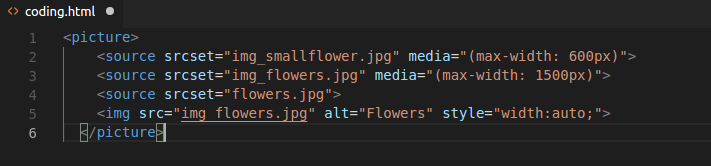
**HTML ,Show Different Images Depending on Browser Width**

 <picture> element allows you to define different images for different browser window sizes.

<https://www.w3schools.com/html/tryit.asp?filename=tryhtml_responsive_picture>



**image sprites web app**

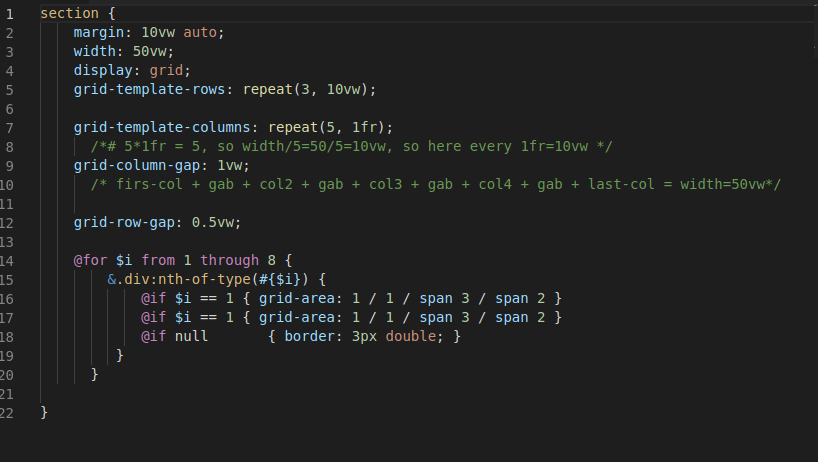
An image sprite is a collection of images put into a single image.

A web page with many images can take a long time to load and generates multiple server requests.

Using image sprites will reduce the number of server requests and save bandwidth.

[http://css.spritegen.com](http://css.spritegen.com/)

**css Grid fr**





Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

All the commands start by git

Advantages :

- it is free

- create snapshot

- snapshot to snapshot

- local storing or online server (it is called “repository مستودع”, push and pull)

- update the code and save the old too

**1- create the repository folder and make it as Home directory.**



**2- make it as local git repository.**

*git init*

note: the git commands will apply only under this home directory (Git repository)

**3- to be sure if it is Master**

*git status*



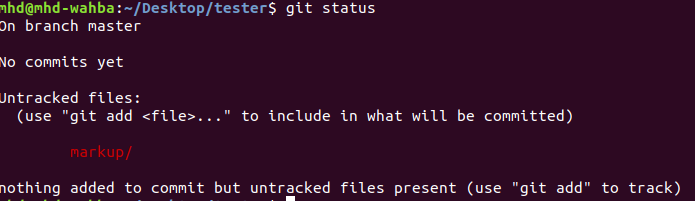
**4-**



**5-**



**6-**



**7-**

*git add –-all*

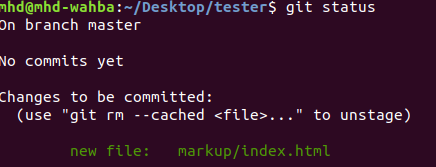
or

*git add* FileName



**git add** . adds all modified and new (untracked = not saved) files in the current directory and all subdirectories to the staging area (a.k.a. the index) that which files need to make the Snapshot , thus preparing them to be included in the next **git** commit .

**8-**



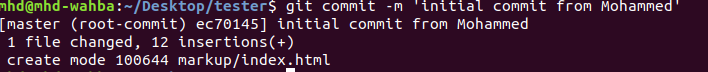
**9- create git acount**

*git config --global user.email* [best.pid@gmail.com](mailto:best.pid@gmail.com)



**10- create Snapshot**

*git commit -m 'initial commit from Mohammed'*





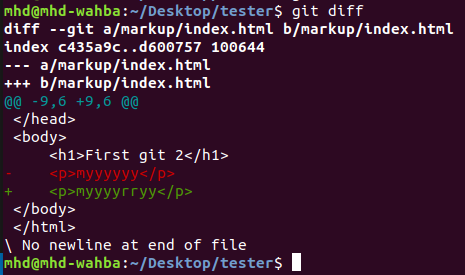
**11- to show all commits**

*git log*  … more details

*git log – -oneline -<number>* …. short details.. *<number> is number of last commit*

**12- to show what it is changed before to add**

*git diff*

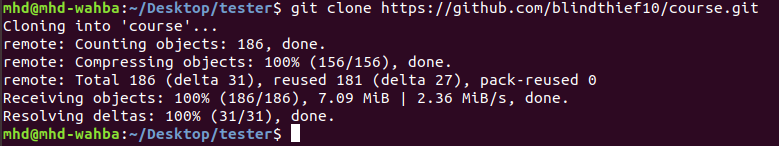
****

**13- to backup from last snapshot**

*git revert HEAD … not on Master , not after push*

**14- to clone(download) file from github.com**

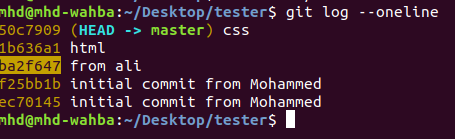
*git clone* [https://github.com/blindthief10/co](https://github.com/blindthief10/course.git)u[rse.git](https://github.com/blindthief10/course.git)



**to delete the cloned(downloaded) file**

*rm -rf <file name> ….. m -rf course*

**- to backup to certain snapshot …. will delete the previous commits**

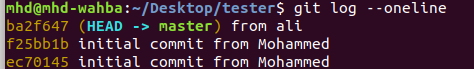


*git reset –-hard <ID> …. hard will delete also the last stage area*

*git reset –-soft <ID> …. soft will not delete also the last stage area*

Stage area : if done ‘git add - -all’ ,and not done ‘git commit -m ‘mhd’’

note: we never done this on master, but on only branches.



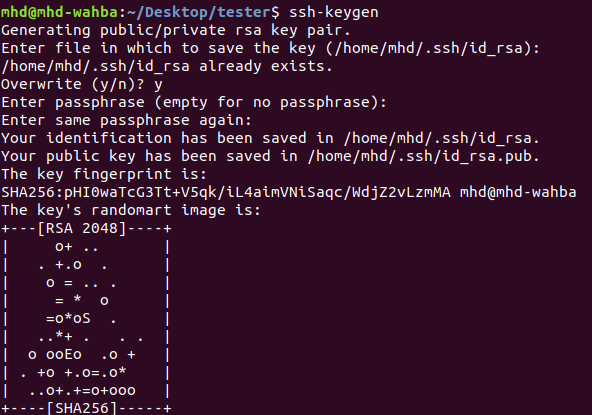
Note that it was deleted was before of the last ID reset.

**- to backup to certain snapshot …. but without delete the previous commits**

*git checkout <ID>*

*git branch -b <new branch name>*

**- to make ssh connection setup between local git repository and remote git repository**

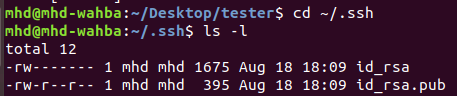
*step 1 -* **ssh-keygen generation**

*1- ssh-keygen*

*three times : enter*

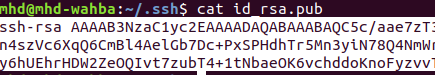
*2- cd ~/.ssh*

*ls -l*

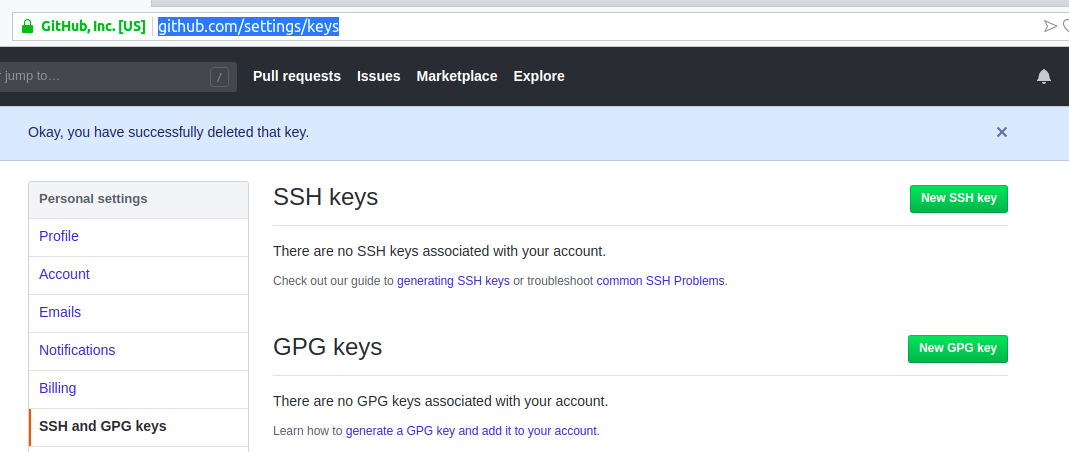
**

*3- cat id\_rsa.pub*

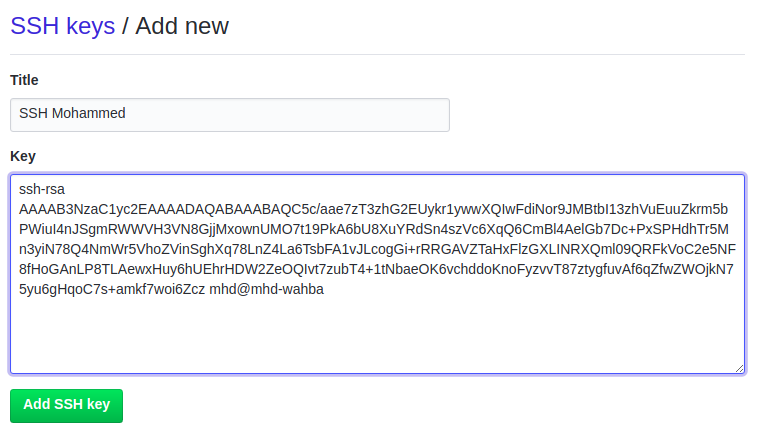
and then the copy this public key



*4-* go to [*https://github.com/settings/keys*](https://github.com/settings/keys)



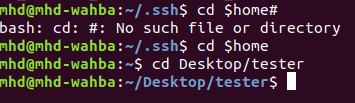
then copy the public key

then enter your account’s password

*5-* change the home directory to git repository

*cd $home*

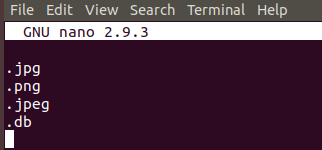
*cd Desktop/tester*

**

*6-* create .gitignore file to ignore file types ( no push to online git repository)

*touch .gitignore*

*nano .gitignore*

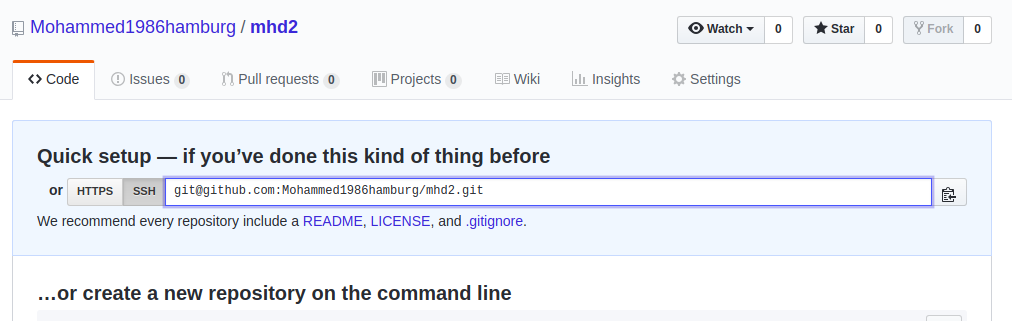


*7-* add and commit .gitignore file to local git repository

*git add --all*

*git commit -m 'add .gitignore file'*

8- copy the SSH the url of new online github repository



*step 2- git remote add origin <url of new remote git repository>*

**- to show remote repository**

*git remote -v*

**- to remove remote repository**

*git remote remove <name>*

*step 3- git push --set-upstream origin master …..* to create file and connection between remote origin repository and local mgit aster branch

**- to push from local master to remote repository**

*git push*

*the summary to push:*

*git add –-all*

*git commit -m 'my message'*

*git push*

**- to create branch**

*git branch …* to see the branches ,\* means which branch I am using

*git branch <name> ….* to add new branche

*git checkout name …..* swich to the name branch

*git push --set-upstream origin name* ….. to create file and connection between remote origin repository and local name branch

*the summary to push:*

*git add –-all*

*git commit -m 'my message'*

*git push …. or git push - –force*  ( never used in Master)

**- to delete branch**

*git branch -d name*

**- to merge branch to master**

*- git checkout master*

*- git diff master name*

*-git merge name*

-*git push*

**- to Pull from online repository ( Fetch+Merge with local )**

*git pull*

or pull from different remote branch name

*git pull origin <remote branch name>*

*\**git pull is shorthand for git fetch followed by git merge <branch>

**- to go back one step (redo )**

*git reset HEAD …. ignore last change on repository*

*git checkout - - <file name.\_\_> ……ignore last change on this file*

**- to Marge a current commit with last commit**

*git commit - -amend …. but it will generate new ID*

**- to create new branch and switch to it**

*git checkout -b <new branch>*

**- to merge all remote branches together , to create the final file project**

*step 1 on my branch -to decide which the right code-*

………… wait for anther branch to push………

*-git pull origin <my partner branch> ….. to download the remote files of <my partner branch> .. this local*

*-git checkout <my name branch>*

*-git merge <my partner branch>*

… check the codes and accept which it…

*-git add - -all*

*-git commit -m ‘message’*

*-git push*

then every branch have to do the above

*step 2 on my master- to merge to master*

*-git checkout master ….*

*-git merge <my name branch>*

*-git add - -all*

*-git commit -m ‘message’*

*-git push*

**- to see all the actions and commands**

*git reflog*

**- to undo the file if not done ‘git add - -all’ ,and not done ‘git commit -m ‘mhd’’**

*git checkout - - <file name>*

**- to undo the file if done ‘git add - -all’ ,and not done ‘git commit -m ‘mhd’’**

*git reset HEAD <file name>*

*git checkout - - <file name>*

**-** *rebase* **to merge but keeps the commits of your current branch on top**

*git rebase jake …... rebase is merges , but Rebase brings all commits from the specified branch but keeps the commits of your branch on top of them.*

**- to go back to the commit with hash eb49uf2 safely without deleting all the in-between commits**

*Git checkout <ID>*

**-** *cherry-pick* **To bring a specific commit from different branch. This commit will go on top of your commits.**

*git cherry-pick <ID>*

**-** *stash* **to Save temporarily untracked files without commit**

*git stash …. Save temporarily untracked files without commit*

*git stash pop ….. Bring the stashed files back to life*

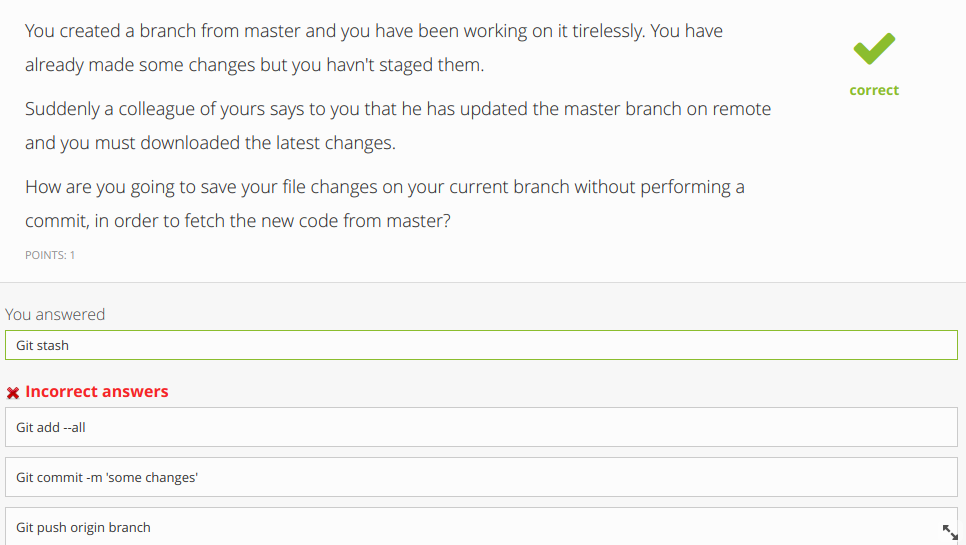
**-to unstage a file that i have accidentally staged**

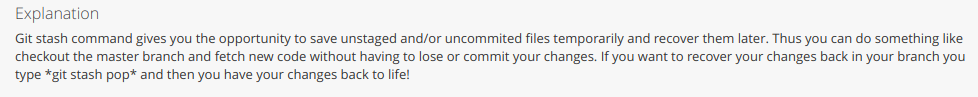
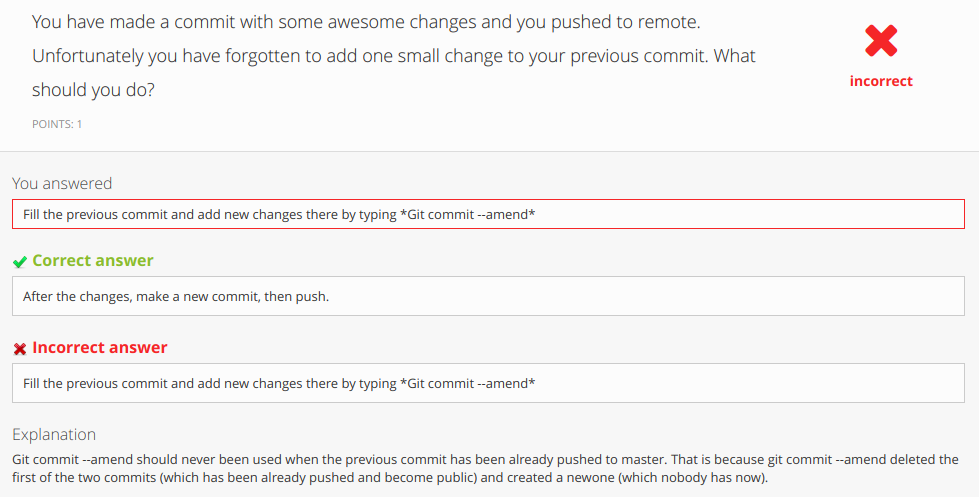
*git reset HEAD <file name>*

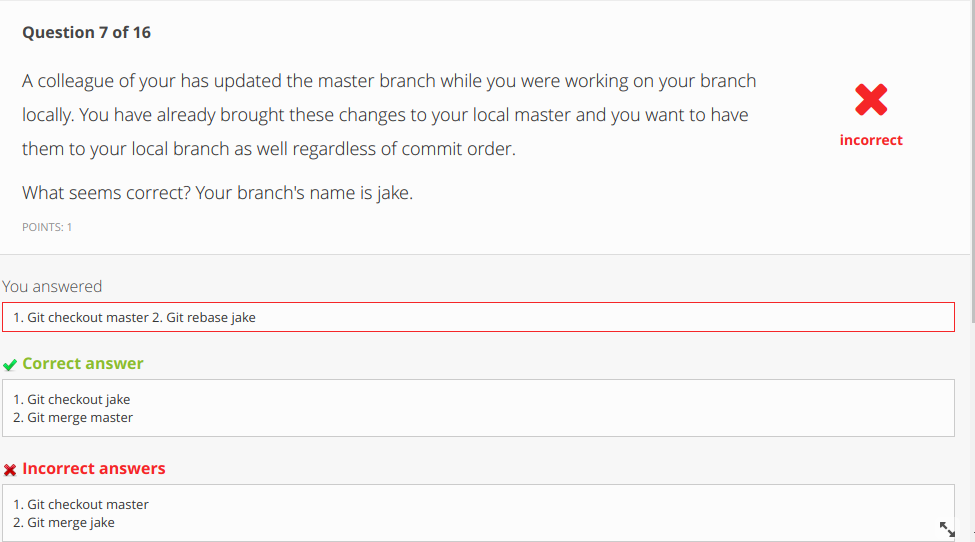
*ex:*

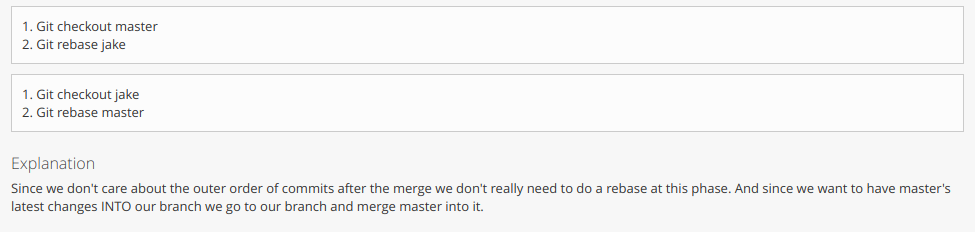
*git reset HEAD jake.html …... Unstage jake.html*

*--------------------------------------------------------------------------------------*

**

**

**

**

*------------------------------------------------------------------------------------------------------------------------*

JavaScript

**css Example**

Css Example

**css Example**

Css Example 6 7 11 13 14 15

**css Example**

Css Example