## Terminal Codes

1. xinput list - List of devices
2. xinput float [] - remove device
3. xinput reattach [] [] - reattach device
4. whoami - user name
5. echo “sample” - print sample
6. echo $SHELL - type of shell currently using
7. clear (Ctrl + L) - clear the terminal
8. history - view all command history
9. history [10] - view last 10 command history

### Directory Manipulation

1. pwd - print working directory
2. ls - list files in the current directory
3. ls c\* - list items starts with “c”
4. ls [-a] - show hidden files and directories

. - current directory

.. - parent directory

1. ls [-l], ls [-al] - view file or folder (show details)
2. cd , cd ~ - home directory
3. cd / - root directory
4. cd - - previous directory
5. cd . - change to current directory
6. cd .. - change to parent directory
7. cd ../.. - change to grand parent directory
8. cd /home - change directory by absolute path

### File Manipulation

1. mkdir dep - make directory “dep”
2. mkdir “dep 10” - make directory with spaces
3. rm file\_name - remove file
4. rm -r folder\_name - remove folder
5. mv old\_folder/file\_name new\_name - rename folder or file
6. touch file\_name - create empty file
7. nano text\_file\_name - open text document using nano. (text viever in UNIX)
8. cp existing\_file\_path&name new\_file\_path&name - copying file
9. cp -r folder\_name&path new\_path&name - copying folder
10. mv file/folder\_directory&name existing\_directory\_path - move file or folder

### Permisions

1. sudo …. - temporary root access for the following code
2. sudo su - start sudo session
3. exit - exit sudo session
4. chmod permission file/folder - change folder/file permission
   1. permission – 700, 600, 500, 300
5. chown user\_name file\_name - change ownership of a file
6. chown -R user\_name folder\_name - change ownership of a folder

JDK 17 (/opt/java/jdk-17)

JDK 11 (/opt/java/jdk-11)

JDK 8 (/opt/java/jdk-8)

NodeJS (/opt/nodejs/nodejs-18)

Apache Maven (/opt/apache/apache-maven-3.x.x)

intellij IDEA Ultimate (/opt/jetbrains/itellij-idea)

DataGrip ((/opt/jetbrains/data-grip)

VS Code (/opt/vscode)

MySQL (apt install mysql-server)

Java FX Scene Builder

### Installations

Editing “/home/gayashan/.profile” or “/etc/profile” is not recommended here as we need to use multiple java versions in the system.

If we need to use multiple software versions in the operating system don’t set the environment variables.

Therefore, use the following method. (“code” is used for VSCode in following sample)

sudo update-alternatives --install /*usr/*bin/code code *installation path (executable path)* 1

{update-alternatives} -

{--install} -

{usr/bin/code} - Shortcut path, where the executable files are stored (usr/bin)

{code} - The command, that want to be added as syntax to run in terminal

{installation path} - The path of the extracted file’s executable file

1 - priority number

To check the previous update-alternavies relative to “java”, use:

update-alternatives --list javac

update-alternatives --list java

To change the version of java used in the operating system, use the following command,

sudo update-alternatives --config java

### JAVA

Compiling the .java file

javac -cp . Demo.java

If class path is similar to . (working directory) the ( -cp . ) part is optional.

Running .class file by JVM

java Demo

## GIT

#### Initialize git

git init

#### Set username and email for git

git config --global user.name "Gayashan\_Dananjaya"

git config --global user.email "[gayashandananjaya3@gmail.com](mailto:gayashandananjaya3@gmail.com)"

#### View my git confguration file

git config --list

You can use the git config –list command to see all the configuration values that are associated with your particular Git installation.

#### Git config: configuration levels

Configuration values can be set at three different levels:

**–local:**

Local values will be applied to the repository in which the git config command is executed. These values are stored in .git/config inside a repository.

**–system:**

System values are applied to all users on a machine. You should set system-level configuration values with caution because it may alter existing configurations. These values are stored in /etc/gitconfig on Linux.

**–global:**

Global values are applied to a particular user on an operating system. They are stored within the ~/.gitconfig file in your home directory.

When you’re first setting up Git, you’ll mostly use the –global level.

#### Cloning (Download) github repository

git clone [url]

Clone (download) a repository that already exists on GitHub, including all of the files, branches, and commits.

#### Check the git staging environment

git status

#### Add files to the staging environment

git add --all Stage all changed (*new, modified, and deleted*) files.

git add -A

git add index.html (adding specific file)

#### Commiting from staging environment to the repository

git commit -m “Type your message here”

#### Commiting without the staging environment

git commit -a -m “Type your message here”

#### Checking the created git branches

git branch

***Creating a new git branch*** (whie staying at the previous branch)

git branch branch-name

#### Switching to an existing branch

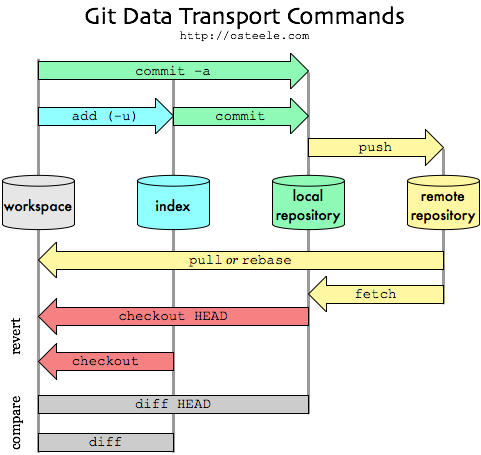
git switch branch-name

git checkout branch-name

#### Creating and switching to a newly created branch

git checkout -b branch-name (branch created from the current branch)

git checkout -b NEW\_BRANCH [FROM\_BRANCH]



#### Renaming branch

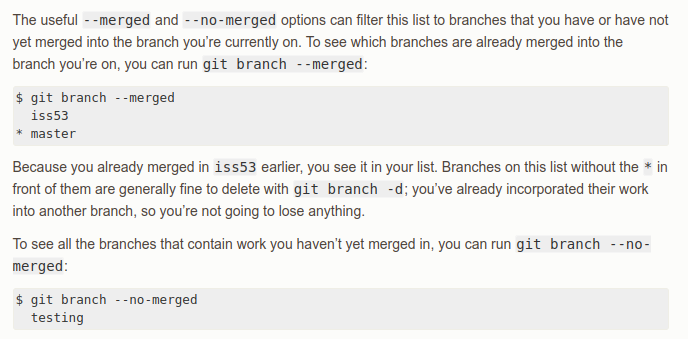
git branch -m new\_name (renaming the current branch)

git branch -m old\_name new\_name (renaming a different branch)

#### Merging a branch to master

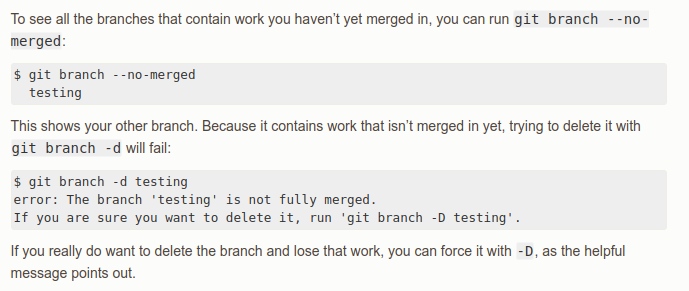
1. switch to master branch

2. git merge branch-name



#### Deleting a branch

git branch -d branch-name



#### Setting an upstream branch

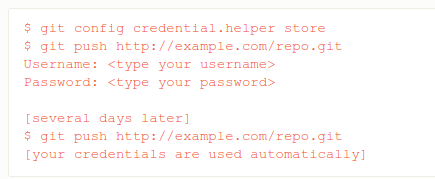
git remote add origin https://github.com/hello-world.git

This specifies that you are adding a remote repository, with the specified URL, as an origin to your local Git repo.

#### Saving the username and password credentials to github to be used automatically

git config --global credential.helper store

The point of this helper is to reduce the number of times you must type your username or password. For example:



#### Setting an upstream branch

git push --set-upstream origin <branch>

git push -u origin <branch> (Shortened version)

<branch> : The local git branch, that need to push

The easiest way to set an upstream branch is to use the "--set-upstream" option when pushing the branch to the remote repository for the first time. This relationship is very helpful for two reasons.

**1st reason;**

Push and pull become a lot easier. With an upstream branch set, you can simply use the shorthand commands "git pull" and "git push" - instead of having to think about the exact parameters like in "git push origin development".

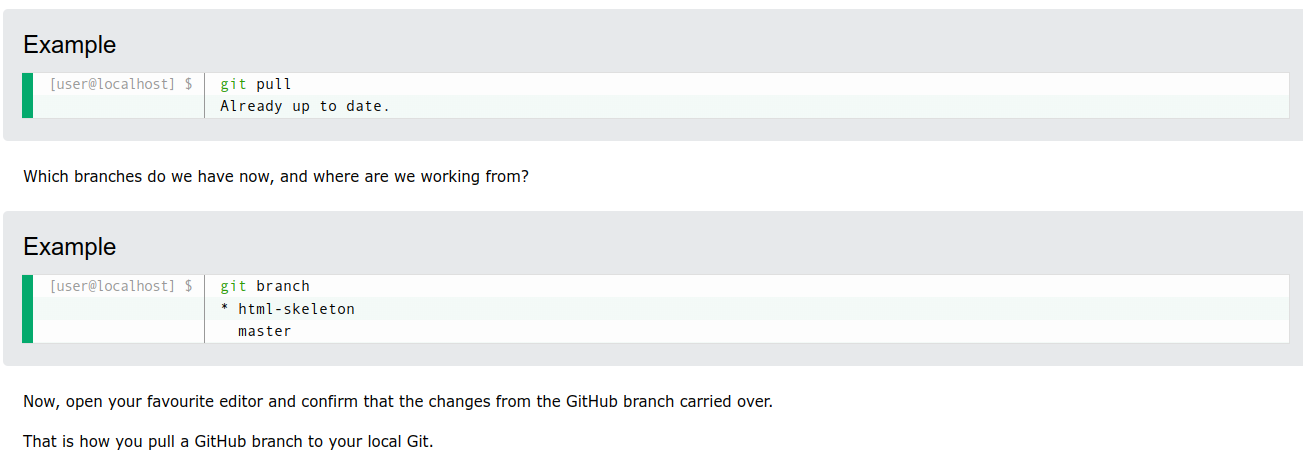
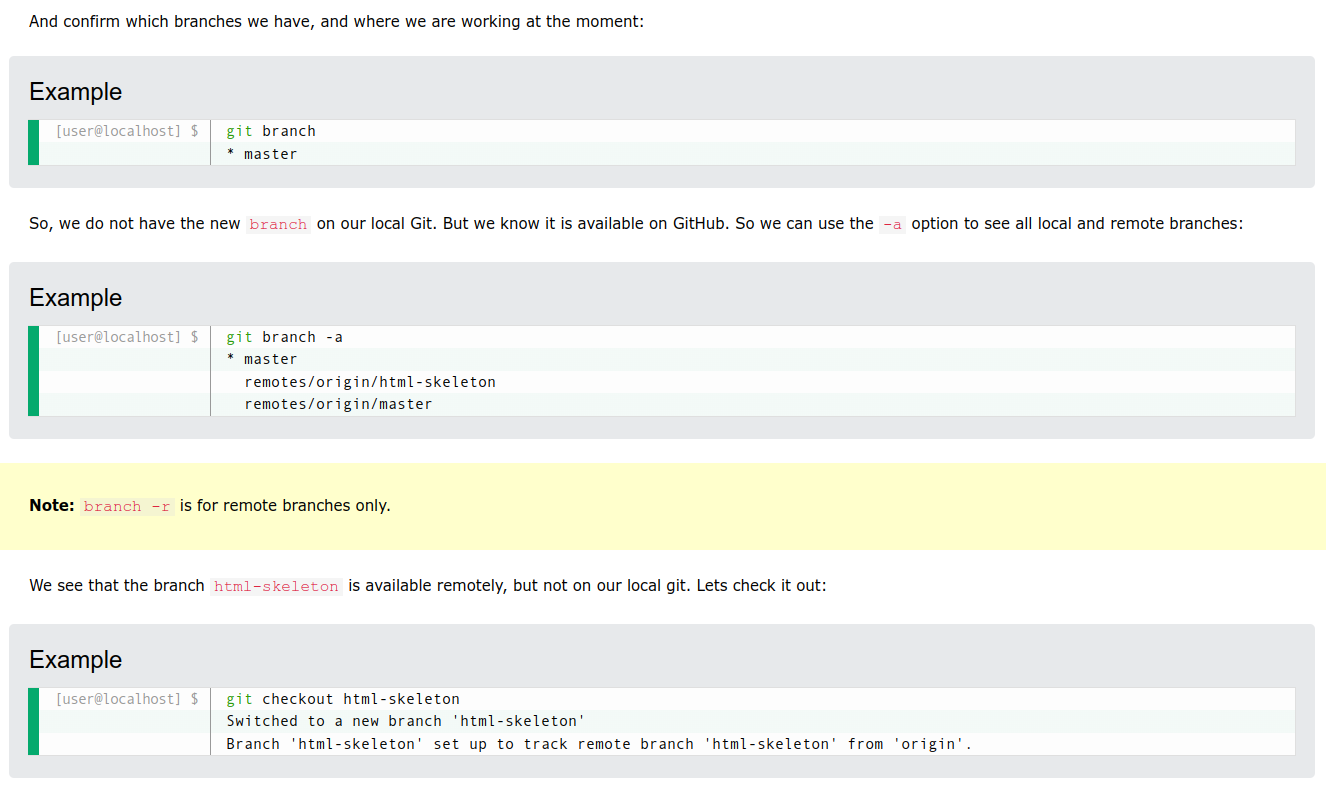
**2nd reason;**

Git can now also tell you about unsynced commits which you haven't pushed or pulled, yet. Here's an example.

(a) if you have 2 commits in your local repository which you haven't pushed to the remote yet, then your local branch is "2 commits ahead" of its upstream branch.

(b) if there are 4 commits on the remote upstream branch which you haven't pulled yet, then your local branch is "4 commits behind" its upstream branch.

#### Pulling a branch from github (not saved in local git)



git pull = git fetch + git merge

git pull: Updates your current local working branch with all new commits from the corresponding remote branch on GitHub

#### Pushing a new branch to github

git push origin branch-name

(branch name is the git branch name and same name branch will be crated in github)

#### Cloning a github repository to git

git clone [url]

Clone (download) a repository that already exists on GitHub, including all of the files, branches, and commits.

.