Contents

[**Commands** 1](#_Toc62669264)

[**BRANCHing** 2](#_Toc62669265)

[Rename a Branch 2](#_Toc62669266)

[Delete a Branch 2](#_Toc62669267)

[Create Branch from GitHub 3](#_Toc62669268)

[Clone the Branch into Local repository 4](#_Toc62669269)

[**Tags** 5](#_Toc62669270)

[**Stash** 6](#_Toc62669271)

[**Git Revert** 6](#_Toc62669272)

# **Commands**

* **git clone <URL> -> To clone the Project from Remote to Local Repository**
* **git log –> git history**
* **git add . 🡪 To add all the newly created or modified files into Staging area in local repository**
* **git add <filename> 🡪 To add the single file name that is created or modified into Staging area in local repository**
* **git commit -m “Test Message” 🡪 To add the modified files from staging area to local repository**
* **You have to Identify yourself to Add files to Remote Repository by executing below commands**
  + **git config --global user.name "<Github Username>"**
  + **git config --global user.email” <Github User email>”**
* **git push 🡪 To Push the changes from local repository to remote repository**
* **git init 🡪 git init** creates an empty Git repository or re-initializes an existing one. It basically creates a**.git** directory with sub directories and template files. Running a **git init** in an existing repository will not overwrite things that are already there. It rather picks up the newly added templates.
  + **git remote –> to see remote repository attached to it**
  + **git remote -v**
  + **git remote add origin** <<https://github.com/skatta3/TestRole.git>> --> To map local repository to remote repository
  + **git push origin master**

Here we need to create a empty repository in github without initialization of README file

* **git status** -> The git status command lists all the modified files which are ready to be added to the local or remote repository.
* **git pull -> To get latest version from Remote Repository**
  + **it auto merges the code from remote Repository into Local repository as long as there are no MERGE Conflicts**
  + **if there are any MERGE Conflicts it won’t auto merge, you have to resolve the Conflicts manually**
* **git reset --hard – To move the committed changes to the history**
* **git mv <Source> <Dest> 🡪 To Rename the file**

# **BRANCHing**

* **git checkout -b uat - Creation of new Branch and switching into it**
  + **Equivalent to**
    - **git branch uat**
    - **git checkout uat**
* **git push --set-upstream origin <NEWBRANCH>** -- To Map the newly created branch in local to Remote

## Rename a Branch

To rename a branch, run the command:

**git branch -m OLD-BRANCH-NAME** NEW**-BRANCH-NAME**

*# Alternative*

**git branch --move OLD-BRANCH-NAME** NEW**-BRANCH-NAME**

## Delete a Branch

Git won’t let you delete a branch that you’re currently on. You first need to checkout a different branch, then run the command:

**git branch -d <BRANCH-TO-**DELETE>

*# Alternative:*

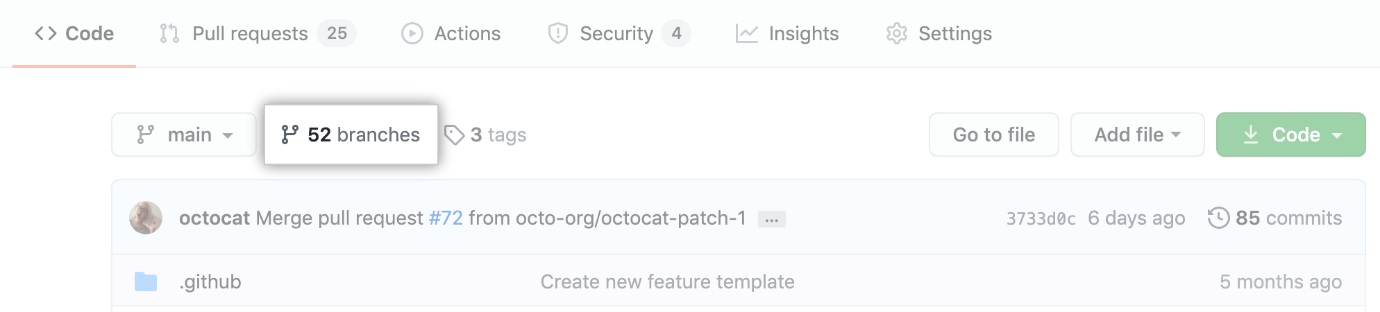
**git branch --delete <BRANCH-TO-DELETE>**

**push the changes to remote Branch**

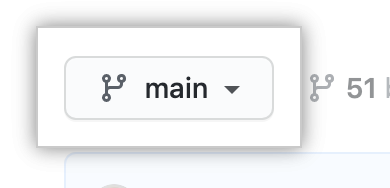
**git push -d origin <Branch-Name>**

## Create Branch from GitHub

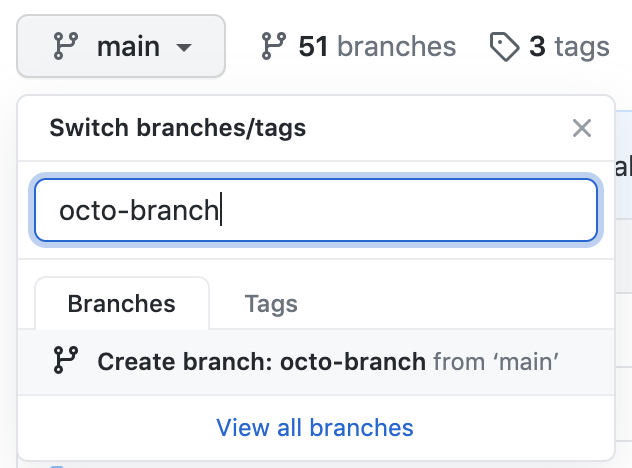
1. On GitHub, navigate to the main page of the repository.
2. Optionally, if you want to create your new branch from a branch other than the default branch for the repository, click  ***NUMBER* branches** then choose another branch:



1. Click the branch selector menu.



1. Type a unique name for your new branch, then select **Create branch**.



## Clone the Branch into Local repository

**git clone -b <BranchName> <URL>**

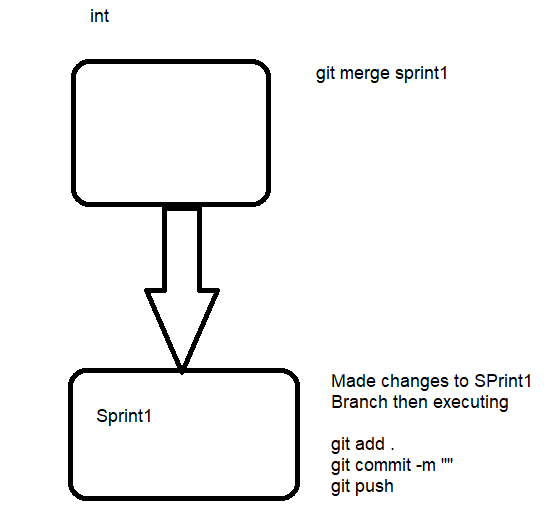
**git diff TESTBRANCH moche2branching**

**Git push**

* **pushes to default branch which is master branch**
* **Git push origin uat – to push to uat branch**

**From int branch**

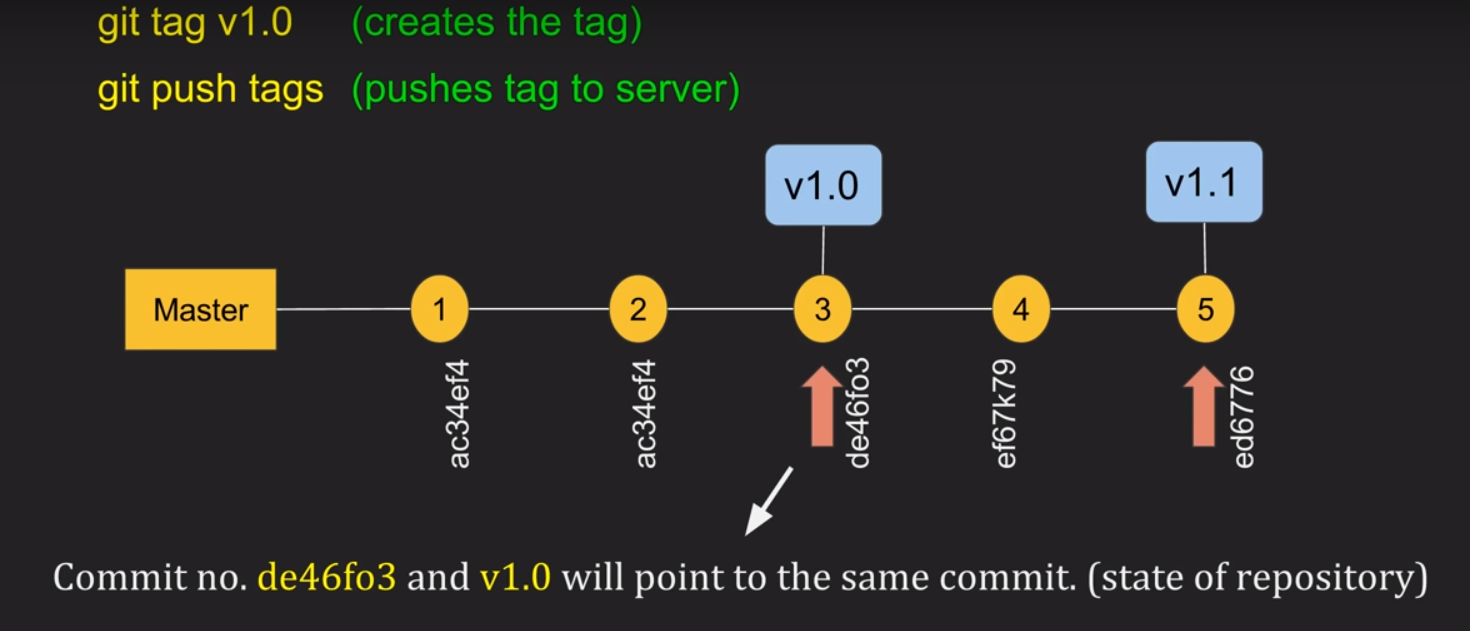
**Git merge sprint1 (Sprint1 Branch created from int branch and added some changes in Sprint1 branch)**



1. **Created Release Branch from Main Branch**
2. **Made changes to Release Branch and pushed the changes to Release Branch**
3. **If we need to merge these changes to Main Branch then switch to Main branch by executing git checkout main**
4. **Then execute git merge Release Branch Name**
5. **Then execute git push command**

# **Tags**

* **git tag <tagname> -- To Create the Tags**
* **git tag or git show <tagname> -- To list the available tags**
* **git push origin <tagname> or git push –tags -- Push the tags to Remote Repository**
* **git tag –d <tagname> OR git push origin –d <tagname>**
* **git checkout –b <branchname> <tagname> -- Create the Branch from the tag**
* **git tag <tagname> <commitID> -- Create the tag from the commit ID**

****

**Tagging is nothing but labelling**

**On this Branch Given Versioning to customer**

**Tagging will be done to commit**

**Git log –oneline -- Shows the logging one line per commit**

**Git log --stat – Display statistics**

**Git log --oneline --graph**

**Git log --graph**

# **Stash**

* **git stash save <message>**
* **git stash list**
* **git stash show –p <stash-id>**
* **git stash drop <stash-id>**
* **git stash pop – Take the code from stash and apply to current repository**

**git stash – to record the state of current working directory when you want to work on other priority task**

**It will present in your local repo only**

**git stash apply – we will get our changes back**

**git stash apply <ID> if you have more than one stash**

**After applied it will not remove stash**

**git stash pop 🡪 it will apply and remove from stash**

# **Git Revert**

**Undo uncommitted Changes**

* **git checkout -- <filename> 🡪 undo single file**
* **git checkout -- . 🡪 undo all files**

**Undo Committed Changes**

* **git revert <commit ID> 🡪 Change is committed**
* **git push**

**Resetting Committed Changes**

* **git reset --hard <Commit ID> 🡪 Remove all the changes after this commit ID**

**Git Stash**

**Stash**

**We deployed Banking application into Production on May 1st**

**You are the Developer responsible for Transactions Module**

**On May 5th You client has given some enhancements in Transactions Module (Task1)**

**You started working on Task 1 from May 6th**

**on May 8th We found a critical defect in Prod in Transactions Module**

**The defect also assigned to you (Task 2)**

**When you stash it moves the incomplete changes/Uncommitted changes into temporary location in your Local Repo and makes your branch clean**

* **git config –list**