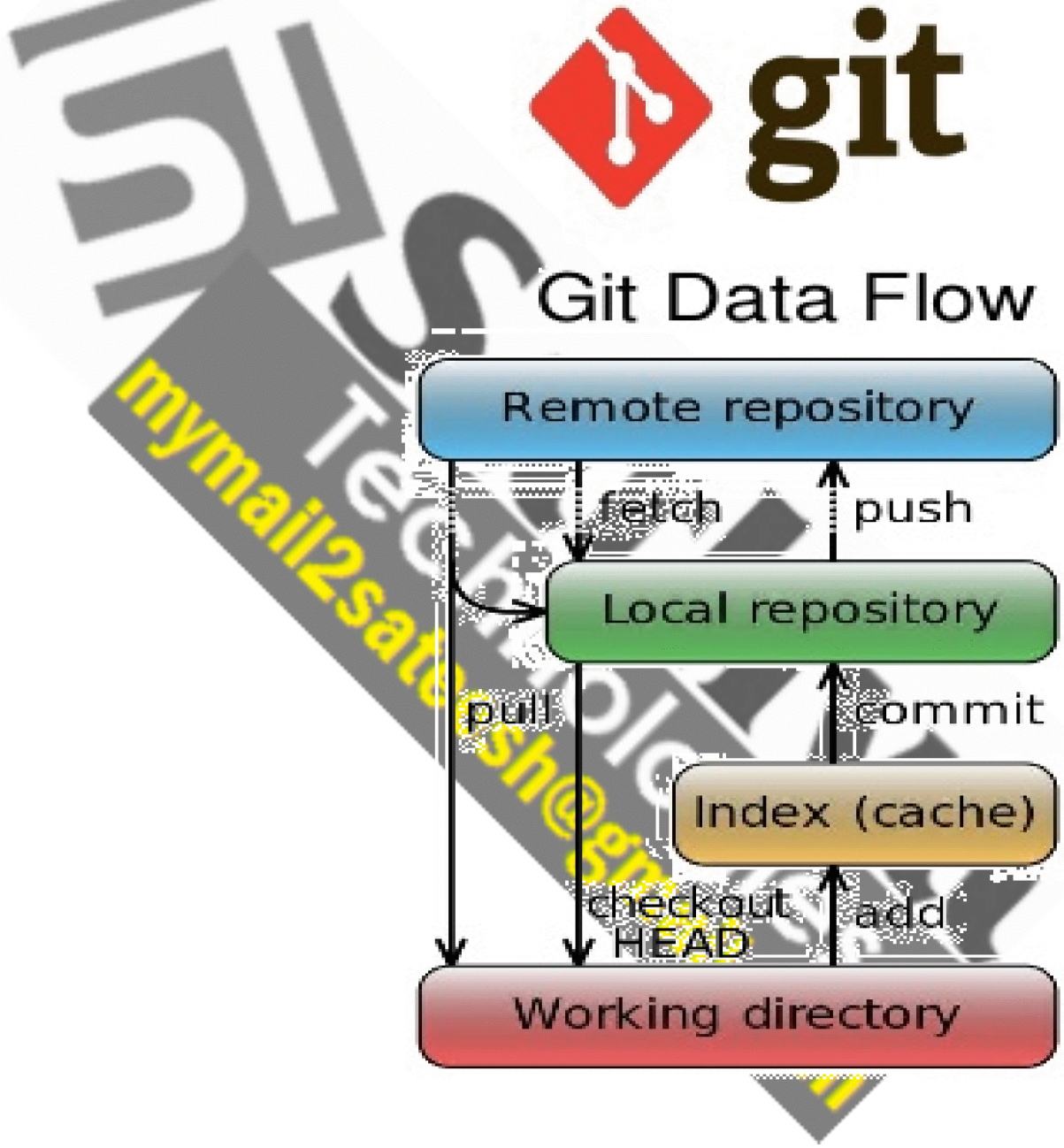
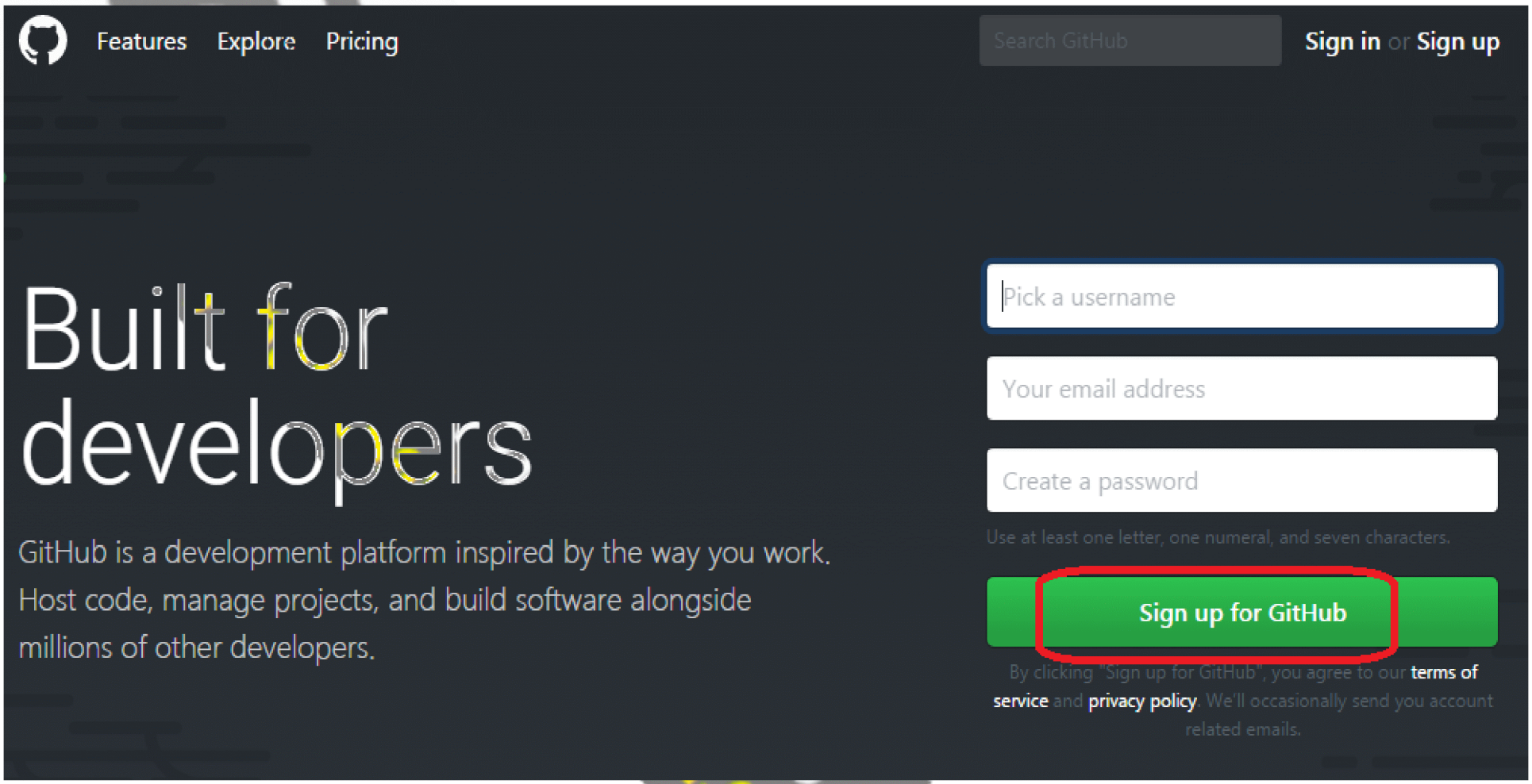
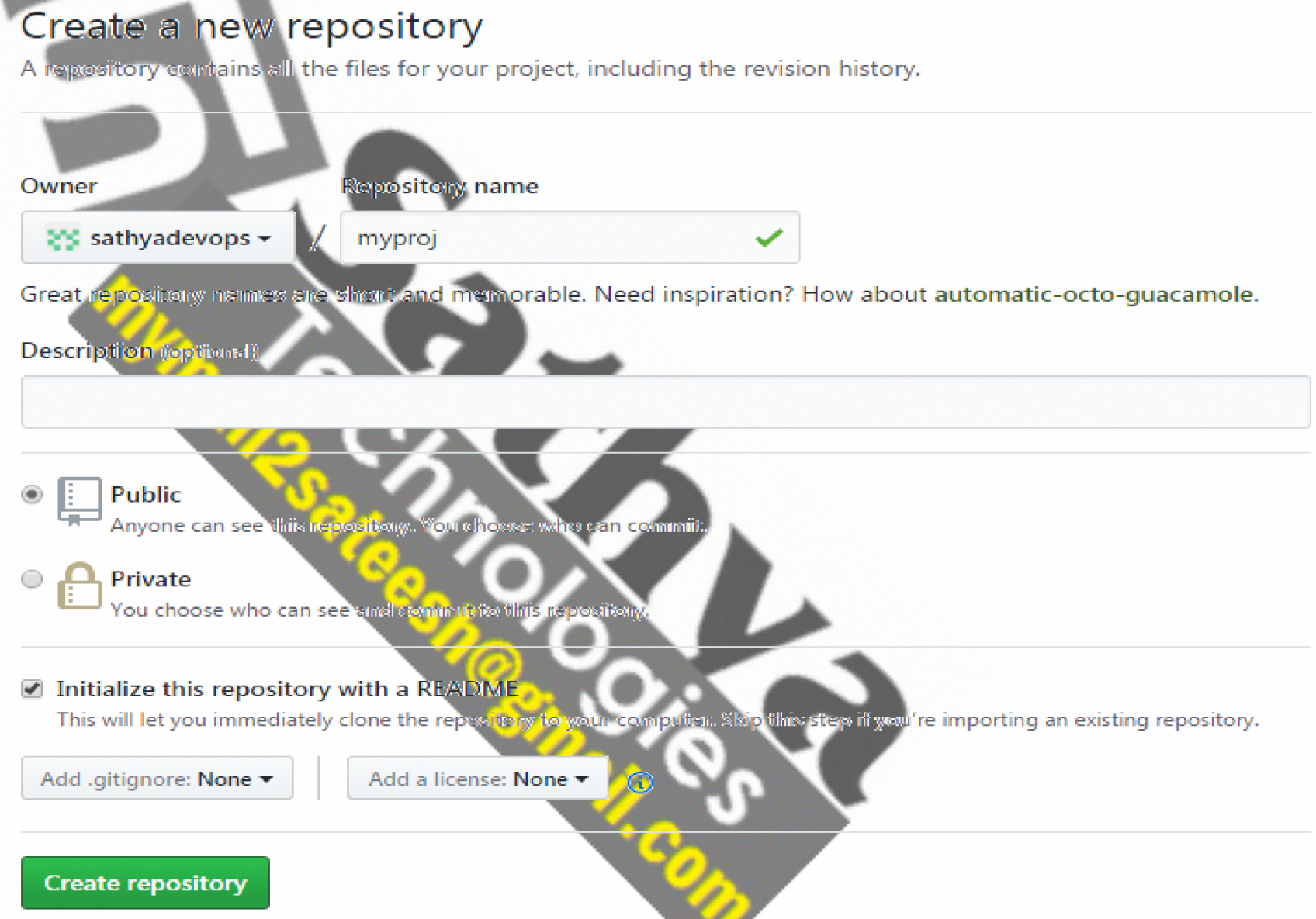

nologies
teesh@gmail.com
git
GitHub


git
Git Data Flow
Remote repository
fetch
push
Local repository
pull
commit
Index (cache)
checkout
HEAD
add
Working directory

Create Account in GitHub



Create Repository



Install Git on Ubuntu 14.04

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 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 –a -m "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 branch

Git - Review Changes

 # git diff

git log

git show

c0f455906befd100192848233fbb896d081e2284

Git – Remote Server

#git remote -v

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

#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 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 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 :

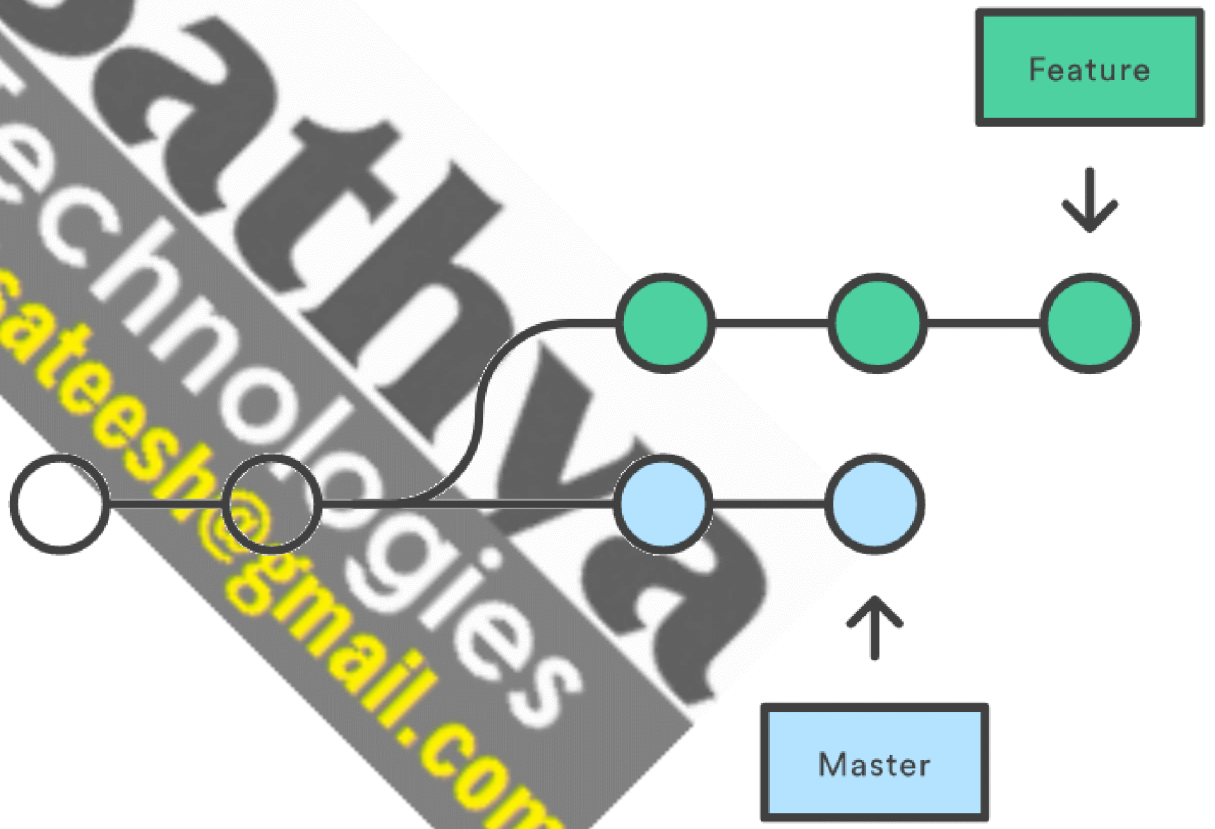
#git pull

#git status –s

Git Merge and Rebase

The Merge Option

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



Git Merge and Rebase

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 feature

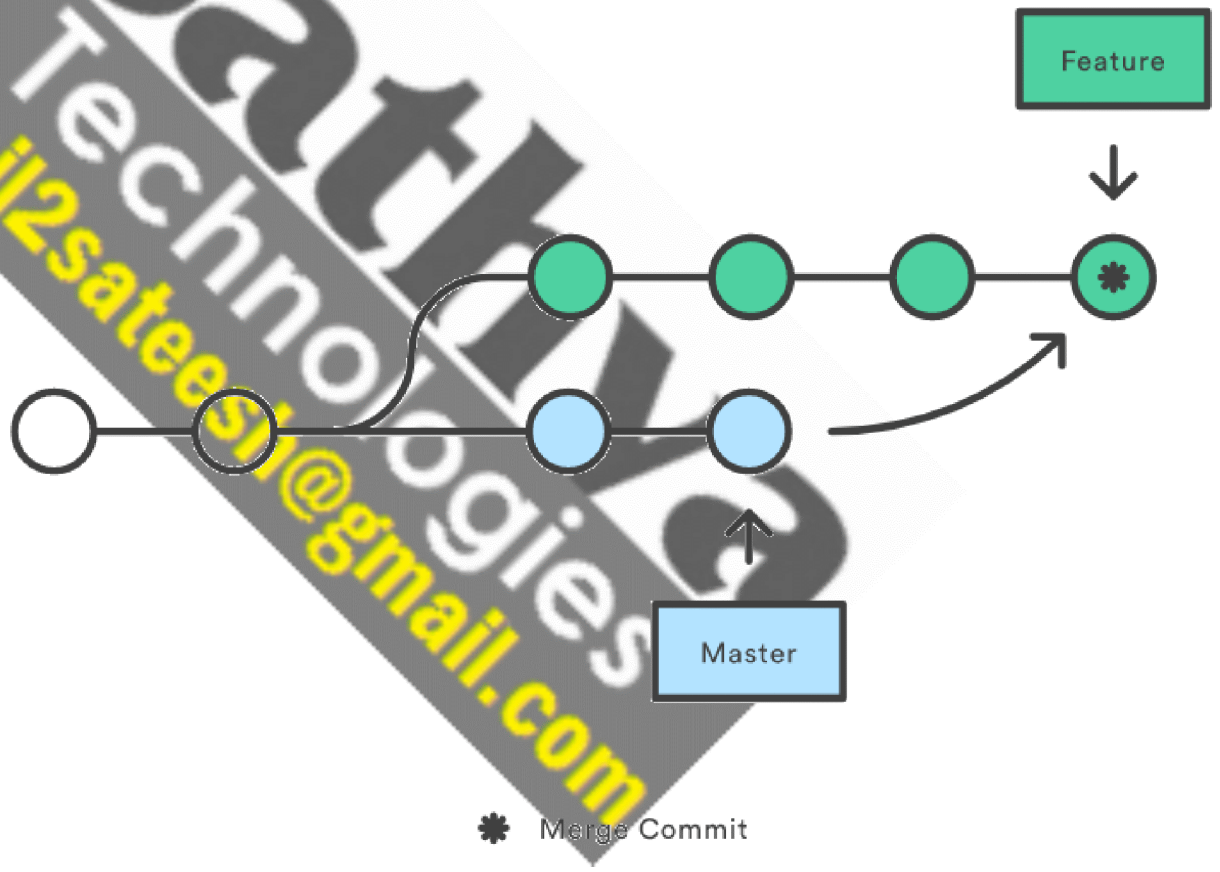
#git merge master

(OR)

#git merge master feature

Git Merge and Rebase

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 Merge and Rebase

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 feature

 # git rebase master

(or)

 # git rebase -i master (interactive rebase)

Git Merge and Rebase

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