**Git Branching – Branches in a Nutshell**

// Is to deviate from the principal work and keep working and you do not care of mess the principal line.



**When you do some changes**



### Creating a New Branch

You must do this with the git branch command:

### $ git branch name

### Switching Branches

run the git checkout command

$ git checkout name

 Divergent history

You can divergent history with commant called

git log

run

git log --oneline --decorate --graph –

**Basic Branching**

To create a new branch

you run

git checkout

command with the

-b

switch:

$ git checkout -b iss53

Switched to a new branch "iss53"

This is shorthand for:

$ git branch iss53

$ git checkout iss53

You can concentrate on your hotfix.

**hotfix**

$ git checkout -b hotfix

Switched to a new branch 'hotfix'

$ vim index.html

$ git commit -a -m 'Fix broken email address'

[hotfix 1fb7853] Fix broken email address

1 file changed, 2 insertions(+)

**master** is fast-forwarded **to hotfix**

1.- delete the hotfix branch (You will not use it anymore)

2.-  the master branch points at the same place.

$ git branch -d hotfix

Deleted branch hotfix (3a0874c).

### Basic Merge Conflicts

When you want to fix some mistake (53) in hotfix branch, maybe you will get

$ git merge iss53

Auto-merging index.html

CONFLICT (content): Merge conflict in index.html

Automatic merge failed; fix conflicts and then commit the result.

to see which files are unmerged at any point after a merge conflict

git status.

If you tun git branch but you do not have arguments:

$ git branch

iss53

\* master

testing

indicates the branch that you currently have checked out

-the \* character

To see the last commit on each branch, run

git branch -v:

$ git branch -v

iss53 93b412c Fix javascript issue

\* master 7a98805 Merge branch 'iss53'

testing 782fd34 Add scott to the author list in the readme

If you want to delete the branch and lose that work

Use

-D

### Long-Running Branches

This use 3 way to merge, merging from one branch into other but many times, over a long period is generally easy to do.





### Topic Branches

Topic branches, this is so easy to do in every project, it does not matter the size. A topic branch is a short-lived branch that you create and use for a single feature or related work.



## Remote Branches

To get a full list of remote references explicitly

git ls-remote <remote>,

or

git remote show <remote>

Remote-tracking branch names take the form

<remote>/<branch>

If you already did some work master branch, and, someone else pushes to git.ourcompany.com

Then your histories will move

To synchronize your work with a given remote, run

git fetch <remote>

To push it up the same way you pushed your first branch. Run

git push <remote> <branch>:

### Pulling

To fetch all the changes on the server that you don’t have yet, run

git fetch

git pull

git fetch

git merge .

### Deleting Remote Branches

You can delete a remote branch , you run

--delete option to git push. If you want to delete your serverfix branch from the server, you run the following:

$ git push origin --delete serverfix

To https://github.com/schacon/simplegit

- [deleted] serverfix

## Rebasing

### We have two ways integrate changes from one branch into another:

* the merge
* rebase



You would check out the experiment branch, and if you want to check master branc, then this will look like this

$ git checkout experiment

$ git rebase master

First, rewinding head to replay your work on top of it...

Applying: added staged command



You can return to master branch and do a fast-forward merge.

$ git checkout master

$ git merge experiment



**A history with a topic branch off another topic branch**

You can take the changes on

client t

hat aren’t on

server (C8 and C9)

and reply

master branch

You need to use

 --onto

option of

 git rebase:

$ git rebase --onto master server client

### Rebase When You Rebase

You can rewrite works that you already write before, you need to notice what happened

It is called “Patch – id” when Git also calculates a checksum that is based just on the patch introduced with the commit.