This document is created to have a good understanding of git and its keywords:

Links:

<https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud>

A good link for understanding of HEAD and Master: <https://people.gnome.org/~federico/news-2008-11.html#pushing-and-pulling-with-git-1>

When we add a new file in a repo and do git status we see something like below:

git status

On branch master

Initial commit

Untracked files:

(use "git add <file>..." to include in what will be committed)

locations.txt

nothing added to commit but untracked files present (use "git add" to track)

In the above example, the untracked file means that the file wasn’t present in the last checkout of the repo or the file wasn’t part of previous commit.

To track the file, we do git add. Git add will put the file into staging area, the staging area is where you prepare a snapshot of a set of changes before committing them to the official history.

If we have done some wrong changes in a file and we want to revert them, then we can use git checkout command to download a copy of the file from repo and replace it in the local, example as below:

[ec2-user@ip-172-31-8-219 git\_practice]$ cat test.txt

ssdsdfgsgfdg

[ec2-user@ip-172-31-8-219 git\_practice]$

[ec2-user@ip-172-31-8-219 git\_practice]$ echo hello> test.txt

[ec2-user@ip-172-31-8-219 git\_practice]$ cat test.txt

hello

[ec2-user@ip-172-31-8-219 git\_practice]$ echo bye > test.txt

[ec2-user@ip-172-31-8-219 git\_practice]$ cat test.txt

bye

[ec2-user@ip-172-31-8-219 git\_practice]$

[ec2-user@ip-172-31-8-219 git\_practice]$ echo tu > test.txt

[ec2-user@ip-172-31-8-219 git\_practice]$ cat test.txt

tu

[ec2-user@ip-172-31-8-219 git\_practice]$ git checkout -- test.txt

[ec2-user@ip-172-31-8-219 git\_practice]$ cat test.txt

Ssdsdfgsgfdg

Git pull command working:

The git pull command first runs git fetch which downloads content from the specified remote repository. Then a git merge is executed to merge the remote content refs and heads into a new local merge commit.

The git fetch command can be confused with git pull. They are both used to download remote content. An important safety distinction can be made between git pull and get fetch. git fetch can be considered the "safe" option whereas, git pull can be considered unsafe. git fetch will download the remote content and not alter the state of the local repository. Alternatively, git pull will download remote content and immediately attempt to change the local state to match that content. This may unintentionally cause

the local repository to get in a conflicted state.

What is HEAD and Master –

They both are refs or pointers. HEAD is generally the pointer to the latest commit of the current branch, whereas Master in general is the first branch of any repo.

How can we come into detached-head state-> by checking out an old commit.

Detached-HEAD in general the HEAD represent like below:

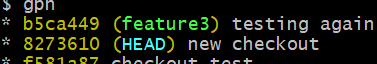
HEAD->branchname

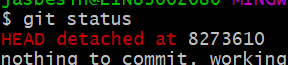






But when, for any reason, the HEAD is not pointing to the branch, it represents as detached head state. Which means that HEAD is not pointing to the latest commit of the branch, like below:





This state can be changed by switching back to any branch using git checkout.

To save the work done in a detached-HEAD state, we can create a new branch this state, which will attach the detached HEAD to the newly created branch and then merge it to the target branch.

Repository URLs

Git supports many ways to reference a remote repository. Two of the easiest ways to access a remote repo are via the HTTP and the SSH protocols. HTTP is an easy way to allow anonymous, read-only access to a repository. For example:

http://host/path/to/repo.git

But, it’s generally not possible to push commits to an HTTP address (you wouldn’t want to allow anonymous pushes anyways). For read-write access, you should use SSH instead:

ssh://user@host/path/to/repo.git

# **Rewriting history**

Git has several mechanisms for storing history and saving changes. These mechanisms include: Commit --amend, git rebase and git reflog.

**Git commit --amend:**

the amend option is being used to edit the last commit on the branch. Thid actually overwrite the previous commit and add a new commit at the same position. It helps us in below scenarios:

1. We want to edit the message of out last commit. When we execute ‘git commit –amend”, it will prompt us a message window, where we can type our new commit message.
2. We have added multiple files in the last commit but missed one or more file to add or forgot to add something in any files that was commited. For that we have to stage those files first using the git add command and then we can use amend, this will overwrite the previous commit by adding the missed files.

**Git commit –amend –no-edit** -> this command will amend without prompting the commit message.

**Git Rebase:**

Rebase is one of two Git utilities that specializes in integrating changes from one branch onto another. The other change integration utility is git merge. It has two modes: "manual" and "interactive"

Rebasing is the process of moving or combining a sequence of commits to a new base commit.

