1. What is Git and GitHub?
2. What is the difference between Git and GitHub?
3. How do you create a new repository on GitHub?
4. What is a commit in Git?
5. How do you stage and commit changes in Git?
6. What is a branch in Git and why is it useful?
7. How do you create a new branch in Git?
8. What is a pull request on GitHub?
9. How do you fork a repository on GitHub?
10. How do you clone a repository from GitHub to your local machine?
11. What is the purpose of the git clone command?
12. How do you push changes from your local Git repository to GitHub?
13. How do you pull changes from a remote repository in Git?
14. How do you resolve merge conflicts in Git?
15. How do you revert a commit in Git?
16. How do you create a new branch for a feature or bug fix?
17. What is the purpose of a .gitignore file and how do you use it?
18. How do you revert a commit that has already been pushed to a remote repository?
19. How do you review changes made in a pull request on GitHub?
20. How do you merge changes from one branch into another branch using a pull request?
21. How do you resolve conflicts when merging a pull request on GitHub?
22. How do you close an issue in GitHub using a commit message?
23. What is Git branching strategy and why is it important in collaborative development?
24. Explain the difference between the "feature branch" and "release branch" in a Git branching strategy.
25. What is the purpose of a "develop" branch in the Git branching model?
26. What is the "master" branch in Git and how is it typically used in a branching strategy?
27. Describe the "pull request" workflow in the Git branching model.
28. What are some popular Git branching models or strategies (e.g., GitFlow, GitHub Flow, GitLab Flow)?
29. Explain the concept of "hotfix" branches in a Git branching strategy and when they are typically used.
30. How does a Git branching strategy facilitate parallel development of multiple features or bug fixes?
31. What actions should be taken to ensure code quality and stability in a Git branching model?

==================================================================================

Create a Public GitHub Repository with Branches and Commits

Instructions:

* Create a new public repository on GitHub called "my-project".
* Clone the repository to your local machine.
* Create a new branch called "feature-branch".
* Switch to the "feature-branch".
* Create a new file called "script.py" and add some code to it.
* Commit the changes to the "feature-branch".
* Push the "feature-branch" to the remote repository.
* Switch back to the main branch (often called "main" or "master").
* Merge code to main.
* Create another branch called "bug-fix".
* Update the "script.py" file to fix a bug.
* Commit the bug fix changes.
* Push the "bug-fix" branch to the remote repository.
* Merge the "bug-fix" branch into the main branch.
* Push the updated main branch to the remote repository.
* Your task is to perform the above steps on GitHub and describe the commands and steps you would use to complete each task.

Share the links of your Public Github repo.

1. Make a flow chart based on your understanding that shows the connection between Infra (Iac), Application Code, Repository (along with branches), Azure DevOps (Boards, Project settings, Pipelines, Environments), Azure Portal Resources.

OR

Explain the DevOps process flow diagram?

2. What are different types of Authentications and Authorization ways used in DevOps?

List the differences between Service Connection, Service Principle, Managed Identity (& its types), PAT (Personal Access token).

3. What is tagging Process? What do you mean by Tag on Branch? On which branch TAG should be created and why? How is Tag created?