

GIT COMMANDS

1.Troubleshooting : (403 error)

Invalid status for POST <https://pointcache.github.com/RaginBear/CaelDominis.git/info/lfs/objects/batch>: 403

Solution :

```
$ git remote rm origin
$ git remote add origin https://github.com/RaginBear/CaelDominis (new repo url)
```

What is Git ?

Version Control tool (Source Code management Tool)

SVN - SubVersion
GIT

How we operate git :-

- 1. Command Line
- 2. Graphical

Difference B/W GIT and SVN

SVN	GIT
➤ Centralized Repository	➤ Distrubuted Repository
➤ Code Develops at Centrailized Repo	➤ Code Develops at Local Centrailized Repo
➤ Internet is needed always	➤ Only Needed to push code

Downloading Git :-

<https://git-scm.com/downloads>

Default path of Git In windows
C:\Program Files\Git

Debian/Ubuntu:-

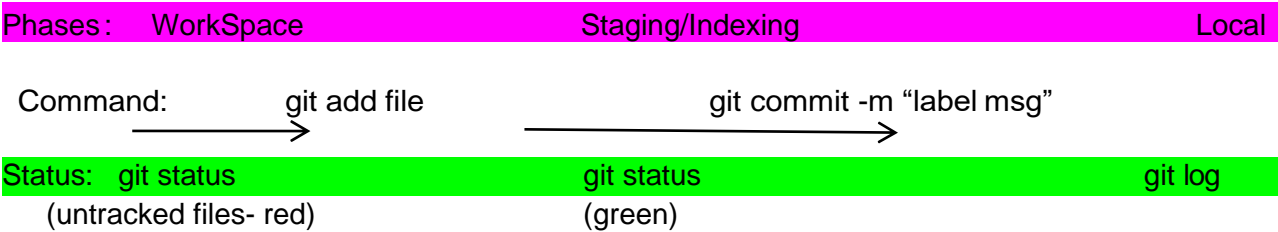
```
sudo apt-get update
sudo apt-get install git -y
git --version
```

Uninstall git in windows:-

Control panel --> Programs --> Uninstall Program

Uninstall git in Ubuntu:
sudo apt-get remove

Phases In GIT : -



- git init - Initialize git repo
- ls - list files and directories
- ls -la - list hidden directories
- git status - to know which phase

Configure the username, replace First Last :-

git config --global user.name "First Last"
git config --global user.email "example@example.com"


Confirm with below command whether Configured or not :-


git config --list

Creating GitHub Account :-

Link : <https://github.com/>

Click on sign up and provide the basic details and verify email address

 Step 1:
Set up your account

 Step 2:
Choose your subscription

Create your personal account

Username *

This will be your username. You can add the name of your organization later.

Email address *

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Verify account

7013882648

devopsbyrush@gmail.com

Getting code from Github Repository:-

git clone <https://github.com/devopsrishi/helloworld.war.git>

Create a new repository on the command line:-

```
echo "# helloworld" >> README.md
```

```
git init
```

```
git add README.md
```

```
git commit -m "first commit"
```

```
git remote add origin https://github.com/devopsrishi/helloworld.git
```

```
git push -u origin master
```

Push an existing repository from the command line:-

```
git remote add origin https://github.com/devopsrishi/helloworld.git
```

```
git push -u origin master
```

Task :Create a sample file and upload to gitHub :-

```
touch rushi
```

```
git add rushi -->one file to move to staging area
```

```
git add .
```

```
git add -A --> use any one command to move all files at a time
```

```
git add *
```

```
git status
```

```
git commit -m "my first commit " .
```

```
git log
```

Committing only single file from indexing area:-

```
git commit -m "single file commit " filename
```

Check the Details/ Files in Particular commit ID:-

```
git show commit id
```

Summary:-

```
touch file -->git status --> git add file -->git status --> git commit -m "firstcommit" . -->git log --->git show cid
```

```
-->git push origin master
```

GIT LOGS with commit ids :-

git log

git log --oneline

git log -n

git log --oneline -3

➤ **Checking Logs based with author:-**

git log --author=devopsrishi

git log --author=devopsrishi-3

➤ **Checking Logs based on time:-**

Syntax: git log --since=yy-mm-dd

git log --since=2019-04-15

➤ **Checking logs till date:**

Syntax: git log --until=yy-mm-dd

git log --until=2019-04-15

➤ **Checking logs with decoration of branches:**

git log --oneline --decorate

Rollback Commands Reset:-

Workspace staging/index Local repository

Reset	Commands
➤ Local to staging	➤ git reset --soft commitid
➤ Staging to workspace	➤ git reset head filename
	➤ git reset head *
➤ Local Repo to Workspace	➤ git reset --mixed commitid

Note : Local to staging - give Previous commit id to get present one (eg: To get 4th commit id roll back give 3rd commit id)

After this commit will dis-appear from local

Collaboration Accesess :-

Loginto Repo --> settings -->Collaborators -->ProvideGithub-ID--> Add Collaborator

Branches Concept:-

Master

touch file1 file2

git add commit

git log

git checkout -b mastser1

touch file3 file4

git add commit

git log

Come to master

git merge branchname

Git Branch commands	Explination
➤ git branch	➤ list all branches
➤ git branch branch-name	➤ creates new branch
➤ git checkout branch-name	➤ switching branch
➤ git checkout -b branch-name	➤ creating new branch and switching to new
➤ git branch -D <branch>	➤ delete branch
➤ git merge branchname	➤ merging (come to main branch and merge)

Conflict Errors:-

If we modify same file in two different branches

➤ **Under master:-**

cat >file1 -->hello --> git add commit -->

➤ **Git branch master1**

vi file1 --> good morning --> git add commit -->

➤ **Git checkout master**

Vi file1 --> good evening

Merging now

git merge master1

7013882648

devopsbyrushi@gmail.com

Output:-

Auto-merging file1

CONFLICT (content): Merge conflict in file1

Automatic merge failed; fix conflicts and then commit the result.

Resolving:-

git merge --abort

Solving the issue:-

Open the conflict error file

vi file1

Remove unwanted data

git add --> git commit --> git push

Stash Memory : -

Default memory available in git i.e stash memory. (Temporary storage of files which is available in staging?index)



Workspae



staging/index



Local repository

Lab:-

git stash list

touch newfile --> git add newfile

git status

git stash save "labelname"

Status

git stash list

Note :- newly created stash memory will be **stash@{0}**:

7013882648

devopsbyrushi@gmail.com

Stash memory	Description
➤ git stash list	➤ Listing stash memory in git
➤ git stash save "label-name"	➤ Saving files to stash memory
➤ git stash show -p stash@{0}	➤ See the files in stash memory
➤ Roll back from Stash Memory	➤ pop and apply
➤ git stash pop	➤ default data from Stash@{0} rollback and deletes it
➤ git stash pop stash@{1}	➤ Particular data roll back
➤ git stash apply	➤ default data from Stash@{0} rollback only
➤ git stash apply stash@{1}	➤ Particular data roll back
➤ git stash drop	➤ default data from Stash@{0}
➤ git stash drop stash@{1}	➤ Particular stash to delete it

Alias Commands:-

Alias Command	Description
➤ git config --list	➤ list out alias & user details
➤ git config --global alias.l "log"	➤ Configuring alias command for log
➤ git config --global alias.b "branch"	➤ Alias command for branch
➤ git config --global alias.l1 "log --oneline"	➤ Alias command for git log --oneline

Removing Username

- git config --global --unset user.name

Removing Alias

- git config --global --unset alias.b

GIT IGNORE FILES:-

Some log files keep on modifying for those files I want to hide in workspace

Lab:-

touch file --> git add file --> git commit -m "file" .

again making changes --> vi file

git status (modified file)

git commit -am "modified file"

Hiding files at workspace:-

vi .gitignore

Rushi

Rohith

Now try to add these files and modify this it won't show in status.

