File Discription: GitLab and Git
Name: Swapnil Chougule
Date: 01.10.2022

Contents

1. *Links*

Discription	Link
Linker scripts	https://users.informatik.haw-hamburg.de/~krabat/FH-Labor/gnupro/5_GNUPro_Utilities/c_Using_LD/ldLinker_scripts.html#Commands
Linker scripts	https://users.informatik.haw-hamburg.de/~krabat/FH-Labor/gnupro/5_GNUPro_Utilities/c_Using_LD/ldLinker_scripts.html#Commands
Linker scripts (Video)	https://www.google.com/search?q=linker+script+tutorial&oq=Linker+Script+&aqs=chrome.0.69i59j0j69i57j0l4j69i60.383j0j7&sourceid=chrome&ie=UTF-8#kpvalbx=_3U4_YP-nNZicUtfIqPgL32
STM32 IDE	https://www.st.com/en/development-tools/stm32-ides.html#overview
Linux Basic Commands	https://www.hostinger.com/tutorials/linux-commands
	https://www.youtube.com/watch?v=aw9wHbFTnAQ
GNU Make	https://www.gnu.org/software/make/manual/make.html#toc-How-to-Run-make
JTAG header	https://stm32-base.org/guides/connecting-your-debugger.html
Device setup video	https://www.youtube.com/watch?v=8TAffkgwVYk&t=183s
SEGGER J-Link tools	https://www.segger.com/downloads/jlink/JLink_Linux_x86_64.deb
Ozone	https://www.youtube.com/watch?v=8TAffkgwVYk&t=183s
GCC tool Chain setup	https://www.youtube.com/watch?v=imUiQkO9YHM
MCU interface	https://www.davidkebo.com/microcontroller-interfacing
Hex to Decimal	http://hextodecimal.com/index.php?hex=0F
Linux Commands	https://linuxconfig.org/linux-commands

1. GitLab & Git

- Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows
- Git is version control system to locally track changes in your project/folder and push & pull changes from remote repositories like GitHub, BitBucket, GitLab
- GitLab, GitHub, BitBucket are Services that allow to host your project on a remote repo & have additional features to help in SDLC and CI, CD e.g Managing Sharing Wiki Bug tracking CI & CD

User: **********
Pass: **********

1.1 GitLab installation steps

Step 1: Goto GitLab.com and create an account
Step 2: Sign in to GitLab
Step 3: Create a new project on GitLab

1.2 Git installation steps

Step 1: Download git from https://git-scm.com/ and install git
(# apt-get install git)
(git --version)
Step 2: Git global setup (run the following commands)
git config --global user.name
git config --global user.email
git config --global --list
Step 3: Create a demo project/folder & add to git
git init
git status
git add . // add all untracked/unchecked files to git
git add .file name
git commit -m “msg”
git push -u “url” master
git push -u “https://gitlab.com/swapnil_chougule/project_splitter” master
or
git push -u origin master

1.3 Fork

A fork is a copy of a project Forking a project/repository allows us to make changes without affecting the original project
Step 1: Login to GitLab and goto your project
Step 2: Click on Fork button

1.4 SSH key

- SSH - Secure Shell protocol
- Used for authentication
- By setting ssh key you can connect to GitLab server without using username and password each time

Step 1: Run command
ssh-keygen
- two files (id_rsa and id_rsa.pub) are generated
@ .file:///home/swapnil/.ssh/

Step 2: Login to GitLab Gotot account ＞ Preferences＞ SSH Keys
Step 3: Copy contents of id_rsa.pub

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDAXBaLyZRVWzYCmgf7sVsA05aFFjXg4
yhW+tfT435XO00jTirmaPxRxwJ2D4YFnHoHThLWHNiH72jV3o0I3m5Una0UvsKM
4X0LLFo11UG+kJNhwQTciVcVrtN5YYQmuXKEjVtaKGtKe2U+fTrfJj5H1okcfYJeaEP
UZzBT8S1efS25cg1f8kZmZLggbp26MwF080A0Aq9kLy5XJrgiim/
cpf9jbPLajD4VdA2Qj+hzZIF3wvQ0o61cvelU51WK1jC7lvLJJQkBxkzSpirQATA5nwj
yxa0vwhcHDG5+sTu+0VRlp2MvEGIbeEx6gh0WZsJNv73cvz+vxSTYQM61sKk7
swapnil@swapnil-HP-Z440
Step 4: Verify SSH key is added

1.5 GitLab Runner

GitLab Runner is an application that works with GitLab CI/CD to run jobs in a pipeline.

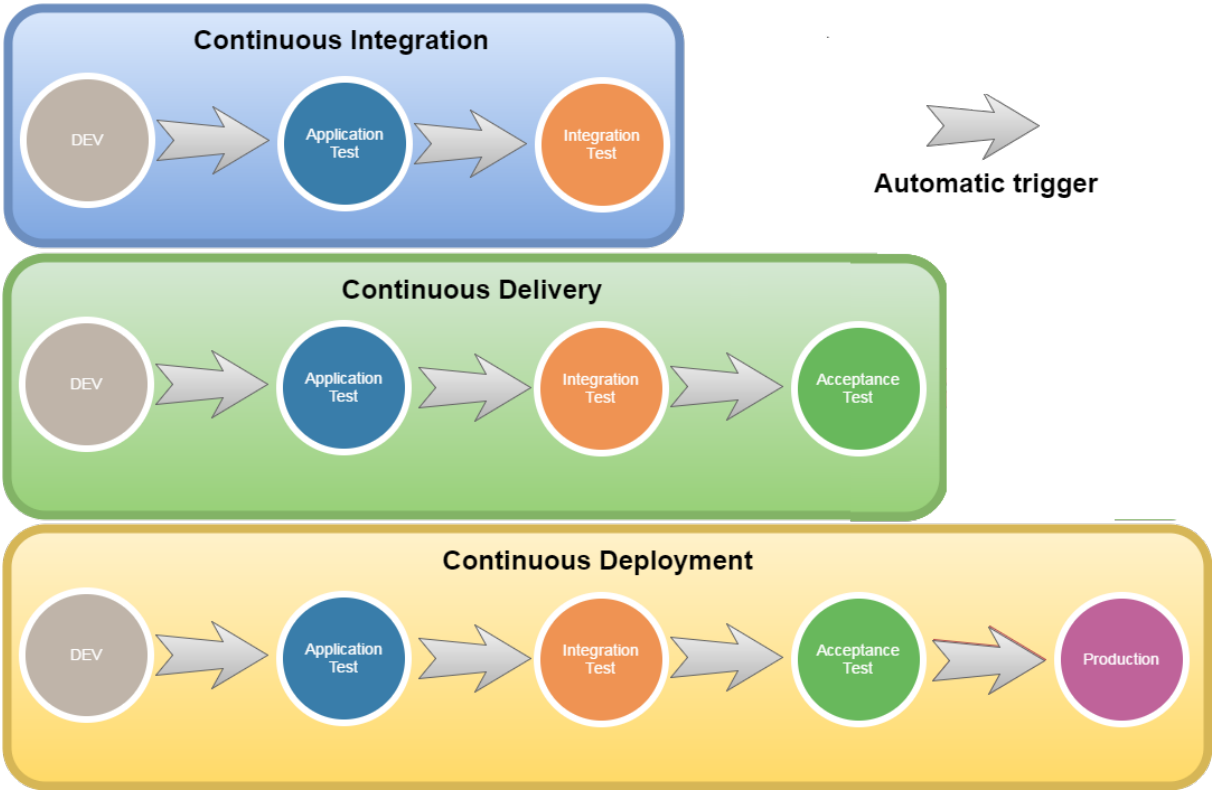
- Open-source continuous integration service included with GitLab
- used to run jobs & send results back to GitLab

Step 1: Install GitLab Runner
- https://docs.gitlab.com/runner/install/linux-manually.html#install

Step 2: Register GitLab Runner
https://docs.gitlab.com/runner/register/index.html

Step 3: Start GitLab Runner
sudo gitlab-runner start
sudo gitlab-runner stop
gitlab-runner --version

1.6 GitLab CI/CD



1. GitLab CI/CD
2. Create .gitlab-ci.yml
3. Run CI/CD pipeline GitLab CI is an open source CI service included with GitLab Since ver 8.0

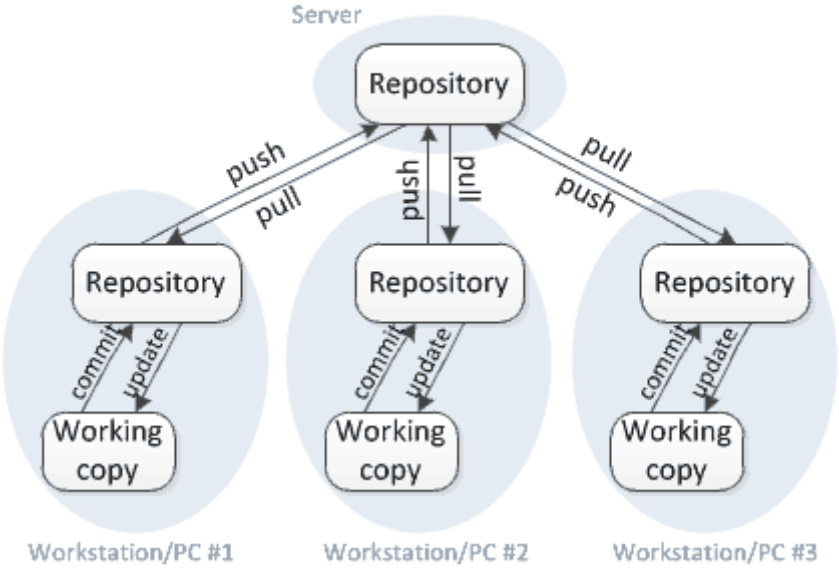
GitLab CI is an open source CI service included with GitLab Since ver 8.0 GitLab CI is an open source Continuous Integration service included with GitLab Only project maintainers & Admin can access the Settings.

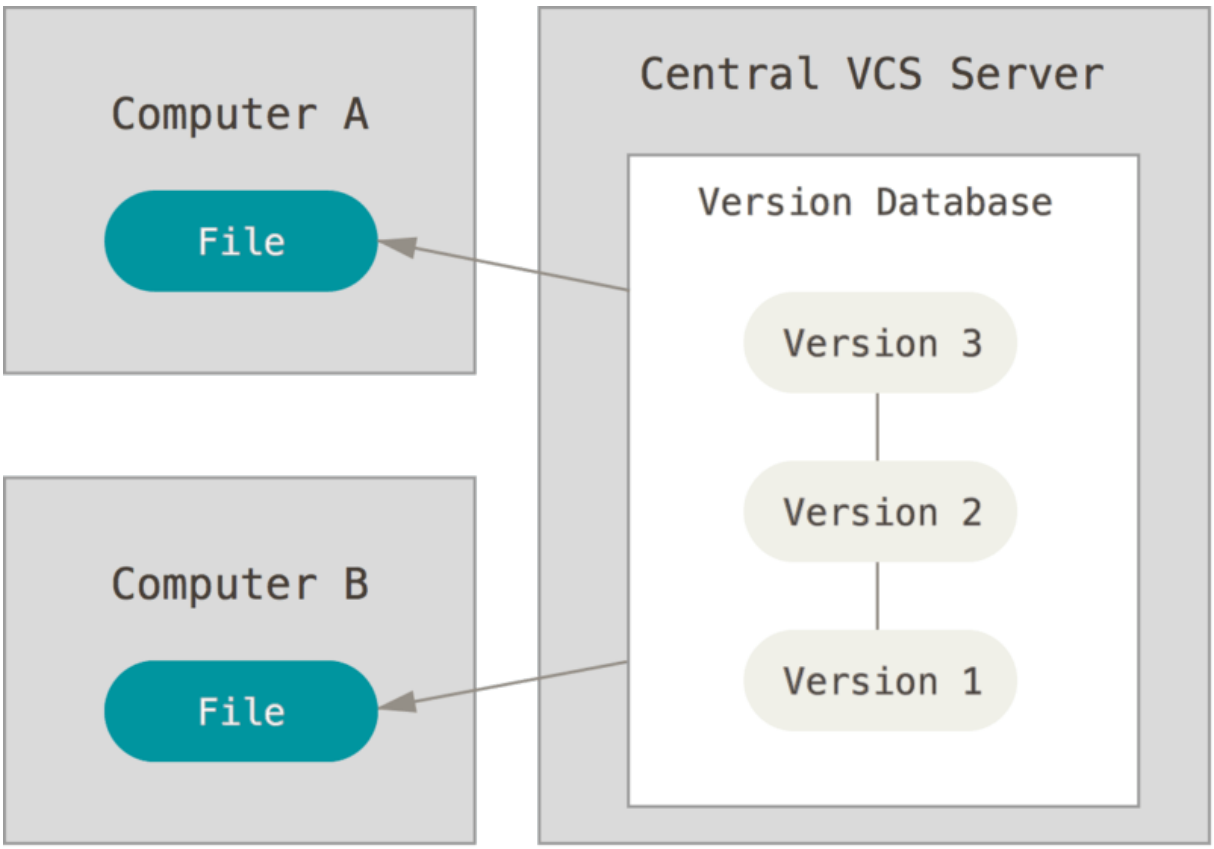
Step 1: Add .gitlab-ci.yml in the root folder of your project/repo
- GitLab CI/CD pipeline are configured using YAML file called .gitlab-ci.yml in each project
- .gitlab-ci.yml file defines the structure and order of the pipeline & determines

1.7 GIT

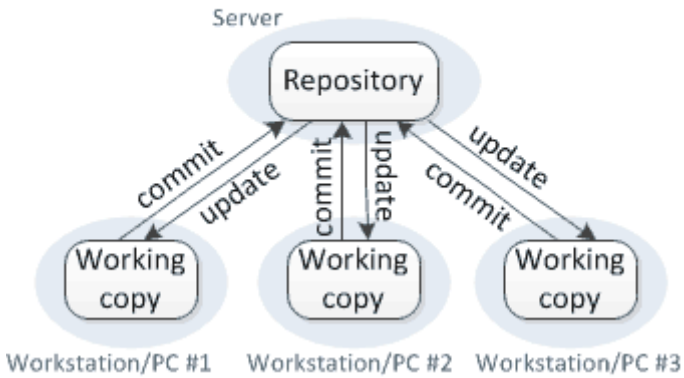
- VCS : version control system
- to track changes in files / folders
- to collaborate in teams
- free and open source
- GIT is Distributed VCS

Distributed version control





Centralized version control



1.7.1 *Git workflow*

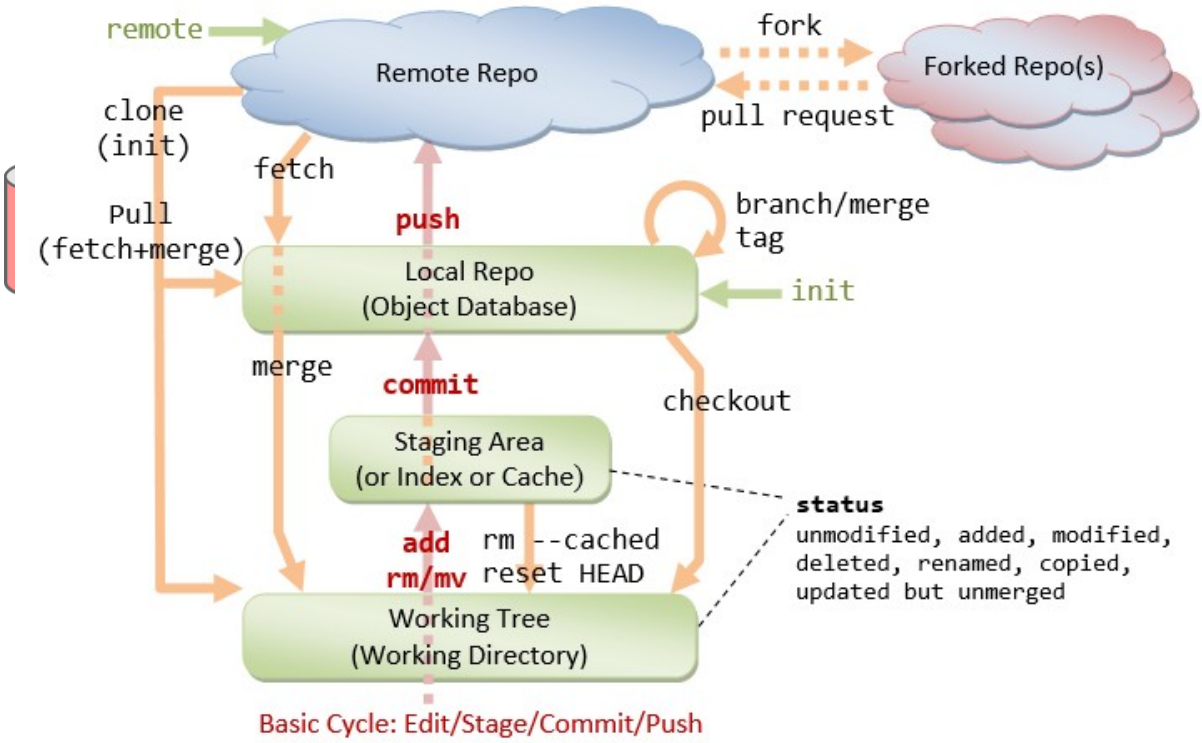


Figure 1: Git storage model

1.7.2 *Git installation:*

Step1: Start by updating the package index:
-sudo apt update

Step2: Run the following command to install Git:

-sudo apt install git

Step3: Verify the installation by typing the following command which will print the Git version:

-git --version

1.7.3 *Git Folder tracking*

Step 1 : Add file/folders to git - tracking
- goto the location of the folder/project
- git init
- git status
- git add
- git commit -m “.....”

- git remote add origin “location of remote repo”
- git push -u origin master
- git log
- git --help

1.7.4 *Git Branch*

Step 1 : Create branch
git branch “branch name”

Step 2 : Checkout branch
git checkout “branch name”
git checkout “Branch1” //Switched to branch 'Branch1'
git checkout “master” //Switched to branch 'master'

Step 3 : Merge new branch in master branch
Note: while merging to master first checkout master
git merge “branch name”
git merge “Branch1”

Step 4 : Delete branch
git branch -d “branch name” // delete from local
git branch -d “Branch1”
git push origin —delete “branch name” //delete from remote
git push origin —delete “Branch1”

Step 1 : Github - Repository - Settings - integration & services - add email
Step 2 : Test and validate by making some change in the project

1.7.5 *Git Tags*

Step 1: Checkout the branch where you want to create the tag
git checkout "branch name"
example : git checkout master

Step 2: Create tag with some name
git tag "tag name"
example : git tag v1.0
git tag -a v1.0 -m "ver 1 of proj_Splitter" // (to create annotated tags)

Step 3: Display or Show tags
git tag
git show v1.0
git tag -l “v1.*” // show all tags with v1.

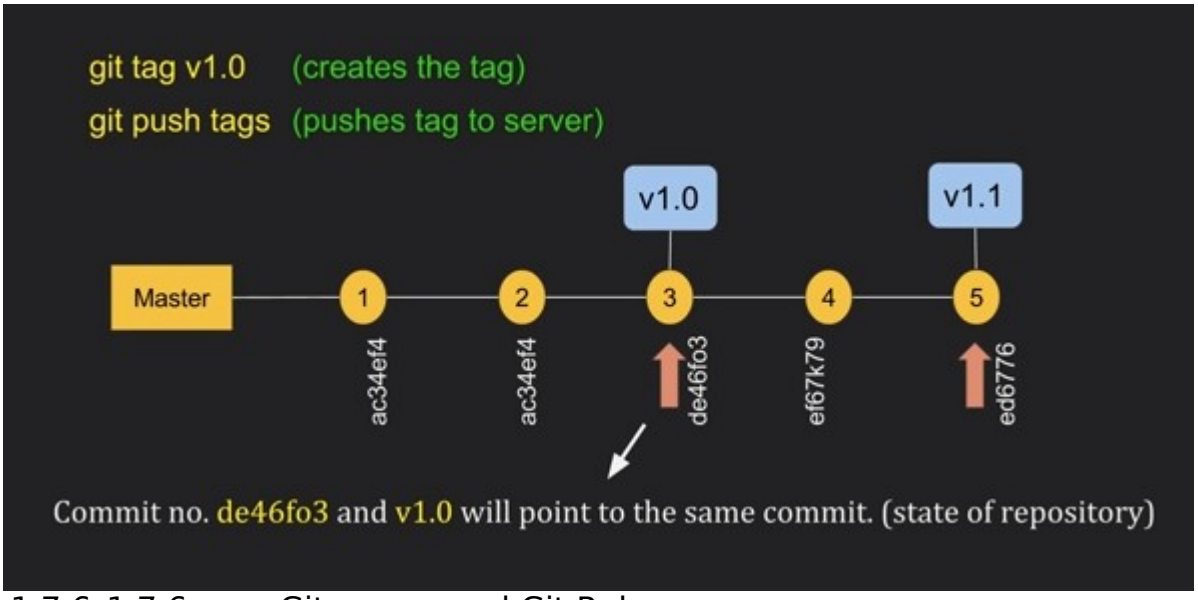
Step 4: Push tags to remote
git push origin v1.0
git push origin --tags

git push --tags // to push all tags at once

Step 5: Delete tags (if required only) to delete tags from local :
git tag -d v1.0
git tag --delete v1.0

- to delete tags from remote :
git push origin -d v1.0
git push origin --delete v1.0
git push origin :v1.0

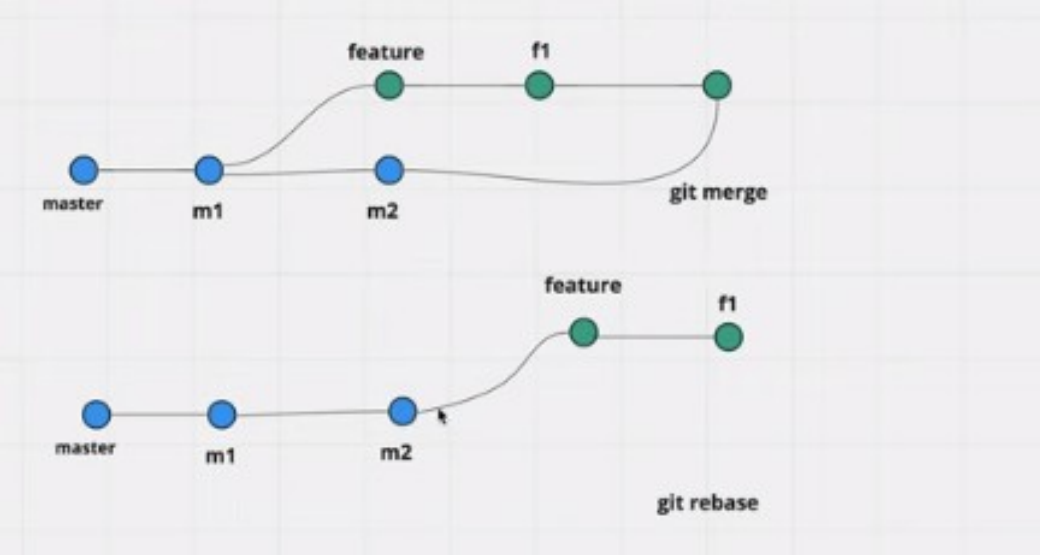
- to delete multiple tags at once:
git tag -d v1.0 v1.1 (local)
git push origin -d v1.0 v1.1 (remote)



1.7.6 1.7.6 Git merge and Git Rebase

git merge :
- its is a non-destructive operation
- Existing branches are not changed in any way
- Creates a new merge commit in the feature branch

git rebase:
- Moves the entire feature branch to begin on the tip of the master branch
- Re-writes the project history
- We get much cleaner and linear project history



Online course: https://automationstepbystep.com/online-courses/
Commands: https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html
Commands: https://www.git-tower.com/learn/git/commands/
Book: https://git-scm.com/book/en/v2
Documentation: https://git-scm.com/docs/
Basics:https://www3.ntu.edu.sg/home/ehchua/programming/howto/Git_HowTo.html
Commands: https://www.atlassian.com/git/tutorials/svn-to-git-prepping-your-team-migration

Commands:

git config --global user.name
git config --global user.email // shows user email ID
git config --global --list // shows user name and email ID
git init //This creates a new subdirectory named .git that contains all of our necessary repository files — a Git repository skeleton.
git status // determine which files are in which state
git add . // adds all the files to working directory
git add filename.txt // adds the specific file to working directory
git add *.txt // all files with extension .txt will be addad
git commit -m "Message : commit for project splitter" // Commit with a message
git push -u “https://gitlab.com/swapnil_chougule/project_splitter” master
or
git push -u origin master // Push your work on the remote repository
git --version
touch filename.txt // creat a text file in directory
git log
1git --help
git branch “branch name”

git checkout “branch name”
git checkout “Branch1” //Switched to branch 'Branch1'
git checkout “master” //Switched to branch 'master
Note: while merging to master first checkout master
git merge “branch name”
git merge “Branch1”
git branch -d “branch name” // delete from local
git branch -d “Branch1”
git push origin —delete “branch name” //delete from remote
git push origin —delete “Branch1”
git fetch origin

Git task	Notes	Git commands
Tell Git who you are	Configure the author name and email address to be used with your commits.Note that Git strips some characters (for example trailing periods) from user.name.	git config --global user.name "Sam Smith" git config --global user.email sam@example.com
Create a new local repository		git init
Check out a repository	Create a working copy of a local repository:	git clone /path/to/repository
	For a remote server, use:	git clone username@host:/path/to/repository
Add files	Add one or more files to staging (index):	git add
Commit	Commit changes to head (but not yet to the remote repository):	git commit -m "Commit message"
	Commit any files you've added with git add, and also commit any files you've changed since then:	git commit -a
Push	Send changes to the master branch of your remote repository:	git push origin master
Status	List the files you've changed and those you still need to add or commit:	git status
Connect to a remote repository	If you haven't connected your local repository to a remote server, add the server to be able to push to it:	git remote add origin
	List all currently configured remote repositories:	git remote -v
Branches	Create a new branch and switch to it:	git checkout -b
	Switch from one branch to another:	git checkout

Git task	Notes	Git commands
	List all the branches in your repo, and also tell you what branch you're currently in:	git branch
	Delete the feature branch:	git branch -d
	Push the branch to your remote repository, so others can use it:	git push origin
	Push all branches to your remote repository:	git push --all origin
	Delete a branch on your remote repository:	git push origin :
Update from the remote repository	Fetch and merge changes on the remote server to your working directory:	git pull
	To merge a different branch into your active branch:	git merge
	View all the merge conflicts:View the conflicts against the base file:Preview changes, before merging:	git diff git diff --base git diff
	After you have manually resolved any conflicts, you mark the changed file:	git add
Tags	You can use tagging to mark a significant changeset, such as a release:	git tag 1.0.0
	CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using:	git log
	Push all tags to remote repository:	git push --tags origin
Undo local changes	If you mess up, you can replace the changes in your working tree with the last content in head:Changes already added to the index, as well as new files, will be kept.	git checkout --
	Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this:	git fetch origin git reset --hard origin/master
Search	Search the working directory for foo():	git grep "foo()"

Table link: https://www.atlassian.com/git/tutorials/svn-to-git-prepping-your-team-migration