MEAN Stack Notes

Technologies

1. GIT
2. HTML
3. CSS
4. Javascript
5. ES6+
6. Typescript
7. Angular
8. MySQL
9. MongoDB
10. Node.js & Express.js
11. DevOps
12. Micro Front end
13. Case study.

GIT:

It is a version controlling system.

It helps to collaborate each others work in a team.

There will be two repositories in the GIT

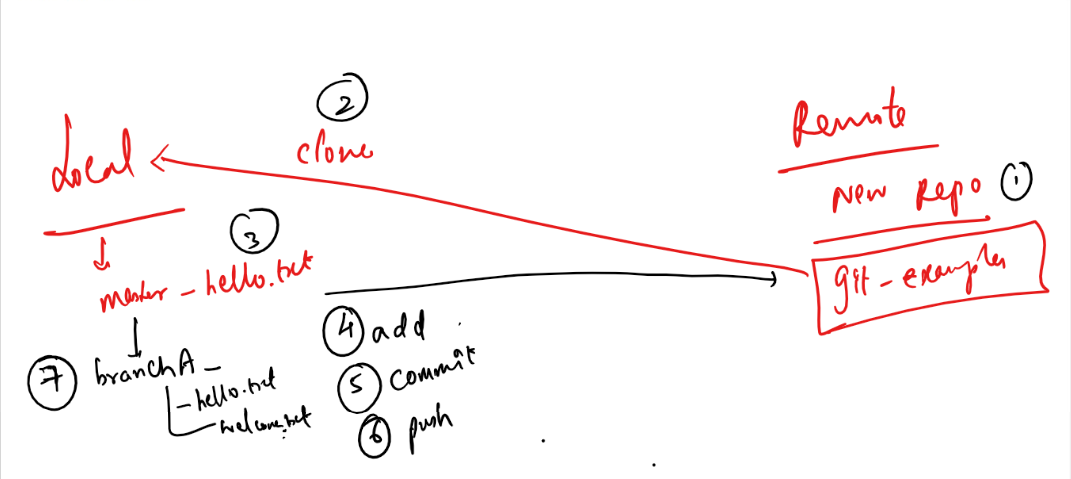
1. Remote: Will have access to everyone
2. Local: Local to the particular individual

Git provides commands to update each others work

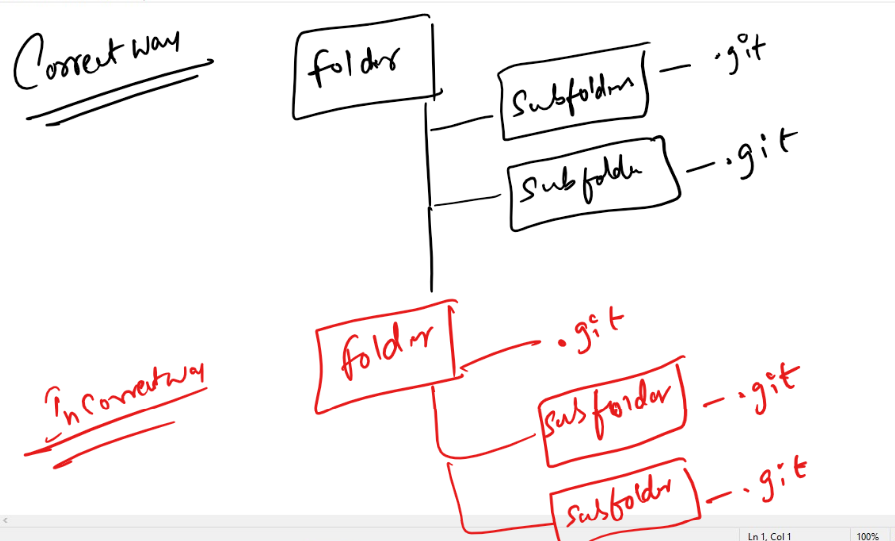
1. push: It tries to update the remote repository from the local repository
2. pull: It tries to update the local repository from the remote repository
3. clone: It creates a copy of remote repository in the local repository, it is done for the first time before you can enter push/pull
4. merge: It is used to manually merge the updates in the local repository from the local branch
5. add: It is used to add the changes to the staging area, only the changes in the staging area can be committed and pushed
6. commit: it is used to commit the changes, it creates one unique id which is used at the time of merging
7. status: It is used to see the changes done in the local repository, it shows the changes in the staging area and the changes not in the staging area

Git branches:

It is a pointer of particular commits, by default git gives you one branch called master/main branch, but every individual must work in custom branch not in the master branch

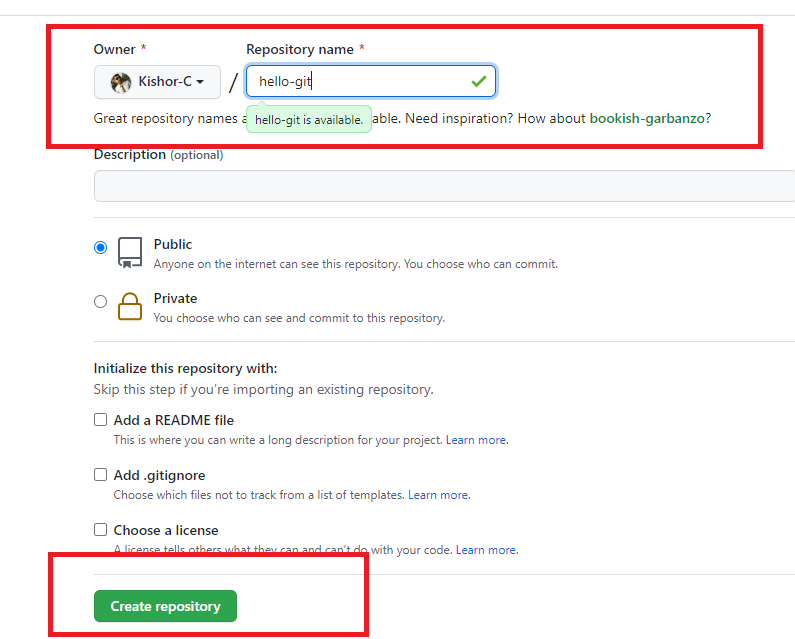


You must have a single .git folder which represents the git repository in the parent directory, but nested repositories you must not have ie.., an enclosing folder having .git and the sub folder also havening .git.

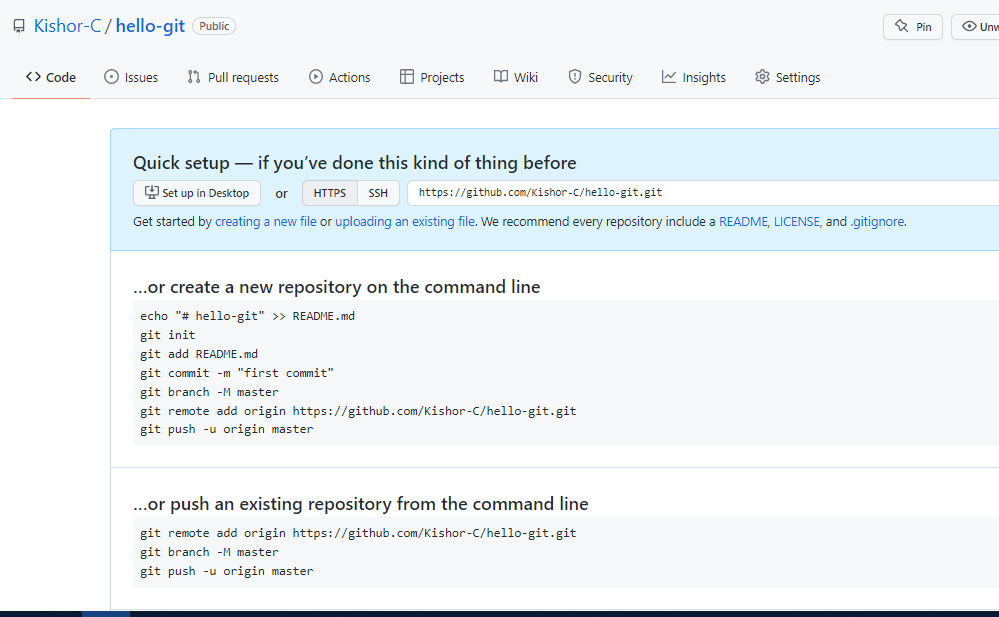


Steps to work with GIT

1. Create a GIT account & login
2. Create a remote repository & name it

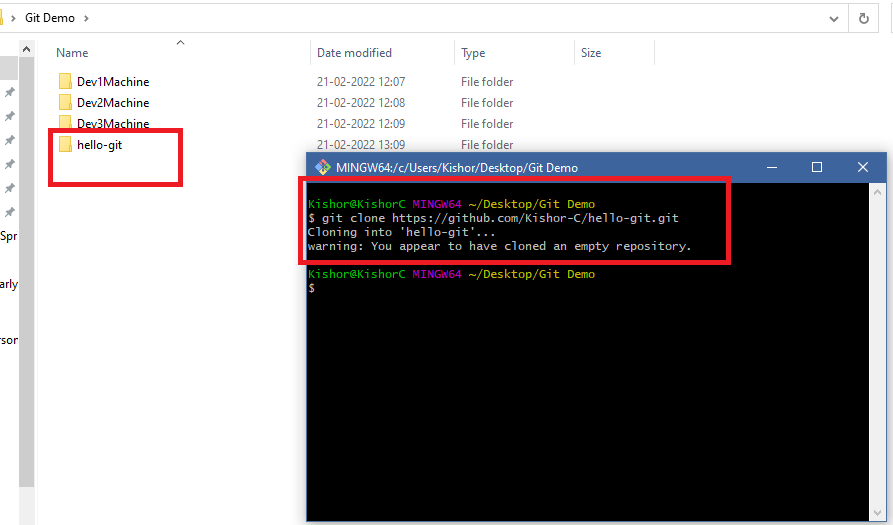


1. After you create the repository you will see the empty repository in the git website



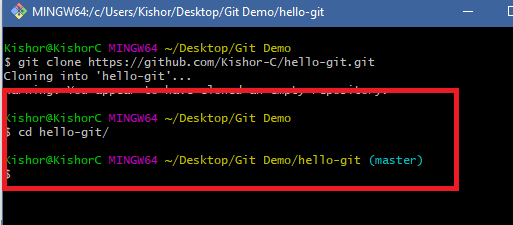
1. Clone the Remote repository in the local machine

Command: git clone URL



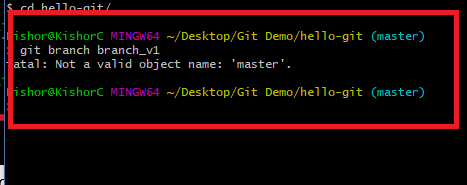
Note: You don’t see the branch name in the GIT bash, because you need to navigate to hello-git

1. Navigate to the local repository cloned



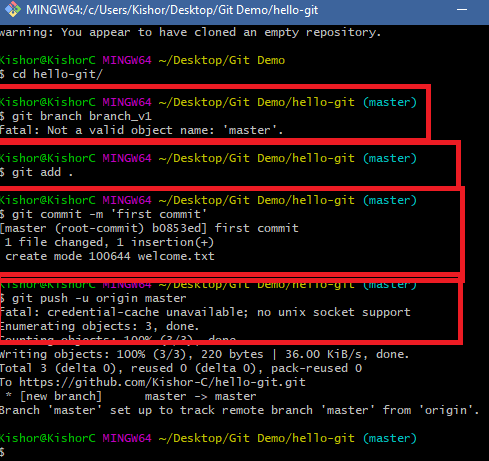
1. Create a custom branch when you want to do any changes/updates

Command: git branch branch\_name

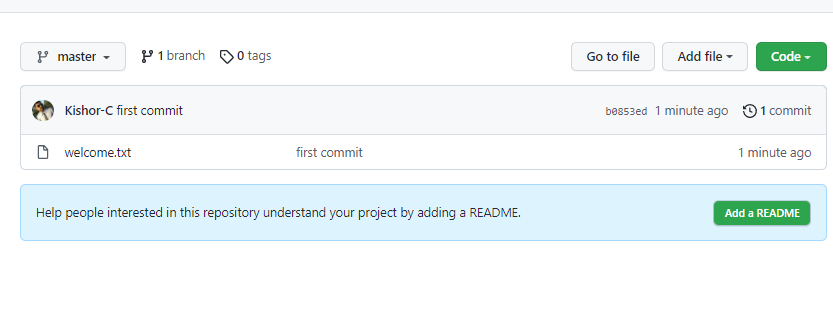
Command: git checkout branch\_name

Note: You may get this fatal error, when there are not commits in the master, i.e., if its empty repository you can’t create branch

To avoid this in the master branch create 1st commit

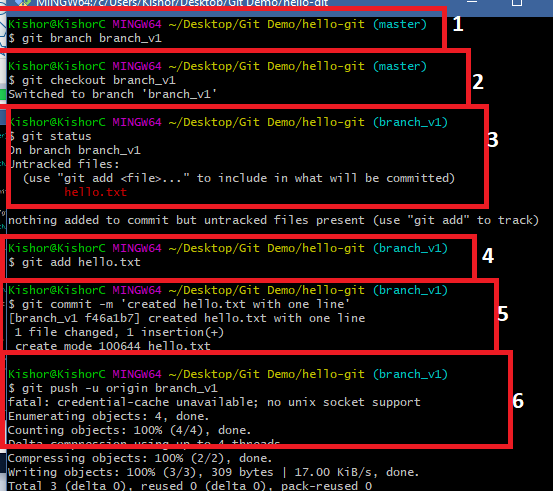


In the Remote you can see that first commit

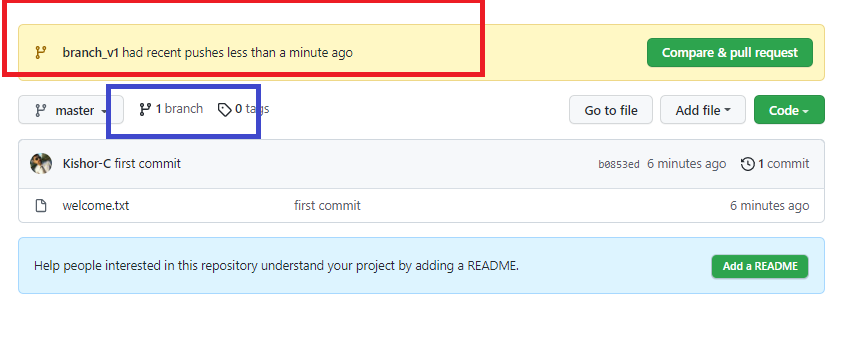


Note: It doesn’t ask you to create pull request, as we pushed master branch

1. Now we can create a custom branch and push that branch to the Remote repository

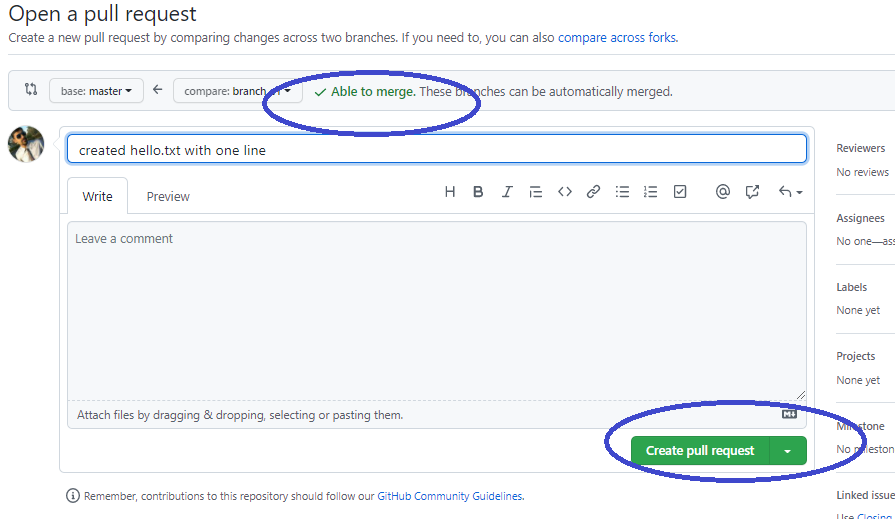


1. Now you can see a pull request in the Remote repository

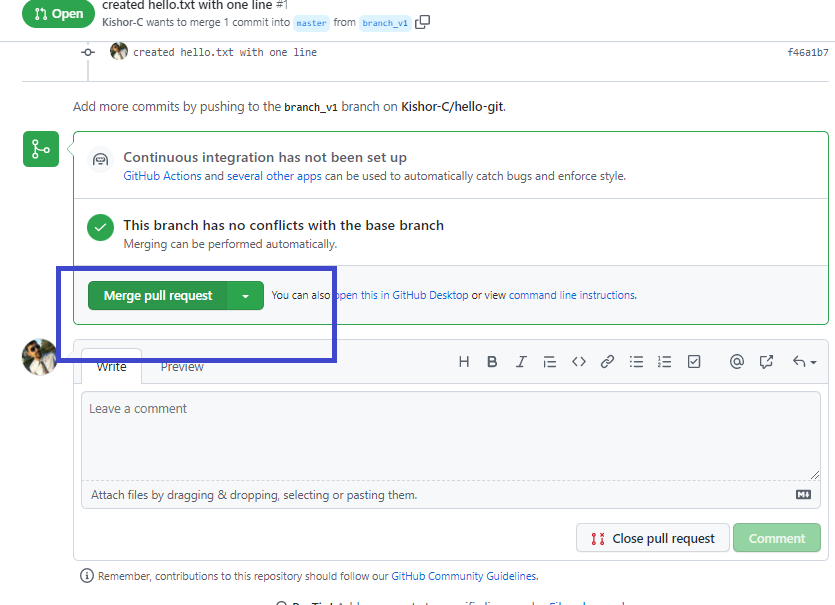


Note: Sometimes you may not see the recent push branch with Compare & Pull request, in that case you click on the branch highlighted in blue color

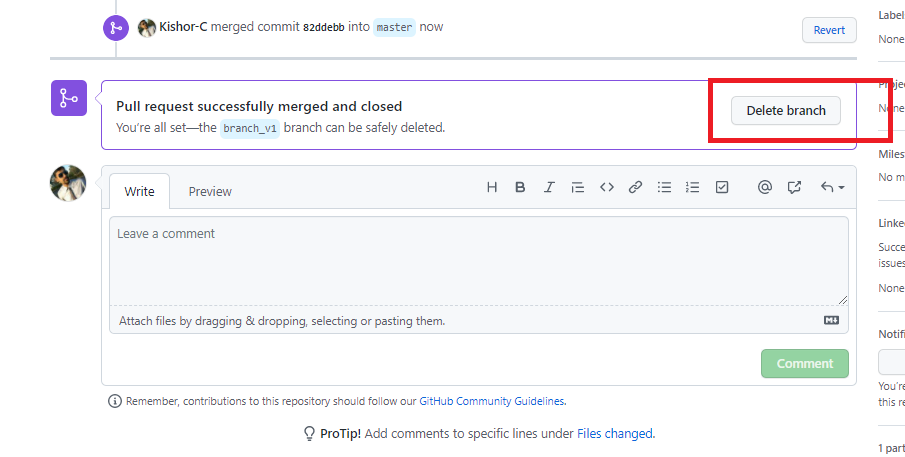
1. You can click on compare & pull request to update the remote or close pull request when there’s a conflict, in either of the case you must delete custom branch in remote repository.



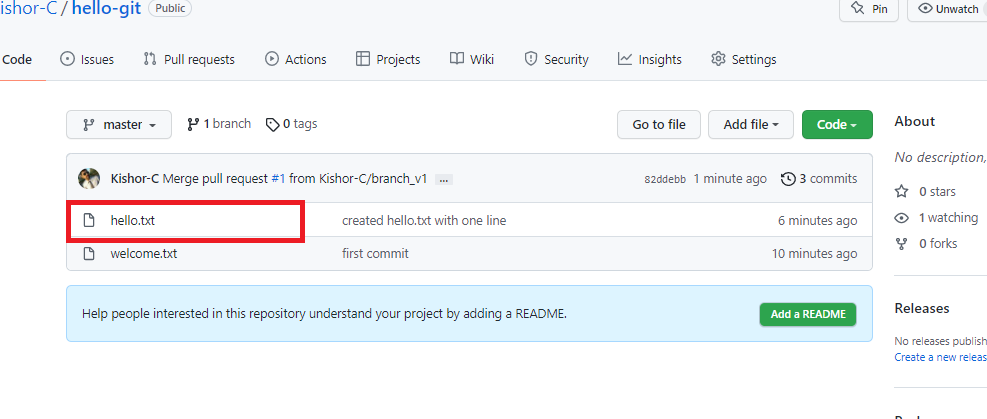
Merge the branch & delete



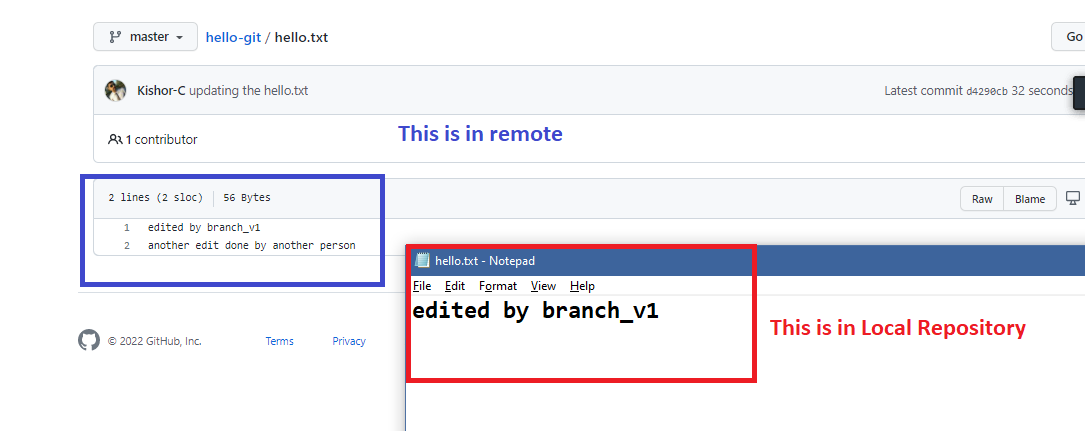
Delete branch



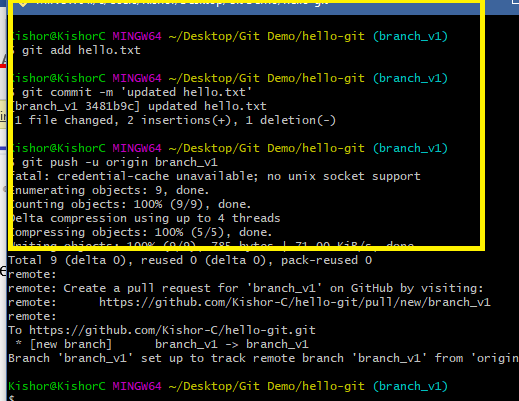
1. You can see the changes in the remote



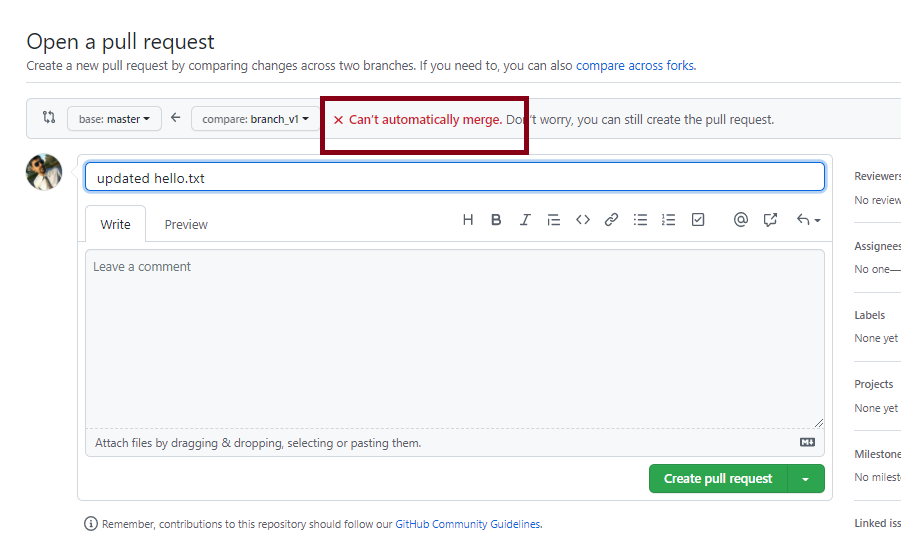
1. If you get a conflict then the local repository must pull the changes of master to its master branch, suppose for the below changes we get conflict when we try to update



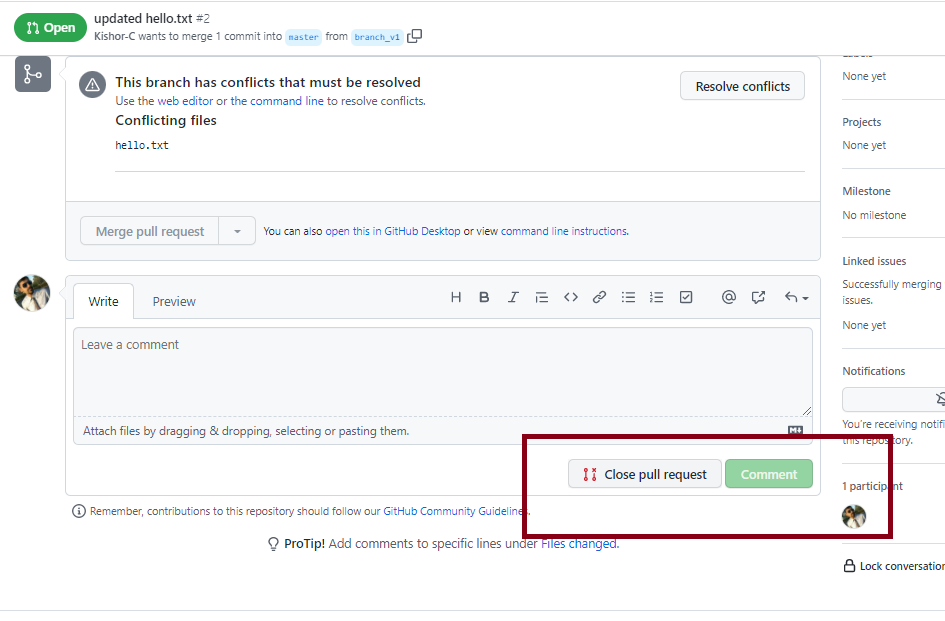
1. Add some content to the hello.txt & try to push the branch



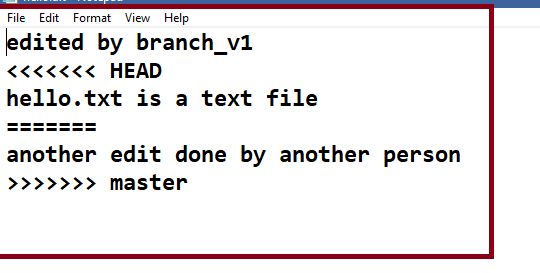
1. In the remote we see a pull request, but it can’t merge as remote master & the custom branch has conflict



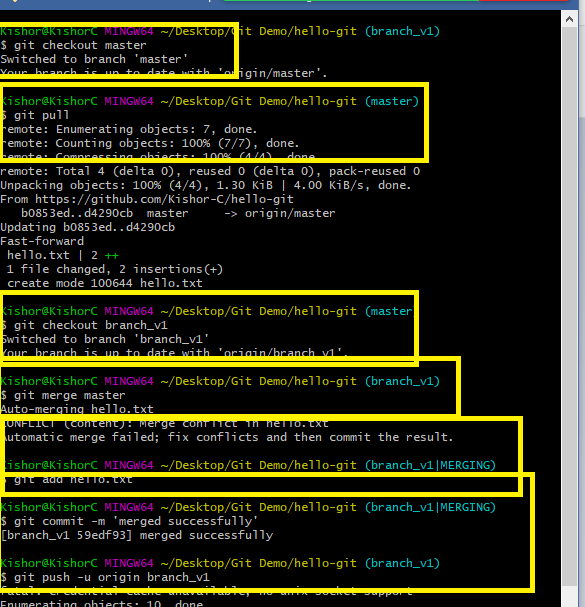
1. Close the pull request & delete the branch in the remote



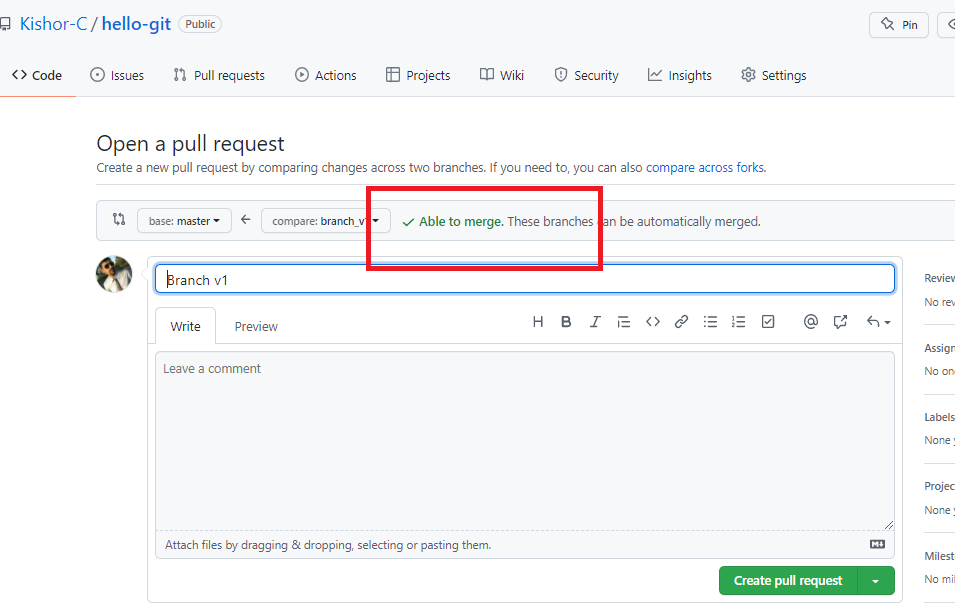
1. Local master must pull the remote master, then the custom branch should merge the local master and resolve the conflict
   1. Checkout to master: git checkout master
   2. Pull the remote master: git pull
   3. Checkout to custom branch: git checkout branch\_v1
   4. Merge the master with custom branch: git merge
   5. You will get conflict, edit that file, add & commit & push, you will the conflict file as below:



* 1. All the steps performed.



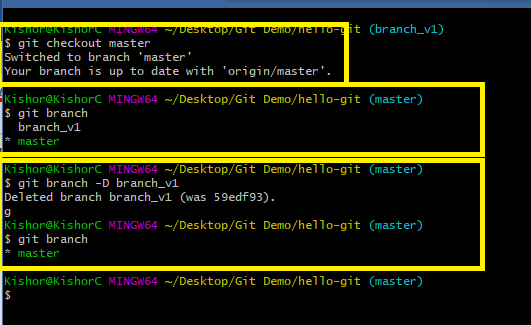
1. Now you can see the pull request in the remote

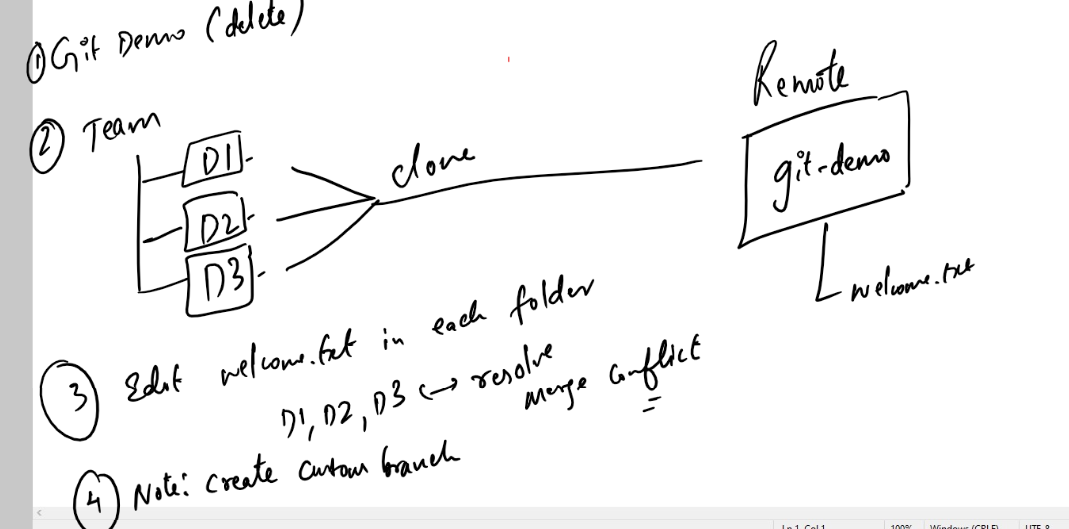


1. Create pull request & merge it and delete the branch in the remote, if required delete it in the Local also

Command: git branch -D branch\_name

Note: Checkout to different branch to delete



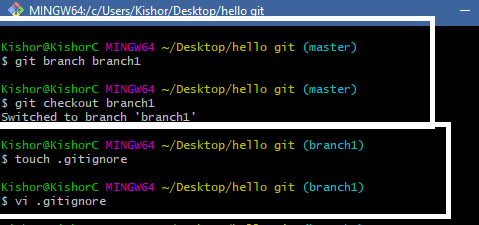


Summary:

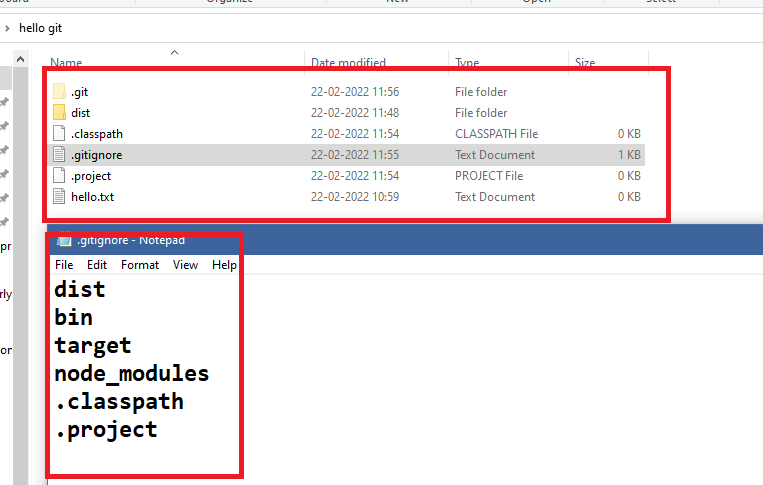
1. There are 2 ways you can create the local repository
2. using git clone remote-url
3. using git init & add remote url in the local repository
4. Important commands of Git
   1. git clone: creates a copy of remote repository in the local machine
   2. git init: creates a local repository
   3. git log: shows all the commits done with date, time, message & author information
   4. git status: shows all the tracked & untracked updates
   5. git add: adds the update to the staging area
   6. git commit: commits the updates in the staging area
   7. git push: pushes the branch to the remote from the local repo
   8. git pull: pulls the branch from the remote to the local repo
   9. git merge: merges the branch with another checked out branch

.gitignore:

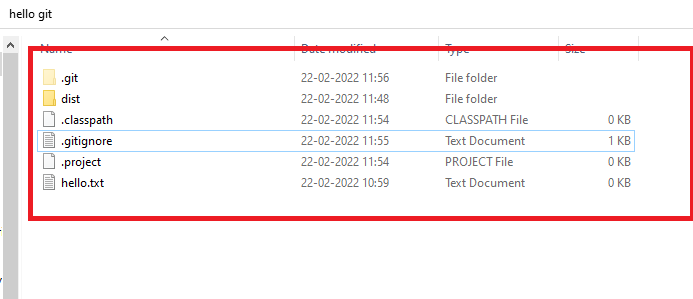
It is a file used by git to ignore the files/folders to be tracked.



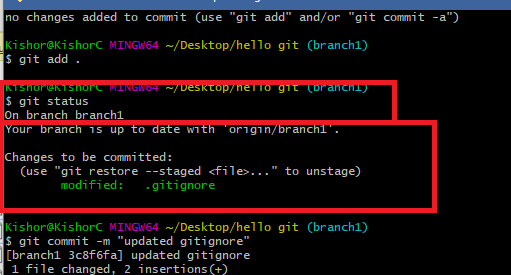
Add the content to .gitignore as mentioned below



Create some folders and files like dist, .classpath, .project and so on in the local repository

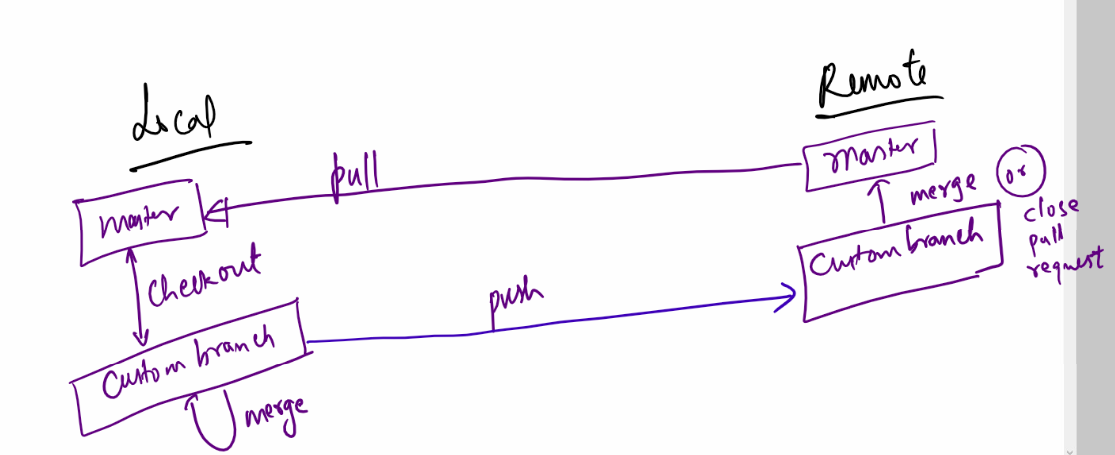


If you use git status you can see dist, .project & .classpath wouldn’t be tracked because its mentioned in gitignore



This .gitignore will be pushed to the remote, so that other developers would also get this file, it means everyone will have .gitignore file in their local machine, but the list of files/folders mentioned in the .gitignore wouldn’t be tracked at all from any machine.

Git Cycle



Git Organization:

It allows group of people to collaborate their work, only the members within group can update the repository inside the organization.

Steps:

1. Create the organization
2. Name the organization & fill all the details
3. Add the members by entering their username of git
4. Select the required options what you are going to with the organization, like manage code, collaborate work, team-size and so on
5. Change the role of members to owner.
6. Create one repository which all these members can access.

Activity

1. Create a team of 5 members (Trainer will do it)
2. Choose a team leader (Trainer will do it)
3. Team leader must create organization
4. Team leader will add members & members will accept the invitation
5. Team leader will change each members role
6. Team leader will create a repository
7. Team leader will create a file called hello.txt & pushes the master to the remote
8. Everyone in the team including the team leader edits hello.txt with their names and pushes their custom branch
9. Team leader will merge their custom branch or close the pull request depending on the scenario
10. Someone in the team will create another file welcome.txt and each members should pull that file and write their names again in welcome.txt and push that update to the remote, team leader will again merge or close the pull request

Team members

|  |  |
| --- | --- |
| Team Leader | Team Members |
|  | Prathamesh |
|  | Abdul |
| Apurva | Abhishek |
|  | Apurva |
|  | Archit |
|  |  |
|  | Azad |
| Bhalchandra | Bhalchandra |
|  | Deepak |
|  | Gopal |
|  | Jayesh |
|  |  |
|  | Jayesh Saindane |
|  | Jitesh |
| Jitesh | Jui |
|  | Nikhil Pawar |
|  | Nitesh |
|  |  |
|  | Pankaj |
|  | Pravi |
| Pankaj | Rahul Khenat |
|  | Rohit |
|  | Shaqib |
|  |  |
|  | Shubham |
| Nikil Hatwar | Sourabh |
|  | Sumit |
|  | Vikas |
|  | Nikhil Hatwar |