pralee
Technology & Games

Git

BASIC COMMANDS
FOR BEGINNERS

/praleetechnology

/praleetechnology

/@praleetechnology

/company/praleetechnology

www.praleetechnology.com

Git Installation

WINDOWS

Go To https://git-scm.com

Download For Windows
And Execute
Downloaded File

Follow The Instructions &
Click On Next

Install Git With Default
Options For Now

LINUX

```
sudo apt-get update
sudo apt-get install git
```

Open Terminal and
Run Commands

MAC OS

Install Homebrew
(if you don't have)

Run Command

```
brew install git
```

Configure & Init

CONFIGURATION

To commit (saving your work) you have to configure your name and email

```
git config --global user.name “Your Name”
git config --global user.email “your@email.com”
```

```
pralee@dheeraj-ubuntu:~$ git config --global user.name "Dheeraj Sapkale"
pralee@dheeraj-ubuntu:~$ git config --global user.email "dsapkale@gmail.com"
pralee@dheeraj-ubuntu:~$
```

INIT (CREATING LOCAL REPOSITORY)

```
git init
```

Creates local repository in the current directory

```
git init project-1
```

Creates a new directory => **project-1** and creates a local repository in it

CLONE (CREATE FROM GITHUB/REMOTE URL)

```
git clone https://github.com/dsapkale/dcapp.git
```

Will fetch repository from given **URL** and create a local repository in a folder with the same name of repository name in the current directory

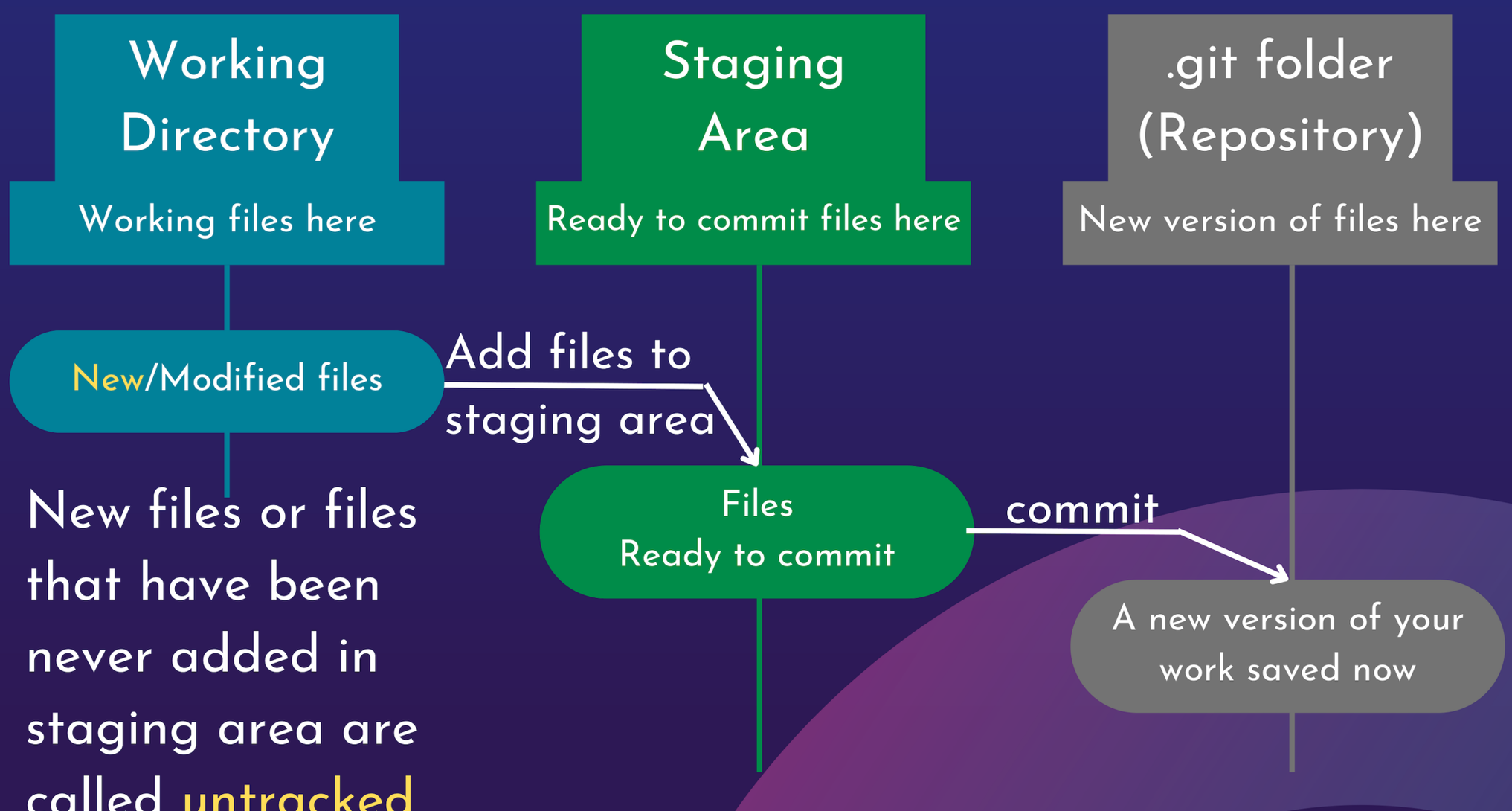
```
git clone https://github.com/dsapkale/dcapp.git myapp
```

Will fetch repository from given **URL** and create a local repository in a folder named myapp

Add & Save

CONCEPT

After init or clone, you will have a hidden folder named **.git** which is your actual repository that maintains your changes



COMMANDS

`git add Hello.java` Adding **file** to the staging area

```
pralee@dheeraj-ubuntu:~/learn-git$ git add Hello.java
pralee@dheeraj-ubuntu:~/learn-git$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Hello.java

pralee@dheeraj-ubuntu:~/learn-git$
```

Added into the Staging area & ready to Save/Commit

`git status` Shows the status of untracked/modified/staged files for your next commit in current directory

Add & Save

```
git commit -m "hello world created"
```

Used to commit/save your staged files as a new snapshot(new version) with a **descriptive message.** The message helps you to identify this version


```
git diff
```

Shows changes of current work but not staged

Shows changes of staged but not commited

```
git diff --staged
```

No output will be shown if there are no changes made in any of the areas you are looking for

Add Options

`git add *.java` To add all files which are of **.java** extension

All commands are used to add/stage all (new/modified/deleted) files

```
git add .
git add -A
git add --all
```

```
git add -u
git add --update
```

Both commands are used to add/stage modified and deleted files only

To add/stage new and modified files only

`git add . --ignore-removal`

```
git commit -a
git commit --all
```

To automatically add/stage modified and deleted files and commit

e.g.

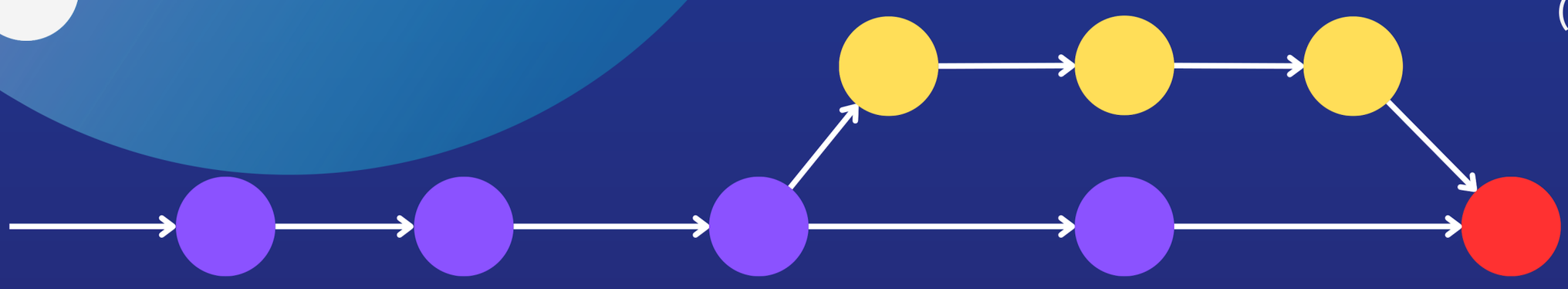
`git commit -a -m “your message”`

`git commit --all -m “your message”`

`git commit -am “your message”`

All commands will automatically add/stage files that have been modified and deleted and then commit

Branch (Local)



Using branches you can work independently without disturbing the main work and later you can merge it into the main work

`git branch` Shows existing branches of local repository

Creates a new branch named **feature-1** `git branch feature-1`

Branch Name

```
pralee@dheeraj-ubuntu: ~/branches
pralee@dheeraj-ubuntu:~/branches$ git branch
* master
pralee@dheeraj-ubuntu:~/branches$ git branch feature-1
pralee@dheeraj-ubuntu:~/branches$ git branch
  feature-1
* master
pralee@dheeraj-ubuntu:~/branches$
```

Current Branch

* master shows that you are currently working in the master branch

Both commands are used to change working **branch**

```
git switch feature-1
git checkout feature-1
```

switch is a branch specific command introduced in Git v2.23

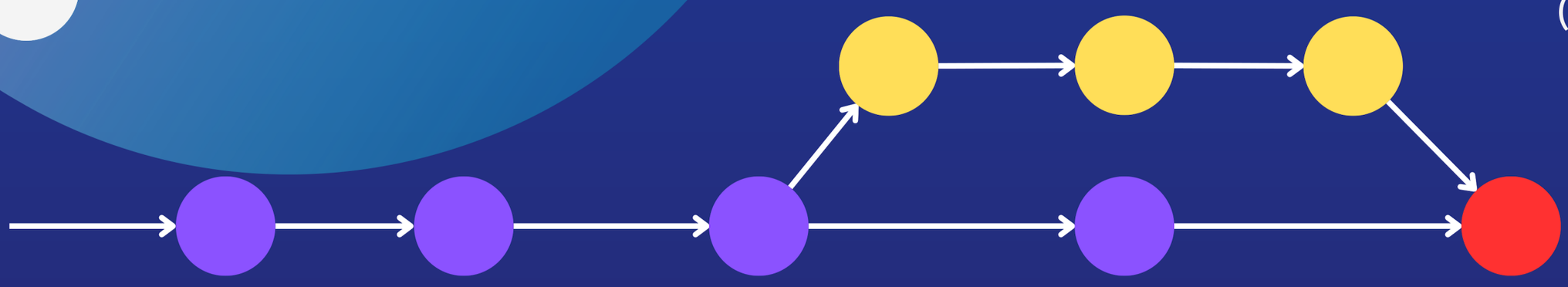
checkout is a multi-purpose command

```
pralee@dheeraj-ubuntu: ~/branches
pralee@dheeraj-ubuntu:~/branches$ git switch feature-1
Switched to branch 'feature-1'
pralee@dheeraj-ubuntu:~/branches$ git branch
* feature-1
  master
pralee@dheeraj-ubuntu:~/branches$
```

Current Branch

Switched into **feature-1** branch using **switch** command

Branch
(Local)



```
pralee@dheeraj-ubuntu: ~/branches
pralee@dheeraj-ubuntu:~/branches$ git checkout master
Switched to branch 'master'
pralee@dheeraj-ubuntu:~/branches$ git branch
  feature-1
* master
pralee@dheeraj-ubuntu:~/branches$
```

Switched into the **master** branch using **checkout** command

```
git switch -c feature-2
```

Create and switch into **feature-2 branch**

Create and switch into **feature-3 branch**

```
git checkout -b feature-3
```

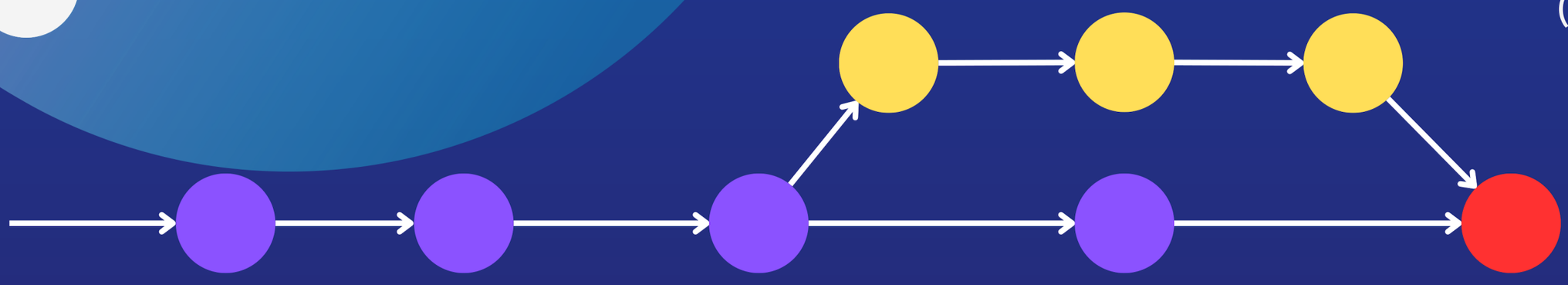
All commands that have been used to create a branch, create a new branch from the latest commit of the currently selected branch

old new

```
git  branch -m feature-1 feature-one
git  branch --move feature-2 feature-two
```

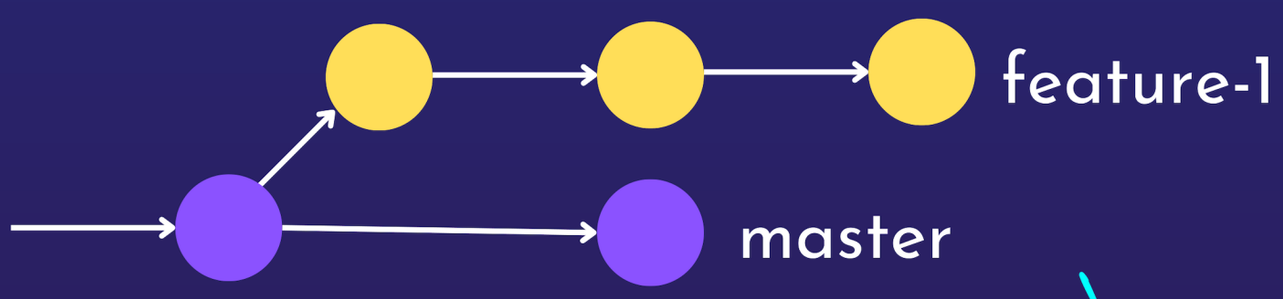
Both commands are used to rename the branch from **old** to **new** name

Branch
(Local)

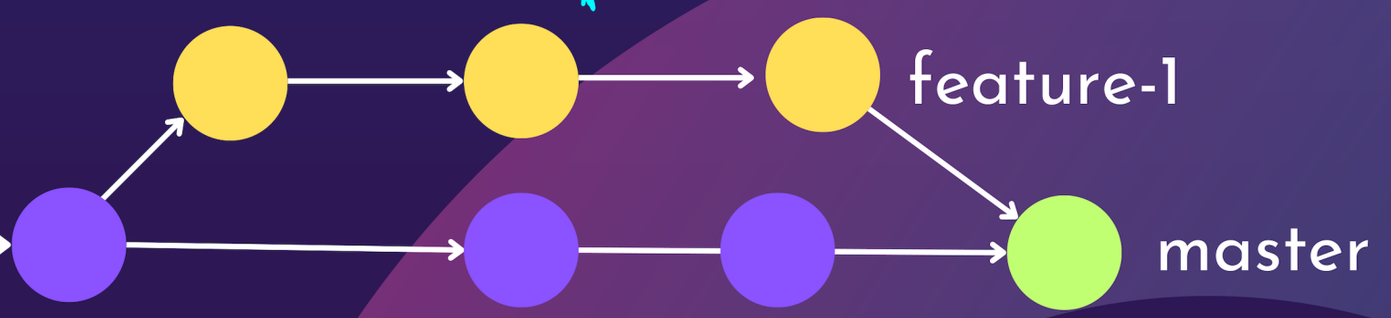


```
git merge feature-1
```

Merges the specified **branch** into current branch



here current branch is master



all commands delete the given **branch** of local repository

```
git branch -d feature-1
git branch --delete dev
git branch -D feature-2
```

-d => delete if it has already been merged

--delete => works same as -d works

-D => delete irrespective of its merged status

You will receive an error if you try to delete the current branch

```
pralee@dheeraj-ubuntu:~/java$ git branch
* feature-1
  master
pralee@dheeraj-ubuntu:~/java$ git branch -d feature-1
error: Cannot delete branch 'feature-1' checked out at '/home/pralee/java'
pralee@dheeraj-ubuntu:~/java$
```

Current Branch

View History

git log

Shows commit history of current branch

```
pralee@dheeraj-ubuntu:~/branches$ git log
commit 3f5d088da3d6028a1a439faf9bb7591b0d573c8a (HEAD -> master)
Author: Dheeraj Sapkale <dsapkale@gmail.com>
Date:   Thu Sep 14 15:57:27 2023 +0530

    form updated

commit 60aad80a8fae7202f8b3af18f9bf996d652f0b06
Author: Dheeraj Sapkale <dsapkale@gmail.com>
Date:   Thu Sep 14 15:06:29 2023 +0530

    login form created

commit 8a44960649090ff823d6e3836a09d1abdf5395ed
Author: Dheeraj Sapkale <dsapkale@gmail.com>
Date:   Thu Sep 14 15:04:33 2023 +0530

    readme created
```

Commit ID

git log -2

Shows last **n (count)** commits of current branch

```
pralee@dheeraj-ubuntu:~/branches$ git log -2
commit 3f5d088da3d6028a1a439faf9bb7591b0d573c8a (HEAD -> master)
Author: Dheeraj Sapkale <dsapkale@gmail.com>
Date:   Thu Sep 14 15:57:27 2023 +0530

    form updated

commit 60aad80a8fae7202f8b3af18f9bf996d652f0b06
Author: Dheeraj Sapkale <dsapkale@gmail.com>
Date:   Thu Sep 14 15:06:29 2023 +0530

    login form created
pralee@dheeraj-ubuntu:~/branches$
```

git log --oneline

Shows all commits
1 commit details
in one line

```
pralee@dheeraj-ubuntu:~/branches$ git log --oneline
3f5d088 (HEAD -> master) form updated
60aad80 login form created
8a44960 readme created
a26bfde initial commit
pralee@dheeraj-ubuntu:~/branches$
```

hash code

Short form of Commit ID

git log -2 --oneline

Shows last two commits, **in one line**

git log --since=5.days

Shows all commits of last 5 days, you can use weeks, months, years and more...

git log --graph --oneline

Shows all commits/branches with graphical (* & lines) representation, **in one line**

If commits are out of screen view => q to exit, enter to continue

View History

```
git log -p
git log --patch
```

Both commands show the commits along with diff information

```
pralee@dheeraj-ubuntu:~/java$ git log --patch
commit 588eea2d4103c53b8932247df82bf3b797a4cb78 (HEAD -> master)
Author: Dheeraj Sapkale <dsapkale@gmail.com>
Date:   Thu Sep 14 18:44:28 2023 +0530

    typo corrected

diff --git a/Hello.java b/Hello.java
index 0fd05d0..c7d6934 100644
--- a/Hello.java
+++ b/Hello.java
@@ -1,5 +1,5 @@
 class Hello{
     public static void main(String args[]){
-        System.out.println("Hello Pralee");
+        System.out.println("Hello Pralee");
     }
 }
```

```
pralee@dheeraj-ubuntu:~/java$ git log --stat
commit 3a6658e887ca4abe55bad7afa10b7af8756e8d17 (HEAD -> master)
Author: Dheeraj Sapkale <dsapkale@gmail.com>
Date:   Thu Sep 14 19:03:36 2023 +0530

    addition class created

 Addition.java | 9 +++++++++
 1 file changed, 9 insertions(+)

commit 588eea2d4103c53b8932247df82bf3b797a4cb78
Author: Dheeraj Sapkale <dsapkale@gmail.com>
Date:   Thu Sep 14 18:44:28 2023 +0530

    typo corrected

 Hello.java | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
git log --stat
```

Shows the info of every file in the commit and what is modified, `+++` for insertions and `----` for deletions

```
git log feature-1..
```

Shows commits that are present on current branch and not on/merged into **feature-1** branch

```
pralee@dheeraj-ubuntu:~/java$ git log feature-1..
commit fb84d79fb570d120000cde100d78c3689ac63820 (HEAD -> master)
Author: Dheeraj Sapkale <dsapkale@gmail.com>
Date:   Thu Sep 14 19:21:47 2023 +0530

    add method modified for 3 inputs
pralee@dheeraj-ubuntu:~/java$ git log ..feature-1
commit 86678aa9cfd222f3cdc9f3712e929203eced3752 (feature-1)
Author: Dheeraj Sapkale <dsapkale@gmail.com>
Date:   Thu Sep 14 19:32:41 2023 +0530

    message removed from addition
pralee@dheeraj-ubuntu:~/java$
```

```
git log ..feature-1
```

Shows commits that are present on branch **feature-1** and not on/merged into current branch

```
pralee@dheeraj-ubuntu:~/java$ git log --oneline
fb84d79 (HEAD -> master) add method modified for 3 inputs
3a6658e addition class created
588eea2 typo corrected
babe319 hello world created
pralee@dheeraj-ubuntu:~/java$
```

Present on master but not on feature-1

```
pralee@dheeraj-ubuntu:~/java$ git log --oneline
86678aa (HEAD -> feature-1) message removed from addition
3a6658e addition class created
588eea2 typo corrected
babe319 hello world created
pralee@dheeraj-ubuntu:~/java$
```

Present on feature-1 but not on master

```
git reflog
```

Shows all the operations made on your repository

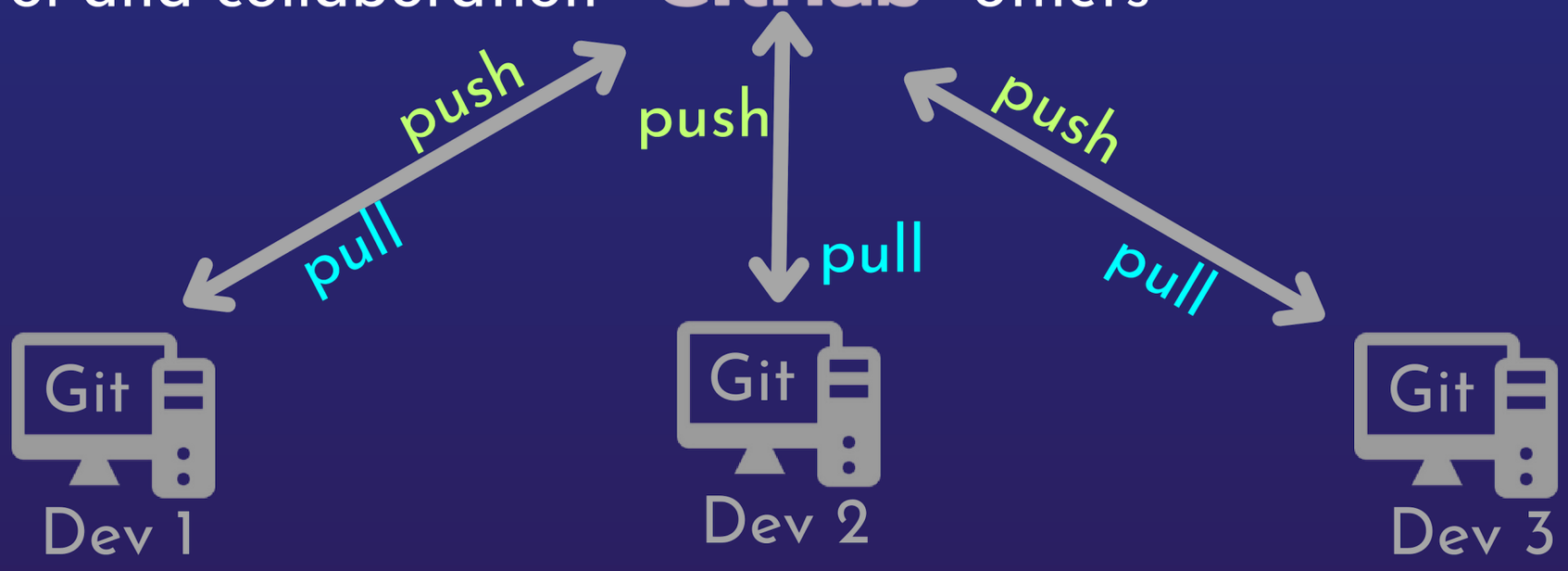
Remote/GitHub

Now we are ready to put our work on remote server/GitHub

CONCEPT

A web-based code hosting platform for version control and collaboration

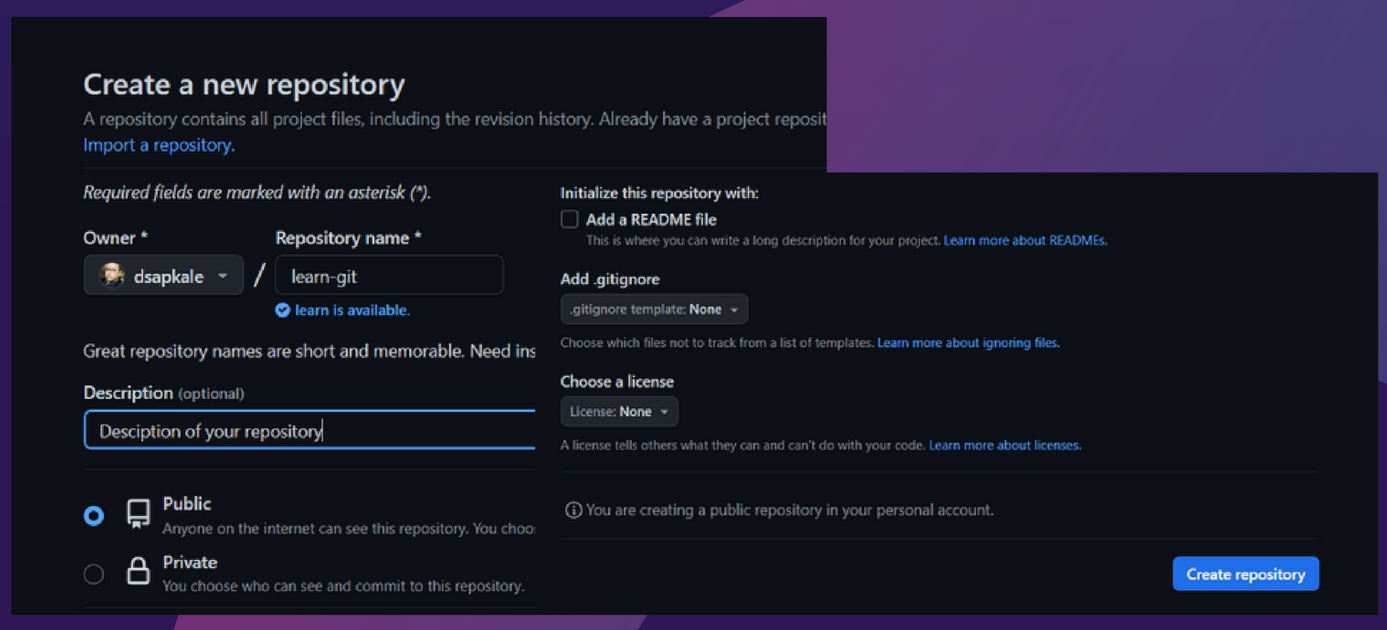
It allows to store and share repositories with others



CREATE REMOTE/GITHUB REPOSITORY

Go to https://github.com/

Login and create a new repository



Now you got the **URL** of your remote repository just add it

ADD REMOTE TO LOCAL REPOSITORY

```
git remote add origin https://github.com/dsapkale/learn-git.git
```

(origin: Remote Name; https://github.com/dsapkale/learn-git.git: URL)

No need to add, if you have created repo using `git clone url`

```
pralee@dheeraj-ubuntu:~/java$ git remote
origin
pralee@dheeraj-ubuntu:~/java$
```

`git remote`

Shows all remote names/aliases

Remote/GitHub

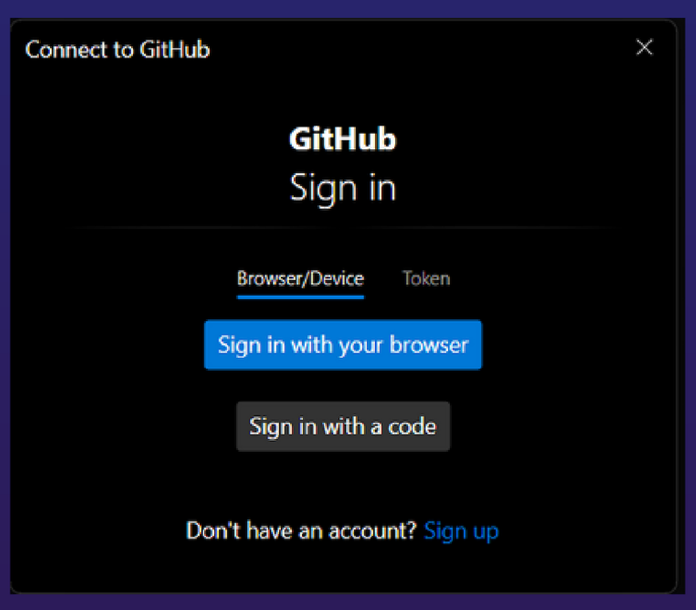
PUSH YOUR LOCAL REPOSITORY TO REMOTE

`git push -u origin master`

upstream

Will push/send your local repository into the remote repository

-u or --set-upstream used to set default remote branch for local branch and track it. Just use `git push` next time



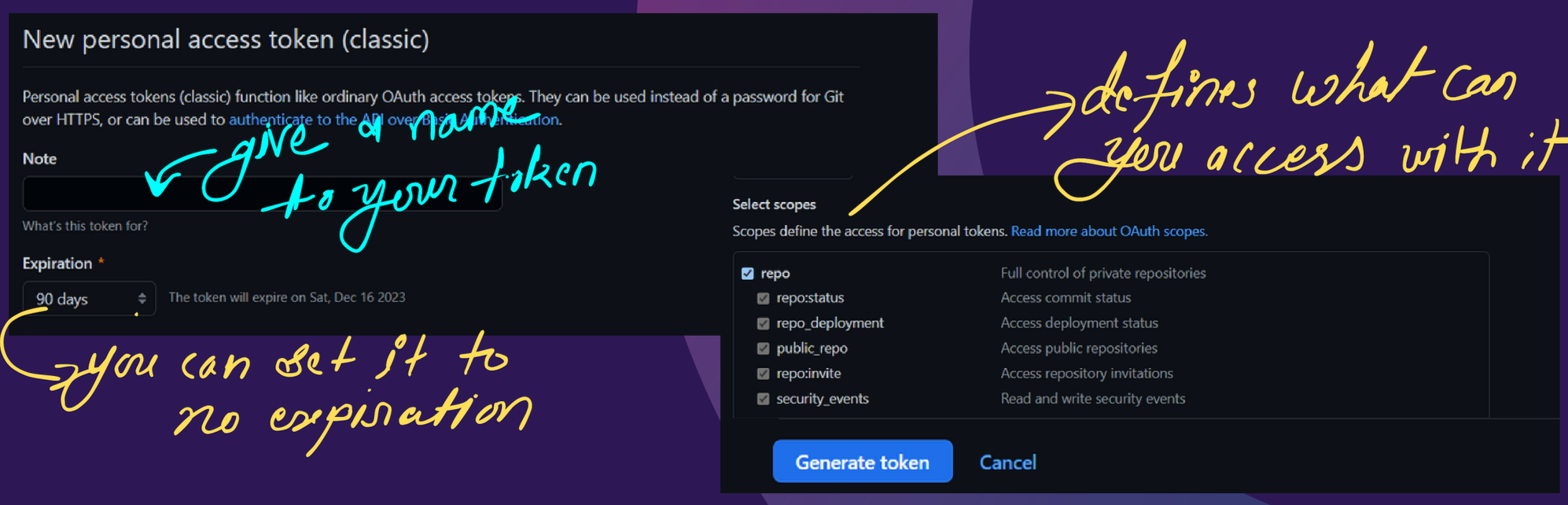
When you push first time you need to sign in with GitHub credentials

Windows OS will pop up sign in dialog just sign in with your browser

For Linux & Mac OS use token

```
pralee@dheeraj-ubuntu: ~/java
pralee@dheeraj-ubuntu:~/java$ git push -u origin master
Username for 'https://github.com': dsapkale
Password for 'https://dsapkale@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-g
it/about-remote-repositories#cloning-with-https-urls for information on currentl
y recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/dsapkale/learn-git.git/'
pralee@dheeraj-ubuntu:~/java$
```

Go to GitHub settings => developer settings => personal access tokens => Tokens(classic) and generate new token



Copy the generated token(it shows only once), and push again
Remember, you have to paste the token when the terminal asks for the password run `git config --global credential.helper store` before push otherwise git ask username & password every time

Sync Repository

`git fetch` fetch down all the information from git remote

fetch down all the info from specified remote name/alias `git fetch origin`

`git fetch -v` fetch down all the information from git remote
-v or --verbose => shows details what fetched

fetch changes and merge with upstream branch `git pull`

`git push -u origin feature-1` Push local branch to remote and set it as an upstream

Push local changes to the remote `git push`

if your current branch has no upstream then you have to push it by specifying the remote and branch

`git push origin feature-2` Push local branch to the remote origin

SYNC YOUR REMOTE BRANCH

`git branch -r`

`git branch -a`

`git branch -v -a`

```
pralee@dheeraj-ubuntu:~/java$ git branch
* master
pralee@dheeraj-ubuntu:~/java$ git branch -r
  origin/feature-1
  origin/master
pralee@dheeraj-ubuntu:~/java$ git branch -a
* master
  remotes/origin/feature-1
  remotes/origin/master
pralee@dheeraj-ubuntu:~/java$ git branch -v -a
* master                  fb84d79 [behind 2] add method modified for 3 inputs
  remotes/origin/feature-1 8c93d5d testing
  remotes/origin/master    8c93d5d testing
pralee@dheeraj-ubuntu:~/java$
```

-r => view remote branches, -a=> view all remote & local branch
-v => verbose mode => view with commit details

Sync Repository

local branch

remote branch

```
git checkout -b feature-1 origin/feature-1
```

From Git v2.23

```
git switch -c feature-1 origin/feature-1
```

Both commands create a local branch feature-1 and set up to track remote branch origin/feature-1

-vv => shows hash, commit message and tracking remote branch

```
pralee@dheeraj-ubuntu:~/java$ git switch -c feature-1 origin/feature-1
branch 'feature-1' set up to track 'origin/feature-1'.
Switched to a new branch 'feature-1'
pralee@dheeraj-ubuntu:~/java$ git branch -vv
* feature-1 8c93d5d [origin/feature-1] testing
  master    fb84d79 [origin/master: behind 2] add method modified for 3
pralee@dheeraj-ubuntu:~/java$
```

tracking remote/upstream branch

```
git push origin feature-1 --delete
```

```
git push origin :feature-1
```

Both commands delete remote branch feature-1
Remote rejects the operation, if you are trying to delete remote default/base branch

```
git merge origin/feature-1
```

to merge remote branch feature-1 into current local branch

```
git merge feature-2 origin/master
```

to merge remote branch master into the existing local branch feature-2

Back To Changes

`git checkout Hello.java`

Discard the changes of the given **file** in the working area

`git checkout fb84d79 Hello.java`

to get back to the version of the given **file** at the specified **commit hash**

to view hash, run
git log --oneline

```
pralee@dheeraj-ubuntu:~/java$ git log --oneline
359d015 (HEAD -> master, newb, feature-1) before rebase
ef96d2c hello changed for checkout
a9e75ec (origin/newb) Update Hello.java
8c93d5d (origin/feature-1) testing
f83456d Update Hello.java
fb84d79 add method modified for 3 inputs
3a6658e addition class created
588eea2 typo corrected
babe319 hello world created
```

`git checkout fb84d79`

checkout all the files to the specified commit hash

Now you can experiment with your work without affecting any of your working branch

```
pralee@dheeraj-ubuntu:~/java$ git checkout fb84d79
Note: switching to 'fb84d79'.

You are in 'detached HEAD' state. You can look arou
pralee@dheeraj-ubuntu:~/java$ git branch
* (HEAD detached at fb84d79)
  feature-1
  master
  newb
pralee@dheeraj-ubuntu:~/java$
```

```
pralee@dheeraj-ubuntu:~/java$ git checkout master
M       Hello.java
Previous HEAD position was fb84d79 add method modified for 3 inputs
Switched to branch 'master'
pralee@dheeraj-ubuntu:~/java$ git branch
  feature-1
* master
  newb
pralee@dheeraj-ubuntu:~/java$
```

Switch back to the master branch

run git checkout branch-name from where you were detached to get back to the last working HEAD

Remove, Move

```
pralee@dheeraj-ubuntu:~/java$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Hello.java

pralee@dheeraj-ubuntu:~/java$ git rm Hello.java --cached
rm 'Hello.java'
pralee@dheeraj-ubuntu:~/java$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    Hello.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Hello.java
```

deleted but present in working directory.

`git rm Hello.java --cached`

to delete file from staging area not from working area

deleting from staging area = unstage

to unstage you can also use `git restore Hello.java --staged`

`git rm Hello.java -f`

to delete a file from working directory and stage it

```
pralee@dheeraj-ubuntu:~/java$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Hello.java

pralee@dheeraj-ubuntu:~/java$ git rm Hello.java -f
rm 'Hello.java'
pralee@dheeraj-ubuntu:~/java$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    Hello.java

pralee@dheeraj-ubuntu:~/java$ ls
Addition.java
```

deleted from Working directory

only 1 file exists.

`git mv old_name new_name` to rename the file

e.g. => git mv **Addition.java Calc.java**

renamed and ready to commit changes

```
pralee@dheeraj-ubuntu:~/java$ ls
Addition.java  Hello.java
pralee@dheeraj-ubuntu:~/java$ git mv Addition.java Calc.java
pralee@dheeraj-ubuntu:~/java$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    Addition.java -> Calc.java
        new file:   Hello.java

pralee@dheeraj-ubuntu:~/java$
```

mv => move => b/c it changes an existing file path and stages the move

Rewrite History

`git commit --amend -m "message"`

Overrides last commit with new message

```
pralee@dheeraj-ubuntu: ~/react
pralee@dheeraj-ubuntu:~/react$ git log --oneline
c234ecf (HEAD -> master) initial commit
pralee@dheeraj-ubuntu:~/react$ git commit --amend -m "react project initialized"
[master f7fafcf] react project initialized
 Date: Mon Sep 18 10:58:33 2023 +0530
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 App.jsx
pralee@dheeraj-ubuntu:~/react$ git log --oneline
f7fafcf (HEAD -> master) react project initialized
pralee@dheeraj-ubuntu:~/react$
```

Overridden

`git reset d317678`

resets your work to the given hash and puts your current files(changes) in the working area

```
pralee@dheeraj-ubuntu: ~/react
pralee@dheeraj-ubuntu:~/react$ git log --oneline
3baf321 (HEAD -> master) signup component created
eada7ff login component created
d317678 index page added
f7fafcf react project initialized
pralee@dheeraj-ubuntu:~/react$ git reset d317678
pralee@dheeraj-ubuntu:~/react$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be
        Loging.jsx
        Signup.jsx

nothing added to commit but untracked files present (u
pralee@dheeraj-ubuntu:~/react$ git log --oneline
d317678 (HEAD -> master) index page added
f7fafcf react project initialized
pralee@dheeraj-ubuntu:~/react$
```

in working directory

all Commits have been gone, made after this commit

`git reset d317678 --hard`

if you don't want current files(changes) in the working area

`git revert d317678`

reverts changes from the specified commit hash and puts it in a new commit

```
pralee@dheeraj-ubuntu: ~/react
pralee@dheeraj-ubuntu:~/react$ git log --oneline
e99ea34 (HEAD -> master) login signup added after reset
d317678 index page added
f7fafcf react project initialized
pralee@dheeraj-ubuntu:~/react$ git revert d317678
[master b8ca86f] Revert "index page added"
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 Index.html
pralee@dheeraj-ubuntu:~/react$ git log --oneline
b8ca86f (HEAD -> master) Revert "index page added"
e99ea34 login signup added after reset
d317678 index page added
f7fafcf react project initialized
pralee@dheeraj-ubuntu:~/react$
```

One extra Commit is created with specific commit changes

.gitignore

A file named .gitignore in your working/project folder is used to tell git which files to ignore when committing a project to GitHub

Some patterns to ignore files

```
logs
build
node_modules
dist
```

ignore the folder and everything inside it that is named with the given text

ignore all .txt files present anywhere in the project

```
*.txt
```

```
/*.doc
```

ignore all .doc files present in the root directory

ignore all .cpp files present in lib folder(present anywhere)

```
**/lib/*.cpp
```

```
/lib/**/old
```

ignore the folder named old and its content if old is anywhere inside the folder lib and lib is present in the root directory

ignore the folder named old and its content if old is anywhere inside the folder lib and lib is present anywhere

```
**/lib/**/old
```

Tagging

Tagging is generally used to capture a point in history that is used for a marked version release.

```
pralee@dheeraj-ubuntu: ~/react
pralee@dheeraj-ubuntu:~/react$ git log --oneline
b8ca86f (HEAD -> master) Revert "index page added"
e99ea34 login signup added after reset
d317678 index page added
f7fafcf react project initialized
pralee@dheeraj-ubuntu:~/react$ git tag v2.0
pralee@dheeraj-ubuntu:~/react$ git log --oneline
b8ca86f (HEAD -> master, tag: v2.0) Revert "index page added"
e99ea34 login signup added after reset
d317678 index page added
f7fafcf react project initialized
```

tag created

`git tag v2.0`

to assign a tag (version) to the latest commit

`git tag v1.0 d317678` Assign a tag to the specified commit

```
pralee@dheeraj-ubuntu:~/react$ git tag v1.0 d317678
pralee@dheeraj-ubuntu:~/react$ git log --oneline
b8ca86f (HEAD -> master, tag: v2.0) Revert "index page added"
e99ea34 login signup added after reset
d317678 (tag: v1.0) index page added
f7fafcf react project initialized
```

`git tag -d v1.0` Delete the specified tag

```
pralee@dheeraj-ubuntu:~/react$ git tag
v1.0
v2.0
pralee@dheeraj-ubuntu:~/react$ git tag -d v1.0
Deleted tag 'v1.0' (was d317678)
```

`git tag`

Shows all tags

`git push --tag` Sends your tags to the remote repository

```
pralee@dheeraj-ubuntu:~/java$ git push --tag
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/dsapkale/learn-git.git
 * [new tag]         v2.0 -> v2.0
```

pralee
Technology & Games

Thank You

Keep learning

/praleetechnology

/praleetechnology

/@praleetechnology

/company/praleetechnology

www.praleetechnology.com