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).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  | Typically created within the context of an Azure DevOps project. It is established during the pipeline or release definition setup. | Generated by users manually within the Azure DevOps portal |
|  |  |  | Tied to the lifecycle of the Azure DevOps project or organization. Creation, updates, and deletions can be managed within the Azure DevOps portal. | Manually generated by users, with expiration and renewal managed by the user. |
|  |  |  | Associated with a specific service principal, but the permissions are typically managed through the Azure DevOps portal. Permissions are often linked to the service principal's role in Azure DevOps, not directly through Azure RBAC. | Scoped to specific tasks within Azure DevOps (e.g., read, write, build, release) and can be fine-tuned based on the user's needs. |
|  |  |  | Primarily used within Azure DevOps pipelines and release definitions to connect to Azure services during build and deployment processes. | Typically used by individuals for manual interactions with Azure DevOps, such as accessing repositories, triggering builds, or managing resources. |
|  |  |  | Typically scoped to an entire Azure subscription or specific resource groups. The scope is often managed at the Azure DevOps project level. | Scoped to specific tasks within Azure DevOps and may have a limited scope based on the user's role and permissions. |

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?

In version control systems, a tag is a reference to a specific point in the commit history of a branch. Tags are used to mark specific releases, milestones, or points of interest in the development timeline. They provide a convenient way to identify and reference a specific commit, making it easier to manage and navigate the project's history.

**Tagging Process:**

1. **Select a Commit:**
   * Choose a specific commit in your version control system that you want to tag. This is often a commit that represents a release or a significant point in the project's development.
2. **Create a Tag:**
   * Create a tag at the chosen commit. This tag serves as a reference to that specific state of the codebase.
3. **Tag Naming Conventions:**
   * Tags often have naming conventions, such as version numbers (e.g., v1.0.0), release names, or other identifiers that make it easy to understand the purpose of the tag.
4. **Push Tag to Remote Repository (Optional):**
   * If you're using a distributed version control system and want to share the tag with others, you may need to push the tag to the remote repository.

**Tag on Branch:**

* A tag is typically created on a specific branch at a particular commit. It's a snapshot of the code at that moment in time.
* In Git, you can create a tag on any branch, including the main development branch (e.g., **main** or **master**) or a release branch.

**Which Branch Should a Tag be Created On and Why?**

1. **Main Branch (e.g., main or master):**
   * Tags on the main branch are commonly used to mark releases or significant milestones in the overall project.
2. **Release Branch:**
   * In some workflows, a separate release branch may be created specifically for preparing and stabilizing a release. Tags on a release branch help to isolate release-related changes.
3. **Feature Branch (Less Common):**
   * It's less common to tag feature branches because these branches are typically used for ongoing development and may undergo changes before they are merged into the main branch.

**How is a Tag Created?**

Git Command Line:

To create a lightweight tag:

git tag <tag\_name> <commit\_hash>git tag <tag\_name> <commit\_hash>

To create an annotated tag (with additional information like a message):

git tag -a <tag\_name> -m "Tagging version <version\_number>" <commit\_hash>

GitHub/GitLab/Bitbucket (Web Interface):

* On many platforms like GitHub, GitLab, or Bitbucket, you can create tags through the web interface. There is usually an option to create a new release or tag, allowing you to choose a specific commit.

Remember to push the tags to the remote repository if you want others to access them:

bashCopy code

git push origin <tag\_name>

Always follow the best practices and conventions of your team or project when creating and managing tags. Tags play a crucial role in versioning and release management, providing a clear and unambiguous reference to specific points in your project's history.

Top of Form