# git

GitHub

# 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 :

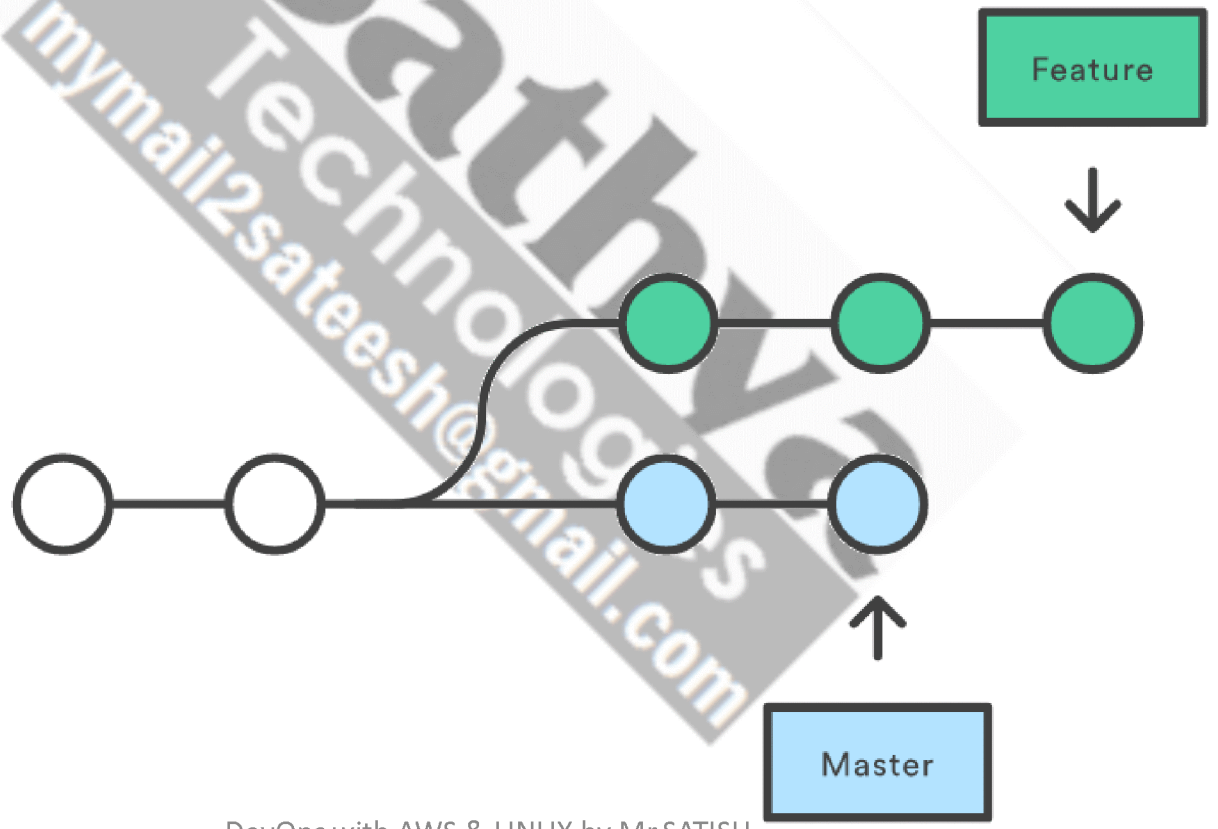
#git pull

#git status –s

# Git 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

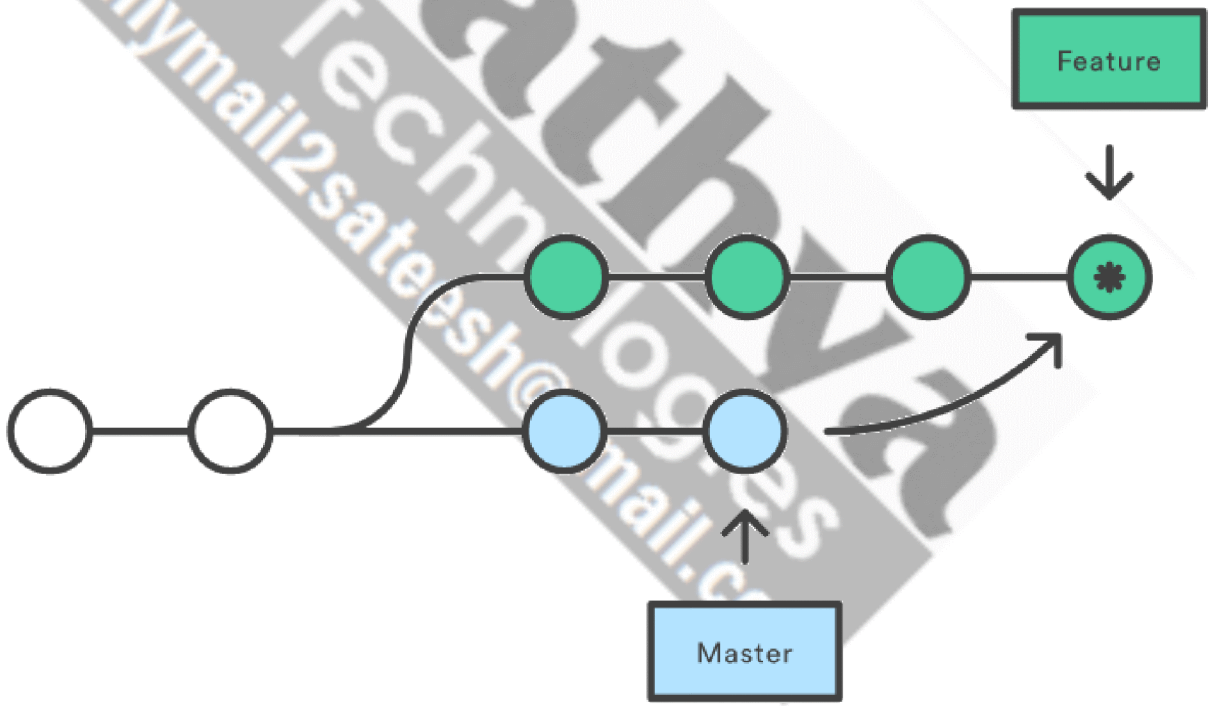
Let's say you have created a branch for the purpose of developing a single feature. When you want to bring those changes back to master, you probably want **merge**.

#git checkout feature

#git merge master

# 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:








Merge Commit

# 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.

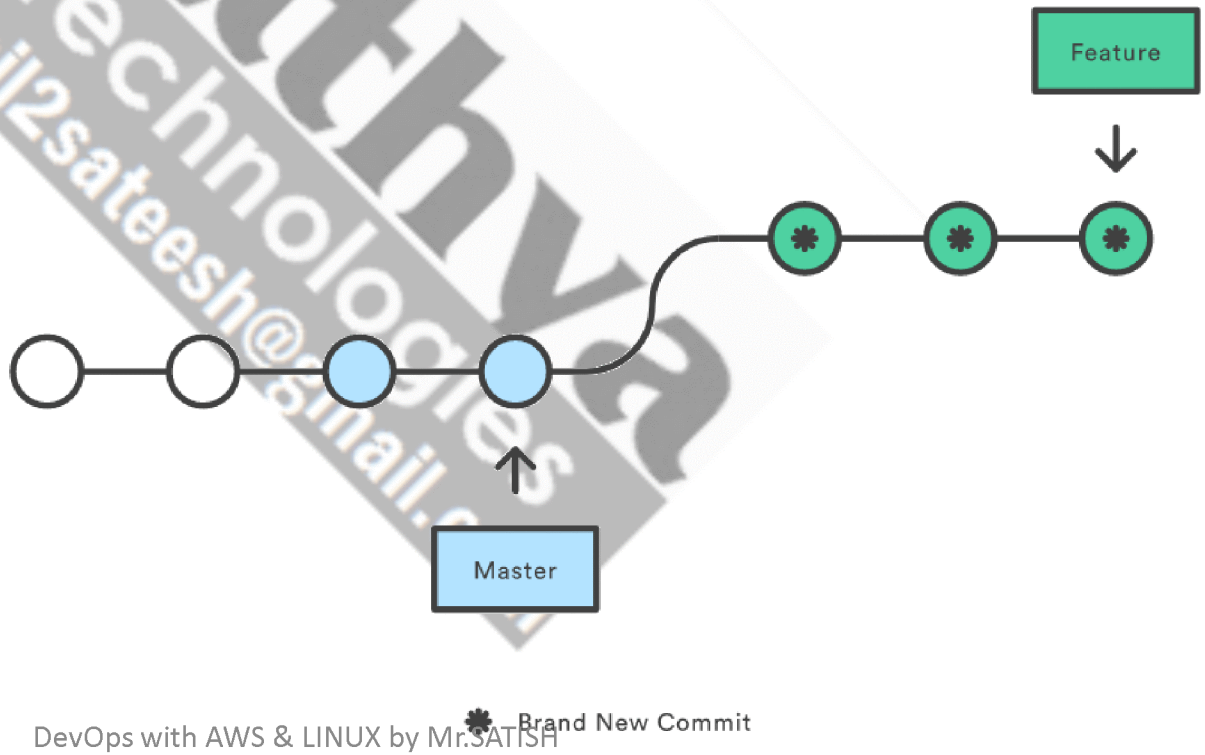
As an alternative to merging, you can rebase the feature branch onto master branch using the following commands:

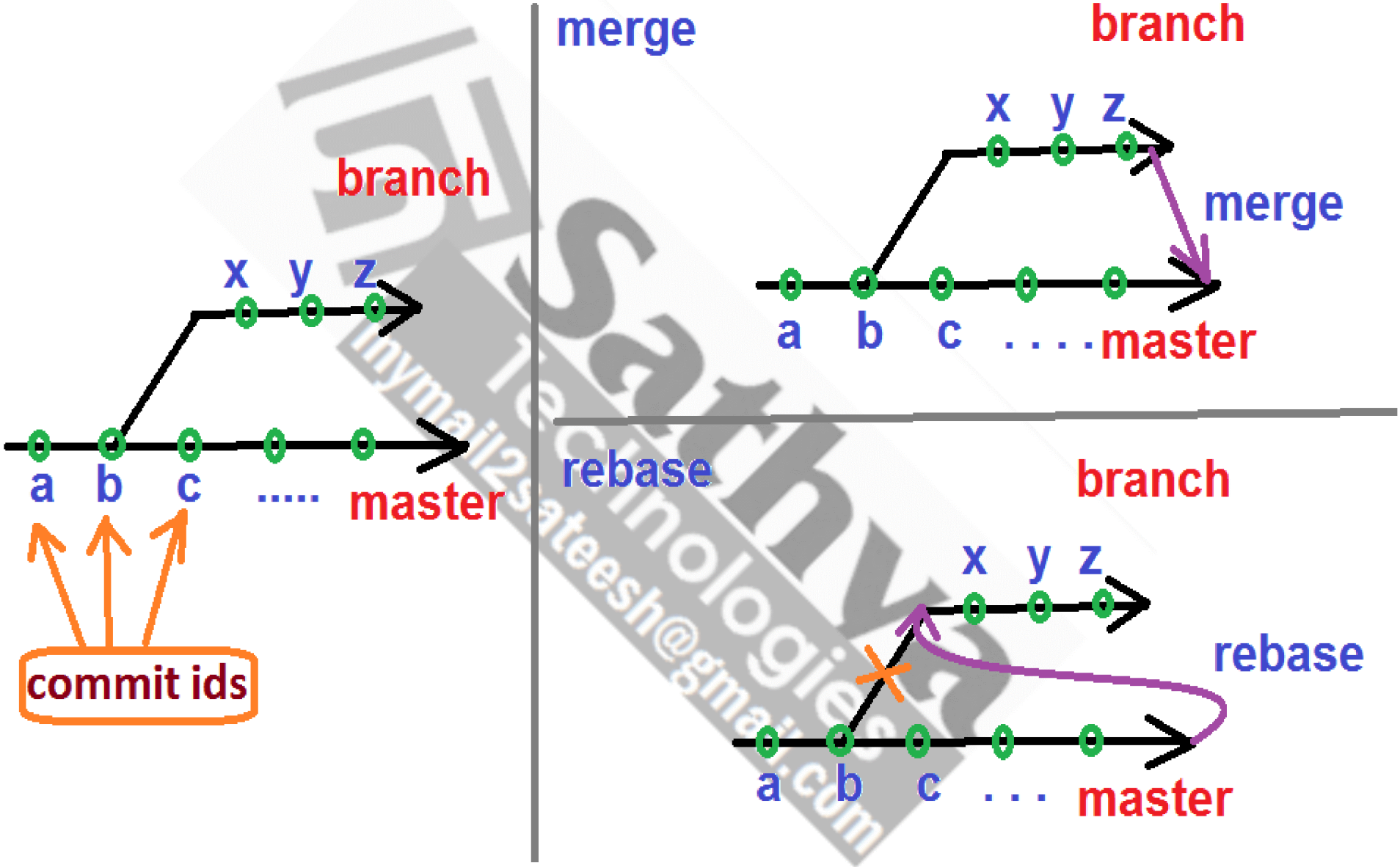
**# git checkout master**

**# git rebase feature**

# 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





## 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^