GIT ADVANCED

Git Advanced Commands

GIT PUSH USING SSH

- Generating Key
  - ssh-keygen -t rsa -b 4096 -C "sonam.gravity@gmail.com"
- starting agent in the background
  - eval $(ssh-agent -s)
- set up private key ass ssh-client
  - ssh-add ~/.ssh/id_rsa
- it will ask you to enter passphrase. copy public key for adding it to GitHub
  - clip < ~/.ssh/id_rsa.pub
- GitHub --> settings --> SSH & GPG key --> add new SSH key and paste the copied key
- Once its configured you can set origin with ssh command rather that using https and that it.
- You can push code on GitHub directly using ssh.
- so your origin setup look like this: git remote add origin git@github.com:sonam-niit/TestSSH.git

GIT REBASE

Git **rebasing** is a powerful command used to reapply commits from one branch onto another.

It offers a cleaner commit history compared to a merge.

It is helpful when collaborating on shared repositories.

WHEN TO USE REBASE

- **Keep a clean commit history:**
  - When working on a feature branch, rebase before merging it into the main branch to avoid merge commits and keep the history linear.
- **Integrate changes from the main branch:**
  - If your feature branch is behind the main branch, rebase it to include the latest changes.
- **Squash commits:**
  - Combine multiple small commits into one to create a cleaner commit history.
- **Avoid merge bubbles:**
  - Rebase keeps the history linear, whereas merging creates a "bubble" or "branch-off" effect in the graph.

WHERE REBASE IS USEFUL

- **Feature branch updates:**
  - Your team is actively working on the main branch, and you want to bring your feature branch up to date with it.
- **Interactive rebasing to clean up commits:**
  - You made multiple commits like "Fix typo" or "Add missing semicolon." Squash these into a single, meaningful commit.
- **Rewriting history before sharing:**
  - Before pushing your branch to a remote repository, rebase to rewrite commits and make the history concise.

AVOID USING REBASE

- **Avoid rebasing public/shared branches:**
  - Rebasing rewrites commit history, which can cause issues for other developers if the branch is already shared.
- **Rebase only when you fully understand its impact:**
  - Use it on private branches or local changes to avoid disrupting team workflows.

SCENARIO

- You are working on a feature branch called test, but new changes have been added to the main branch.
- You want to rebase your test branch onto the updated main branch to include the latest changes.
- Check out branch: git checkout test
- git rebase main
- Git automatically do below process
  - Move the commits from test temporarily.
  - Apply the commits from main to test.
  - Reapply your test commits on top of the updated main branch.
- Push Rebased Branch: git push --force

GIT STASH

- The git stash command temporarily saves your changes in a **stash**.
- What is Stash: a stack of uncommitted changes
- so you can switch branches or work on something else without committing them.
- Afterward, you can reapply those changes to your working directory.

STASH COMMANDS

- Save stash: git stash
- List stashes: git stash list
- Apply stash: git stash apply
- Pop a stash: git stash pop
- Drop a stash: git stash drop
- Clear all stashes: git stash clear
- Save untracked files in stash: git stash -u

SCENARIO

- **Switching Branches While Working on a Feature**
- You are working on feature-branch and have made changes to a file (file1.js).
- Before completing your work, a teammate asks you to fix an urgent bug on the main branch.
- Instead of committing your incomplete changes, you can stash them.
- Checkout to test branch: git checkout fetaure/test
- Suppose you modify file1.js and create a new file file2.js.
- Git status: you can see one modified and one untracked file.

- No you want to checkout to main branch to complete some urgent work but before leaving the branch execute below commands
- git stash –u
- Git stash list
- Checkout to main branch: git checkout main
- Complete the work and commit
- Go back to feature/test branch
- Apply the git stash: git stash apply
- Remove the stash: git stash drop

BENEFITS OF GIT STASH

- Temporarily saves incomplete work without committing.
- Helps maintain a clean working directory for branch switching.
- Allows multitasking without losing progress.

FORKING A REPOSITORY

- Forking a repository in GitHub creates a personal copy of someone else's repository under your account.
- Benefits:
- Make changes to a project without affecting the original repository.
- Submit improvements or bug fixes by creating pull requests from your forked repository.
- Start your own project based on the original repository.

LET'S FORK ONE REPO

- Go to any repository which you want to fork and click on fork button to fork the repository.
- When you fork the repository its copy is getting created in your account.
- To get that code in you local system you can clone that copy repo from your account and set upstream to existing repo.
- git remote add upstream https://github.com/original-author/project-repo.git
- To make some changes you can create branch
- Add and commit the changes and push changes to your branch.
- git push origin feature-branch
- You can submit pull request

ASSIGNMENT 3

ASSIGNMENT 4