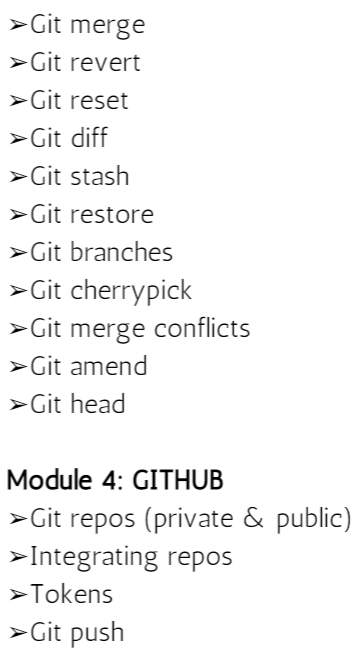
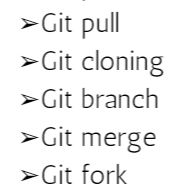
****

****

****

****

**GIT CLASS-1 (**ghp\_idBPLfgPXwA2oFURDIA8zlLKLHwL5M1ZyeHj)

TO INSTALL GIT IN LINUX MACHINE :

command: **yum install git -y**

**[root@ip-172-31-37-230 ~]# yum install git -y**

yum : yellowdog updater modiﬁer

This is a package management tool in linux which is used to install, uninstall and update the tool

To check git version : **git –version**

**[root@ip-172-31-37-230 ~]# git --version**

**git version 2.40.1**

To init a git : **git init .**

Here dot(.) means pwd present working directory

**[root@ip-172-31-37-230 ~]# git init .**

**TO COMMIT A FILE :**

create a ﬁle : **touch aws**

**[root@ip-172-31-37-230 ~]# touch aws**

Track the ﬁle : **git add aws**

**[root@ip-172-31-37-230 ~]# git add aws(now it is in tracking -staging aera)**

To check the status : **git status** Staging aera:-git status

**[root@ip-172-31-37-230 ~]# git status**

**On branch master**

**No commits yet**

**Changes to be committed:**

**(use "git rm --cached <file>..." to unstage)**

**new file: aws**

**add data with vim editor after adding click on git status**

[root@ip-172-31-37-230 ~]# git status

On branch masters

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: aws

Changes not staged for commit:

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

(use "git restore <file>..." to discard changes in working directory)

**modified: aws**

**how to know data is added or not ?**

To commit a ﬁle : **git commit -m “enter a commit message” fllename** Commit all tracking

[root@ip-172-31-37-230 ~]# git commit -m "my first commit" aws

[master (root-commit) a3f4883] my first commit

Committer: root <root@ip-172-31-37-230.ec2.internal>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly. Run the

following command and follow the instructions in your editor to edit

your configuration file:

git config --global --edit

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

**1 file changed, 3 insertions(+)**

create mode 100644 aws

ﬁles : **git commit -m “enter a commit message” .**

To check the history : **git log**

**[root@ip-172-31-37-230 ~]# git log**

**commit a3f4883beb124d884ee97299ba28cd5faffb5d0b (HEAD -> master)**

**Author: root <root@ip-172-31-37-230.ec2.internal>**

**Date: Thu Jul 11 12:46:02 2024 +0000**

**my first commit**

To get commit details along with ﬁlename : **git show commit\_id --name-only**

**[**root@ip-172-31-37-230 ~]# git show a3f4883beb124d884ee97299ba28cd5faffb5d0b --name-only

commit a3f4883beb124d884ee97299ba28cd5faffb5d0b (HEAD -> master)

Author: root <root@ip-172-31-37-230.ec2.internal>

Date: Thu Jul 11 12:46:02 2024 +0000

my first commit

**aws**

Track multiple ﬁles : **git add flle1 flle2**

Track all ﬁles : **git add**

GIT CLASS-2

To untrack a file :**git rm –cached html**

[root@ip-172-31-80-163 ~]# **git rm --cached html**

rm 'html'

To get history of a git : **git log**

To get top 3 commits : **git log -3**

To get only commit-ids and commit messages : **git log –oneline**

[root@ip-172-31-80-163 ~]# **git log --oneline**

acd4b44 (HEAD -> master) html commit

faa7430 commiting dev

5412352 commit aws

abf9e59 commit al files

To get only full commit-ids and commit messages : **git log --pretty=oneline**

[root@ip-172-31-80-163 ~]# **git log --pretty=oneline**

acd4b448ef0f350cfa675b273b3b8894dfa58280 (HEAD -> master) html commit

faa7430d9586bedb56195a2099d028c4cb2946f4 commiting dev

5412352129540d0fb0cd29ce904fa7baca7baf0d commit aws

abf9e59c05d4eb1360b80a3b546bf1cd12178895 commit al files

To get full commits of a single ﬁle : **git log --follow --all ﬁlename**(to know the second commit)

To get history of a commit : **git show commit id** (to know how much info is deleted(-) and inserted(+) while we commit second time )

To get history of a commit in stat format : **git show commit --stat**

[root@ip-172-31-80-163 ~]# **git show d14405f28e00498c3ca2e8b23c21be85c4a752db --stat**

commit d14405f28e00498c3ca2e8b23c21be85c4a752db (HEAD -> master)

Author: root <root@ip-172-31-80-163.ec2.internal>

Date: Sat Jul 13 12:05:21 2024 +0000

seconf commit

aws | 2 ++

1 file changed, 2 insertions(+)

**To** conﬁgure user name : **git conﬁg user.name “our name”[**we **must commit** after changing user name ]

To conﬁgure email id : **git conﬁg user. email “our-email”**

**[root@ip-172-31-80-163 ~]# git config user.name "Seema"**

**[root@ip-172-31-80-163 ~]# git config user.email "Seema@dev.com"**

[root@ip-172-31-80-163 ~]# git log -3

commit 8ec458e3a2bd20d5614e6008cc4946536c2c83e0 (HEAD -> master)

Author: **Seema <Seema@dev.com>**

Date: Sat Jul 13 12:20:00 2024 +0000

**Or**

change it from file

Cd .git/

Vim config file->modify user name and email

cd

Commit any file

Git log -1

Now user name and email will be changed

To edit the latest commit message : **git commit --amend -m “message”**

To edit author for latest commit : **git commit --amend --author “name** <mail>”

To commit the changes with prev commit : **git commit --amend --no-edit**

**Story:-**The above is used to commit files (when we miss any files ,to add that file we used this cmd)

Create 5 files and committed 4 files only

[root@ip-172-31-80-163 ~]# **git commit -m "all tracking file" file{1..4}**

[master 9616c61] all tracking file

4 files changed, 0 insertions(+), 0 deletions(-)

create mode 100644 file1

create mode 100644 file2

create mode 100644 file3

create mode 100644 file4

[root@ip-172-31-80-163 ~]# **git commit --amend --no-edit**

[master 4f16e12] all tracking file

Date: Sat Jul 13 13:41:09 2024 +0000

5 files changed, 0 insertions(+), 0 deletions(-)

create mode 100644 file1

create mode 100644 file2

create mode 100644 file3

create mode 100644 file4

create mode 100644 file5

[root@ip-172-31-80-163 ~]#

**Now 5th files is attached to commit by using the command and the cmd will focus on all staging arear files -only previous commit this cmd will use**

**Git Day3:-**

**Delete Files:-**

**2** types of deletions

1. Soft commit- only commit (file will be exist in staging aera-only commit will be deleted)
2. Hard Commit-Commit and action(data) both will delete

To delete latest commit :-**git reset –hard HEAD~1**

[root@ip-172-31-85-14 ~]# **git reset --hard HEAD~1**

HEAD is now at d250b11 mysecond linux

[root@ip-172-31-85-14 ~]# **git reset --soft HEAD~1**

[root@ip-172-31-85-14 ~]# git log

commit 236944d9460412d44c3dfb623198ae35b270e37d (HEAD -> master)

Author: root <root@ip-172-31-85-14.ec2.internal>

Date: Sat Jul 13 15:09:25 2024 +0000

my first linux

[root@ip-172-31-85-14 ~]# cat linux

hi am linux this is my first commit

**my second commit (**second commit data is not deleted **)**

**To delete all** commits at a time :-**git update-ref -d HEAD**

[root@ip-172-31-85-14 ~]# **git update-ref -d HEAD**

[root@ip-172-31-85-14 ~]# git log

fatal: your current branch 'master' does not have any commits yet

To get deleted commits list :-**git reflog**

[root@ip-172-31-85-14 ~]# **git reflog**

34266da (HEAD -> master) HEAD@{0}: reset: moving to HEAD~1

34f50cb HEAD@{1}: commit: my dev

34266da (HEAD -> master) HEAD@{2}: commit: my aws

ae272da HEAD@{3}: commit (initial): my linux

0613ce9 HEAD@{5}: commit: my linux

249106f HEAD@{6}: commit: my linux

236944d HEAD@{7}: reset: moving to HEAD~1

661d93d HEAD@{8}: commit: my second linux

236944d HEAD@{9}: reset: moving to HEAD~1

d250b11 HEAD@{10}: reset: moving to HEAD~1

7dd9349 HEAD@{11}: commit: my Third linux

d250b11 HEAD@{12}: commit: mysecond linux

236944d HEAD@{13}: commit (initial): my first linux

To Restore the deleted commit :- **git cherry-pick commit id**

[root@ip-172-31-85-14 ~]# **git cherry-pick 34266da**

**For Restoring M**ultple commits:- git cherry-pick commitid1,commitid2,commitid3…

**Branches: -**

**Default**  Branche is **MASTER**

**First we must do one commit to know in which we are there**

**[root@ip-172-31-81-255 ~]# git branch**

**\* master**

**To**  create a new branch :-**git branch branch name** or git checkout -b branch name (this will automatically switched with new branch )

[root@ip-172-31-81-255 ~]# **git branch Devops**

[root@ip-172-31-81-255 ~]# git branch

Devops

**\* master**

To switch to another new branch :- **git checkout Devops**

[root@ip-172-31-81-255 ~]# **git checkout Devops**

Switched to branch 'Devops'

[root@ip-172-31-81-255 ~]# git branch

**\* Devops**

master

To get commits and files frpm another branche:-**git merge Master(**all files will merge**)**

If we want to get data from devops

Then we have give cmd frm master

To get data from dev to master

To take particular files and commit from one branch to another branche:-**git cherry-pick commit ids1 commit ids2……**

[root@ip-172-31-87-149 ~]# git log --oneline

adc28fe (HEAD -> master) **Merge branch 'Seema'**

ac9ed30 linux commit

172d494 windows commit

887d72e (Seema) macos commit

89f154c windows commit

543bd3a linux commit

3d8bc88 aws commit

To delete Branch :-**git branch -d branch name**

[root@ip-172-31-87-149 ~]# **git branch -d Seema**

Deleted branch Seema (was 887d72e).

[root@ip-172-31-87-149 ~]# git branch

**\* master**

**To delete branch we have to delete it from another branch**

**git branch -d branch name**

**git branch -D branch name**

When we use Capital **‘D’ forcefully data will be removed**

Staging Future

10 loc 10loc

5 commits

Witout mergeing data if want to delete future branch we use D

[root@ip-172-31-87-149 ~]# git branch -d seema

error: The branch 'seema' **is not fully** merged.

If you are sure you want to delete it, run **'git branch -D seema'**.

To change branch name :-**git branch -m branch old name new name**

[root@ip-172-31-87-149 ~]# git branch nani

[root@ip-172-31-87-149 ~]# git branch

\* master

nani

[root@ip-172-31-87-149 ~]# **git branch -m nani Seema**

[root@ip-172-31-87-149 ~]# git branch

Seema

\* master

**Merge Conflict :-**Differ data in 2 branches , while merging conflict will be raised

**Branch1 Branch2**

**aws (hit this is aws) aws (2365867)**

**How to create conflict**

**Branch1 ->commit**

**Create another branch 2-overrider data->commit**

**Again branch1->add data ->commit**

**While merging conflict will be coming**

**[root@ip-172-31-83-107 ~]# vim aws**

**[root@ip-172-31-83-107 ~]# git add aws**

**[root@ip-172-31-83-107 ~]# git commit -m "resolved"**

**If you don’t want any conflict:-git merge –abort**

**Stash: -data will be removed temporarily**

**Don’t know whether to commit feature branch, lack of confidence on code to commit**

**3files:3 commits 3files:3 commits**

**Extra data**

**Master feature**

**Once files is modified Without committing we cant move to another branch**

**So stash is used to remove data temporarily**