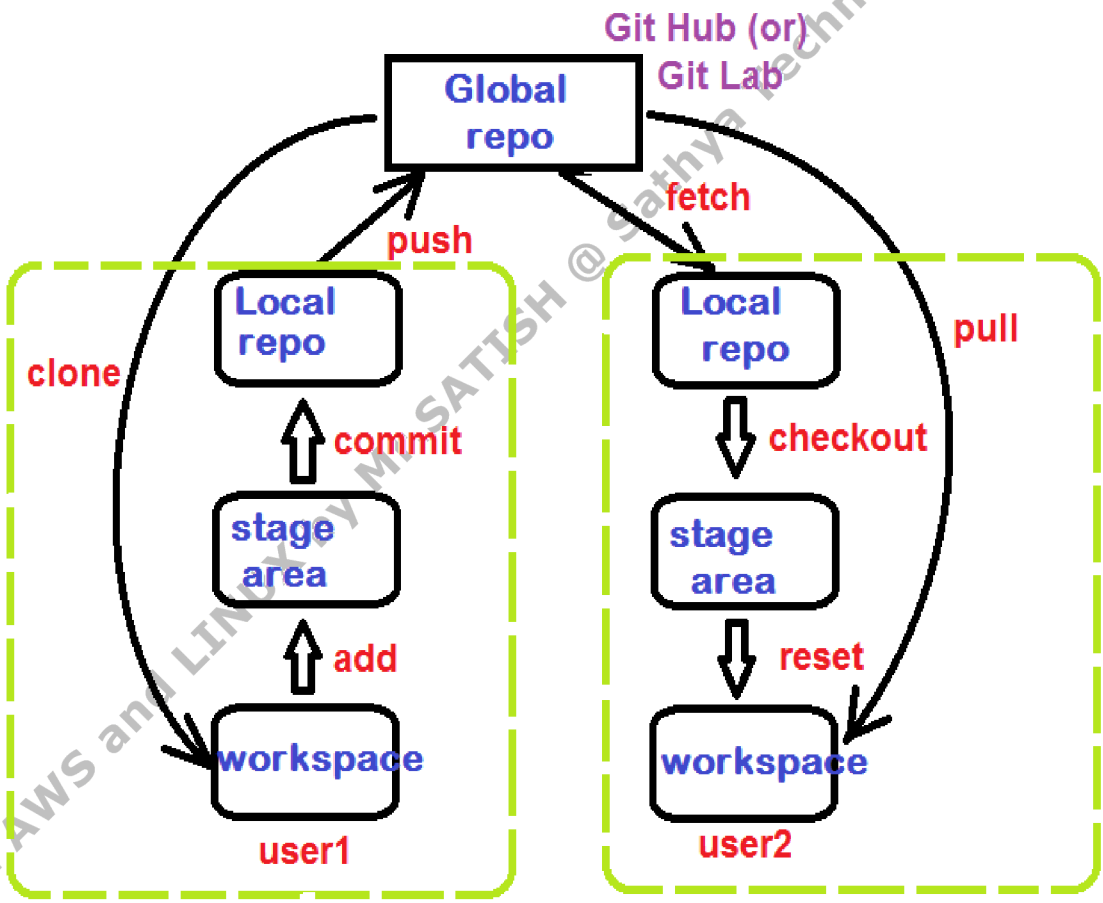
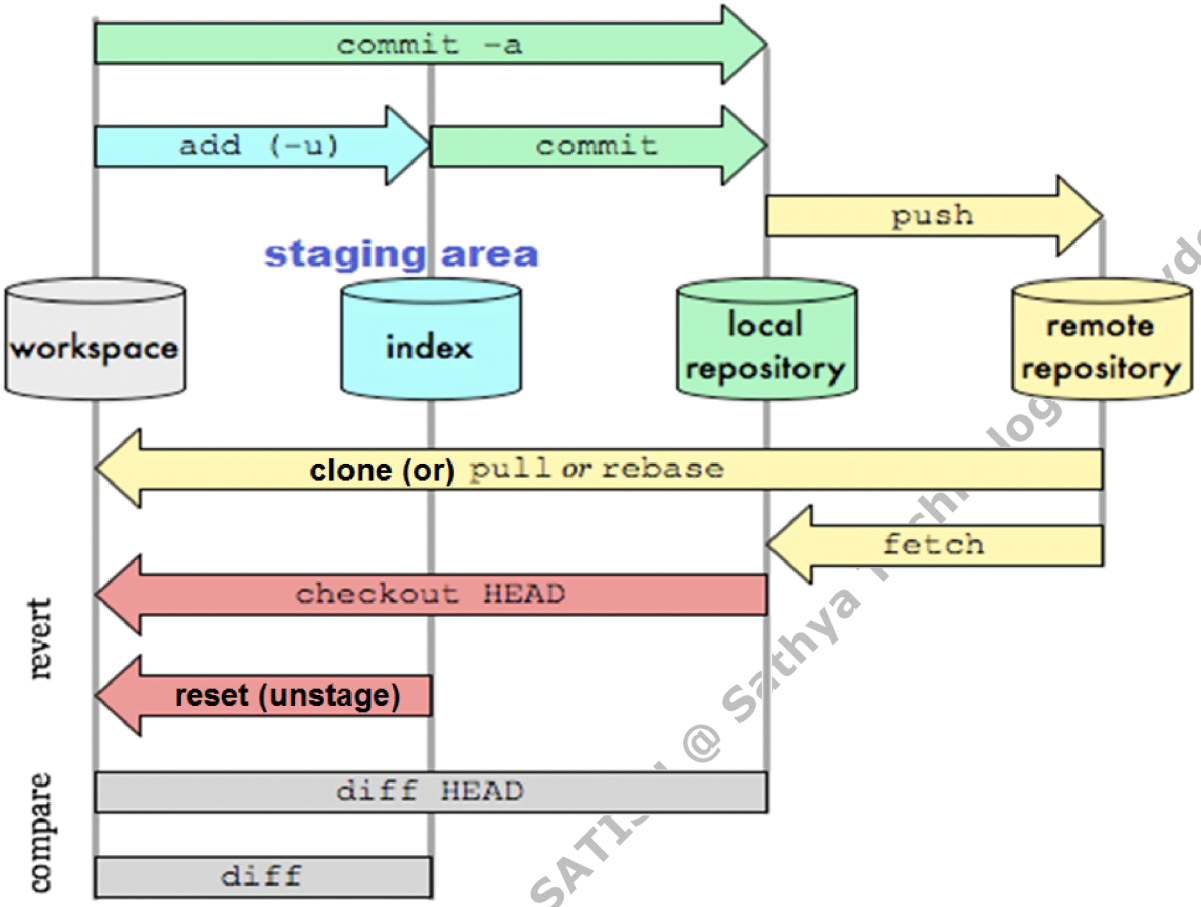
Introduction

- Git is an open source, distributed version control system (VCS) designed to handle everything from small to very large projects with speed and efficiency.
- Git is a distributed revision control and source code management system with an emphasis on speed.



What about GIT ?

- Git was initially designed and developed by Linus Torvalds for Linux kernel development. Git is a free software distributed under the terms of the GNU General Public License version 2.
- It's commonly used for source code management (SCM)

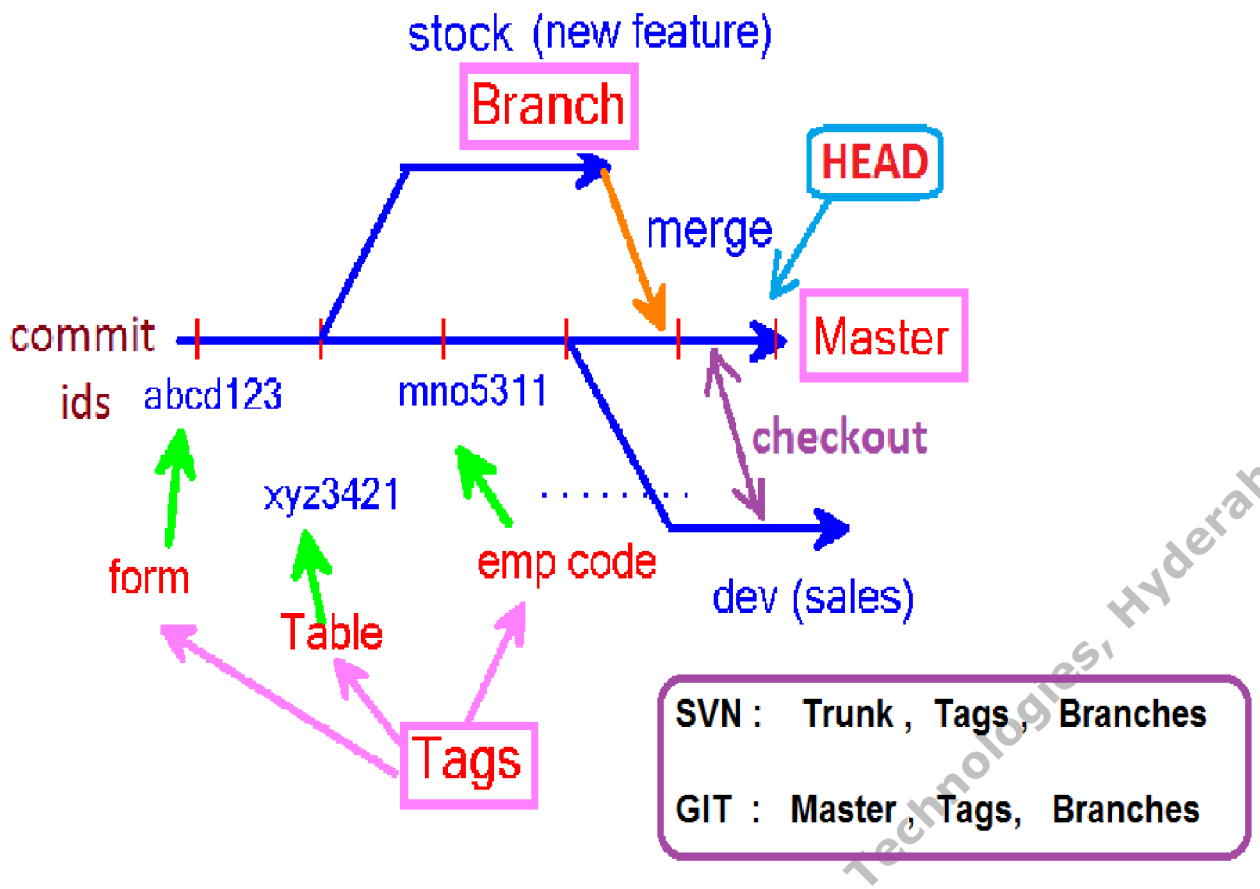


GIT Terminology

Repository ("Repo") :Git repository as a directory that stores all the files, folders, and content needed for your project.

Branch : A version of the repository that diverges from the main working project. Branches can be a new version of a repositor.

Clone : A clone is a copy of a repository or the action of copying a repository.

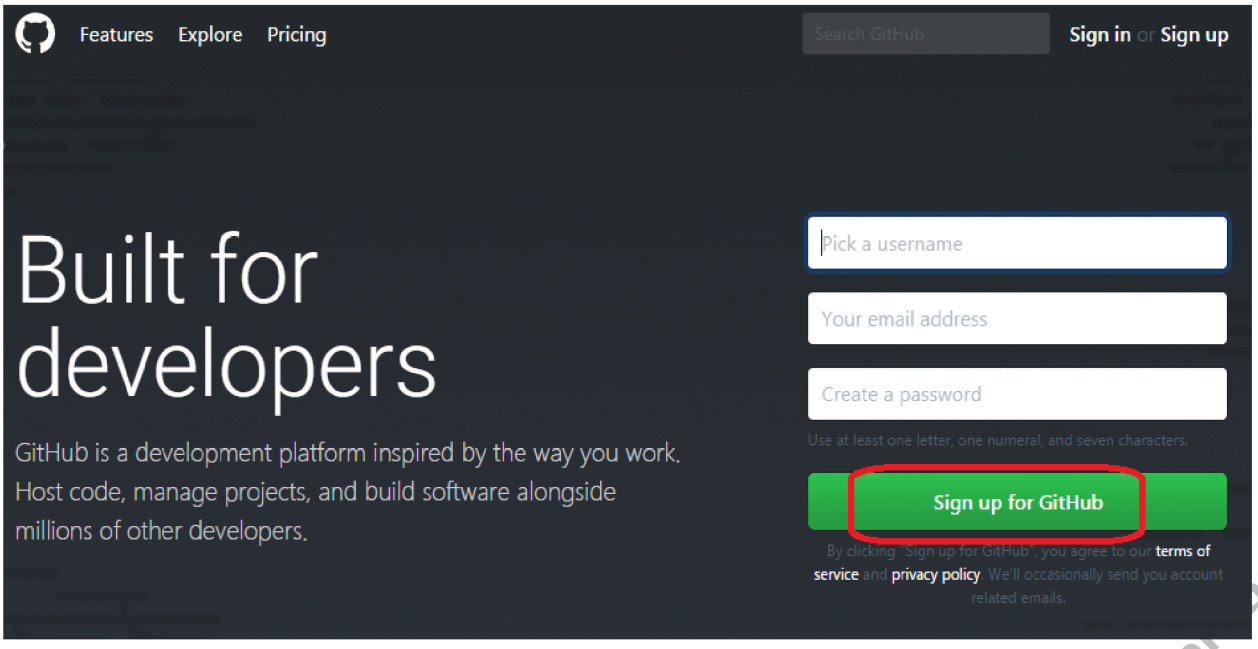


Master : The primary branch of all repositories. All committed and accepted changes should be on the master branch. You can work directly from the master branch, or create other branches.

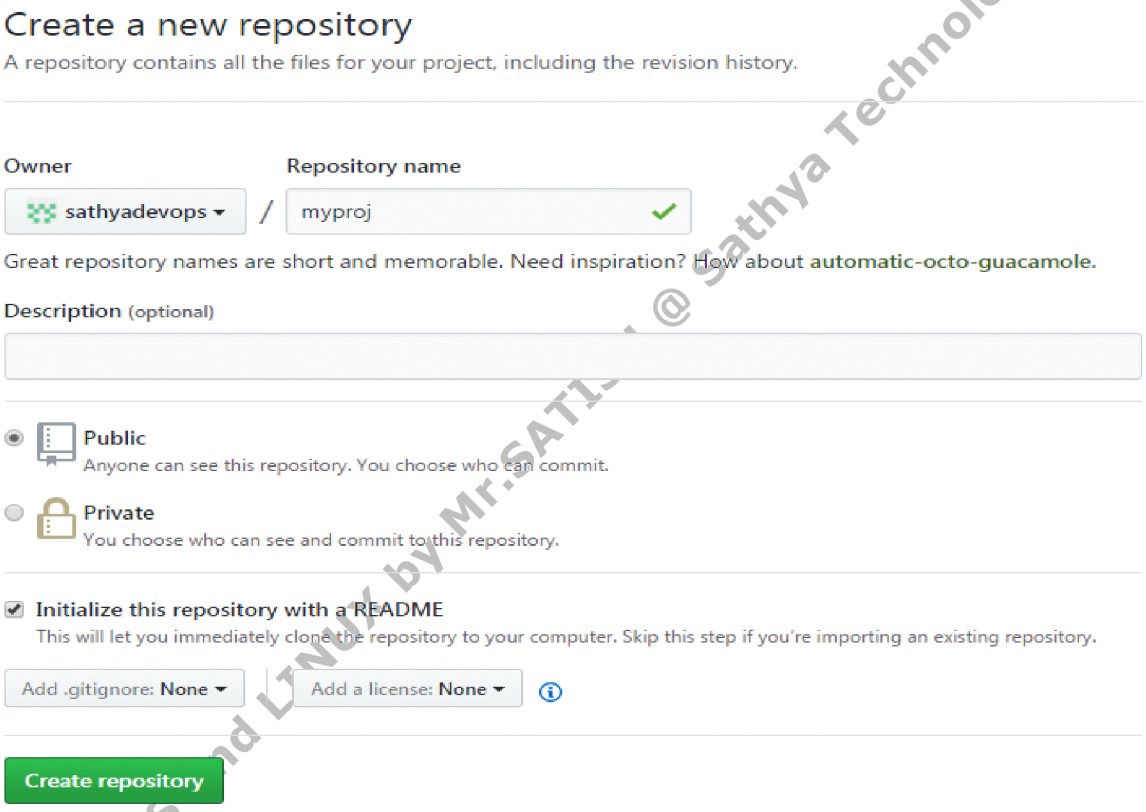
Checkout: The git checkout command is used to switch branches in a repository.

Merge : Taking the changes from one branch and adding them into another (traditionally master) branch.

Create Account in GitHub



Create Repository



Install Git on Ubuntu 20.04

Step 1: Installation

```
#apt-get update
#apt-get install git-core –y
#git --version
```

Step 2: Configuration

```
#git config --global user.name sathyadevops
#git config --global user.email sathyadevops1@gmail.com
#cat .gitconfig (or) #git config --list
```

Step 3: Create GIT repository

```
#mkdir /repos
#cd /repos
#git init
#ls –a
#git clone https://github.com/sathyadevops/myproj.git
```

Step 4: Working with Git Repository

```
#echo "Welcome to Git" >> README.md
#git status
```

to add a file to cache (staging Area)

```
#git add README.md
#git status
```

to move a file from Staging Area to Local Repo

#git commit -m "initial commit"

to Add amd Commit a file at a time

#git commit -a -m “initial commit“

to push the code to Central Repo(master)

#git push -u origin master

To changed files in your working repository

#git status -s

To show all git commits

#git log

#git log –p

#git log --since=12-03-2017 --until=13-03-2017

#git log --oneline

To push Code to GitHub by using key:

Step1: Generate key

#ssh-keygen

Step2: add public key to github project

github → project → settings → deploy keys

Step3: set remote github url

Syntax:

git remote set-url origin git@github.com:<Username>/<Project>.git

Ex:

#git remote set-url origin git@github.com:sathyadevops/newproj.git

To made changes to tracked files

```
#git diff
#git log
#git log -1
#git diff  57af6s43d..9wg5c2ys3
```

To list all branches

```
#git branch
```

to work with branches:

```
#git branch branch1
#git checkout branch1
#git branch
#vi index
        new line from branch
#git commit –am "new line from branch"
#git push -u origin branch1
```

check in browser → github

to merge the branch code into master

```
#git checkout master
#git merge branch1
#cat index.html
#git push -u origin master
```

to delete a Branch:

```
#git branch  -d  branch1
```

to delete a Branch without merging the Data:

#git branch -D branch1

#git push origin --delete br1

(to delete remote branch)

Git - Review Changes

git diff

git log

git show c0f455906befd100192848233fbb896d081e2284

Git – Remote Server

#git remote -v

#git checkout -- . (to revert all the changes)

Install git package:

sudo su -

apt-get update

apt-get install git-core -y

git --version

git version 2.17.1

git init --> to create an Empty git repo

Create a Global repo:

visit ---> www.github.com --> create a New Account

and do create a New Git Repo. and do Git clone

https://github.com/sathyadevops/dev830am

```
# git clone https://github.com/sathyadevops/dev830am.git
Cloning into 'dev830am'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

Method-1: Generate Token:

https://github.com/sathyadevops/ --> Settings --> Developer settings

select "Personal access tokens (classic)" --> create a New token

TOKEN: ghp_lgduMvXFoO9BAvNkaG6ixxx....

```
 # cd dev830am
 # vi demo.java
class Demo
{
  public static void main(...)
  {
   s.o.print(" Hello world ");
```

```
  }

}
```

```
# git add                          ---> to move code to stage area

# git commit  -m  "java code"    ---> to save code in local repo

# git push                         ---> to move code to Global repo
```

```
Username for 'https://github.com': sathyadevops

Password for 'https://sathyadevops@github.com':

Counting objects: 3, done.

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 344 bytes | 344.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/sathyadevops/dev830am.git

  3a4f7ed..11f86ff  main -> main
```

Method-2: using SSH-Keys:

```
# ssh-keygen  (to generate a Key-pair)

# cd ~/.ssh/

# ls

authorized_keys  id_rsa  id_rsa.pub  known_hosts

# cat id_rsa.pub  (copy public key)

    ssh-rsa AAAAB3NzaC1xxxx... root@ip-172-31-40-23
```

visit --> https://github.com/sathyadevops/dev830am

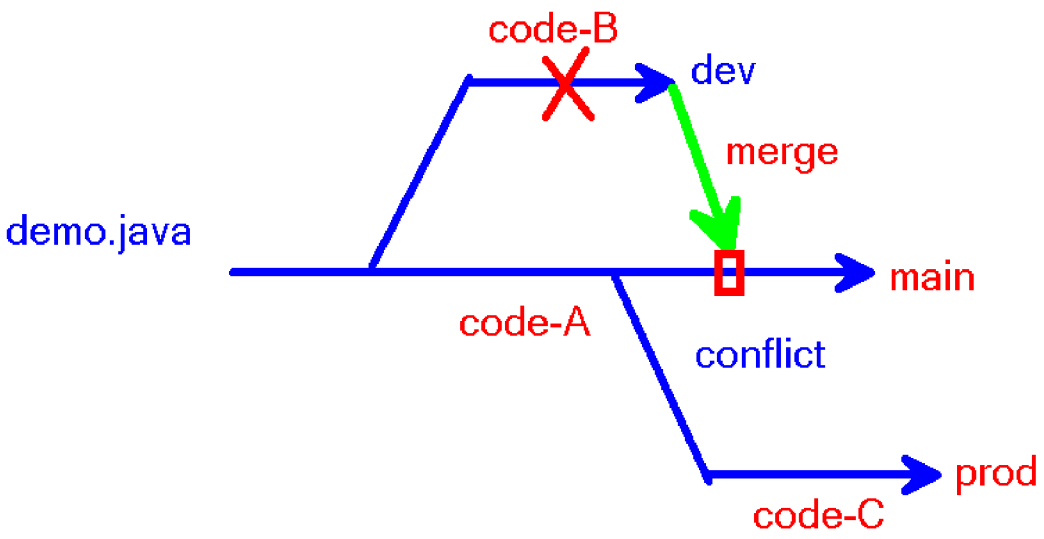
settings --> Deploy keys --> add key -->

Title: mykey

Key: paste key --> Allow write access --> add key

```
# git remote set-url origin git@github.com:sathyadevops/dev830am
# vi sample.c
void main()
{
  .... code ....
}

# git add sample.c
# git commit -m "c-code"
# git push

Counting objects: 3, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 339 bytes | 339.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:sathyadevops/dev830am
   11f86ff..2faec6f  main -> main
```

Git Merge Conflict :




```
# git branch
# git branch dev
# git checkout dev
# git branch
# vi demo.java
```

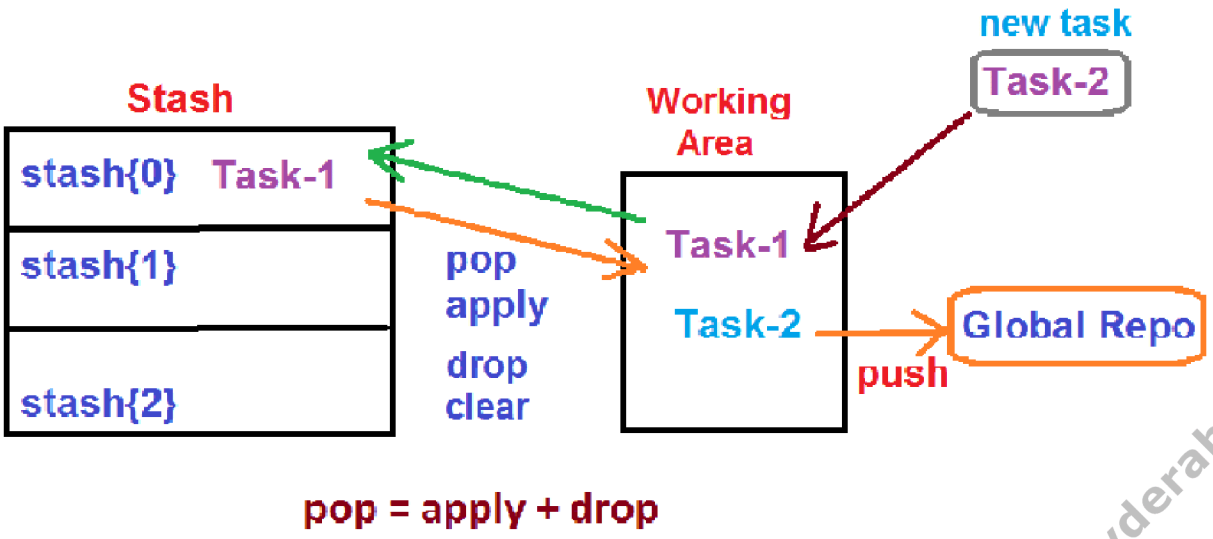
..... BRANCH-CODE .....

```
# git add demo.java
# git commit -m "branch code"
# git push origin dev
# git checkout main
# vi demo.java
```

..... MAIN-CODE .....

```
# git add demo.java

# git commit -m "main code"


# git merge dev      ----> it leads to code conflict

# vi demo.java

..... accept the code changes .....

# git commit -am "code merged"

# git push



# git branch -d dev                ---> to delete a local branch

# git push origin --delete dev   --> to delete a Global branch


# git branch prod

# git checkout prod

          (or)

# git checkout -b prod            ---> to create and switch to branch

# git branch

# git branch -D prod              ---> to delete branch forcefully
```

Git Stash



- git stash temporarily shelves (or *stashes*) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on.
- Stashing is a way to pause what you're currently working on and come back to it later.

#vi index.html

<h1> Hello World </h1>

<h2> New line is added </h2>

git diff

Stash your changes away with:

git stash (or)

git stash save "message"

git diff

cat index.html

<h1> Hello World </h1>

To List multiple layers of stashes

git stash list

git stash show

You're back to your original working state

git stash apply

git stash apply stash@{0}

git stash pop

cat index.html

<h1> Hello World </h1>

<h2> New line is added </h2>

We can manually delete stashes :

git stash drop stash@{1}

delete all of the stored stashes

git stash clear

To Move a file to another Dir :

#cd gitproj

#mkdir mydir

#git mv demo.c mydir/

#git status –s

#git commit –m "new dir"

#git push origin master

To Rename a File :

#git mv demo.c sample.c

#git status –s

#git commit -am “file renamed”

#git push orign master

To Remove a file from git Repo :

#git rm sample.c

#git status –s

#git commit -am “file removed”

#git push orign master

To Pull the Changes from git Repo :

#git pull

#git status –s

Git Stash Practice:

vi sample.c

..... CODE-A .....

git stash save "Add(), sub() code"

git stash list

git stash apply stash@{0}

vi sample.c

..... CODE-B .....

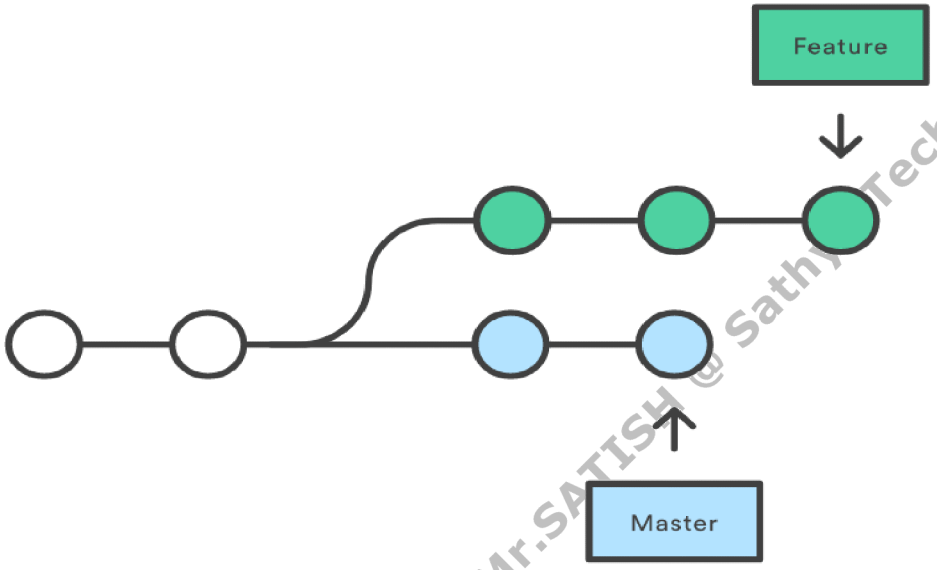
git stash save "mul(), div() code"

```
# git stash list

# git stash pop stash@{0}

# git stash list

# git stash drop stash@{0}

# git stash clear
```

Git Merge and Rebase

The Merge Option

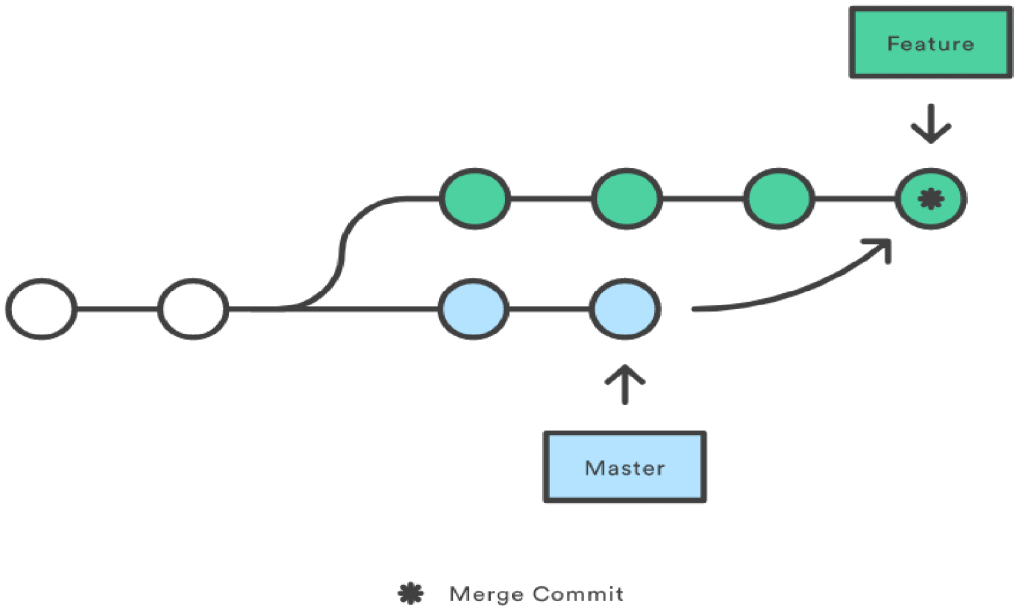
Merge takes all the changes in one branch and merges them into another branch in one commit.



Let's say you have created a branch for the purpose of developing a single feature. When you want to bring those changes back to master, you probably want merge.

```
#git checkout master

#git merge feature
```

This creates a new "merge commit" in the feature branch that ties together the histories of both branches, giving you a branch structure that looks like this:

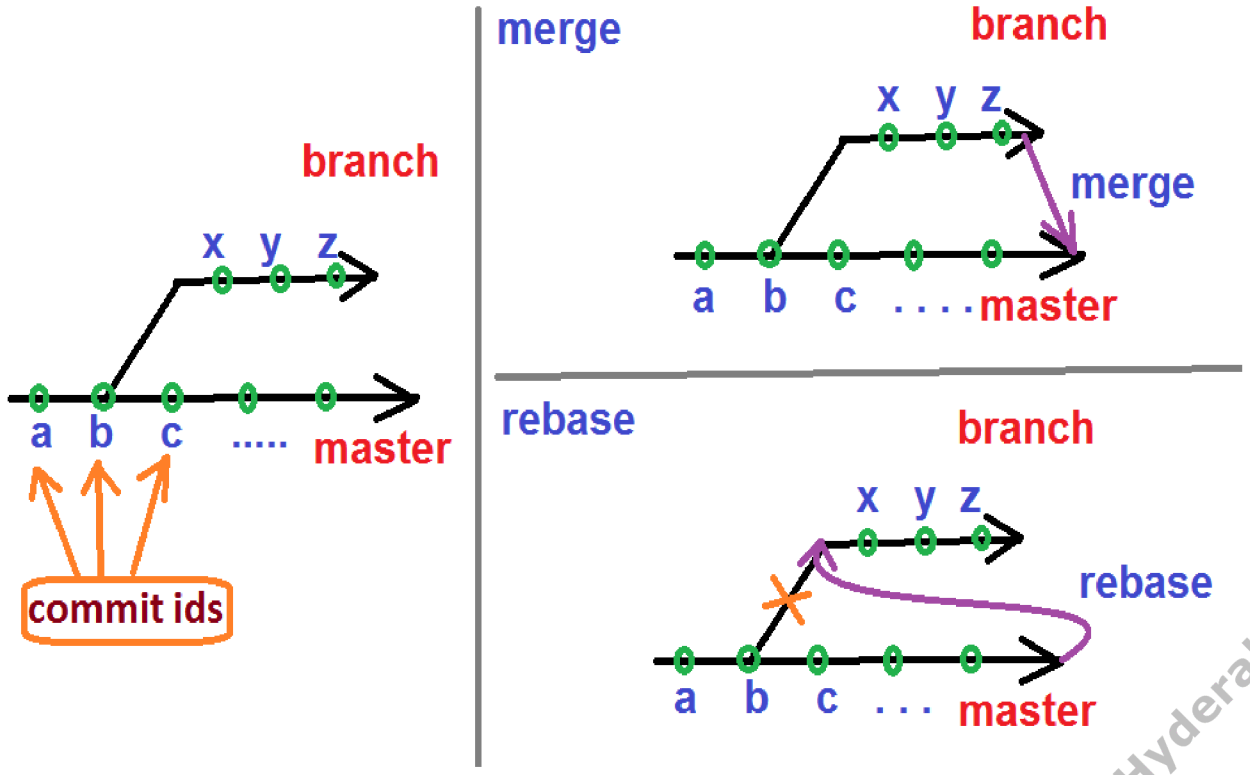


Git Rebase: As its name suggests, *rebase* exists to change the "base" of a branch, which means its origin commit. It replays a series of commits on top of a new base.

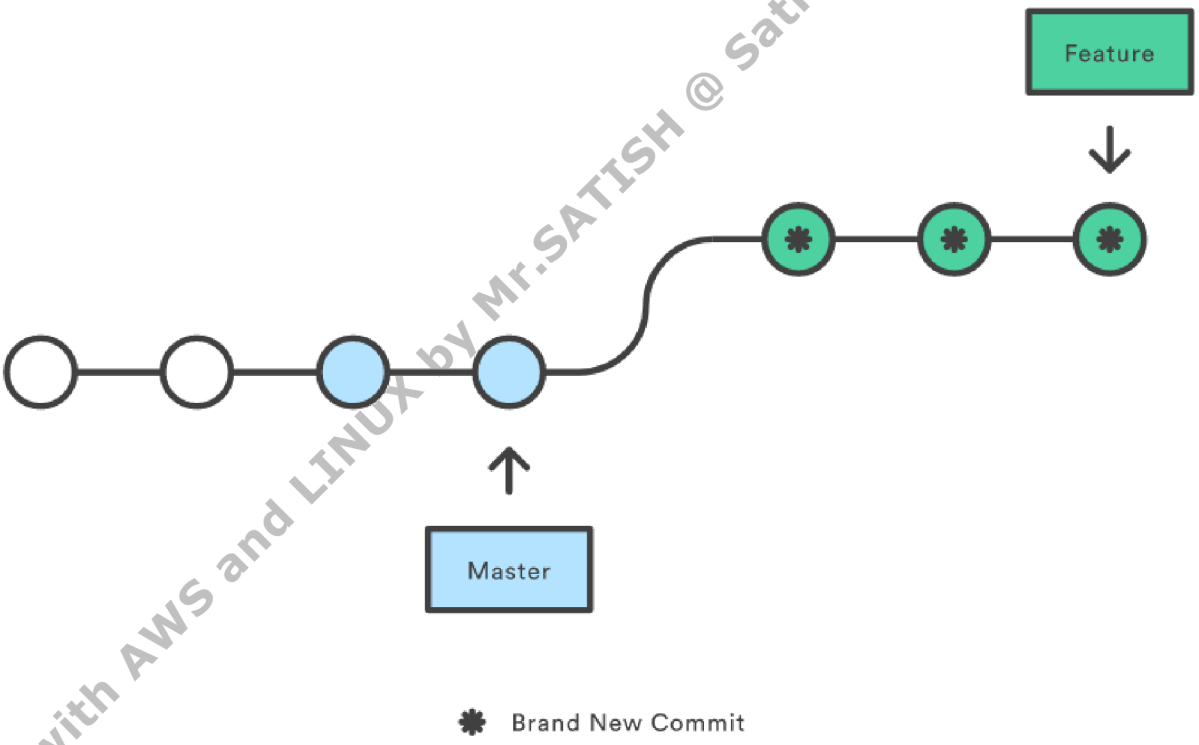
As an alternative to merging, you can rebase the feature branch onto master branch using the following commands:

git checkout master

git rebase feature



This moves the entire feature branch to begin on the tip of the master branch, effectively incorporating all of the new commits in master. But, instead of using a merge commit, rebasing *re-writes*the project history by creating brand new commits for each commit in the original branch



Git Commit Revert

#git reset --hard HEAD^1

(or)

#git reset --hard HEAD~

Git HARD and SOFT reset

HARD reset : If you don't want to keep your changes that you made:

git reset --hard HEAD^

SOFT reset: If you want to keep your changes:

git reset --soft HEAD^

To restore a deleted branch

Step-1: Find the old-branch commit ids using

git reflog

Step-2: To restore the branch, use:

git checkout -b <branch> <recent commit id>

git log --online

GIT Cherry-pick

git branch prod

git checkout prod

git log --oneline

vi devops.txt

.....test code.....

git add devops.txt

```
# git commit -m "devops tools"

# git log --oneline

# git checkout main

# git cherry-pick 05829e5

# git cherry-pick 05829e5  0894v2

# git cherry-pick --continue

# git cherry-pick –abort

# git push origin main
```

Git Rebase Practice:

```
#  vi bank.java

class Bank

{

  s.o.print("Bank code")

  s.o.print("Bank code")

}


# git add bank.java

# git commit -m "bank code"

# git push


# git checkout -b devops
```

```
# vi bank.java
void Depoiste()
{
  bal = bal + amount;
}
# git commit -am "Deposite() code"

# vi bank.java
void Withdraw()
{
  bal = bal - amount;
}
# git commit -am "Withdraw() code"

# git log --oneline
# git checkout main
# git log --oneline
# git rebase devops
# git log --oneline
5cc893d (HEAD -> devops) Withdraw() code
753efd0 Deposite() code
9766285 (origin/main, origin/HEAD, main) bank code
b3936b3 add() sub() code
```

57a99a7 c-cdoe

7846347 code merged

26ad6a1 Main code

2faec6f c-code

11f86ff java code

3a4f7ed Initial commit

cat bank.java

git status

git push

Git Commit Revert

#git reset --hard HEAD^1

(or)

#git reset --hard HEAD~

Git HARD and SOFT reset

HARD reset : If you don't want to keep your changes that you made:

git reset --hard HEAD^

SOFT reset: If you want to keep your changes:

git reset --soft HEAD^

Git commands to revert a commit

Hard reset:

```
# vi sample.c
void main()
{
  ...... TEST CODE .....
  ...... TEST CODE .....
}

# git commit -am "RESET EX."
# git log --oneline
# git reset --hard HEAD^
# git log --oneline

# vi sample.c
void main()
{
  ...... TEST CODE .....
  ...... TEST CODE .....
}
# git commit -am "RESET EX."
```

git log --oneline

ecc742e (HEAD -> main) RESET EX.

5cc893d (origin/main, origin/HEAD, devops) Withdraw() code

753efd0 Deposite() code

9766285 bank code

b3936b3 add() sub() code

57a99a7 c-code

git reset --soft HEAD^

git log --oneline

5cc893d (HEAD -> main, origin/main, origin/HEAD, devops) Withdraw()

code

753efd0 Deposite() code

9766285 bank code

b3936b3 add() sub() code

57a99a7 c-code

To restore a deleted branch

Step-1: Find the old-branch commit ids using

 # git reflog

Step-2: To restore the branch, use:

 # git checkout -b <branch> <recent commit id>

 # git log --online

```
# git checkout -b prod
Switched to a new branch 'prod'

# git branch
  devops
  main
* prod
# vi sample.c
void Test-Code-1()
{
  ::::: TEST-CODE-1 :::::
  ::::: TEST-CODE-1 :::::
  ::::: TEST-CODE-1 :::::
}
# git commit -am "Test-Code-1"
[prod 980e8cf] Test-Code-1
 1 file changed, 33 insertions(+)

# vi sample.c
void Test-Code-2()
{
  ::::: TEST-CODE-2 :::::
```

::::: TEST-CODE-2 :::::

}

```
# git commit -am "Test-Code-2"
[prod 9177666] Test-Code-2
 1 file changed, 6 insertions(+)


# git push origin prod
Counting objects: 6, done.
Compressing objects: 100% (6/6), done.
 * [new branch]      prod -> prod


# git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
# git branch
* main
  prod


# git branch -D prod
Deleted branch prod (was 9177666).
# git reflog
192dce6 (HEAD -> main, origin/main, origin/HEAD) HEAD@{0}: checkout: moving from prod to main
```

9177666 (origin/prod) HEAD@{1}: commit: Test-Code-2

980e8cf HEAD@{2}: commit: Test-Code-1

git checkout -b prod 9177666

Switched to a new branch 'prod'

git log --oneline

9177666 (HEAD -> prod, origin/prod) Test-Code-2

980e8cf Test-Code-1

GIT Cherry-pick

 # git branch prod

 # git checkout prod

 # git log --oneline

 # vi devops.txt

.....test code.....

 # git add devops.txt

 # git commit -m "devops tools"

 # git log --oneline

 # git checkout main

 # git cherry-pick 05829e5

 # git cherry-pick 05829e5 0894v2

 # git cherry-pick --continue

 # git cherry-pick –abort

 # git push origin main

```
# git log --oneline
9177666 (HEAD -> prod, origin/prod) Test-Code-2
980e8cf Test-Code-1
192dce6 (origin/main, origin/HEAD, main) code
5cc893d (devops) Withdraw() code
753efd0 Deposite() code

# git cherry-pick 980e8cf
[main 87d80c3] Test-Code-1
 Date: Sat Nov 19 03:27:01 2022 +0000
 1 file changed, 33 insertions(+)

# git cherry-pick 9177666
[main 4672d33] Test-Code-2
 Date: Sat Nov 19 03:27:45 2022 +0000
 1 file changed, 6 insertions(+)

# git log --oneline
4672d33 (HEAD -> main) Test-Code-2
87d80c3 Test-Code-1
```