Git-Day-3

19 November 2023 11:52

==
Branch
==

- Normally when you create a repository by default master/main branch will be get created inside the repository.

- The master branch will contains stable & production ready code for deployments. We shouldn't tamper this code at all.

- Suppose if we allow developers for directly pushing the code on master branch there will be a chance of disturb the production ready/stable code on main branch.

- so if the code on main branch is not stable then the deployments also will not work correctly.

- To overcome such kind of issues, Whenever developers want to work on new feature Git allows to create a new branch & on this new branch developers can parallel work without disturbing the main line branch code. Once the feature development completed feature branch will merged to master.

- Now let's work on these branching practically,
  - Clone pets clinic repo

  - By default present we are in master branch & its having 3 files.

  - Create new branch called **feature-1**
    git branch feature-1

  - Switch to new branch
    Git checkout feature-1

  - Check on which branch currently you are in
    git branch

  - List all the branches present in local repository & remote repository
    git branch --all

  - Let's add a new file & create commit, now feature branch having 4-commits & master having 3-commits

  - Let's merge the feature branch into master
    git checkout master
    git merge feature

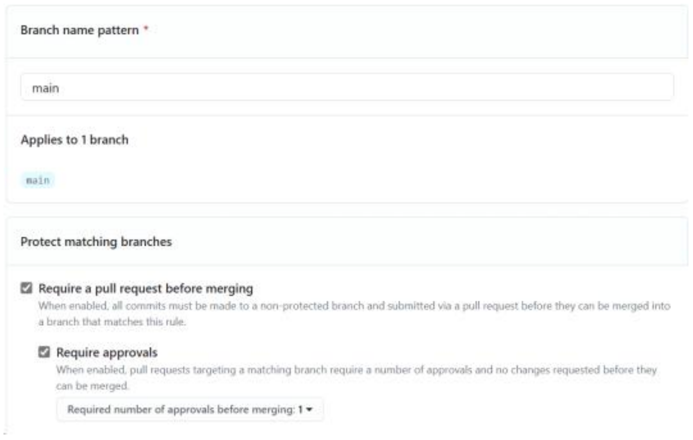
  - Delete the feature branch
    git branch -D feature-1

- **Did you notice one issue here? while merging the code from feature branch into master branch**
  - Also **developers are directly not allowed to push the code from local repository master branch to remote repository master branch**.
  - **Code is not reviewed by other developers/lead** since the changes are present in Local machine.

- Now we will see how to overcome above two issues.

- **How to protect master branch on GitHub from direct code push from developers.**
  - To protect main branch from direct push, we have to setup protection policies
    Repo --> Settings --> Branches

  - Require a pull request before merging --> Users can't do direct push & only pull request is the way to merge code to master.
    Require approvals --> The code must be reviewed & approved by other developers.
    Do not allow bypassing the above settings --> No one have possibility to skip rules like above
    Lock --> Whenever code freeze is there, we can lock the branch. So users can't merge pull requests.



Branch name pattern *

main

Applies to 1 branch

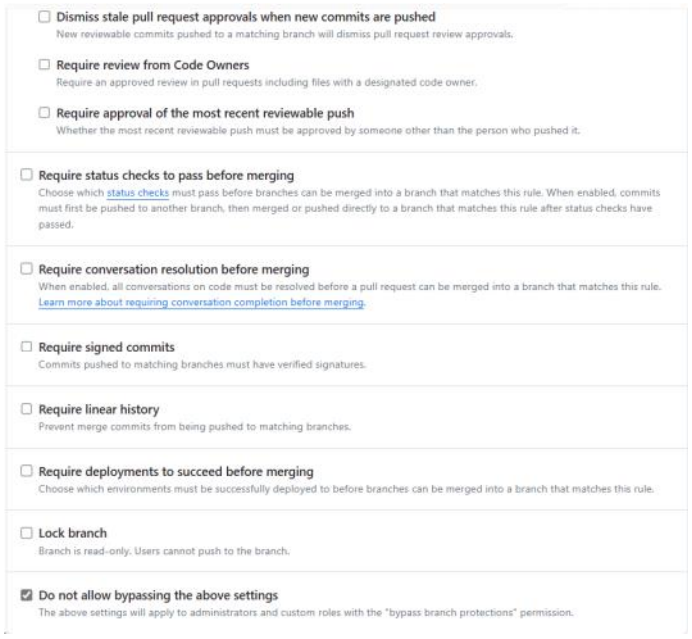
main

Protect matching branches

☑ Require a pull request before merging
When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☑ Require approvals
When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.

Required number of approvals before merging: 1



- ☐ **Dismiss stale pull request approvals when new commits are pushed**
  New reviewable commits pushed to a matching branch will dismiss pull request review approvals.
- ☐ **Require review from Code Owners**
  Require an approved review in pull requests including files with a designated code owner.
- ☐ **Require approval of the most recent reviewable push**
  Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

- ☐ **Require status checks to pass before merging**
  Choose which status checks must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.
- ☐ **Require conversation resolution before merging**
  When enabled, all conversations on code must be resolved before a pull request can be merged into a branch that matches this rule. Learn more about requiring conversation completion before merging.
- ☐ **Require signed commits**
  Commits pushed to matching branches must have verified signatures.
- ☐ **Require linear history**
  Prevent merge commits from being pushed to matching branches.
- ☐ **Require deployments to succeed before merging**
  Choose which environments must be successfully deployed to before branches can be merged into a branch that matches this rule.
- ☐ **Lock branch**
  Branch is read-only. Users cannot push to the branch.
- ✓ **Do not allow bypassing the above settings**
  The above settings will apply to administrators and custom roles with the "bypass branch protections" permission.

► Now next fix the other issue by raising pull request

===

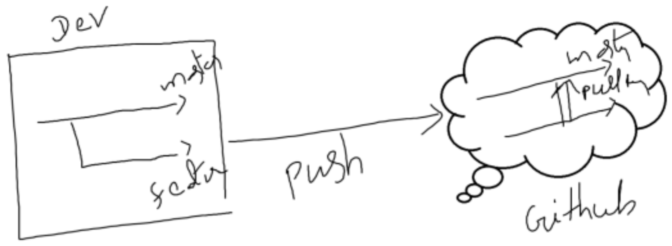
GIT PULL REQUEST

===

► Git pull request will help us show the difference between personal branch & mainline branch. This will helps the peer developers to review changes.

► Also provide feasibility to suggest comment on the code developed if any modifications needed.

► Now let's assume



- This is a developer machine & this is a GitHub repo with master branch
- If developer wants to work on new feature, he will create a branch called feature-1 in local repository.
- Once feature development completed, developer will push the feature branch into GitHub repo.
- Now In order to merge the feature branch into master in GitHub, pull request need to be raised.
- Once the pull request raised it can be reviewed by other developers & can be merged into master.

► Let's do this practically now.

- **Clone repo**
  git clone https://github.com/chaitanyaredd/pull-request-demo.git

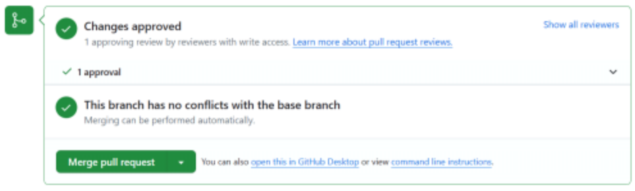
- **Create a feature branch** in local repository based out of master branch
  git checkout -b feature1

- **Add new file & commit in feature1 branch**
  echo "This is file1" > file1.out
  git add .
  git commit -m "Adding first file"

- Push the feature1 branch to GitHub
  git push origin feature1

- **Raise PR** in GitHub
  - Review the PR by another developer(student).
  - Make sure the new developer must have collaborator access(settings --> Collaborator).

- Merge the Pull request to master branch.



Changes approved

1 approving review by reviewers with write access. Learn more about pull request reviews.

Show all reviewers

✓ 1 approval

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also open this in GitHub Desktop or view command line instructions.

===

Git Fetch

===

- Before we discuss on Git Fetch command let's discuss what are all the different branches we have in GIT.

- In Git mainly we have three kind of branches
  - Local branch
  - Remote branch &
  - Remote tracking branch

- **What is local branch?**
  The branch that we have created in the **local repository** & local repository presented in our laptop

- **What is remote branch?**
  The branch that we created in the **GitHub** that is called as a remote branch.

- What is remote tracking branch?
  - **It's local copy of the remote branch.**
  - How can we see the remote tracking branches?
    git branch -r

- Git **Fetch** command will **download the changes from the remote branch** and **updates its corresponding remote tracking branch**.

- In this case the changes are downloaded from remote repository to local repository but not merged it with the local repository branch.

- **Now let's see scenario practically**
  - Create a repository in a GitHub with three commits in master branch and clone that repository into the local machine.

  - Now we have two repositories one is local repository and another one is remote repository in GitHub.

  - Now list the branches, git branch -a
    Here we can see the local branch as master &
    remote tracking branch as origin/master &
    remote branch as master that is present in the GitHub

  - Now let's make some changes in the master branch GitHub.

  - Next step if you run git fetch command the changes whatever present in master branch of GitHub will download it to them remote tracking branch in the local repository.

  - Can we see the changes that we have downloaded?
    No we cannot see those changes that we have downloaded into local repository from master local branch.
    cat filename
    See no changes are coming.

  - Normally we should not edit the remote tracking branches.

  - To make visible the changes that are presented in the remote tracking branches to your local branch we have to run git merge command
    git merge origin/master.

===

GIT PULL

===

- Git pull command is combination of **fetch** as well as the **merge** command.

- It means that whenever developer runs the fetch command changes are downloaded from remote branch(master) to remote tracking branch(origin/master) & then after remote tracking branch will get merged to local branch.

- Now let's see this scenario practical
  - Update the remote repository with the few commits.

  - Run git pull command on master branch
    git pull origin master
    cat filename

    See we can see the changes.