**On terminal app,**

**To install homebrew,**

$ /bin/bash -c "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh>)"

**For brew version control,**

$ brew --version

**For git version control,**

$ git --version

**To install git,**

$ brew install git

**or download and install via installer**

[www.git-scm.com/download/mac](http://www.git-scm.com/download/mac)

**See the available configurations**

$ git config --list

**Configurations process**

$ git config --global user.name ”your name”

$ git config --global user.email ”your email adress”

**To learn configured user name and user email**

$ git config --global user.name

$ git config --global user.email

**To see pathway**

$ pwd

**To see file content you are in**

$ ls

**Clear the terminal**

$ clear

**To go xx pathway**

$ cd xx

**To see the hidden files**

$ ls -a

**To go previous pathway**

$ cd ..

**To make an “xx” directory**

$ mkdir xx

**Create an a.xx file**

$ touch a.xx

**Add something from terminal to inside of a.xx file**

$ vi a.xx

and

push the “i” button

for exit, first push the “ESC” button and then “colon (:)” button. After that write “wq” and push “enter (return)”

**For open the a.xx file on süblime text**

$ subl a.xx

**For open the vs code**

$ code . a.xx

**For remove the a.xx file**

$ rm a.xx

**For remove the all the folder you are in**

$ rm -r

**Create a “xx” named folder**

$ mkdir xx

**Create git repository in folder**

$ git init

Staging area indexleme alanıdır.

**To learn commit, branch status and newly created contents**

$ git status

**Create “aa” named txt or desired extention file**

$ touch aa.txt

**Create “aa” named txt or desired extention file in “xxx” directory**

$ touch xxx/aa.txt

**Add aa file to staging area**

$ git add aa.txt

**Remove aa file from staging area(transfer to untracked files)**

$ git rn –cached aa.txt

**Send all files in folder to staging area for indexing**

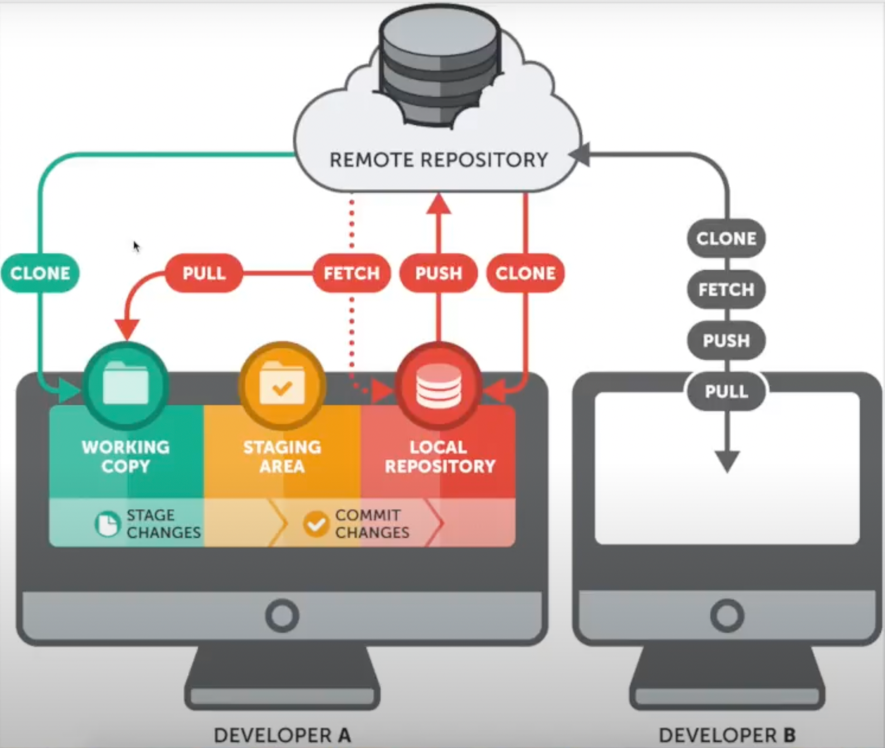
$ git add .

**See the log, see the commits**

$ git log

**For simpler report, see the commits**

Git log --oneline

****

**Save file content to local repository from staging area with “first commit” note in master branch**

$ git commit -m “first commit”

**Go to xx commit**

$ git reset “xx commit id” --hard

**Bring xx commit staging area**

$ git reset “xx commit id” --soft

**Deleting the change made in the commit/version**

$ git revert “xx commit id”

**Change the commit note**

$ git commit--amend -m “new message”

**Create a branch, named “feature”**

$ git branch feature

**For branch listing**

$ git branch -a

**Switch to feature named branch**

$ git checkout feature

**Create a branch, named “feature-1” and switch it**

$ git checkout -b feature-1

**Delete a branch, named “feature”**

$ git branch -d feature

**Delete a branch, named “feature” forcefully**

$ git branch -D feature

**Merging the "feature" group with the group we are currently in**

$ git merge feature

**Github registration**

Select public repository and fill personal informations, mail verification

Click new repository, fill repository name, give description optionally

Select public or private

Add README

Add gitignore. , add license and create repository

Copy the https link

**Save files and commits to remote repository in “xxx” branch**

$ git push “paste https link here” xxx

fill user name and password

**Save https link to repository as “origin” nickname**

$ git remote add origin ”paste link here”

**See the saved https links**

$ git remote -v

**Save files and commits to remote repository saved https nickname as “origin” in “xxx” branch**

$ git push origin xxx

**Clone any repository**

$ git clone “paste https of repository”

Create pull request as contributer and merge pull request as manager

**To fork a repository**

Via click manually fork any project to own account

If modification is necessary modify

Do all previous processes the same way on forked project’s https adress

**Create pull request to forked repository of original**

After modify on forked repository click new pull request

Click create pull request, give title and make comment

**To handle pull requests**

Click pull requests

For review the changes made, click files changed

We can make comment and submit review to other editors

via the merge pull request and then confirm merge we can complete merging

**Save new modifications on remote repository to local repository**

Also we can create new file button, necessary modifications and click Commit new file with description and then

$ git fetch “paste here https adress”

“fetch” doesn't do any file transferring. It's more like just checking to see if there are any changes available.

For copy we need to,

$ git merge “https adress/”branch name”

**Save new modifications on remote repository to local repository**

Also we can create new file button, necessary modifications and click Commit new file with description and then

$ git pull “https adress/”branch name”

“pull” brings (copy) those changes from the remote repository.

**1-Create .gitignore**

$ touch .gitignore

**1-For hide files**

Write the files and directories you want to hide into the .gitignore file

**Don’t hide “aa.php” file, except all other files in same “bb” directory**

In the .gitignore file write,

bb/\*

!bb/aa.php

**Readme**

We can create readme.md extended file in local repository and make modifications before send it to remote repository.

Also on github ui, add a README button.

**For readme standart syntax rules**

<https://docs.github.com/en/github/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>

**For create tables on readme standart syntax rules**

<https://docs.github.com/en/github/writing-on-github/working-with-advanced-formatting/organizing-information-with-tables>

**Stash**

The git stash command takes your uncommitted changes (both staged and unstaged), saves them away for later use, and then reverts them from your working copy.

$ git stash save “notes about save process”

**See the stash list currently we have**

$ git stash list

**See the stash details**

$ git stash Show -p “stash id(0, 1, 2 ..)”

**Delete the last or specified stash**

$ git stash drop

$ git stash drop “stash id”

**Bring back the last or specified stash**

$ git stash pop

$ git stash pop “stash id”

**Alias**

For add a shorthand for a common Git command or set of Git commands

**To use status command as st**

$ git config --global alias.st status

$ git st

**To use “commit -m” as co**

$ git config --global alias.co “commit -m”

$ git co “message”

**To see alias list**

$ git config --list

**Undo changes made to the xx.txt file.**

$ git checkout -- xx.txt

**To see detailed modifications**

$ git diff