WEEK 3 scenario based questions answer

scenario based questions on working with remote repository in collaboration.

### 1. ****Scenario: You've cloned a repository and made some changes to a local branch. Now you want to push these changes to the remote repository, but you're getting an error saying "rejected - non-fast-forward." How would you resolve this?****

* **What they’re looking for**: Understanding of pushing changes, remote branches, and potential conflicts.
* **Key concepts**: git pull, git fetch, git rebase, git merge

**Sample answer**:  
"The error typically occurs when your local branch is behind the remote branch, meaning someone else pushed changes after you cloned the repo. To resolve this, I'd first fetch the latest changes from the remote with git fetch origin. Then, I'd rebase my local changes onto the latest changes from the remote by running git rebase origin/main. After that, I would push my changes to the remote repository."

### 2. ****Scenario: You’ve been working on a feature branch, and now you need to push it to the remote repository. However, the remote repository already has a**** main ****branch. How do you push your feature branch without affecting the**** main ****branch?****

* **What they’re looking for**: Understanding of branching and pushing to a remote repository.
* **Key concepts**: git push, git branch, git remote

**Sample answer**:  
"I would first ensure I am on the feature branch by running git branch. If I’m not on the feature branch, I would switch to it with git checkout feature-branch. Then, I'd push my feature branch to the remote with git push origin feature-branch. This will create the branch on the remote without affecting the main branch."

### 3. ****Scenario: You cloned a remote repository, but after a while, the repository’s structure changed and new branches were added. How would you keep your local repository updated with the latest changes from the remote repository?****

* **What they’re looking for**: Familiarity with syncing local and remote repositories.
* **Key concepts**: git fetch, git pull, git remote

**Sample answer**:  
"To update my local repository, I would run git fetch origin to fetch the latest changes from the remote repository. This command will update my local reference of the remote branches but won't merge them into my working directory. To integrate those changes, I could either switch to the main branch and run git pull origin main, or I could choose to merge or rebase specific branches based on my workflow."

### 4. ****Scenario: A colleague has pushed some changes to the**** main ****branch, but you have local changes in the same branch. You want to pull their changes, but you want to avoid merge conflicts. What steps would you take?****

* **What they’re looking for**: Knowledge of handling merge conflicts and safe workflows for pulling remote changes.
* **Key concepts**: git pull, git fetch, git rebase, git merge

**Sample answer**:  
"First, I would run git fetch origin to get the latest changes from the remote. After fetching, I would rebase my local changes onto the updated main branch with git rebase origin/main. This ensures a clean, linear history and minimizes merge conflicts. If there are any conflicts during the rebase, I would resolve them and continue the rebase with git rebase --continue. Once the rebase is done, I would push my changes to the remote."

### 5. ****Scenario: You accidentally pushed a sensitive file (e.g., API keys) to the remote repository. How would you fix this situation?****

* **What they’re looking for**: Understanding of removing sensitive data from Git history and the repository.
* **Key concepts**: git reset, git revert, git filter-branch, git rebase, history rewriting, .gitignore

**Sample answer**:  
"To remove the sensitive file from the repository’s history, I would use git filter-branch or the newer tool git filter-repo to rewrite the commit history and remove the file from all previous commits. Once done, I would force-push the rewritten history to the remote repository with git push --force origin main (after notifying my team to avoid disrupting their work). **I would also add the sensitive file to .gitignore to prevent it from being committed again."**

### 6. ****Scenario: You’re working on a feature branch, and your manager requests that you integrate the latest changes from**** main ****into your feature branch. What steps would you take?****

* **What they’re looking for**: Understanding of integrating changes from the main branch into a feature branch.
* **Key concepts**: git merge, git rebase, git fetch, git pull

**Sample answer**:  
"I would start by fetching the latest changes from the remote with git fetch origin. Then, I’d checkout my feature branch if I’m not already on it (git checkout feature-branch). To integrate the changes, I would either merge or rebase main into my feature branch. If I want to keep the history clean, I would use git rebase origin/main to rebase my feature branch on top of the latest main. If I prefer a simpler approach, I could use git merge origin/main to merge the changes from main into my feature branch."

### 7. ****Scenario: You cloned a remote repository, but later you find that you need to push your changes to a different remote repository. How do you configure your local repository to push to this new remote?****

* **What they’re looking for**: Knowledge of handling multiple remotes and changing remote URLs.
* **Key concepts**: git remote add, git remote set-url, git remote -v

**Sample answer**:  
"To push my changes to a different remote, I would first check the existing remotes by running git remote -v. If the new remote is not listed, I would add it with git remote add new-remote <new\_repository\_url>. If the new repository URL is just replacing the current one, I would use git remote set-url origin <new\_repository\_url>. Then I could push my changes to the new remote with git push new-remote branch-name."

### 8. ****Scenario: After running**** git pull****, you notice that your local branch is behind the remote branch. How would you proceed to bring your local branch up to date without losing your local changes?****

* **What they’re looking for**: Understanding of safe strategies to keep both local and remote branches in sync.
* **Key concepts**: git pull, git fetch, git rebase, git merge

**Sample answer**:  
"I would start by running git fetch origin to fetch the latest changes from the remote without modifying my local branch. Then, I would either use git rebase origin/branch-name to rebase my local commits on top of the fetched commits or git merge origin/branch-name to merge the remote changes into my local branch. Rebase would be preferred if I want to keep a cleaner commit history."

### 9. ****Scenario: You’re working on a project with multiple collaborators, and you notice that your local changes conflict with changes that have been pushed by others. How would you resolve the conflicts?****

* **What they’re looking for**: Understanding of how to handle merge conflicts in a collaborative environment.
* **Key concepts**: git merge, git rebase, conflict resolution, git status

**Sample answer**:  
"When I encounter a merge conflict, I would first use git status to see which files are conflicting. Then, I would open those files and manually resolve the conflicts. After resolving the conflicts, I would add the resolved files to the staging area with git add <file-name>. Finally, if I was merging, I would commit the merge, and if I was rebasing, I would continue the rebase with git rebase --continue. Once all conflicts are resolved and committed, I would push the changes to the remote repository."

### 10. ****Scenario: You’ve pushed a feature branch to a remote repository, but now you need to delete the branch from the remote. How would you do that?****

* **What they’re looking for**: Knowledge of cleaning up remote branches after a feature is complete.
* **Key concepts**: git push --delete, git branch -d

**Sample answer**:  
"To delete a branch from the remote, I would run the command git push origin --delete feature-branch. This will remove the branch from the remote repository. If I also want to delete it from my local machine, I would use git branch -d feature-branch to delete the local branch (assuming it has been merged). If the branch hasn’t been merged, I’d use git branch -D feature-branch to forcefully delete the branch."

-----

Scenario :

You are working on a collaborative project hosted on GitHub with a team of four developers. The main branch is main, and each developer works on their own feature branches.

You are assigned to implement a new UI component in a branch called feature/ui-update. Meanwhile, another team member has made a critical bug fix and pushed it directly to the main branch. When you try to push your changes, your push fails due to upstream changes.

Additionally, a teammate has submitted a patch file with a small CSS fix and shared it via email. You need to apply the patch, test it, and merge everything into main without breaking any functionality.

### ****1. What command will you use to bring your local main branch up to date with the remote repository before merging it into your feature branch?****

bash

CopyEdit

git checkout main

git pull origin main

This will ensure your local main branch is up to date with the latest changes from the remote repository (e.g., the critical bug fix your teammate made).

### ****2. How will you update your**** feature/ui-update ****branch to reflect the latest changes from main?****

You have two main options: **merge** or **rebase**.

#### Option A: Merge (safer for shared branches)

bash

CopyEdit

git checkout feature/ui-update

git merge main

#### Option B: Rebase (keeps a cleaner history but may require resolving conflicts)

bash

CopyEdit

git checkout feature/ui-update

git rebase main

Use rebase only if the branch is not yet pushed/shared with others.

### ****3. Which command should you use to attempt pushing your local feature branch again, and what should you do if it fails due to conflicts?****

Try to push:

bash

CopyEdit

git push origin feature/ui-update

If it fails due to upstream changes:

* Pull with rebase to avoid unnecessary merge commits:

bash

CopyEdit

git pull --rebase origin feature/ui-update

* Resolve any conflicts manually (Git will tell you which files are in conflict).
* After resolving, continue the rebase:

bash

CopyEdit

git add .

git rebase --continue

Then push again:

bash

CopyEdit

git push origin feature/ui-update

### ****4. How do you apply a**** .patch ****file provided by your teammate and include it in your commit history?****

Assuming you received a file named fix.patch:

bash

CopyEdit

git apply fix.patch

Then stage and commit the changes:

bash

CopyEdit

git add .

git commit -m "Apply CSS fix from patch file"

✅ This will include the patch content in your Git history.

### ****5. After successful testing, describe the steps (with commands) to merge your feature branch into main and push it to GitHub.****

Assuming you're ready to merge:

bash

CopyEdit

# Make sure you're up to date

git checkout main

git pull origin main

# Merge the feature branch

git merge feature/ui-update

# Push the merged changes to GitHub

git push origin main

You can optionally delete the feature branch locally and remotely:

bash

CopyEdit

git branch -d feature/ui-update # delete locally

git push origin --delete feature/ui-update # delete remotely

**Patch**

#### 🔹 To create a patch from the last commit:

bash

CopyEdit

git format-patch -1 HEAD

This will create a file like:

sql

CopyEdit

0001-Your-commit-message.patch

#### 🔹 To specify an output directory:

bash

CopyEdit

git format-patch -1 HEAD -o patches/

This creates the patch file inside a folder called patches.

### ✅ Bonus: Combine into one file manually

If you really want to redirect all patches into a **single file**, you can do this:

bash

CopyEdit

git format-patch -1 HEAD

cat 0001-\*.patch > my\_patch\_file.patch

Now my\_patch\_file.patch contains the same content and can be shared or applied.