**Git-hub Interview Questions**

1. What is the procedure of pushing the changes from local to git hub?
2. What is the difference between git pull and git fetch?
3. What is the difference between Dvcs and Cvcs?
4. Why you are using branching strategy in git hub ?
5. What is git tagging?
6. What is git rebase?
7. What is merge conflict in github?

**Maven**

1. Tell me about maven build life cycle ?

**Sonar Qube**

1. What is the use of sonarqube?
2. How did you integrate sonarqube with Jenkins?

**Here are detailed explanations for your Git, Maven, and SonarQube interview questions.**

**GitHub Interview Questions**

**1. How do you push changes from local to GitHub?**

To push changes to GitHub from your local repository, follow these steps:

1. Initialize Git (if not already initialized)
2. git init
3. Add a remote repository (only once per project)
4. git remote add origin <repository\_url>
5. Check the current branch
6. git branch
7. Add changes to the staging area
8. git add .
9. Commit changes with a meaningful message
10. git commit -m "Commit message"
11. Push changes to GitHub (main branch example)
12. git push origin main

If pushing for the first time:

git push -u origin main

**2. Difference between git pull and git fetch?**

| Command | Function | When to Use? |
| --- | --- | --- |
| git fetch | Retrieves changes from a remote repository but does not merge them into your working branch. | When you want to see changes before merging. |
| git pull | Fetches updates and automatically merges them into the current branch. | When you want to update your local branch immediately. |

Example:

git fetch origin

git merge origin/main # Merge fetched changes manually

git pull origin main # Fetch + merge in one step

**3. Difference between DVCS and CVCS?**

| Feature | CVCS (Centralized Version Control System) | DVCS (Distributed Version Control System) |
| --- | --- | --- |
| Repository Type | Single central repository | Each user has a full repository copy |
| Internet Dependency | Requires internet to commit changes | Can commit offline |
| Examples | SVN, Perforce | Git, Mercurial |
| Speed | Slower | Faster due to local commits |
| Backup | Risky (single point of failure) | Safe (multiple copies exist) |

**4. Why use branching strategy in GitHub?**

Branching allows developers to work independently on different features without affecting the main code. Common strategies:

* Feature Branching – Develop new features in separate branches (feature-login, feature-ui)
* Git Flow – Uses develop, feature, release, and hotfix branches
* GitHub Flow – Uses main and short-lived feature branches
* Trunk-Based Development – Developers commit directly to main

Example:

git checkout -b feature-new-ui

git add .

git commit -m "New UI changes"

git push origin feature-new-ui

**5. What is Git Tagging?**

Tags are used to mark specific points in the Git history, usually for releases.

* Creating a tag:
* git tag -a v1.0 -m "Version 1.0 release"
* Listing tags:
* git tag
* Pushing a tag to GitHub:
* git push origin v1.0

**6. What is Git Rebase?**

Rebase moves a branch to a new base commit, keeping a linear history.

Example:

git checkout feature-branch

git rebase main

This applies the changes from feature-branch on top of main.

**7. What is a merge conflict in GitHub?**

A merge conflict occurs when Git cannot automatically merge changes due to overlapping modifications.

Fixing a Conflict:

1. Identify conflicts using:
2. git status
3. Open the conflicting file and resolve conflicts manually.
4. Mark conflict as resolved:
5. git add .
6. git commit -m "Resolved merge conflict**"**

**Maven Interview Questions**

**8. Maven Build Lifecycle**

Maven has three primary lifecycle phases:

| Phase | Description |
| --- | --- |
| clean | Removes target directory (mvn clean) |
| default | Builds and tests the project (mvn compile, mvn test) |
| site | Generates project documentation (mvn site) |

Key commands:

mvn clean package # Clean, compile, and package the code

mvn install # Installs project into local repo

**SonarQube Interview Questions**

**9. What is the use of SonarQube?**

SonarQube is a tool for code quality analysis and static code review. It helps in:

* Detecting bugs, security vulnerabilities, and code smells
* Maintaining coding standards (e.g., Java, Python, JavaScript)
* Generating reports for DevOps teams
* Integrating with CI/CD pipelines (e.g., Jenkins, GitHub Actions)

**10. How to integrate SonarQube with Jenkins?**

1. Install SonarQube Plugin in Jenkins
   * Go to Manage Jenkins → Plugins → Install SonarQube Scanner Plugin.
2. Configure SonarQube in Jenkins
   * Manage Jenkins → Global Tool Configuration → Add SonarQube server details.
3. Set up a Jenkins Pipeline Add the following in Jenkinsfile:
4. pipeline {
5. agent any
6. stages {
7. stage('SonarQube Analysis') {
8. steps {
9. script {
10. def scannerHome = tool 'SonarQubeScanner'
11. withSonarQubeEnv('SonarQube') {
12. sh "${scannerHome}/bin/sonar-scanner"
13. }
14. }
15. }
16. }
17. }
18. }