GIT – Version control system (VCS) by using Amazon Web Services

Introduction:

- You can have code in local machine as well as in remote location.
- For remote location, we can use – GitHub/GitLab/bitbucket/AWS CodeCommit (so that we can interact with our team instead of working individually)

Code Management (Version Control)

- Version Control or the source code control is a practice of tracking and managing the changes of the software code.
- The VCS (Version Control System) helps team to work effectively and efficiently.
- VCS keeps track of every change happens to the code in a special kind of database and the developers can go back to their previous code state if they need.
- When a lot of developers work together on a code, there are good chances to mix the code and a lot of choices can happen due to changes made by many people.
- Version Control helps us to mitigate this problem.

GIT

- Git is the most common and widely used version control system in the world.
- It is an open-source system.
- Git was developed by Linus Torvalds in 2005.
- Git is an example of Distributed Version Control System (DVCS).
- DVCS means rather than having one place for full version history of the software.
- Here, you can have a separate copy of the code in your local machines as well with the full history of changes.
- Git works on the branching strategy, which means you can have many branches from your code just like a tree having branches connected to its trunk.
- By using Git, we can interact with our team instead of working individually.

What is GitHub/GitLab/Bitbucket?

- All these are code hosting platform for version control and collaboration.
- GitHub is a cloud-based solution for code versioning whereas you can use cloud-based as well as install the GitLab/Bitbucket in your environment.
- All these platforms use in git command line as the interactive program.

Some widely using commands

1. Clone a git repository

   --> git clone <repo URL>

2. Checking status of repository

   --> git status

   This command to show the status of git in a particular repo

3. Create a file

   --> touch <filename>

   --> touch file1

   --> touch file2 file3 file4

4. Adding or Removing files

   --> git add <filename>

   Git will start tracking the changes (stage our changes)

   --> git add file1

   It will add newly created file in the project/repository

   --> git add .

   It will add all newly created files in the project/repository

   --> git rm <filename>

   --> git rm file1

   It will remove the file(s) from the project/repository

5. Committing the changes locally

--> git commit -m "commit message"

It will commit our changes and keep a track of author of the changes and the time

--> git commit -m "This is my repo for testing/project purposes"

6. Checking the commit history

--> git log

7. Pushing the local changes to the remote repository

--> git push

LAB 1: (Creating EC2 instance)

1. Login to AWS console.
2. Create a server with Amazon Linux 2 AMI/RHEL/Ubuntu.
3. Connect the server only with Putty/GitBash.

LAB 2: (Create repo in local machine)

1. Create a folder on my local machine
2. initialize this folder using git init command
    a. --> git init <folder name>
3. Go inside this folder and run git status command to check the status,
4. created some empty files using touch command
    a. --> touch test
5. Again, run git status to see the changes, you can notice that the file is available but not tracked by Git.
6. Run git add <filename> to stage this change (git will start tracking this file), you may check it using git status once again.
7. Now commit our changes by running git commit -m "added test files"
8. Run git status once again and it will show you that the working tree is clean.

LAB 3: (Creating repo in remote location – GitHub)

1. In remote location – GitHub.
2. Create a new repository by clicking on **new** button
    a. Provide repo name
    b. Select whether it is a private or public repo (recommended is private)
    c. Initialize the repo by adding a **README.md** file.
3. Click on create repo and done.

LAB 4: (Working with Remote repo)

- Take the remote repo to your local machine
- Make the changes
- Commit the changes
- Push the changes to remote

1. Pick the clone URL of the repository from the GitHub repo.
2. Go to your local machine and clone this repo using git clone command
    a. git clone <repo URL that you will get by clicking on code button in GitHub>
3. You will be asked to provide username and password
4. Here you will get one error for password-based authentication depreciation.
5. Now, we need to create the personal access token to work with this repo.
    a. At the top right corner click on the user icon
    b. Go to settings
    c. Developer settings
    d. Personal access token
    e. Click on generate token
    f. Provide a note
    g. Expiration date and scope (select the very first checkbox for scope)
    h. Click on generate token
    i. Copy this token and keep it safe
6. Clone the repo again in your local machine, and this time provide the token in place of password while cloning.
7. Once cloned, go to the repo folder and add some sample files. we can use touch command to create empty files.
    a. touch file1 file2

8. Stage these changes by running git add file1 file2
9. Commit these changes by running git commit -m "<any message>"
10. git push --> It will ask you for username and password, provide the username and personal access token in place of password which we created above.
11. Go to the remote repo and see, you will be able to find your new changes.

LAB 5: (Pushing a locally created repo to GitHub)

1. Create one repo in our local machine and initialize it locally (we have done it in **Lab – 2**)
2. If not done then do **Lab – 2**
3. Create one remote repo with the same name as local repo in GitHub and do not initialize it.
4. Come to your local machine and run the following commands from inside your local repo (**Lab – 2**)
    a. git branch -M main (to change the name of branch as master branch is now known as main branch)
    b. git remote add origin <URL of your remote repo>
    c. git push -u origin main (to push your local branch to remote repo)

LAB 6: (Creating a new branch from your main branch)

1. Go to the repository at the place of main and click on the branch dropdown.
2. Type the name of your new branch that you want to create and click on create button.
3. A new branch will be created for you.
4. Make some changes in this branch directly from the console
5. Here you will see that your changes are only applied to your new branch but not to the main branch.

LAB 7: (Pull all the branches in your local machine)

1. Go to your local machine where you have the copy of your remote branch (**Lab – 4**).
2. Run the command "git pull" to pull all the new changes such as branches from the remote location
3. Run "git branch -a" to list down all the branches.
    a. The branch which starts with remotes/ ---> are remote branches
    b. The branch without remotes/ ---> they are available on your local copy a s well
4. Checkout to the feature branch or the branch that you created in (**Lab – 6**).
    a. git checkout <branch name>
5. Make sure that you are on the new branch by running **git status** or **git branch** command (notice the * mark)
6. Make some changes in this branch such as adding the files "touch file3 file4"
7. git status
8. git add <filename>
9. git commit -m "<msg>"
10. git push
11. Go to your GitHub portal once again
12. Here see that the new changes are only available in your feature branch but not in the main branch.

Lab 8: (Merge our feature branch with main branch)

1. Go to your GitHub repository
2. Check the changes in your feature branch (**Lab – 7**)
3. Go to the pull request tab and click on create **Pull Request** (**PR**)
4. Put your main branch (where u want to merge / destination) and the feature branch in the respective blocks
5. Click on create pull request and it will ask for a comment, just click again on create pull request.
6. Go to pull request tab once again and click on the pull request available there
7. Click on review changes and then merge
8. Once done, you can delete this feature branch if required or you can simply ignore it.
9. Go to your code in main branch and see the changes are now visible here.

Lab 9: (Go to local machine)

1. Go to your local machine where you have the copy of your remote repo (**Lab – 7**).
2. Checkout to the main branch
    a. git checkout <branch name>
3. Now run the command "git pull" to pull all the new changes such as branches from the remote location
4. Here see that the new changes are only available in your main branch

Recommended process followed by developers

- In main branch code
- Create a new feature branch
- You make the changes in this feature branch
- We create a Pull Request (PR)
- Share this **PR** with your **lead/manger** or the **person** who **reviews the code** before **merging**
- The reviewer will check your code and merge it with the main branch code.