GIT:--

install git:

https://git-scm.com/downloads

create account on github/bitbucket/gitlab:

https://github.com/

https://bitbucket.org/product

<https://about.gitlab.com/>

--------------------------What is git

git is a version control tool, using git we can deploy any specific version as per the requirement.

git is a client that we can install on the system, while github is a remote repository where you can store the code.

there are 2 types of github:-

1. Company managed GitHub ->

- purchase github plan (https://docs.github.com/en/billing/managing-billing-for-your-github-account/upgrading-your-github-subscription)

- install Github (internal server/ec2)

2. Microsoft managed GitHub (community) :- Github manages the backend server

--------------------Git Commands:

create a folder on your local system (desktop)

go inside that folder -> right click and open git bash here

git init ----> initialized local git repository (whenever you create any new folder then you need to initialize otherwise not needed for existing repo)

this command creates a .git folder -> to store your logs locally/commits id/messages etc

vi jenkinsfile ---------> create a file

git status --------> it tells you the status of file (file is committed or not)

git add --> this moves file from working to staging area ---> when you are done with the changes

syntax: git add <file1> ---> single file

or

git add <file1> <file2> <file3> ---> multiple file can also be moved from working to staging area

or

git add . --> all the files in your current directory/folder

git status ---> green color/ changes to be committed/git has started tracking the file

git config --global user.name "prakash" -----> one time command to tell git who are you

git config --global user.email "prakash@gmail.com" ----> just or identification

syntax: git commit -m "message"

where m : message

git commit -m "jenkinsfile file created"

git log -----> it lists all the commit id (commit IDs are nothing but version)

or

git log --oneline

syntax: git show <commit ID> ----> tells the history/activity performed on that commit id

ex: git show b13d8e4

or

git show b13d8e4fec66bd11570359ee98cecd7e59a41182

{HEAD -> master : HEAD or git is pointing to the master branch-> you have made the changes on master branch}

\*\*\*\* you can modify existing committed file

vi jenkinsfile -----> then add new line or do modification as per your project req

git status

git add jenkinsfile ----> move from working to staging area

git status

\*\*\*\*now you can commit the new changes

git commit -m "modified jenkinsfile, added new k8s stage"

git log ----> you will new commit id (2 commit IDs)

or

git log --oneline

\*\*\*\*\* now we can share our local changes with other team members ?? upload/push changes from local repo to remote repo

- login to the GitHub

- login a remote repository (right top corner -> plus icon -> new repo)

- public/private ---> choose private (so that you can control the access)

once your repo is ready: make sure you have commit Ids

git remote add origin https://github.com/prakashk0301/devops-oct-22.git

we are adding a remote repository "https://github.com/prakashk0301/devops-oct-22.git" as alias called "origin"

keyword "origin" is pointing to github repo...

syntax: git push <alias name> <branch name>

ex: git push origin master -----> we can upload changes to github repo from master branch

you can also work with github URL

git push https://github.com/prakashk0301/devops-oct-22.git master

or git remote add prakash https://github.com/prakashk0301/devops-oct-22.git (any name is fine)

\*\*\*\*\*git remote -v ----> this command tells the alias name

\*\*\*\*Assignment: upload the same code to Bitbuckt and GitLab

- create repo

- run git remote add origin bitbucket-url

- git push -u origin master

—-----------------git remote -v

This command display **all the remote repo and respective alias**

**—---------**we can also configure **ssh key** to upload/download changes (password less method)

We can authenticate using ssh key -> generate ssh key locally -> paste pub to github

ssh-keygen —-> generate private & public key

Copy public key content & paste (github.com-> profile -> setting -> ssh & gpg -> add your key

git remote add origin1 [git@github.com](mailto:git@github.com):prakashk0301/devops-oct-22.git

(not https repo URL)

git push origin1 master

\*\*\*\*\*\* How many branches are there in your org?

Ans: 6 branches

Master

Dev\_branch

Devops\_branch

Feature\_branch

Bug\_branch

release \_branch

{note: you can create many branches not just 6}

\*\*\*How to merge locally

122 git checkout -b local\_branch

123 vi jenkinsfile

124 git add .

125 git commit -m "jenkins file update"

126 vi auto\_deploy.sh

127 git add .

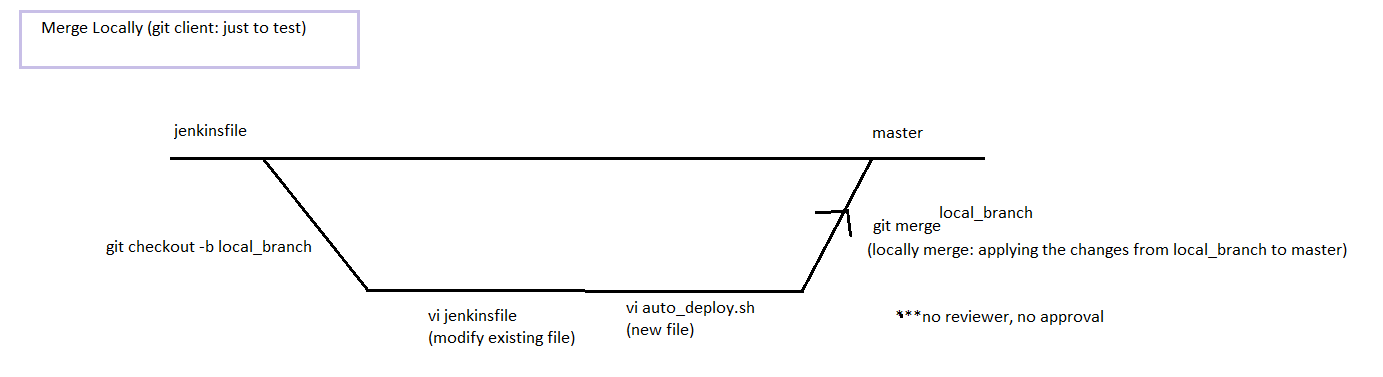
129 git commit -m "created new file to deploy automatically"

130 gitine

131 git checkout ma log --onelster

132 ls

133 git merge local\_branch



**\*\*\* when you mistypes the last commit message or forgot to add a change**

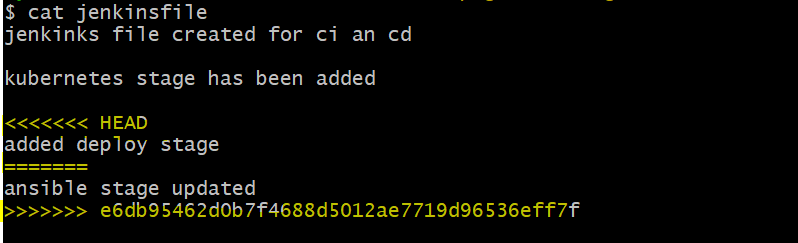
git commit --amend -m “New your message”

It also changes the commit id

**\*\*\*\*\* what do you understand by conflict**

Whenever we execute git merge (merge 2 branches), and when you have diff contents in the file

Whenever you have file level or line level changes then will get conflict



to resolve conflict Just remove unnecessary lines/line/content from your file

Or you can use git merge tool to fix conflict, it compares files

Or you can consider notepad++ compare plugin

Or you can consider visual studio code -> compare

Why does conflict occur???? Whenever we all work on same branch

—-----------**How to download code from Remote (github/gitlab/bitbucket) to Local Repo(laptop)**

git clone <URL>

{if you are on Linux system then download git bash first

apt get install git or yum install git -y

git clone <url>**When to consider**

**when you consider:** when you come back from long vacation, or if are you new to the team/project

—---------------------

**Git pull:** Download changes (file/content) from remote repository to local repository

git pull <alias> <branch>

Ex: git pull origin master

**when you consider:** when there are many team members in a team and all are working on the same code. Then you can pull their changes from remote repository to local repository.

Before you run the git **push** command simply run **git pull** to download the changes made by other team members, so that you all can be in sync.

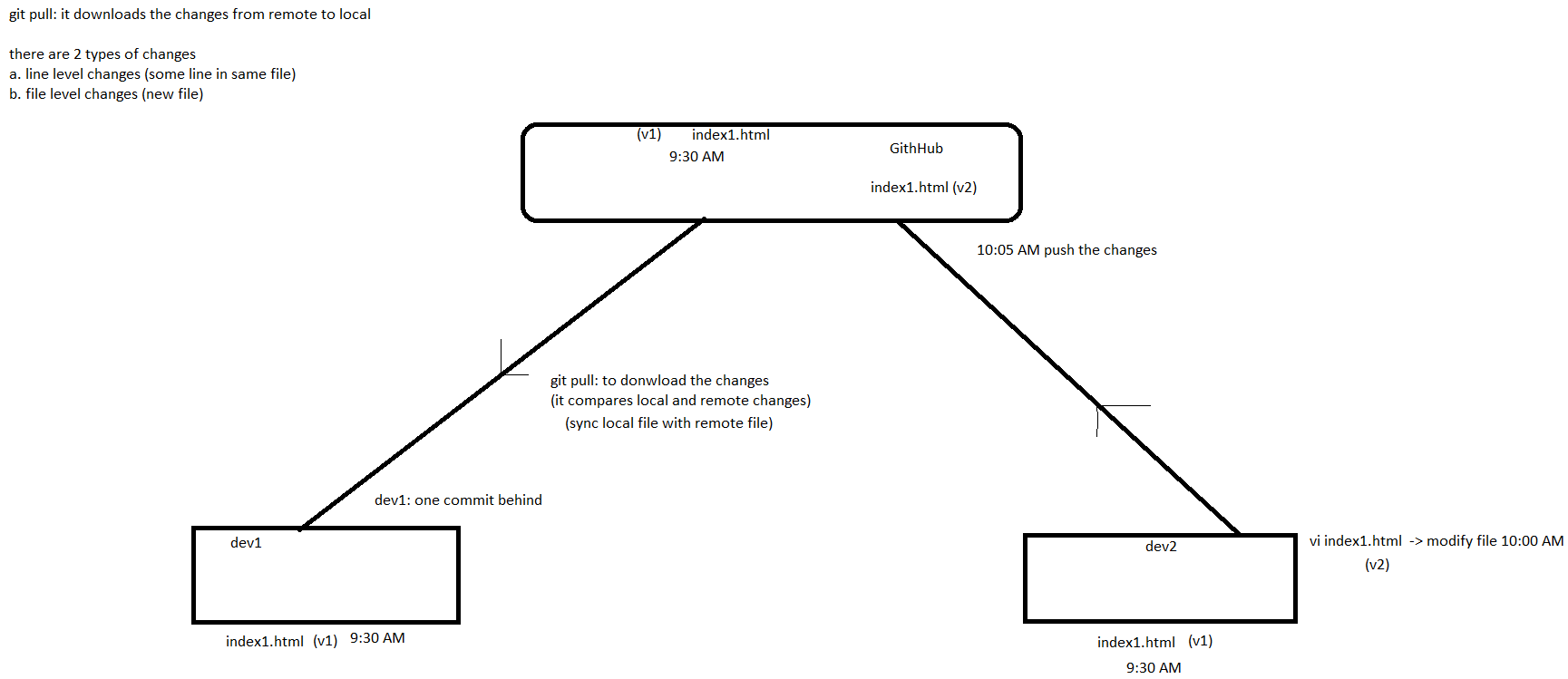
Git pull: git fetch + git merge

-> git pull origin master --force (this may corrupt local repo)

git push origin master --force (this may corrupt remote repo)

\*\*\*\*no one should upload any changes to master branch forcefully

(by implementing the permission)



Q. Diff between git pull and git fetch???

Git pull: it downloads the changes from remote to local on same branch

Git pull = git fetch + git merge

**git fetch**: it also downloads the changes from remote to local but on remote fetched branches.

Once you execute git fetch, this command gonna download changes locally on remote fetch branch now you can switch to that branch ->verify the changes -> if you are okay with the changes then -> merge with your branch

git fetch

git branch -a —--> it displays all the hidden branches

remotes/origin/master —--> this is going to be my fetched branch

git checkout remotes/origin/master

Verify your file if you are okay then switch back to master and merge

git checkout master

git merge remotes/origin/master

—--------------------**git cherry-pick**

Q. how to pick specific changes or commit id ?

Or

Q. how to merge specific changes or commit id

git status

git commit -m "created initial files"

git log --oneline

git checkout -b devops-testing

ls

vi Dockerfile

git status

git commit -m "modified on devops-testing branch"

git add Dockerfile

git commit -m "modified on devops-testing branch"

git log --oneline

vi ansible-playbook.yaml

git add ansible-playbook.yaml

git commit -m "ansible file created on devops testing branch"

git log --oneline

vi terraform.tf

git add terraform.tf

git commit -m "terraform created on devops testing"

git log --oneline

git branch

git checkout master

ls

git log

git cherry-pick 0f24cd8

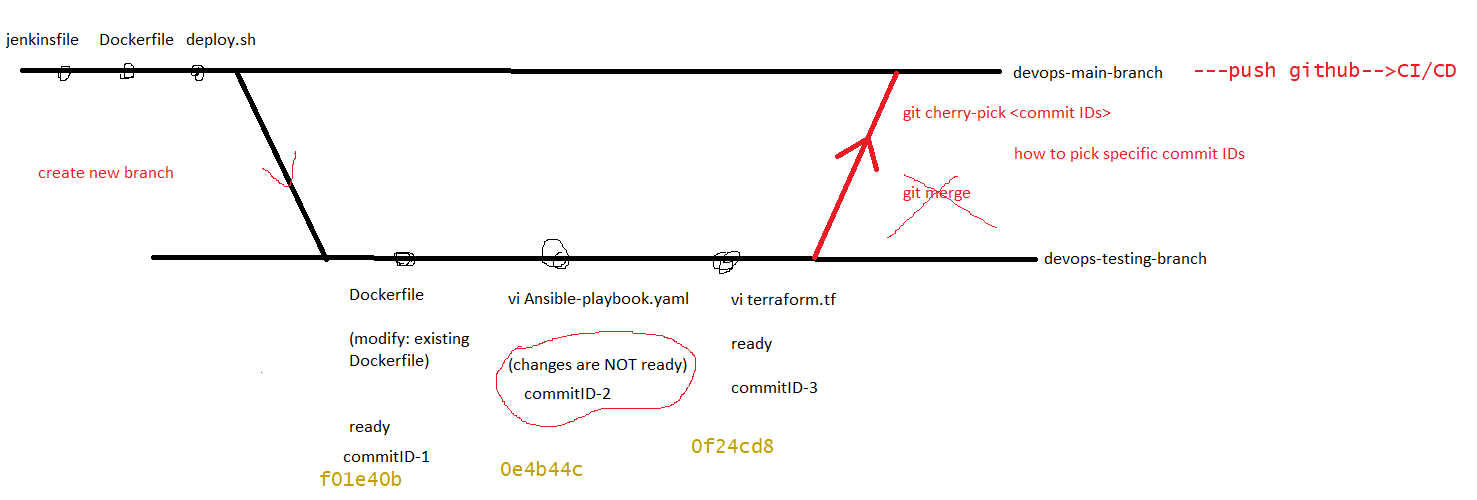
ls

cat Dockerfile

git cherry-pick f01e40b

ls

cat Dockerfile



—--How to revert the changes: **git revert**

If something goes wrong then how to revert the changes (commit ID)

There are 2 possibilities:

1. Undo the changes (git revert) -> undo/revert commit id
2. Create bug-fix branch then ask dev/qa/devops to fix the issue/bug then redeploy immediately

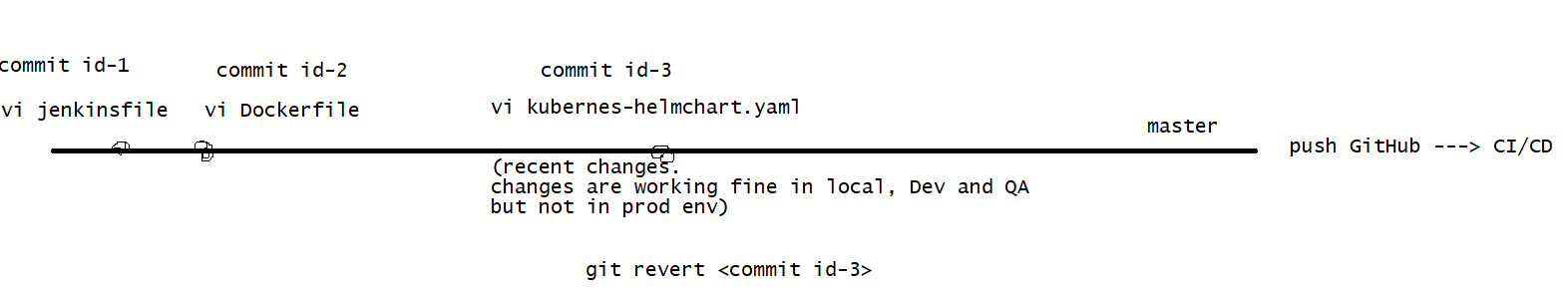
Git revert maintains the git history (when you execute git revert it creates new commit id)

\*\*\*file content related changes can also be reverted

\*\*\* revert is also possible at remote repo (github ) but at branch level

If changes from another branch is creating problem then you can consider revert

(remove/undo the other branch changes) or when you have any pull request then you can consider git revert



—- how many roles are there in Github/gitlab/Bitbucket

owner/Admin → devops team ->

* repo creation/deletion/
* add/remove users
* User can run any command
* You can also push to the master branch directly
* Branches strategy can be defined

Maintainer -> leads/senior/scrum master/managers

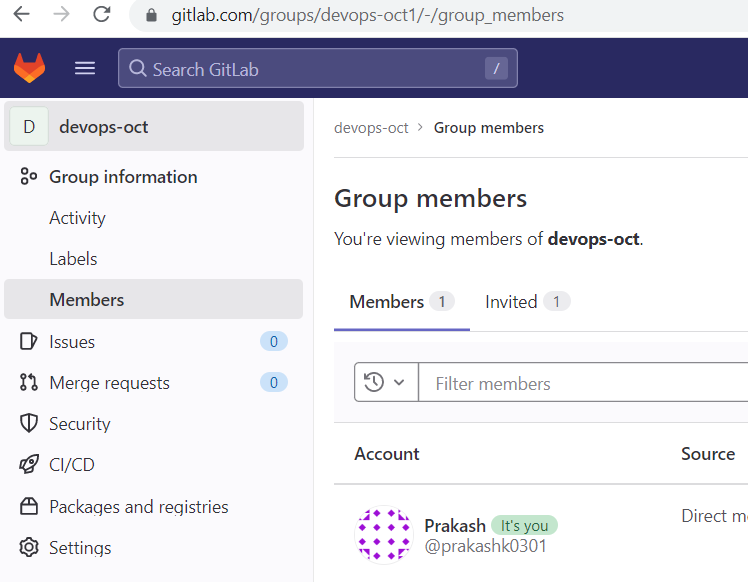
* Upload changes to the master/other branches
* reviewer/ approve merge/pull request
* Can’t create user/permission can’t be granted to other users

Developer -> junior/new team member

* Limited permission
* Can’t push to master branch
* Can upload to other branches

Reader-> the person who just want to read the code/who is not going to contribute

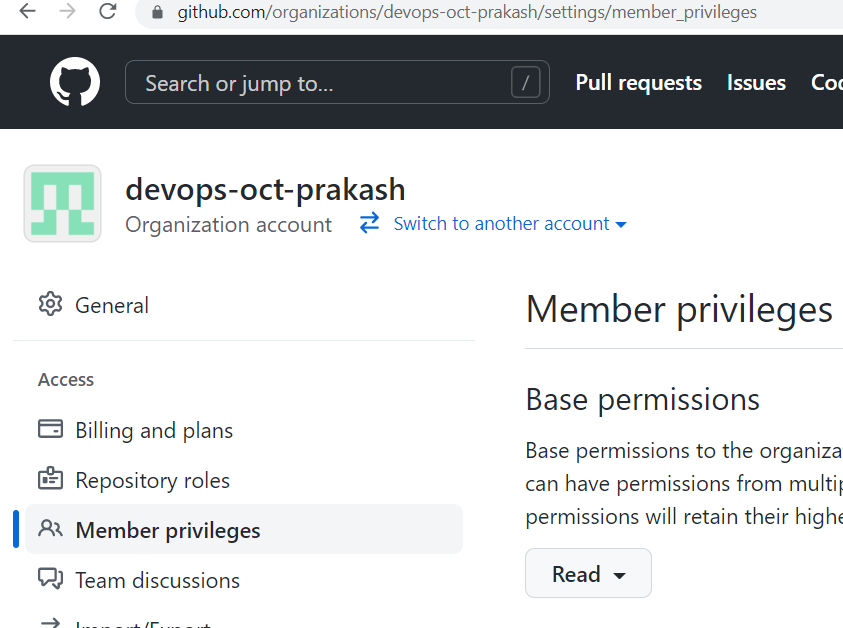
Guest ->



First create group -> create project -> add users to either group or project

—--Github : Admin, Write, Read, No Permission

Create org then project -> you can create teams -> you can add users to the team -> assign permission (member privilege) to the user



—-------for bitbucket: <https://confluence.atlassian.com/bitbucketserverkb/4-levels-of-bitbucket-server-permissions-779171636.html>

—-----git tag: you can tag your commit

Git tag <tag name>

Ex: git tag 1.0.0 —> (major , minor, bug fix/small changes)

Or

git tag 191122 —-> you can point to any commit ID

git tag —> list all the tag

git tag <tag name> —-> history of that tag

—-----------------Git Branching strategy: <https://www.flagship.io/git-branching-strategies/>

We follow [**GitFlow**](https://www.flagship.io/git-branching-strategies/#gitflow)branching strategy in our org

—--------------Best practices