A detached HEAD state occurs when the HEAD points directly to a specific commit instead of a branch. It is a common state during operations like checking out a specific commit.

62.Explain the difference between git pull and git fetch.
Answer: git pull fetches changes from a remote repository and automatically merges them into the current branch. git fetch only retrieves changes but does not automatically merge them, allowing the user to review and decide how to integrate the changes.

63.How do you revert a commit in Git?
Answer: To revert a commit, use git revert <commit_hash> to create a new commit that undoes the changes introduced by the specified commit.

64.What is a Git submodule?
Answer: A Git submodule is a repository embedded within another repository. It allows you to include external repositories as a subdirectory of your project.

65.Explain the purpose of Git hooks.

Answer: Git hooks are scripts that can be triggered at specific points in the Git workflow. They enable custom actions such as pre-commit validation or post-receive deployment.

66.How do you squash commits in Git?
Answer: Use git rebase -i HEAD~n (replace n with the number of commits) and mark commits as "squash" or "s" to combine them into a single commit.

67.Describe the .gitignore file.
Answer: The .gitignore file specifies files and directories that should be ignored by Git. It helps exclude unnecessary files (e.g., build artifacts, logs) from version control.

68.What is Git bisect, and how is it used?
Answer: Git bisect is a binary search tool for finding the commit that introduced a bug. It involves marking known good and bad commits, allowing Git to narrow down the faulty commit.

69.Explain the purpose of Git cherry-pick.
Answer: Git cherry-pick is used to apply a specific commit from one branch to another. It allows you to selectively choose and apply changes.

70.What is Git reflog, and how is it useful?
Answer: Git reflog is a log of reference updates in the repository. It helps recover lost commits or branches and provides a history of recent operations.

71.How do you sign commits in Git?
Answer: Sign commits using git commit -S to add a GPG signature. Configure Git with your GPG key, and use git log --show-signature to verify signatures.

72.What is Git cherry and when would you use it?
Answer: Git cherry is used to find commits that are in one branch but not in another. It is helpful when you want to identify changes that have not been merged between branches.

73.Describe Git rebase --onto.
Answer: git rebase --onto is used to reapply commits onto a new base. It allows you to move a branch or a series of commits from one branch to another.

74.Explain the purpose of Git subcommands like git clean and git reset.

Answer: git clean removes untracked files, and git reset allows you to reset the current branch to a specific commit or undo changes.

75.What is the purpose of the git worktree command?
Answer: The git worktree command allows you to maintain multiple working directories (worktrees) associated with a single Git repository. Each worktree can have its own branch and changes.

76.Describe Git shallow cloning.
Answer: Shallow cloning involves cloning a Git repository with a limited history. It can be done using --depth option with git clone to specify the number of commits to retrieve.

77.How do you configure Git to use an HTTP proxy?
Answer: Set the HTTP proxy for Git using the http.proxy configuration. For example, use git config --global http.proxy http://proxy.example.com:8080.

78.Explain the purpose of Git LFS (Large File Storage).
Answer: Git LFS is an extension to Git that allows for the storage of large files outside the Git repository, reducing the repository size. It is useful for managing binary
files and large assets.

79.What is the Git "refspec"?
Answer: A refspec in Git is a string that defines the mapping between remote and local branches during fetch or push operations. It specifies the source and destination references.

80.Describe Git worktrees and their advantages.
Answer: Git worktrees allow you to work on multiple branches simultaneously without switching back and forth. They provide a clean and isolated environment for each branch.

81.How do you handle confidential information, such as API keys, in a Git repository?
Answer: Confidential information should be stored in environment variables or configuration files outside the repository. Use tools like .gitignore and git-crypt to avoid accidentally committing sensitive data.

82.What is Git rebase -i used for, and how does it work?

Answer: git rebase -i opens an interactive rebase session, allowing you to modify, reorder, or squash commits. It provides a text editor interface where you can choose actions for each commit.

83.Explain the purpose of Git sparse-checkout.
Answer: Git sparse-checkout is used to limit the working directory to specific directories or files, enabling users to check out only the necessary parts of a repository.

84.How do you recover from a detached HEAD state in Git?
Answer: To recover from a detached HEAD state, create a new branch using git checkout -b new_branch_name or switch to an existing branch using git checkout branch_name.

85.What is the Git "rerere" (Reuse Recorded Resolution) feature?
Answer: The rerere feature records previous conflict resolutions, allowing Git to automatically apply them to future conflicts. It stands for "Reuse Recorded Resolution."

86.Explain Git's "worktree add" command.
Answer: The git worktree add command creates a new linked working directory associated with the current repository. It enables you to work on multiple branches simultaneously.

87.How do you amend the last commit message in Git?
Answer: Use git commit --amend to modify the last commit message. This opens the default text editor for you to make changes.

88.Describe the purpose of Git "revert" vs. "reset."
Answer: git revert creates a new commit that undoes changes from a specific commit. git reset is used to reset the current branch to a specified commit, potentially discarding changes.

89.What is the Git "worktree move" command?
Answer: The git worktree move command allows you to move or rename an existing linked working directory created with git worktree add.

90.Explain the concept of Git "subversion" (svn) integration.
Answer: Git can integrate with Subversion repositories using git svn commands, providing a bridge between Git and Subversion version control systems.

91.How do you use Git to show the changes introduced by a specific commit?
Answer: Use git show <commit_hash> to display the changes introduced by a specific commit, including the commit message and diff.

92.What is Git "cherry-pick range"?
Answer: Git cherry-pick range involves picking a range of commits from one branch and applying them to another branch. It can be achieved using git cherry-pick <start_commit>^..<end_commit>.

93.Explain the purpose of the "git bisect" command.
Answer: The git bisect command helps in finding a specific commit that introduced a bug by using a binary search algorithm. It requires marking known good and bad commits to narrow down the problematic commit.

94.How does Git handle line endings, and what are autocrlf and core.autocrlf?
Answer: Git handles line endings based on platform conventions. The core.autocrlf setting controls automatic conversion between Unix and Windows line endings. autocrlf is used for automatic line ending conversion.

95.What is the purpose of "git blame"?
Answer: git blame shows the commit and author information for each line of a file, helping to identify who made changes and when.

96.How can you rename a Git branch?
Answer: To rename a Git branch, use git branch -m <new_branch_name> to rename the current branch, or git branch -m <old_branch_name> <new_branch_name> to rename a different branch.

97.Explain the difference between "merge" and "rebase" workflows.
Answer: The merge workflow combines changes from different branches, creating a new merge commit. The rebase workflow moves or combines a sequence of commits onto a new base, creating a linear history.

98.What is Git "filter-branch" used for?
Answer: git filter-branch is used to rewrite Git repository history by applying filters. It can be used for tasks like renaming files, removing sensitive data, or restructuring the repository.

99.Describe Git "force push" and its implications.

Answer: A force push (git push --force) is used to overwrite remote branch history with local changes. It should be used with caution, as it can lead to loss of commits and disrupt collaboration.

100.How do you squash multiple commits into a single commit during an interactive rebase?
Answer: During an interactive rebase, mark commits as "squash" or "s" to combine them into a single commit. The rebase process will prompt you to edit the commit message before finalizing the changes.

101. What is GIT?
Answer : GIT is a distributed version control system and source code management (SCM) system with an emphasis to handle small and large projects with speed and efficiency.

102. What is Distributed Control System?
Answer : We work in our local machine and later we transfer the code to Centralized repository (GitHub). We don't need to connect to centralized repository to work.

103. What is GIT version control?
Answer :
• GIT version control allows you to track the history of a collection of files (code files).
• It supports creating different versions of file collection. Each version captures a snapshot of the files at a certain point of time and You can revert the collection of files using the snapshot. (You can develop the code in different versions of java. and you can merge in Git)
• VCS allows you to switch between these versions. These versions are stored in a specific place, typically called as repository. (You can switch between different versions of java in between development process)

104. What is a repository in GIT?
Answer : A Git repository contains the history of a files.

105. How can you create a local repository in Git?
Answer : By using # git init command create a local repository.

106. What is 'bare repository' in GIT?
Answer A bare repository in Git just contains the version control information and no working files (no tree) and it doesn't contain the special .git sub-directory.

107. How to configure GitHub repository locally?
Answer :
git config --global user.name "user_name"
git config --global user.email "user_email"

108. How to Create Alias to git commands
Answer :
git config --global alias.lo "log --oneline" -----> To create an Alias to Command
git config --global --unset alias.lo -----> To Remove an Alias
git config --global --unset user.name -----> to remove username

109. What is the git clone?
Answer : To download an existing repository from Centralized (Github) to local system.
git clone <url>

110. What is 'git add'?
Answer : To add files from work area to Index/staging/cache area.
git add <file_name1> <file_name2>

111. What is Staging Area?
Answer : Staging area means "holding area". Before the commits, it can be formatted and reviewed in an intermediate area known as staging or Index Area.

112. What is the use of 'git log'?
Answer : To see the commits. Also, we can find specific commits in your project history- by author, date, content or history.
git log -----> To show the Git Commits
git log -5 -----> To show Recent 5 Commits
git log --oneline -----> To Display the each commit in one line
git log --since=2018-01-21
git log --until=2018-03-18
git log --author="user_name"
git log --grep="Index"
git log --oneline --author="user_name"

113. How can we add modified/updated/edited files to the staging area and commit then at the same time?
Answer : # git commit -a -m "Do Something once more"

114. How to edit an incorrect commit message in Git? Or How can you fix a broken commit?
Answer : # git commit --amend -m "This is your new Git Message"

115. How to get back a commit to staging area?
Answer : # git reset --soft <previous_commit id>

116. How to get back a file from staging area to working area?
Answer : # git reset head <file_name>

117. How to get back a commit to work area?
Answer : # git reset --mixed <previous commit id>

118. What is git reset?
Answer : Reset the current HEAD state to specific state.

119. What is 'head' in git and how many heads can be created in a repository?
Answer : A 'head' is simply a reference to a commit object. In every repository, there is a default head referred as "Master". A repository can contain any number of heads.

120. What is .gitignore file?
Answer : Keep the files names in .gitignore then that files not add and commit, just skip that files while adding and committing.

121. How to see the difference between 2 commits?
Answer : # git diff <commit_id1>..<commit_id2>

122. when file have staging area or file have committed if file is deleted in local repository unfortunately how to get back that file to staging area?
Answer : # git checkout --<file_name>

123. How to create a branch?
Answer : # git branch <branch_name>

124. How to checkout to branch?
Answer : # git checkout <branch_name>

125. How to create branch while checkout?
Answer : # git checkout -b <branch_name>

126. How do you rename the local branch?
Answer : # git branch -m <old_branch_name> <new_branch_name>

127. How to see the branch list?
Answer : # git branch

128. How to see the remote branch list?
Answer : # git branch -r
Or
git remote show origin

129. How to see the local and remote branch list?
Answer : # git branch -a

130. How to delete a branch?
Answer : # git branch -d <branch_name>
Or
git branch -D <branch_name>

131. How to delete a Remote Branch?
Answer : # git push origin -d <branch_name>

132. How to see the difference between 2 branches
Answer : # git diff <branch1>..<branch2 >

133. What is git push?
Answer : git push is to push commits from your local repository to a remote repository.

134. How do you push the files to master branch in remote repo?
Answer : #git push (you must be in master branch)

135. How do you push files from local to particular branch in remote repo?
Answer : #git push origin <branch_name>
(or)
#git push --set-upstream <branch_name>

136. How to push new branch and its data to remote repository?
Answer : #git push <github_repository_path> <branch_name>
(or)
#git push --set-upstream <branch_name>

137. What is git pull?
Answer : Git pull downloads and merges a 'branch data' from remote repository to local repository. It may also lead to 'merge conflicts' if your local changes are not yet committed. Use 'git stash' command to Hide your local changes before git pull.
git pull (git fetch + git merge.)

138. How do you pull a file from particular remote branch?
Answer : # git pull origin <branch_name>

139. How do you download a remote branch to local without merge?
Answer :
git fetch origin <branch_name>
git checkout <downloaded_branchname>

140. What is git Fetch?
Answer : git fetch is only downloads new data from a remote repository, but it doesn't integrate any of the downloaded data into your working files. All it does is provide a view of this data.
git fetch <branch_name>
git fetch origin <branch_name>

141. What is difference between git clone & git pull?
Answer :
If you want to download whole existing repository than use Git Clone.
If you have already repository but you want to take new updates of existing repository than use git pull command.

142. What is git merge?
Answer : Git merge is used to combine two branches.
git merge <branch_name>
Note: you should be in target branch. Then run the command

143. What is git conflict? What is the scenario you will get git conflict error?
Answer : For example, if you and another person both edited the same file on the same lines in different branches of the same Git repository, you'll get a merge conflict error when you try to merge these branches. You must resolve this merge conflict with a new commit before you can merge these branches.

144. How do you resolve merge conflict?

Answer : Will inform the developers regarding this merge conflict. They will change the code and inform us. edit the files to fix the conflicting changes and then add & commit.

145. How do you skip from merge conflict?
Answer : #git merge --abort

146. What is the function of 'git rm'?
Answer : To remove the file from the work area/staging area and also from your disk 'git rm' is used. You can revert a deleted file, if it is deleted using 'git rm'. If you deleted a file 'rm' command then you can't get it.

147. How will you know in GIT if a branch has been already merged into master?
Answer : git branch -merged It lists the branches that have been merged into the current branch.
git branch -no-merged It lists the branches that have not been merged.

148. What is branching? What is the purpose of branching in GIT?
Answer : Git supports branching which means that you can work on different versions of your collection of files. A branch allows the user to switch between these versions so that he can work on different changes independently from each other.

149. What is the criteria you merge two branches?
Answer : We have developed one module in one branch and another module in another branch. After the development, based on the requirement we do merge these two branches. Or One branch is development branch, another branch is test branch.

150. Describe branching strategy you have used?
Answer :
Feature branching
A feature branch model keeps all of the changes for a particular feature inside of a branch. When the feature is fully tested and validated by automated tests, the branch is then merged into master.

Task branching
In this model each task is implemented on its own branch with the task key included in the branch name. It is easy to see which code implements which task, just look for the task key in the branch name.

Release branching

Once the develop branch has acquired enough features for a release, you can clone that branch to form a Release branch. Creating this branch starts the next release cycle, so no new features can be added after this point, only bug fixes, documentation generation, and other release-oriented tasks should go in this branch. Once it is ready to ship, the release gets merged into master and tagged with a version number. In addition, it should be merged back into develop branch, which may have progressed since the release was initiated.

151. What is git stash?

Answer : Stashing takes the Temporary stored state of your working directory.
git stash save "<message>" ------> to store the data into stash
git stash list ------> to see the stash list
git stash apply <stash#> ------> to copy the data into branches
git stash pop <stash#> ------> to move the data into branches
git stash drop <stash#> ------> to delete the particular stash
git stash clear ------> delete the entire stash list

152. When we use git Stash?

Answer :

- If you are checking out from one branch to another branch but you have uncommitted file that you don't want to move then keep that file in stash area.
- When you are merging two branches and you don't want some files to merge, then we move that files to stash area.
- When you are pulling (fetch + merge) a branch/file and you don't want some files to merge, then we move that files to stash area.

153. What is another option for merging in git?

Answer : # git rebasing command is an alternative to merging in git.

154. What is difference between git merge and git rebase?

Answer :

- git merge applies all unique commits from branch A into branch B in one commit with final result.
- git rebase gets all unique commits from both branches and applies them one by one.
- git merge doesn’t rewrite commit history, just adds one new commit
- git rebase rewrites commit history but doesn’t create extra commit for merging

155. How do you undo the last commit?
Answer : # git revert <commit_id>

156. How to Change the URL for a remote Git repository?
Answer : # git remote set-url origin git://this.is.new.url

157. What is pull request?
Answer : Take some changes from a particular branch and bring them into another branch.

158. Why GIT better than Subversion (SVN)?
Answer : Git is an opensource version control system; it will allow you to run 'version' of a project. Multiple developers can check out, and upload changes and each change can then be attributed to a specific developer.

159. How to Lock a branch? why we need to lock a branch?
Answer :

- On GitHub, navigate to the main page of the repository.
- Under your repository name, click Settings.
- In the left menu, click Branches.
- Select the branch you want to mark protected using the drop-down menu.
- Select Protect this branch.

160. How to delete Repository in GitHub?
Answer :

- On GitHub, navigate to the main page of the repository.
- Under your repository name, click Settings.
- Scroll to the bottom of the page and you will find Delete this repository button
- When you click on that button, another pop up will appear, here you need to type in the name of your repository name and click on the button bellow which says: I understand the consequences, delete the repository.

161. how to give an access to a specific person to repository?
Answer :
You can invite users to become collaborators to your personal repository.

- Under your repository name, click Settings.
- In the left sidebar, click Collaborators.
- Under "Collaborators", start typing the collaborator's username.
- Select the collaborator's username from the drop-down menu.
- Click Add collaborator.

- The user will receive an email inviting them to the repository. Once they accept your invitation, they will have collaborator access to your repository.

GIT (Tasks) :-

- Storing scripts (Playbooks & Docker files) in GitHub
- Sharing (Playbooks & Docker files) with Team members with the help of Github
- Managing Github permissions Like working on Public & Private repos.
- Integrating Github with Jenkins in the CI-CD process
- Running Maven Jobs after pulling code from GitHub
- Conducting weekly training sessions （A21 employees in the team）
- Scoring Code (Java, Python...) in Github given by developers a Testers
- Providing GitHub repo accessto alienta when they need
- Creating Private repos while working on secure projects
- Documenting new learning's & Inventions so that other team members/new joiners can refer to them.
- Setting git infrastructure in all team members' machines who are not familiar with git
- Taking older versions of code as we need from GitHub
- Forking & Cloning GitHub Repos
- Setup/remove ssh password less connection to GitHub with all employees'