**GIT and GITHUB**

# 1**.** Install Git

<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

Linux (Debian) : Command: sudo apt-get install git

Linux (Fedora): CommandL sudo yum install git

Mac: <http://git-scm.com/download/mac>

Windows: <http://git-scm.com/download/win>

Download Git from windows site and run the **Git-2.23.0-64-bit.exe** file.

Follow the next steps and click on finish.

Hence Git and GitBash will be installed.

Git terminal looks as below



Using Git Commands

1. To tell Git who you are, run the following two commands:

**$ git config --global user.name "name"**

**$ git config --global user.email "email address"**

The option "--global" means that you set your username and email for Git globally on your computer. No matter how many projects with separate local repositories you create, Git will use the same username and email to mark your commits.

2. To view your Git configurations

**$ git config --list**

3. Create local Git Repository

Go to cd /c/user/my\_project( Create a project folder under user)

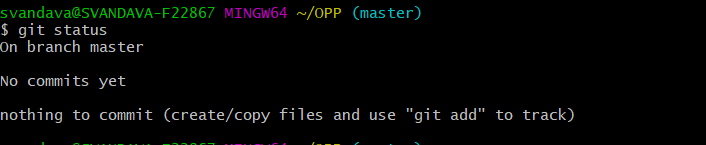
4. **$git init**

This command is asking git to create safe repository under the directory. It creates .git file.

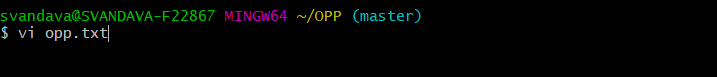


5.**$git status**

This command will give you the status of the files that are in the repository.



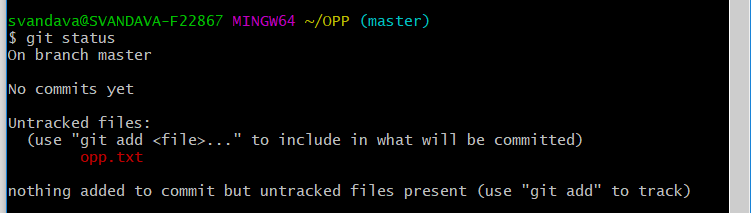
6. To create a file in the Local Repository

vi filename.txt

It will open Vi editor and enter the contents in that file and save the the file

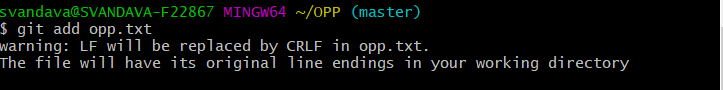
To edit file press “i” and to save “:wq”

If you give **$git status** you can see the file which you have created.



7.To add file to the staging environment.

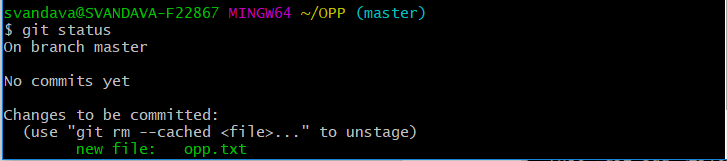
**$git add filename**



a**.$git add .**

This will add all the files to the stating environment.

If you give $git status



b**.$git add –all**

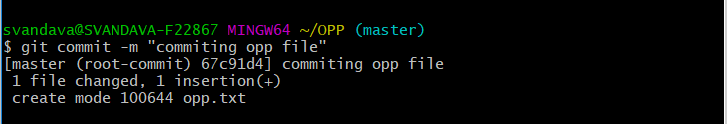
If the root directory has sub directories and to add files present in the subdirectories.

8**.$git rm --cached filename**

To remove file from staging environment.

9. To commit files to repository

**$git commit –m ”reason for commit or commit message”**



After commiting only the file will be added to the repository.

10. **$ git reset file2.txt**

As an alternative to "rm --cached <filename>", you can use the "reset" command:

You can consider "reset" as the opposite of "add".

11. **$ git commit -a -m "commit message"**

To add and commit files at once

12. **$ git reset --soft HEAD^**

To understand what that "HEAD" thing represents, recall that we work in branches. Currently we're in the master branch, and HEAD *points* to this master branch. When we switch to a different branch later, HEAD will point to that different branch. HEAD is just a pointer to a branch.

There will be times when you'll regret committing to a repository. Let's say you've modified ten files, but committed only nine. How can you add that remaining file to the last commit? And how can you modify a file if you've already committed it?

13. **$ git add file-i-forgot-to-add.html**

**$ git commit --amend -m "Add the remaining file"**

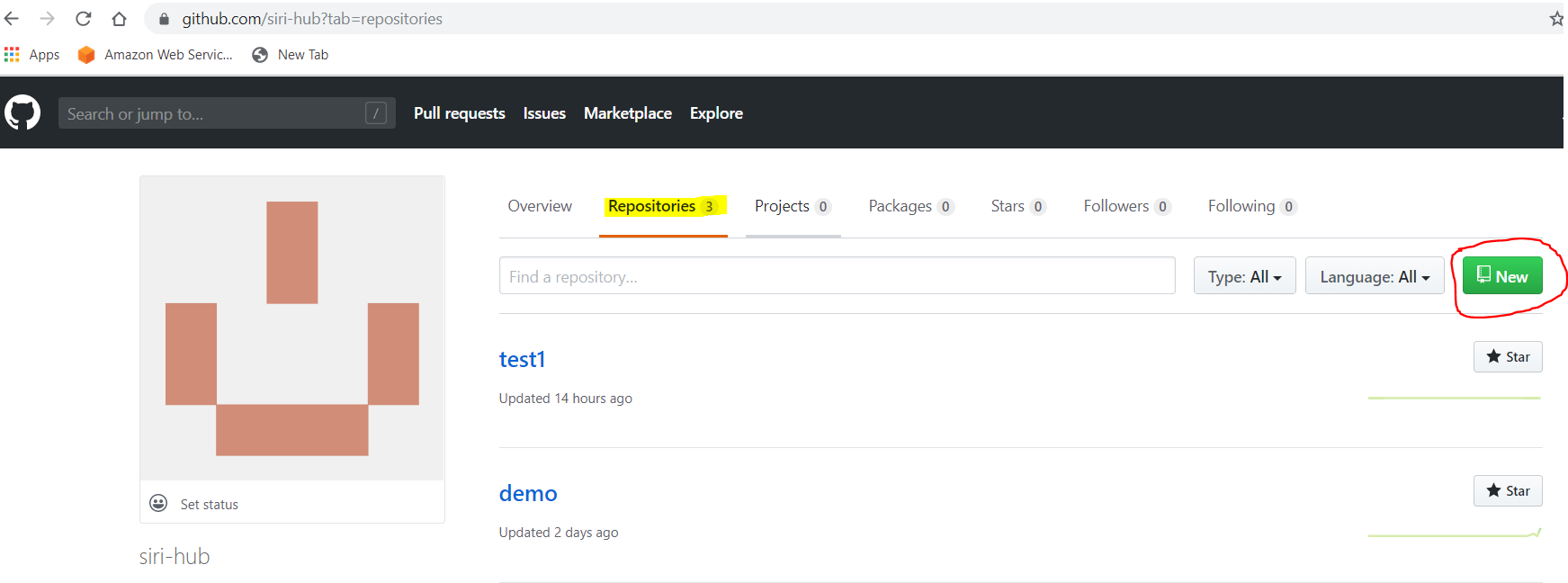
Instead of resetting the HEAD and undoing the last commit, we can rectify a commit by using the "--amend" option when committing to a repository. Just add the remaining file to the staging area and then commit.

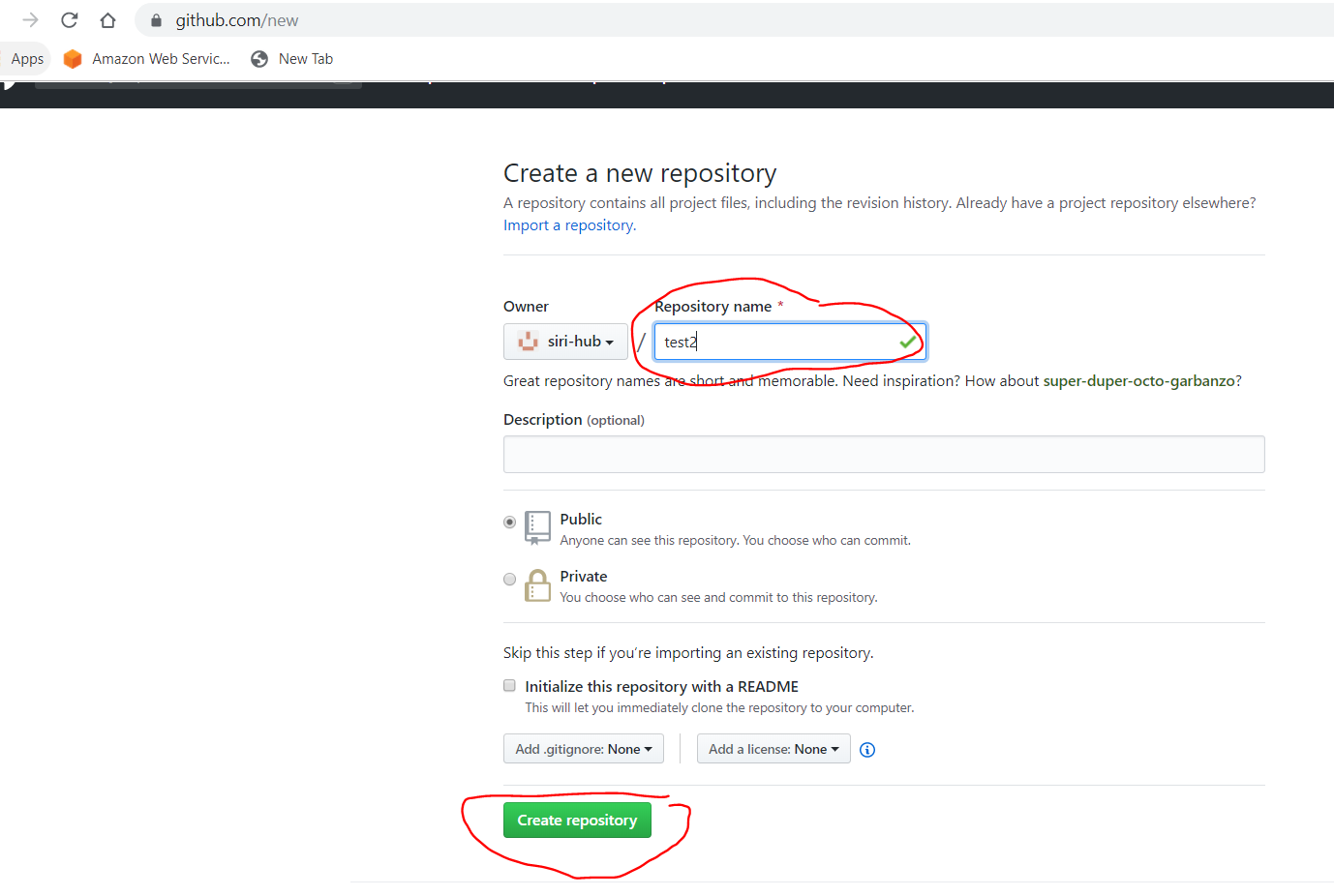
**Push a Local branch to Github**

a.Create Github account on [**www.github.com**](http://www.github.com)

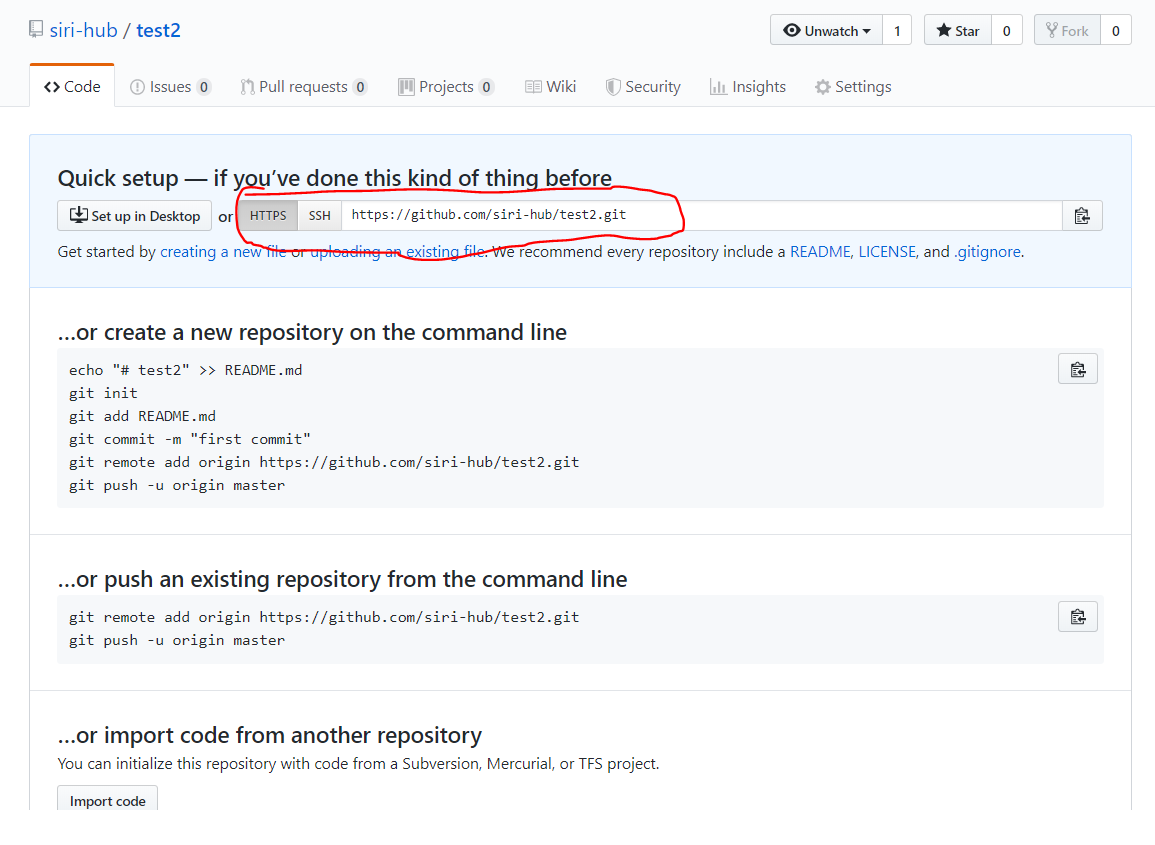
b.Login to github account

Goto repositories🡪click on new🡪give the name of repository🡪click on create repository





c. Copy the SSH link



d.Then go the git terminal, execute the below commands

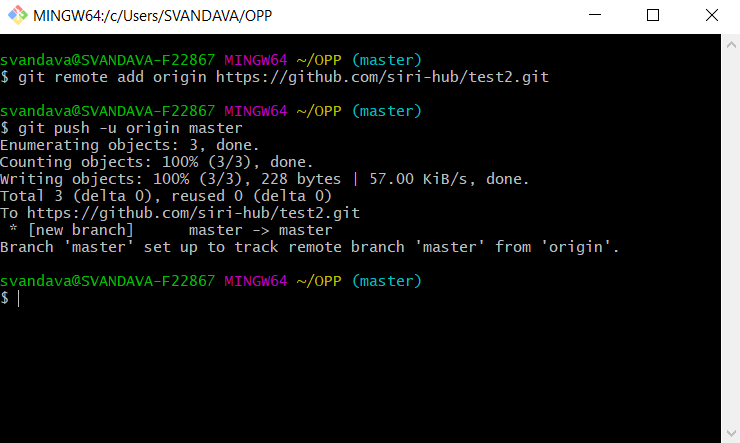
**$git remote add origin “URL copied from GITHUB site”**

**$git push –u origin master**

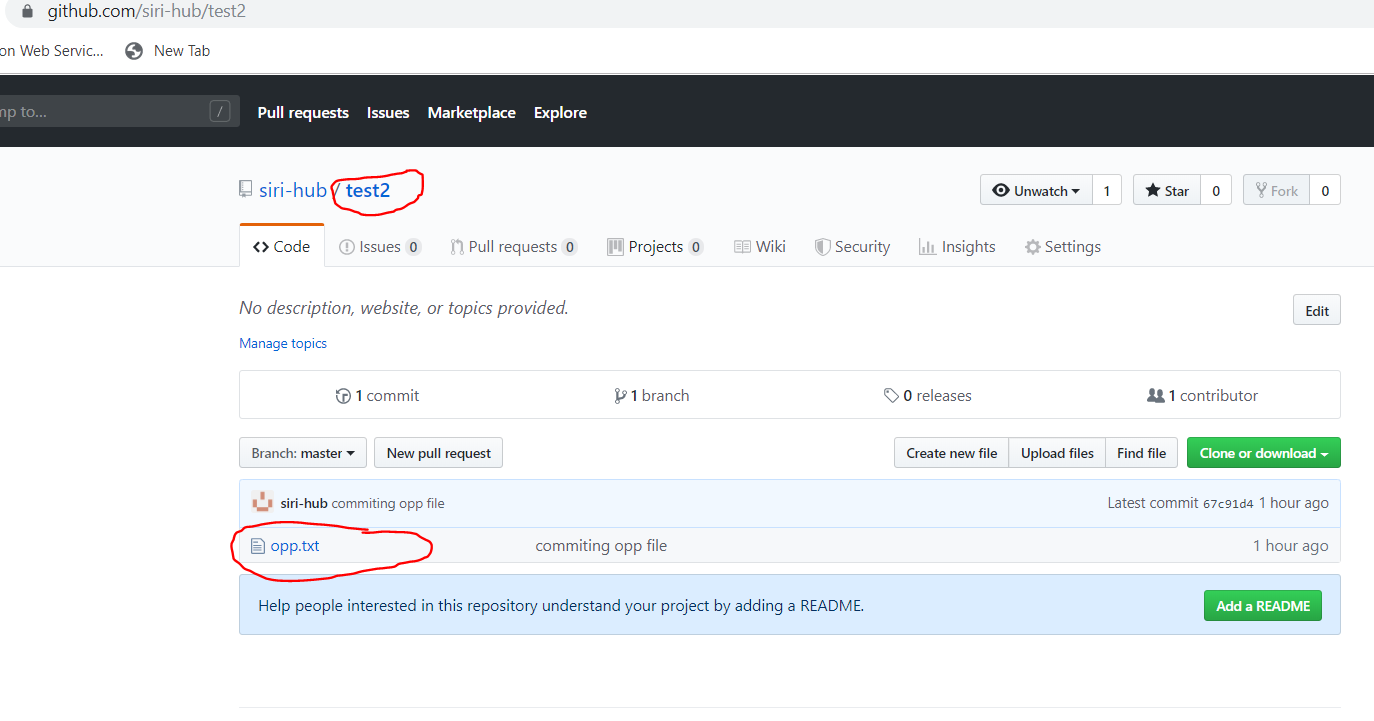
It will push the local branches to remot repository.

For the first time it will ask to login to the github with username and password.

For next times the username and password will be stored and it uses the same**.**

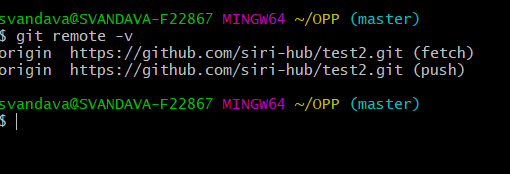


e.Goto github site then you can see the braches added to master from local repository.



**f.$git remote –v**

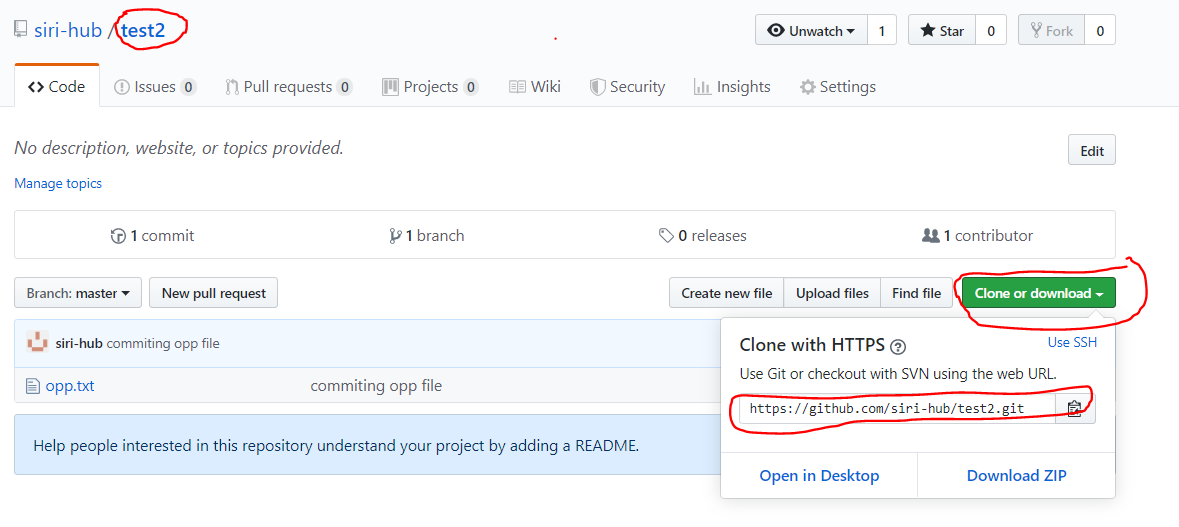
It will list all remote repositories you have connected to.

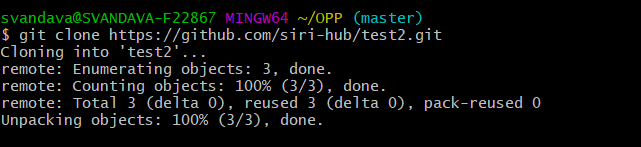


**Cloning Remote Repository to Local machine**

**$git clone your clone URL copied from github site**

This will clone the remote repository and copied to the local machine**.**



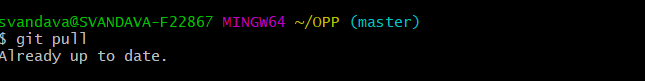


If you want to change the name of the cloning repository you can use below command

**$ git clone git@github.com:YourUsername/your-app.git this-name-is-much-better**

**$git pull**

This will update the cloned repository if there are any changes made after cloning.



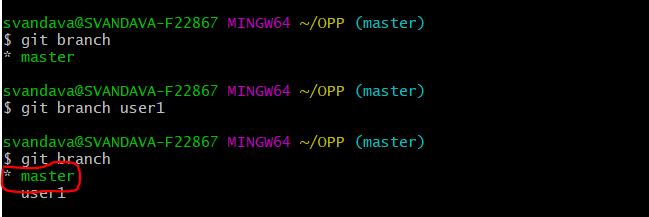
**Working with Branches**

**1.$git branch**

This will display all the branches in the repository

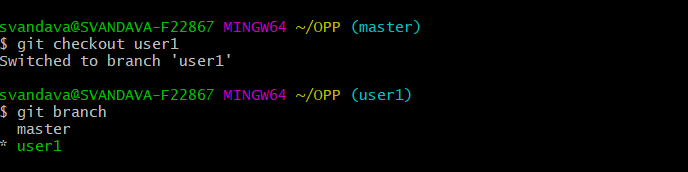
**$git branch branchname**

If we want to add new branch to the repo



The \* sign before the branch master indicates your current working branch, even though you created new branch but you are working in the current branch so you need to switch branch to the new one.

**$ git checkout branchname**

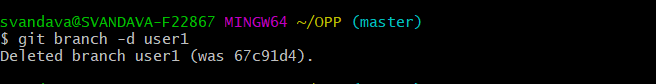


**$ git merge branchname**

If you want to merge the branch to the master

**$ git branch -d branchname**

If you want to delete branch , if you try to delete the branch you are in git doesn’t allow you to do ,so switch to another branch then delete unwanted one.



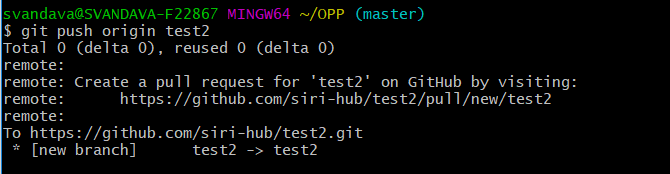
**$git checkout –b branchname**(creates new branch and immediately switches to new branch)

$git push origin branchname

(Or)

git push git@github.com:git/git.git yourbranchname

This will push the branch to the remote repo.



**$git log**

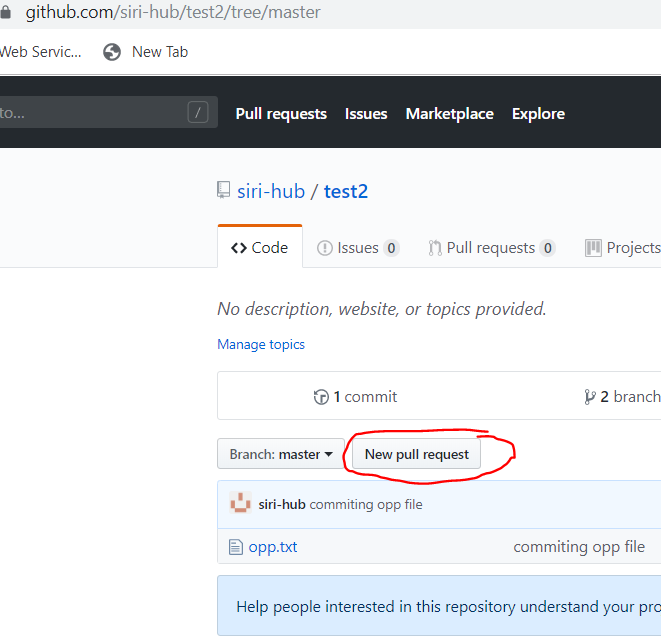
This will gives the log of the repo.

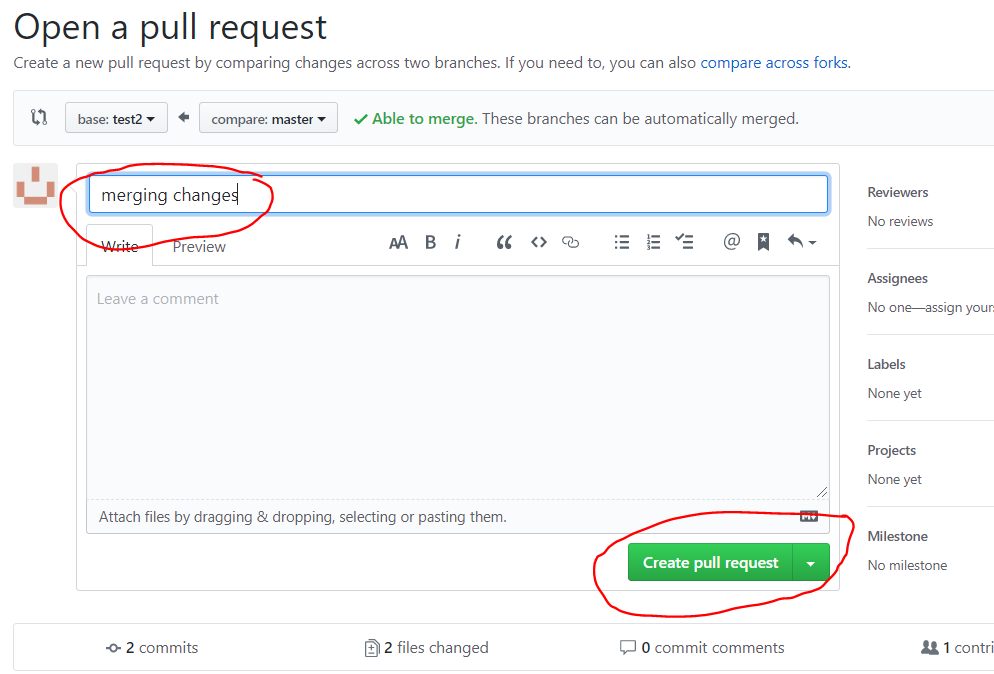
**$ git log --since=2.weeks**

limiting log output for last 2 weeks.

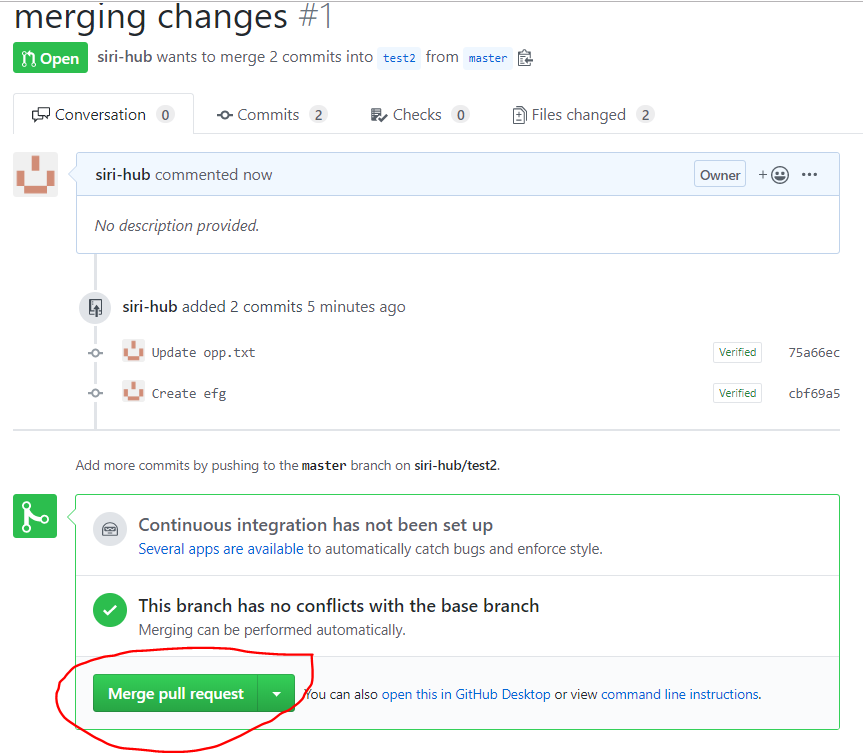
**Creating Pull Request**

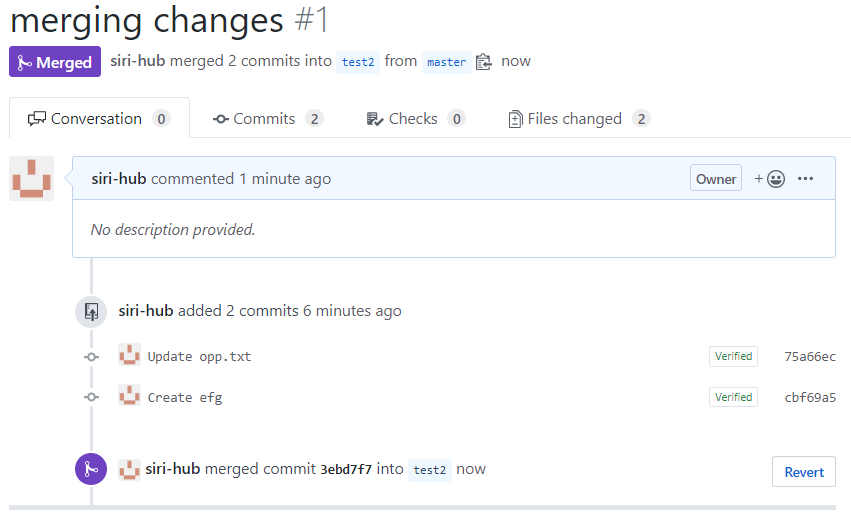
Go to the Github and the repository click on the new pull request, it will opens a pul request give the comments to the pull request and click on the create pull request

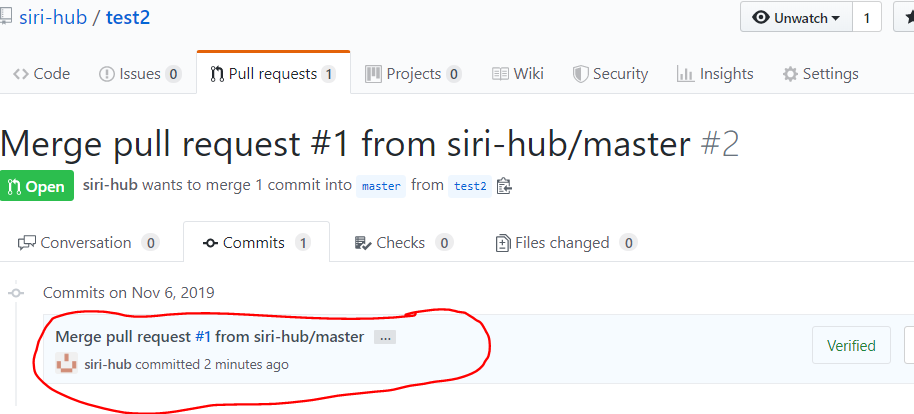




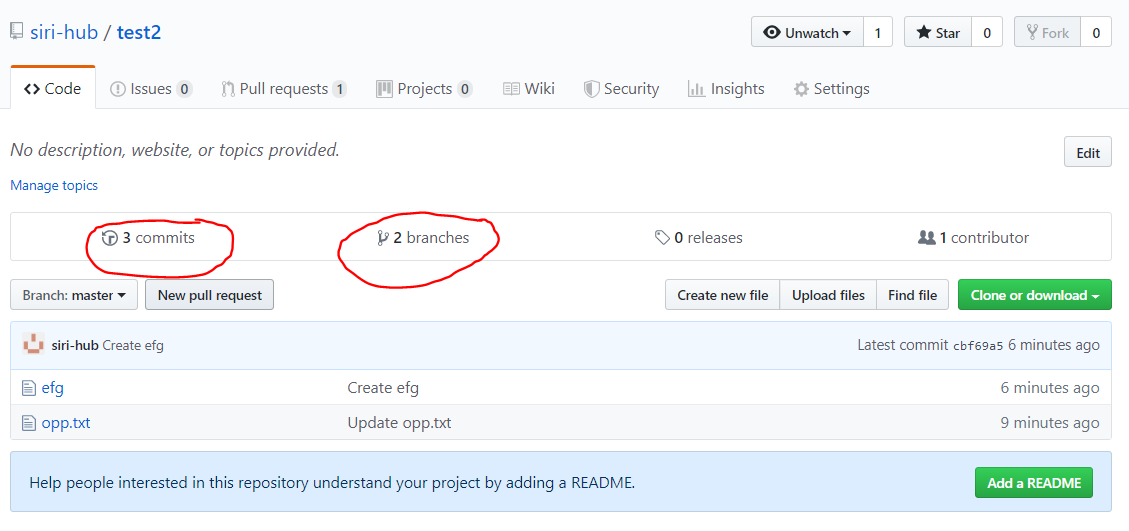
Verify for the new changes made and click on the merge pull request





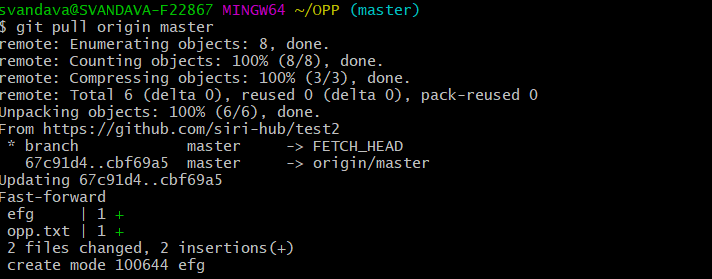


Here we can see no of commits and no of branhes in the repo.



If you want to pull these changes to the local repository

**$git pull origin master** (it will update your local master branch)



$ git remote rename oldname newname

$git remote rm newname

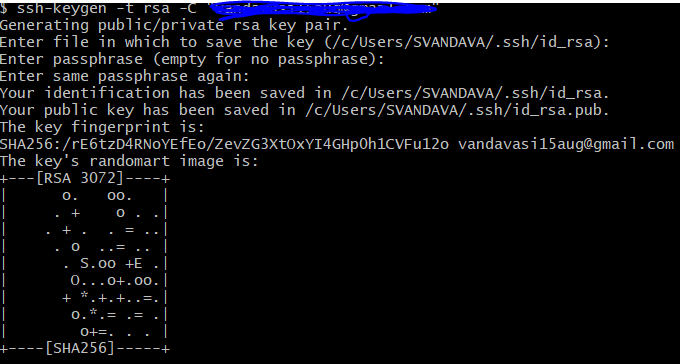
$git show

$git tag –a v1.4 –m “tag message”(-a is for creating tag)

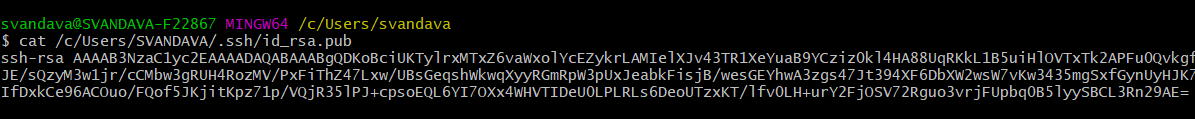
$git tag –d v1.4 (deleting the tag)

**If you are getting any authenticating permission denied private key issue**

$ ssh-keygen -t rsa -C "your email address of git account”



Then copy the public key



Goto the Github.com account🡪settings🡪SSH and GPG keys🡪Add new

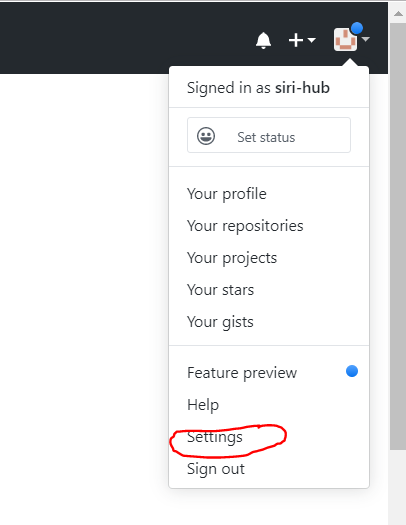
Paste the copied key.

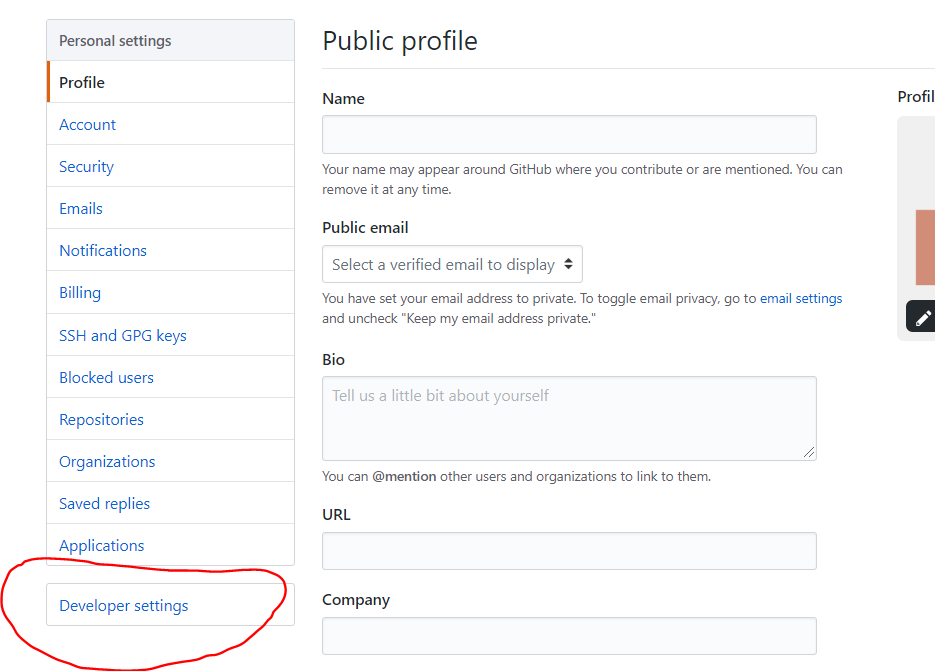
Now go to GitBash terminal , now you can able to work on .

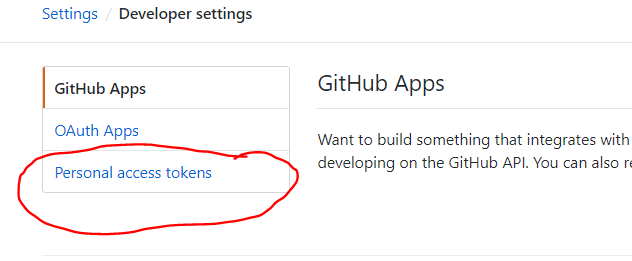
|  |  |
| --- | --- |
|  |  |

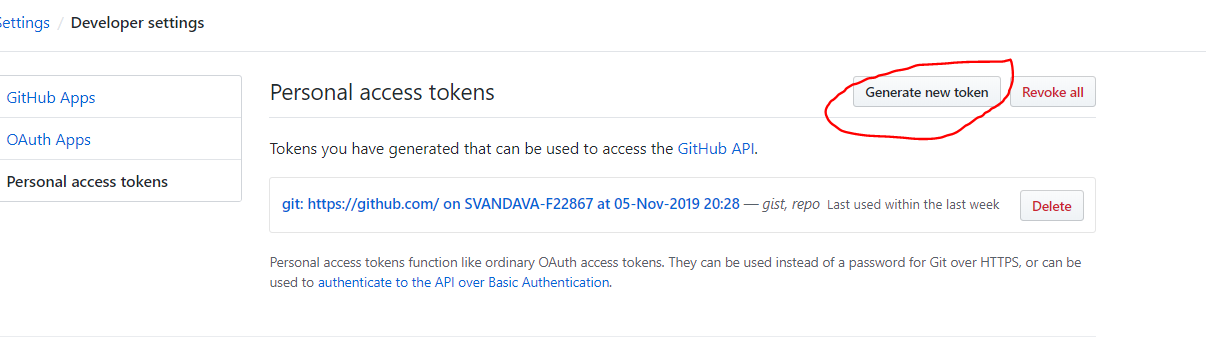
**How to generate personal tokens in github**

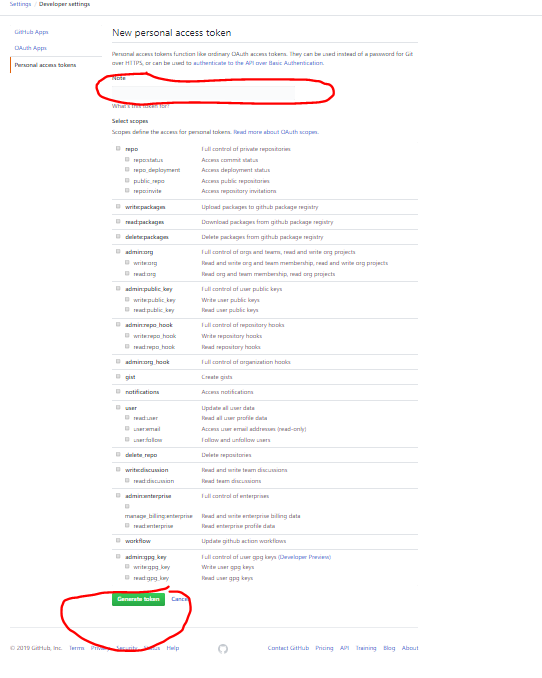
1.Goto Github account 🡪click on settings🡪click on Developer settings🡪personal access tokens🡪generate new token🡪fill in the details🡪click on generate token









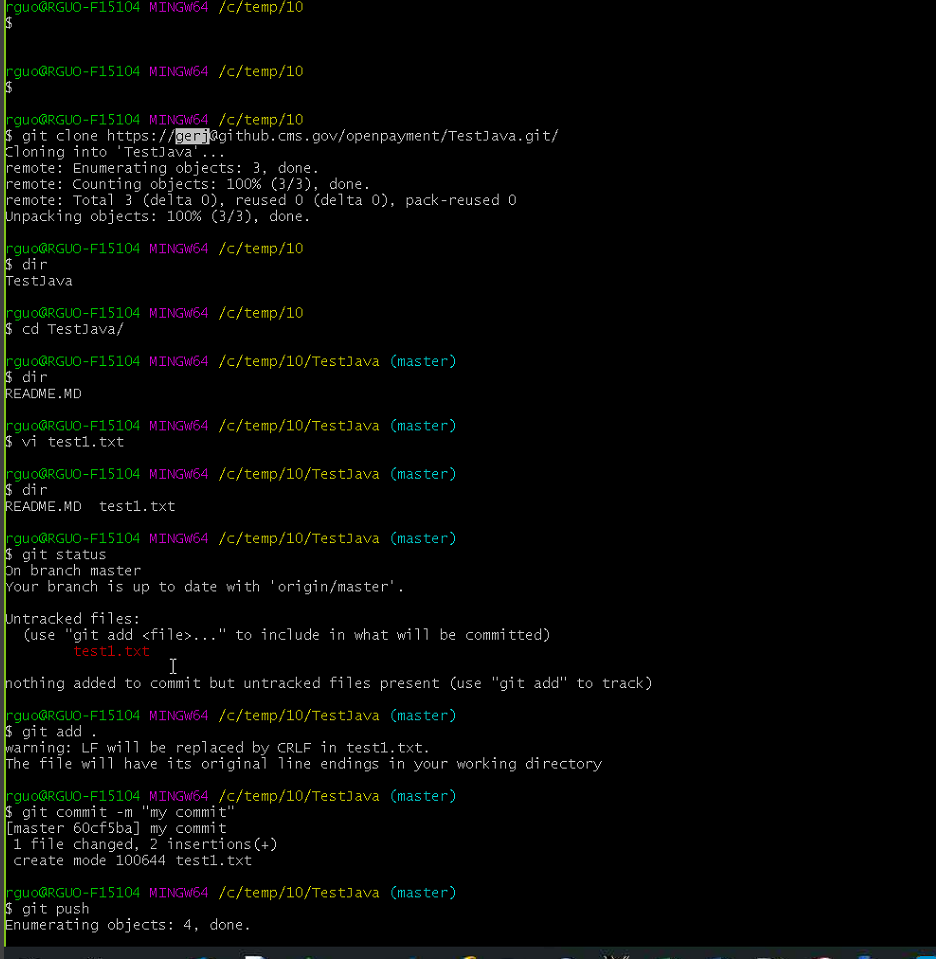


Use the token from Git terminal.

$ git clone https://github.com/*username*/*repo*.git

Username: *your\_username*

Password: *your\_token*



The error may be, the computer has saved a git username and password so if you shift to another account the error 403 will appear. Below is the solution  
For Windows you can find the keys here:

control panel > user accounts > credential manager > Windows credentials > Generic credentials

Next, remove the Github keys.

In mac  
1-In Finder, search for the Keychain Access app.  
2In Keychain Access, search for github.com.  
3-Find the "internet password" entry for github.com.  
4-Edit or delete the entry accordingly.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

git clone https://y2f4@github.cms.gov/openpayment/TestJava.git/

git push -u origin master

git credential-manager uninstall

git credential-manager install

1. Under Programs, click **Uninstall a program**.
2. Right-click the entry named **GitHub**, then click **Uninstall/Change**.
3. Select "Remove the application from this computer."

git remote add origin https://github.cms.gov/openpayment/TestJava.git

**Junkins Pipe line code**

[**https://github.com/javahometech/my-app**](https://github.com/javahometech/my-app)

node {

stage (‘SCM Checkout’) {

git (credentialsId: ‘git-creds’, url: ‘https://github.com/javahometech/my-app’,branch: ‘dev-branch’)

}

stage (‘Mvn Package’) {

def mvnHone = tool name: ‘maven-3’, type: ‘maven’

def mvnCMS = “${mvnHome}/bin/mvn”

sh {mvnCMS}‘mvn clean package’

“tool name: mvn clean package”

}

Stage (‘Build Docker Image’) {

Sh ‘docker build –t kammana/my-app:2.0.0 .’

}

Stage (‘Push docker image’) {

withCredentials ( [string(credetialsId: ‘docker-pwd’, variable: ‘dockerHubPwd’)]) {

sh “docker login –u kammana –p $ {dockerHubPwd} “

}

sh ‘docker push kammana/my-app:2.0.0’

}

Stage(‘Run Container on Dev Server’) {

Sh ‘docker run –p8080”8080 –d -name my-app kammana/my-app:2.0.0’

}

}

**Installing Git in Linux :: https://www.tecmint.com/install-git-centos-fedora-redhat/**

**Deployment Automation**

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

**Jenkins Deploy plugin**

**New Item :: AutomatedDeploymentTest**

Sudo update-alternatives –config java

Sudo yum list |grep java-1.8

Sudo yum install java-1.8.0-openjdk

Java –version

sudo update-alternatives –config java

choose 2 to update the default from java-1.7 to java 1.8

Minimum 4 GB Ram

Google search for download sonatype nexus 3

Download nexus sonatype 3 for Linux, for that copy the link to download

cd /opt/

wget url

untar the file

mv nexus-3.14.0.-04 nexus3

chown ec2-user:ec2-user nexus sonatype-work

starting cd nexus/bin

./nexus start

./nexus status

http://<iP:8081>/ id: admin pw:admin123

configuring nexus as a service.

Run nexus as a service google search.

vi .,bashrc add NEUS\_HOME-/opt/nexus3

source .bashrc

bin/nexus.rc run\_as\_user=”ec2-user”

create a symbolic link

sudo ln -s $NEXUS\_HOME/bin/nexus /etc/init.d/nexus

chkconfig

sudo chkconfig –add nexus

sudo checkconfig –levels 345 nexus on

sudo service nexus start

sudo service nexus status

-----------------------------------------------------

Create docker private registry

---------------------------------------------------

Go to Nexus 🡪 go to settings 🡪 Repostiories 🡪 create a repository

Choose docker hosted, give unique name OPP-Docker-Private-Repostitory

Repository Connections 🡪 Port 8083 (http)

Force basic authentication (make sure Disble to allow anonymous pull is checked).

Enable docker api (checked)

-----------------------------------------------------

Push docker image into the private repostiory

---------------------------------------------------

Create an EC2-instance

Install docker

Sudo yum install docker –y

Start docker service

Sudo service docker start

Update the configuration

vi /etc/docker/daemon.json

{

“insecure-registries”: [ “52.90.181.34:8083” ],

“disable-legacy-registry”:true

}

sudo service docker restart

Log into Docker registsry from the command line.

Sudo docker login –u admin –p admin 123 52;90.181.34:8083

Pull docker image from docker hup

Sudo docker pull alpine:latest

Sudo docker tag alpine:latest 52;90.181.34”8083:alpine:1.0.0

Sudo docker images

Sudo docker push 52.90.181.34:8083/alpine:1.0.0

Open nexus URL and browse for the docker image.

<https://github.com/ValaxyTech/DevOpsDemos/blob/master/SonarQube/Sonal_Integration_with_Jenkins.MD>

<https://www.tothenew.com/blog/how-to-setup-jenkins-for-a-maven-project/>