First Step:

Start the worklog file using script command. This would capture your handson activities for today.

```
osgdev@TG-DevOps-OS004:~/WorkLog$ script Git_wl.log
Script started, file is Day2_wl.log
```

---

1. Install Git on your linux machine:

   For Ubuntu:
   ```
   # apt-get install git
   ```

   For CentOS:
   ```
   # yum install git
   ```

---

2. Check availability of Git

```
[devops@centserver ~]$ git --version
git version 1.8.3.1
```

Write the first line of code to say "Hello World" in Python.

```
[devops@centserver gitlab]$ cat hello.py
print ('Hello World')
```

Execute it:

```
[devops@centserver gitlab]$ python hello.py
Hello World
```

---

3. Create git local repository to version control this code.

```
[devops@centserver gitlab]$ ls -a
.  ..  hello.py

[devops@centserver gitlab]$ git init
Initialized empty Git repository in /home/devops/gitlab/.git/

[devops@centserver gitlab]$ ls -a
.  ..  .git  hello.py

[devops@centserver gitlab]$ git status
# On branch master
#
```

```
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       hello.py
nothing added to commit but untracked files present (use "git add" to
track)

[devops@centserver gitlab]$ git status -s
?? hello.py
```

---

4. Stage the file hello.py for version changes:

```
[devops@centserver gitlab]$ git add hello.py

[devops@centserver gitlab]$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   hello.py
#

[devops@centserver gitlab]$ git status -s
A  hello.py
```

---

5. Now the code is ready for commit. But sent the config file to indicate who is committing the code.

```
[devops@centserver gitlab]$ git config user.name "devop"

[devops@centserver gitlab]$ git config user.email "devop@devops.com"

[devops@centserver gitlab]$

[devops@centserver gitlab]$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
```

```
[user]
        name = devop
        email = devop@devops.com
```

---

6. Commit the code to local repository.

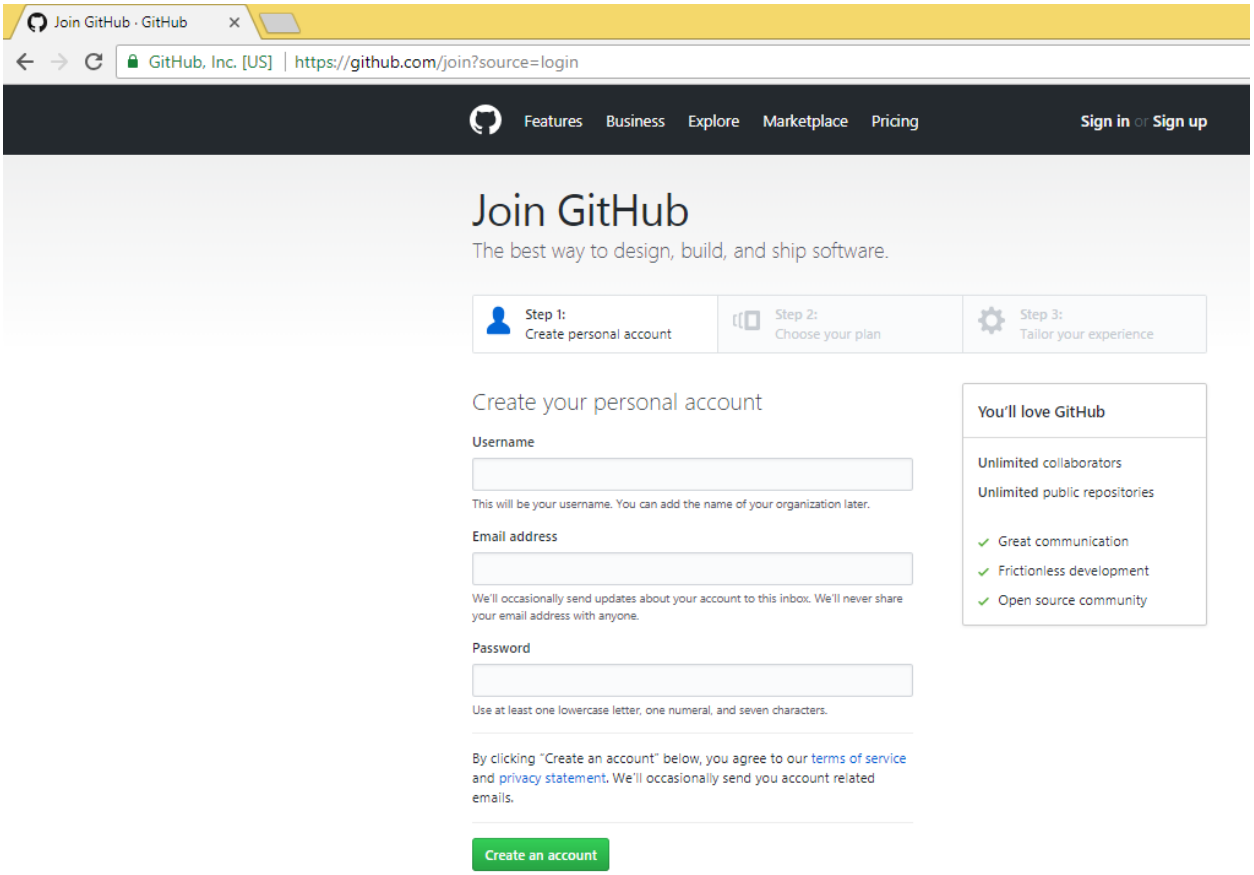
```
[devops@centserver gitlab]$ git commit -m "first commit"
[master (root-commit) 7d8bd35] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 hello.py
```

Check the git log for version committed to local repo:

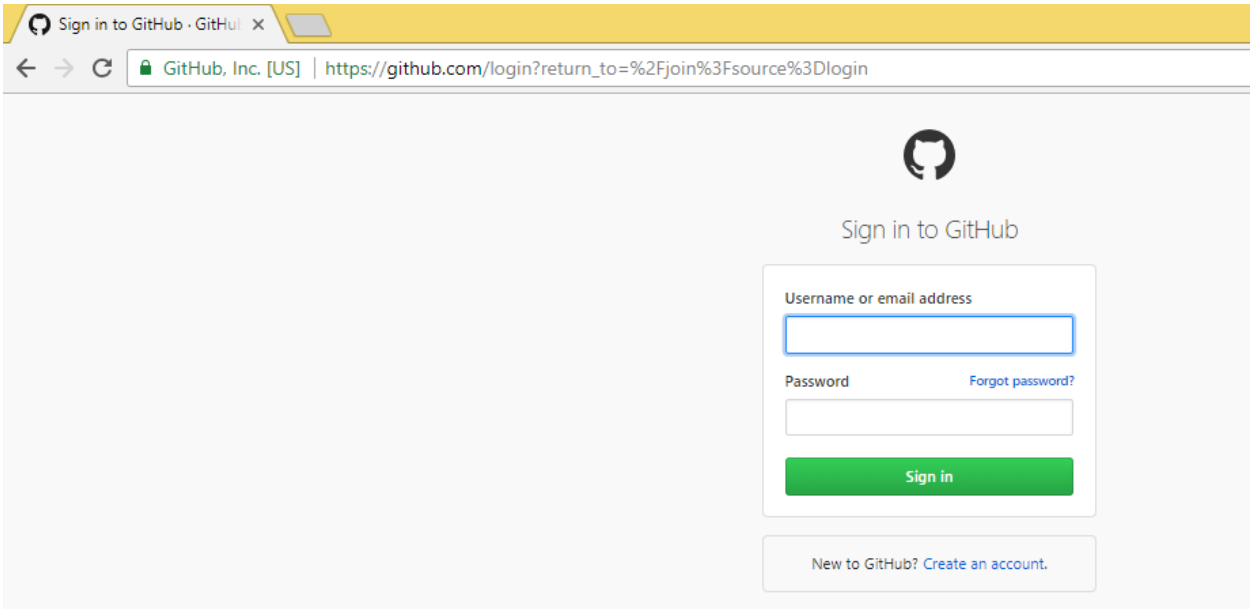
```
[devops@centserver gitlab]$ git log
commit 7d8bd35a28a345f57777d83122508ec4a21f1d79
Author: devop <devop@devops.com>
Date:   Mon Apr 30 12:15:18 2018 +0530

    first commit
```

---

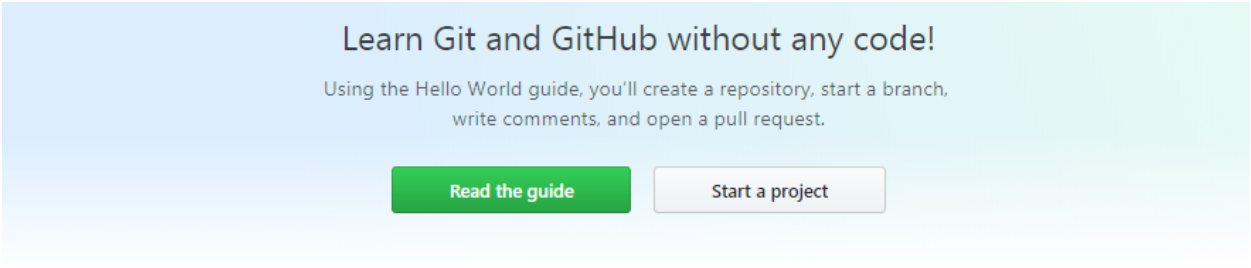
7. Upload to remote repository. Create an account remote repositories like https://github.com. Please note github is an external repository and no code relating to business should be uploaded here.

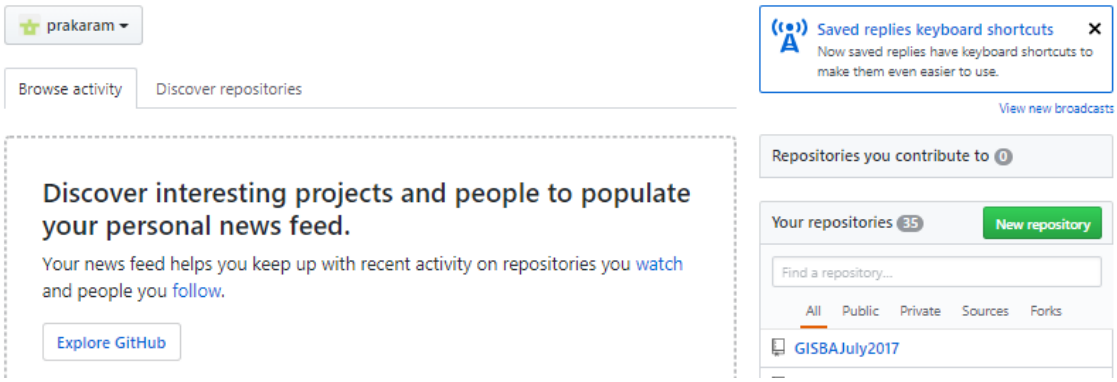


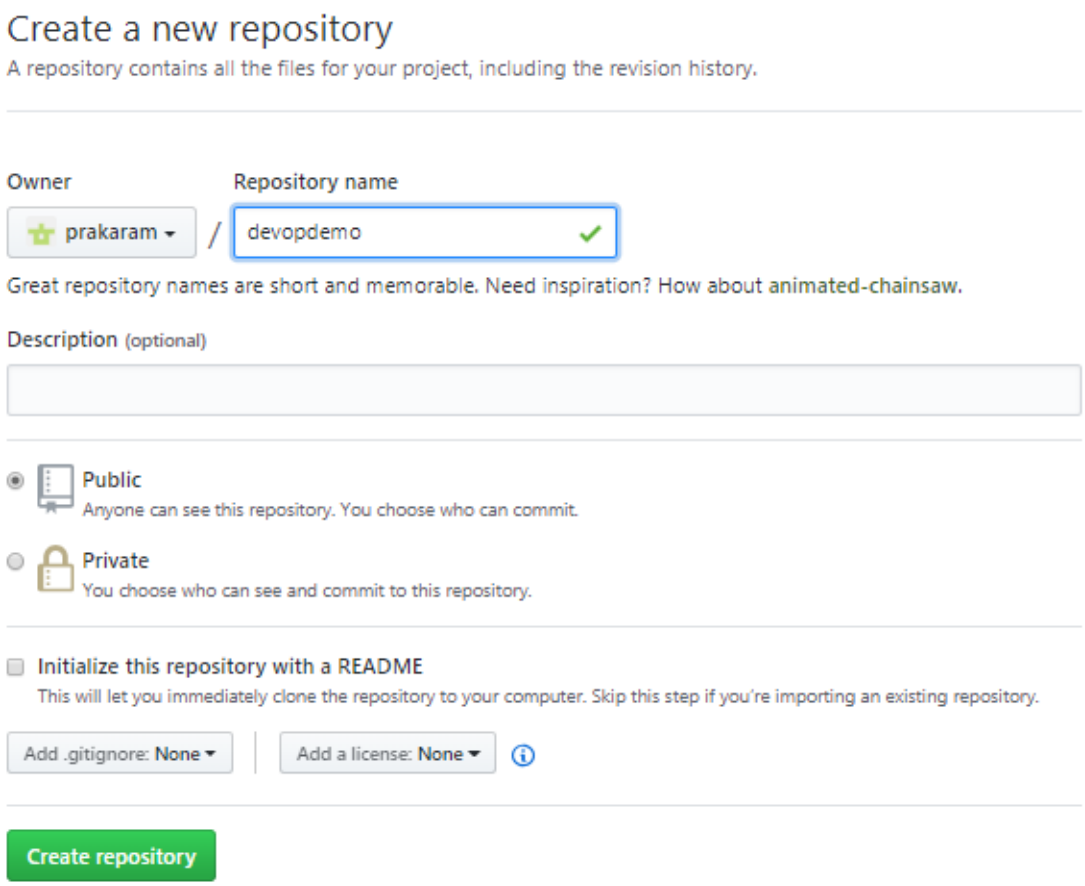
Login to the account:



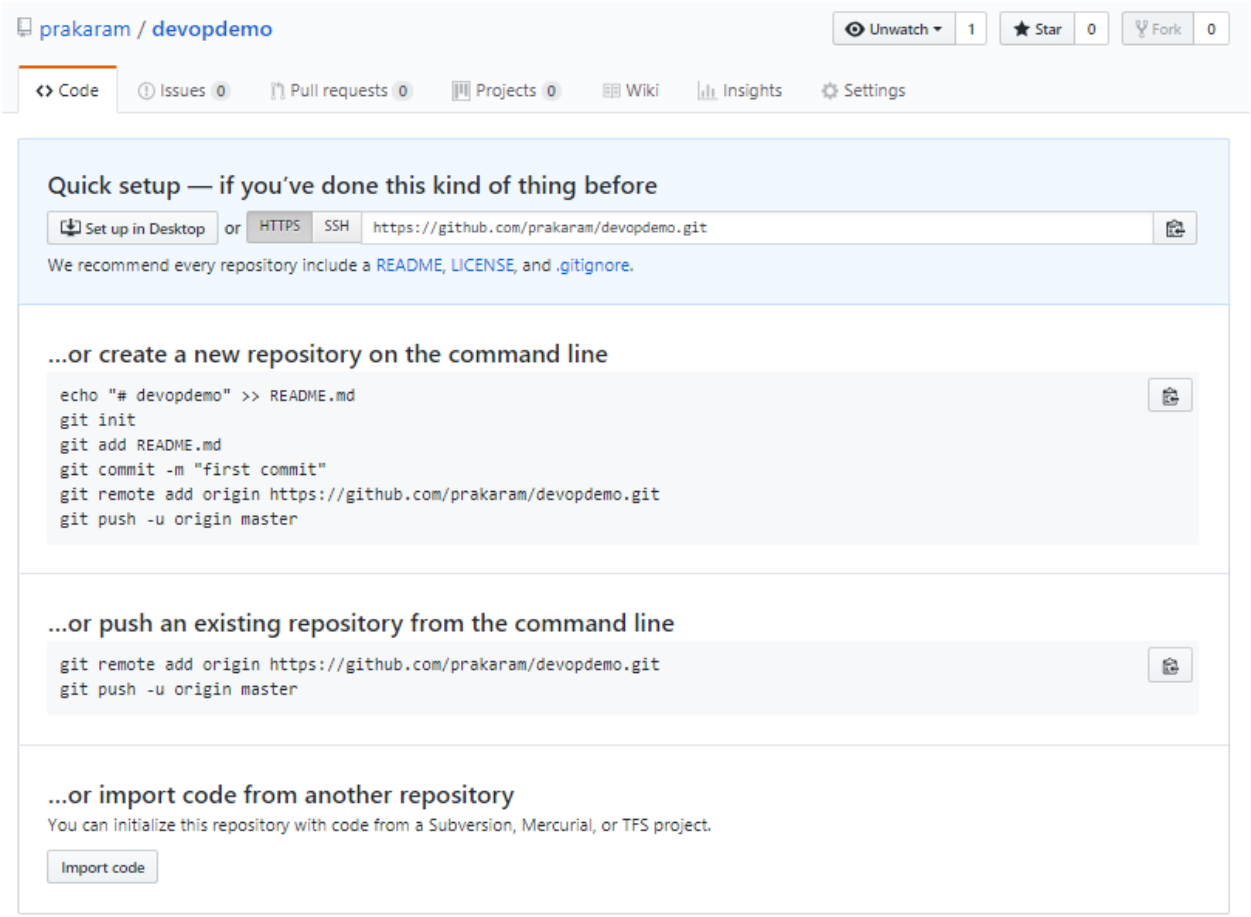
Create a new repository, by clicking on “New Repository”


Learn Git and GitHub without any code!
Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.
Read the guide
Start a project


prakaram
Browse activity
Discover repositories
Discover interesting projects and people to populate your personal news feed.
Your news feed helps you keep up with recent activity on repositories you watch and people you follow.
Explore GitHub
Saved replies keyboard shortcuts
Now saved replies have keyboard shortcuts to make them even easier to use.
View new broadcasts
Repositories you contribute to 0
Your repositories 35
New repository
Find a repository...
All
Public
Private
Sources
Forks
GISBAJuly2017



As you click on “Create repository” you would arrive at following screen:
Collect the https URL from the text box.
In this demonstration the URL is
https://github.com/prakaram/devopdemo.git



---

8. Add the URL of this remote repository to our local repository:

```
[devops@centserver gitlab]$ git remote add origin
https://github.com/prakaram/devopdemo.git

[devops@centserver gitlab]$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
[user]
        name = devop
        email = devop@devops.com
[remote "origin"]
        url = https://github.com/prakaram/devopdemo.git
        fetch = +refs/heads/*:refs/remotes/origin/*
```

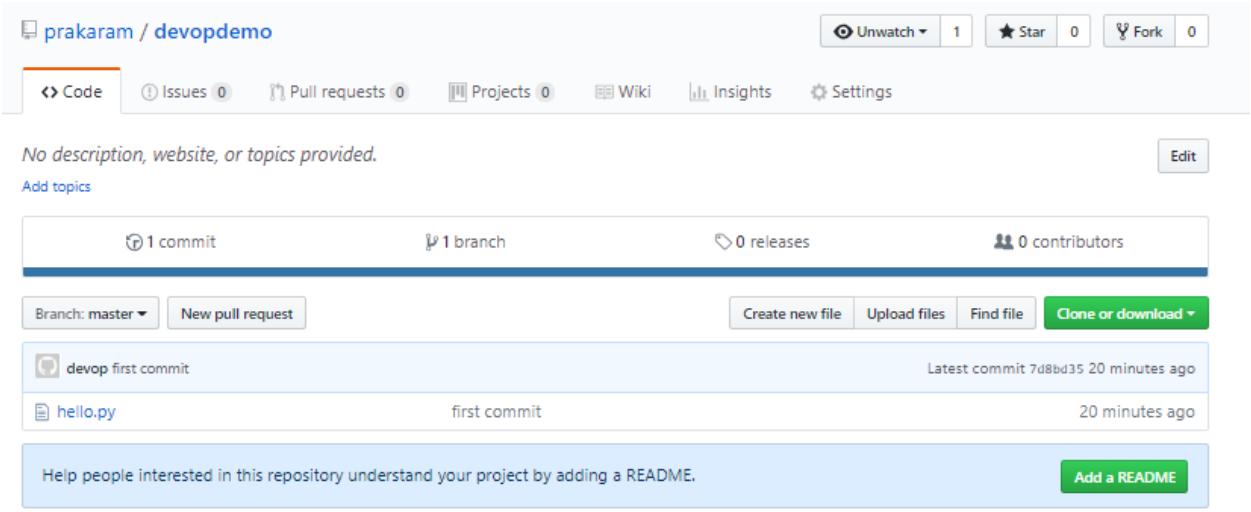
---

9. Push the local repository to this remote repository. Provide your account credentials.

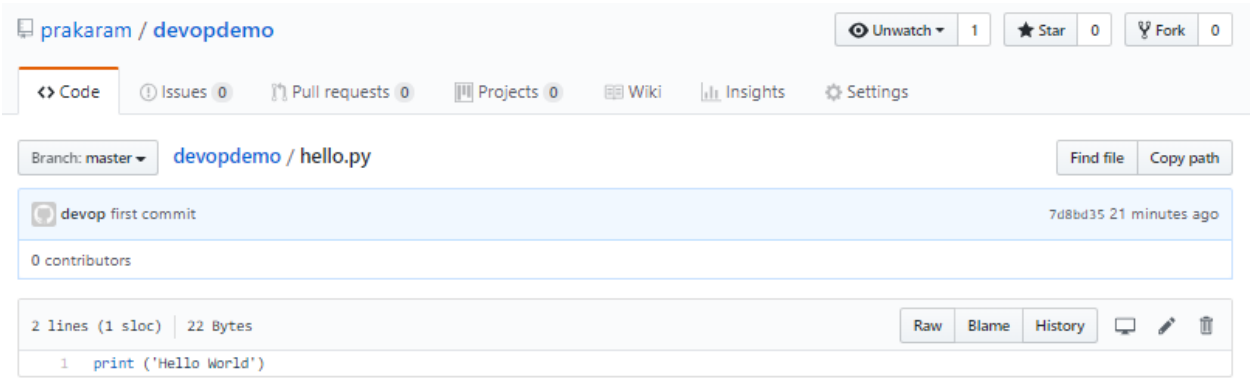
```
[devops@centserver gitlab]$ git push -u origin master
Username for 'https://github.com': prakaram
Password for 'https://prakaram@github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 224 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/prakaram/devopdemo.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

---

10. Watch the code availability on Remote repository. Refresh the page.



Click on hello.py to check the code:

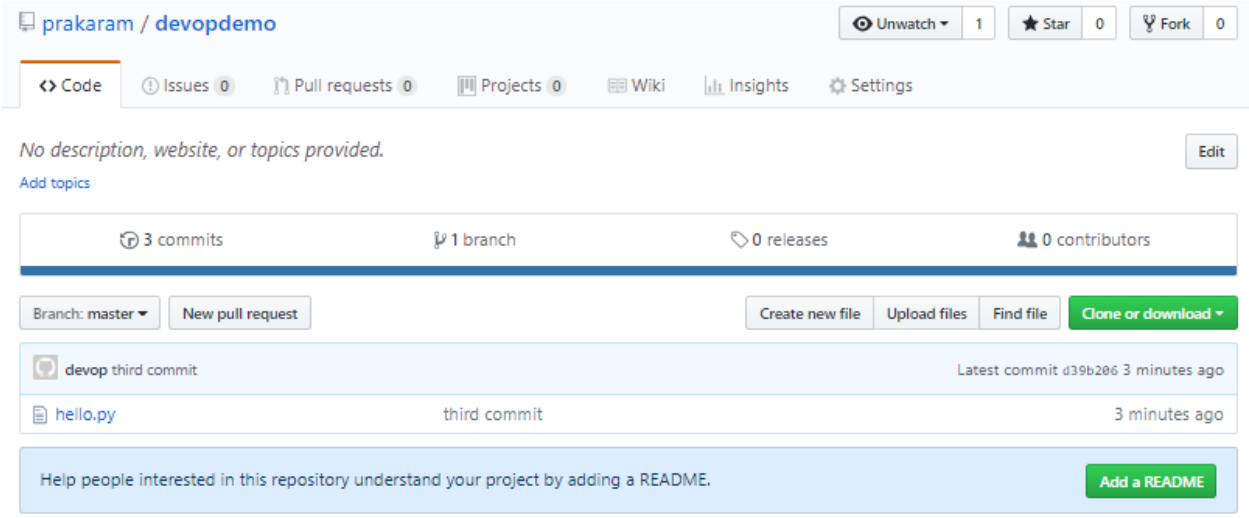


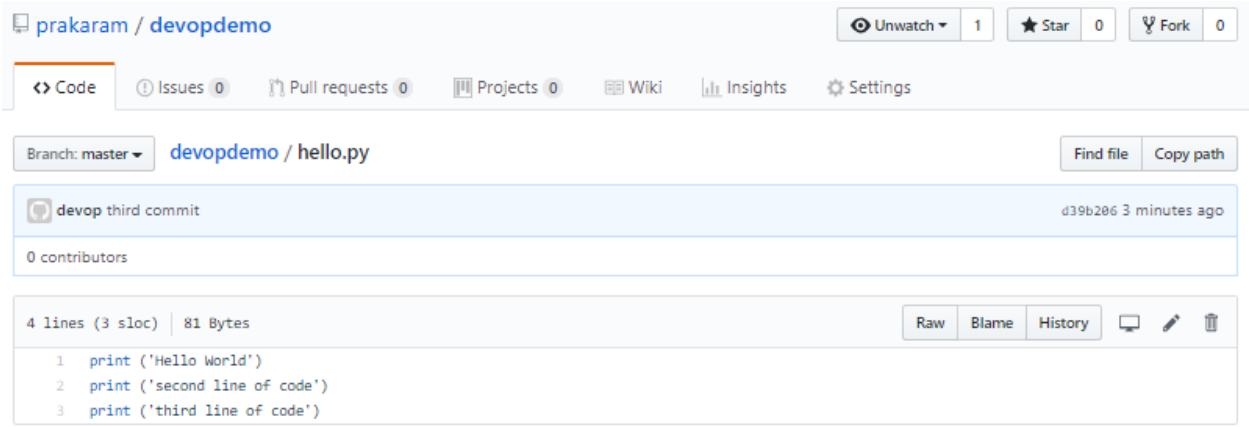
---

11. Make couple of more commits:

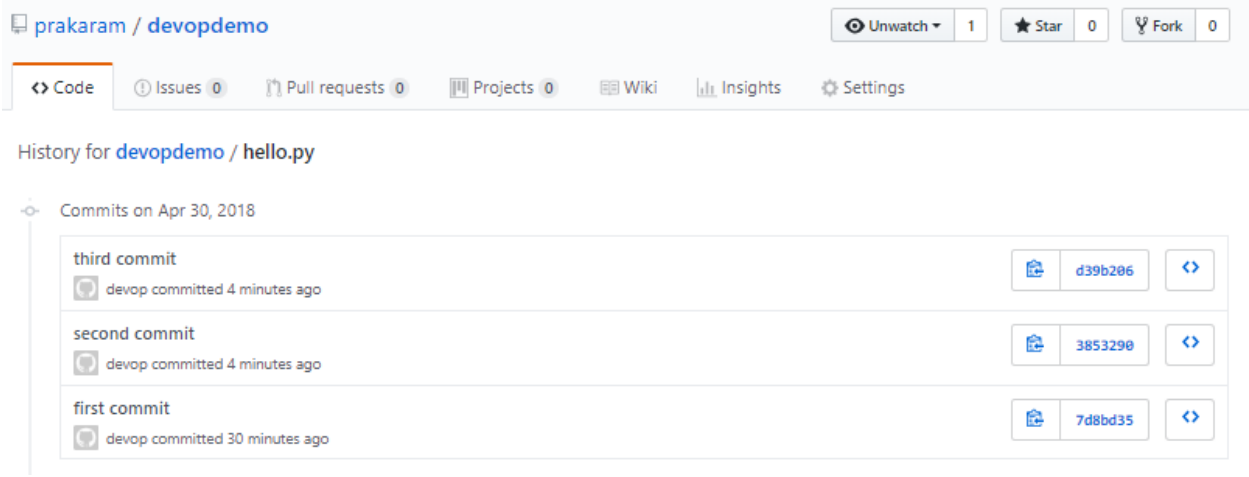
```
[devops@centserver gitlab]$ cat hello.py
print ('Hello World')
print ('second line of code')

[devops@centserver gitlab]$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   hello.py
#
no changes added to commit (use "git add" and/or "git commit -a")

[devops@centserver gitlab]$ git status -s
 M hello.py

[devops@centserver gitlab]$ git add hello.py

[devops@centserver gitlab]$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   hello.py
#

[devops@centserver gitlab]$ git status -s
M  hello.py

[devops@centserver gitlab]$ git commit -m "second commit"
[master 3853290] second commit
 1 file changed, 1 insertion(+)

[devops@centserver gitlab]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')

[devops@centserver gitlab]$ git commit -a -m "third commit"
[master d39b206] third commit
 1 file changed, 1 insertion(+)
```

```
[devops@centserver gitlab]$ git log
commit d39b2067f49b7803e445729c30c1f1a2f6dc44d8
Author: devop <devop@devops.com>
Date:   Mon Apr 30 12:41:24 2018 +0530

    third commit

commit 3853290ee592f8375194162df2045a7512490adf
Author: devop <devop@devops.com>
Date:   Mon Apr 30 12:40:39 2018 +0530

    second commit

commit 7d8bd35a28a345f57777d83122508ec4a21f1d79
Author: devop <devop@devops.com>
Date:   Mon Apr 30 12:15:18 2018 +0530

    first commit

[devops@centserver gitlab]$ git push -u origin master
Username for 'https://github.com': prakaram
Password for 'https://prakaram@github.com':
Counting objects: 8, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 486 bytes | 0 bytes/s, done.
Total 6 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/prakaram/devopdemo.git
   7d8bd35..d39b206  master -> master
Branch master set up to track remote branch master from origin.
```

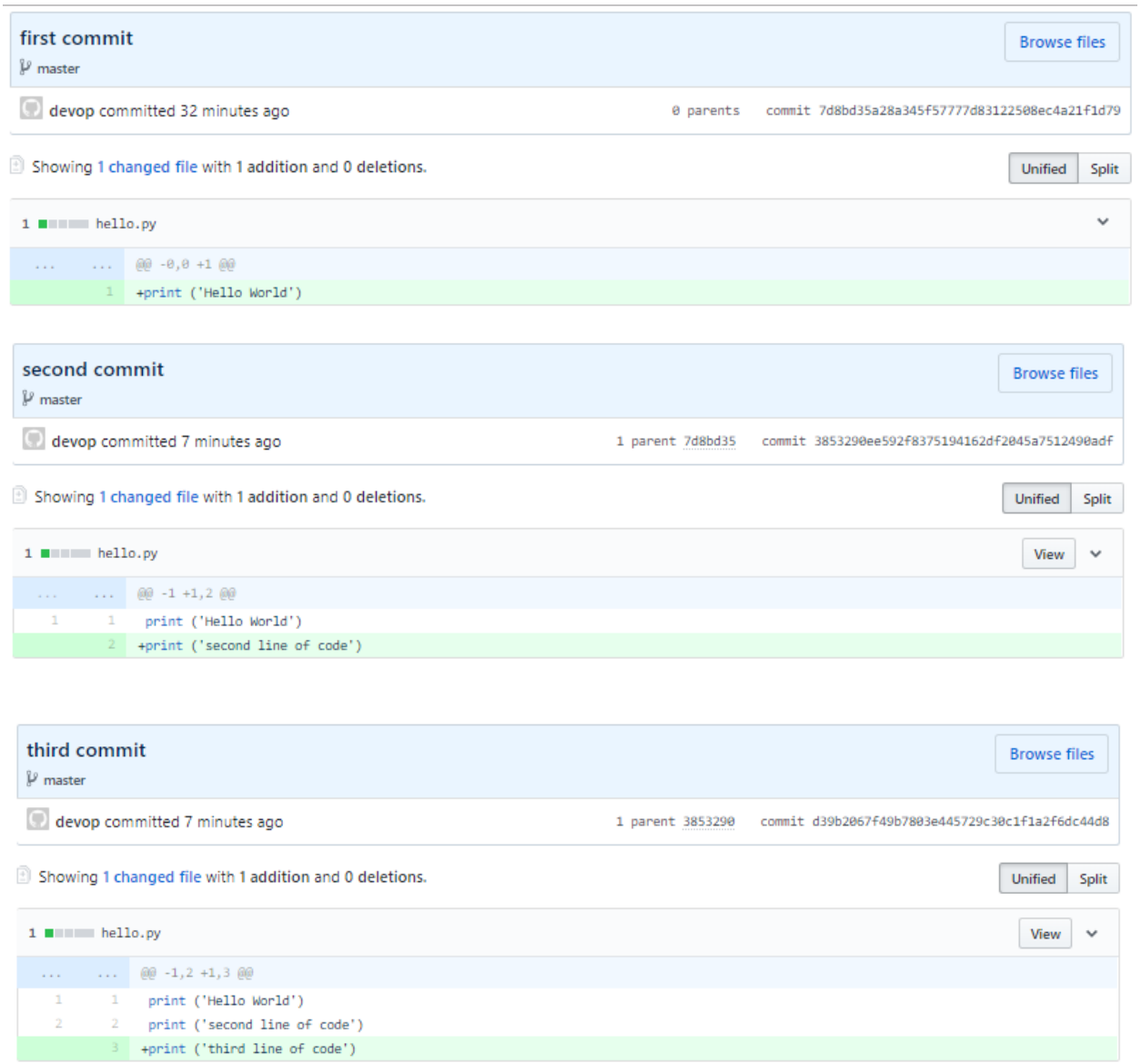




Click on “history” to get different versions of code:



Version number (short code) is displayed on the right click on them to check how code has changed.



---

12. Better log command with pretty printing.

%h: abbreviated hash of the commit
%ad: commit date
%s: comment
%an: name of the author

```
[devops@centserver gitlab]$ git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
d39b206 2018-04-30 | third commit (HEAD, origin/master, master) [devop]
3853290 2018-04-30 | second commit [devop]
7d8bd35 2018-04-30 | first commit [devop]
```

---

13. Moving across the versions:

```
[devops@centserver gitlab]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')

[devops@centserver gitlab]$ git checkout 7d8bd35
Note: checking out '7d8bd35'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at 7d8bd35... first commit

[devops@centserver gitlab]$ git log --pretty=format:"%h %ad | %s%d
[%an]" --date=short
7d8bd35 2018-04-30 | first commit (HEAD) [devop]

[devops@centserver gitlab]$ cat hello.py
print ('Hello World')

[devops@centserver gitlab]$ git checkout 3853290
Previous HEAD position was 7d8bd35... first commit
HEAD is now at 3853290... second commit

[devops@centserver gitlab]$ git log --pretty=format:"%h %ad | %s%d
[%an]" --date=short
3853290 2018-04-30 | second commit (HEAD) [devop]
7d8bd35 2018-04-30 | first commit [devop]

[devops@centserver gitlab]$ cat hello.py
print ('Hello World')
print ('second line of code')

[devops@centserver gitlab]$ git checkout d39b206
Previous HEAD position was 3853290... second commit
HEAD is now at d39b206... third commit

[devops@centserver gitlab]$ git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
d39b206 2018-04-30 | third commit (HEAD, origin/master, master) [devop]
3853290 2018-04-30 | second commit [devop]
```

7d8bd35 2018-04-30 | first commit [devop]

[devops@centserver gitlab]$ cat hello.py print ('Hello World')
print ('second line of code')
print ('third line of code')

---

14. Tag the important version:

```
[devops@centserver gitlab]$ git tag v1.0

[devops@centserver gitlab]$ git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
d39b206 2018-04-30 | third commit (HEAD, tag: v1.0, origin/master, master) [devo
3853290 2018-04-30 | second commit [devop]
7d8bd35 2018-04-30 | first commit [devop]

[devops@centserver gitlab]$ git checkout 7d8bd35
Previous HEAD position was d39b206... third commit
HEAD is now at 7d8bd35... first commit

[devops@centserver gitlab]$ git checkout v1.0
Previous HEAD position was 7d8bd35... first commit
HEAD is now at d39b206... third commit
```

Use the tag for previous version:

```
[devops@centserver gitlab]$ git checkout v1.0^
Previous HEAD position was d39b206... third commit
HEAD is now at 3853290... second commit

[devops@centserver gitlab]$ git checkout v1.0
Previous HEAD position was 3853290... second commit
HEAD is now at d39b206... third commit
```

---

15. Reversing the Staging before commit:

```
[devops@centserver gitlab]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
An unwanted chage

[devops@centserver gitlab]$ git status
# HEAD detached at v1.0
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   hello.py
#
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

[devops@centserver gitlab]$ git add hello.py

[devops@centserver gitlab]$ git status
HEAD detached at v1.0
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)
#
modified: hello.py
#

[devops@centserver gitlab]$ git reset HEAD hello.py
Unstaged changes after reset:
M   hello.py

[devops@centserver gitlab]$ git status
HEAD detached at v1.0
Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)
#
modified: hello.py
#
no changes added to commit (use "git add" and/or "git commit -a")

16. Reversing commit:

```
[devops@centserver gitlab]$ git add hello.py

[devops@centserver gitlab]$ git status
# HEAD detached at v1.0
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   hello.py
#

[devops@centserver gitlab]$ git commit -m "made commit"
[detached HEAD b1f072d] made commit
 1 file changed, 1 insertion(+)

[devops@centserver gitlab]$ git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
b1f072d 2018-04-30 | made commit (HEAD) [devop]
d39b206 2018-04-30 | third commit (tag: v1.0, origin/master, master) [devop]
```

```
3853290 2018-04-30 | second commit [devop]
7d8bd35 2018-04-30 | first commit [devop]
```

Even the unwanted change is committed:

```
[devops@centserver gitlab]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
An unwanted chage

Revert the commit:

[devops@centserver gitlab]$ git revert HEAD
[detached HEAD 5a9b21e] Revert "made commit"
 1 file changed, 1 deletion(-)

[devops@centserver gitlab]$ git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
5a9b21e 2018-04-30 | Revert "made commit" (HEAD) [devop]
b1f072d 2018-04-30 | made commit [devop]
d39b206 2018-04-30 | third commit (tag: v1.0, origin/master, master) [devop]
3853290 2018-04-30 | second commit [devop]
7d8bd35 2018-04-30 | first commit [devop]

Log indicate the reversal of commit, and the same reflected in the file, but the
commit still visible.

[devops@centserver gitlab]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')

Erase the commit from the log:

[devops@centserver gitlab]$ git reset --hard v1.0
HEAD is now at d39b206 third commit

[devops@centserver gitlab]$ git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
d39b206 2018-04-30 | third commit (HEAD, tag: v1.0, origin/master, master) [devo
3853290 2018-04-30 | second commit [devop]
7d8bd35 2018-04-30 | first commit [devop]
(END)
```

Now the commit information is completely erased.
Instead of completely erasing you may also amend the commit by making required change to code.

```
[devops@centserver gitlab]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
An unwanted change

[devops@centserver gitlab]$ git commit -a -m "Made change"
[detached HEAD 8e1546f] Made change
 1 file changed, 1 insertion(+)

[devops@centserver gitlab]$ git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
```

```
8e1546f 2018-04-30 | Made change (HEAD) [devop]
d39b206 2018-04-30 | third commit (tag: v1.0, origin/master, master) [devop]
3853290 2018-04-30 | second commit [devop]
7d8bd35 2018-04-30 | first commit [devop]

Committing again after rectifying the mistake:

[devops@centserver gitlab]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
print ('Made Proper change')

[devops@centserver gitlab]$ git commit -a --amend -m "proper change"
[detached HEAD b305a1f] proper change
 1 file changed, 1 insertion(+)

[devops@centserver gitlab]$ git log --pretty=format:"%h %ad | %s%d [%an]" --date=short
b305a1f 2018-04-30 | proper change (HEAD) [devop]
d39b206 2018-04-30 | third commit (tag: v1.0, origin/master, master) [devop]
3853290 2018-04-30 | second commit [devop]
7d8bd35 2018-04-30 | first commit [devop]

[devops@centserver gitlab]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
print ('Made Proper change')
```

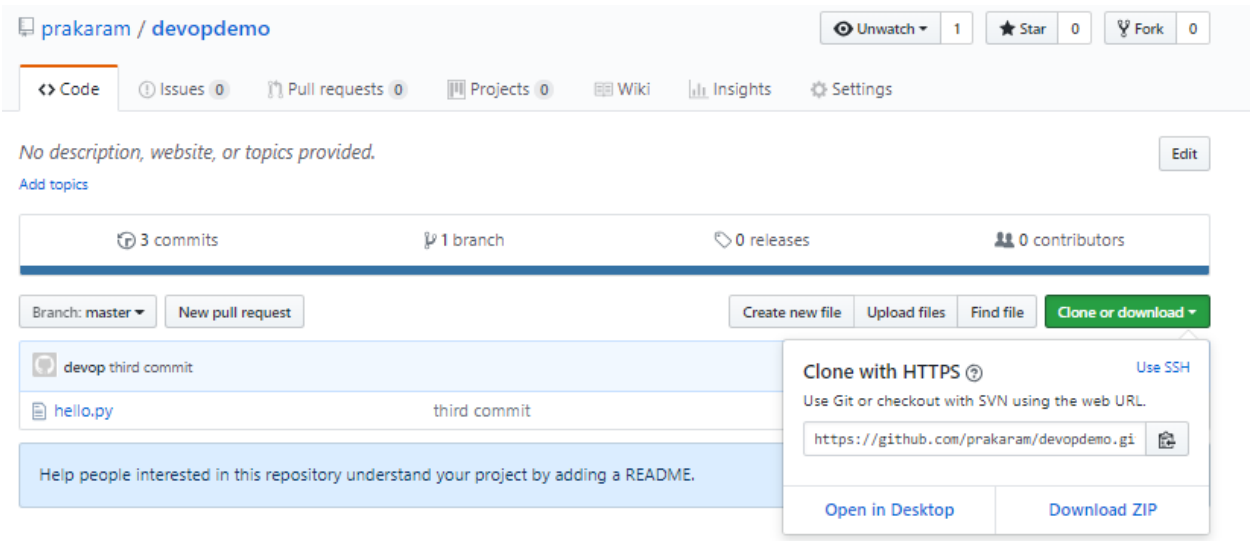
17. Making change to folder

If files are moved, removed using normal mv, rm command then those changes need to be first staged and then committed to local repo. Instead if “git mv” or “git rm” commands are used, the changes are already staged and it just requires only commit

```
[devops@centserver gitlab]$ git status
# HEAD detached from v1.0
nothing to commit, working directory clean

[devops@centserver gitlab]$ mv hello.py ./lib

[devops@centserver gitlab]$ git status
# HEAD detached from v1.0
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working
directory)
#
#       deleted:    hello.py
#
# Untracked files:
```

```
#   (use "git add <file>..." to include in what will be committed)
#
#       lib/
no changes added to commit (use "git add" and/or "git commit -a")

[devops@centserver gitlab]$ mv ./lib/hello.py .

[devops@centserver gitlab]$ git status
# HEAD detached from v1.0
nothing to commit, working directory clean

[devops@centserver gitlab]$ git mv hello.py ./lib

[devops@centserver gitlab]$ git status
# HEAD detached from v1.0
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       renamed:    hello.py -> lib/hello.py
#
```

Get the URL of remote repository:




```
[devops@centserver ~]$ mkdir gitrepo
[devops@centserver ~]$ cd gitrepo
[devops@centserver gitrepo]$ ls -a
.  ..
[devops@centserver gitrepo]$ git init
Initialized empty Git repository in /home/devops/gitrepo/.git/
[devops@centserver gitrepo]$ ls -a
.  ..  .git
```

```
[devops@centserver gitrepo]$ git remote add datacon
https://github.com/prakaram/devopdemo.git
[devops@centserver gitrepo]$ git fetch datacon
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9 (delta 1), reused 9 (delta 1), pack-reused 0
Unpacking objects: 100% (9/9), done.
From https://github.com/prakaram/devopdemo
 * [new branch]      master     -> datacon/master
[devops@centserver gitrepo]$ git checkout master
Branch master set up to track remote branch master from datacon.
Already on 'master'
[devops@centserver gitrepo]$ ls -a
.  ..  .git  hello.py
[devops@centserver gitrepo]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
[devops@centserver gitrepo]$ git log --pretty=format:"%h %ad | %s%d [%an]" --
date=short
d39b206 2018-04-30 | third commit (HEAD, datacon/master, master) [devop]
3853290 2018-04-30 | second commit [devop]
7d8bd35 2018-04-30 | first commit [devop]

[devops@centserver ~]$ rm -rf gitrepo

[devops@centserver ~]$ mkdir gitrepo

[devops@centserver ~]$ cd gitrepo

[devops@centserver gitrepo]$ ls -a
.  ..
[devops@centserver gitrepo]$ git init
Initialized empty Git repository in /home/devops/gitrepo/.git/

[devops@centserver gitrepo]$ ls -a
.  ..  .git

[devops@centserver gitrepo]$ git remote add datacon
https://github.com/prakaram/devopdemo.git

[devops@centserver gitrepo]$ git pull datacon master
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9 (delta 1), reused 9 (delta 1), pack-reused 0
Unpacking objects: 100% (9/9), done.
From https://github.com/prakaram/devopdemo
 * branch            master     -> FETCH_HEAD

[devops@centserver gitrepo]$ ls -a
.  ..  .git  hello.py

[devops@centserver gitrepo]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
```

```
[devops@centserver gitrepo]$ git log --pretty=format:"%h %ad | %s%d [%an]" --
date=short
d39b206 2018-04-30 | third commit (HEAD, master) [devop]
3853290 2018-04-30 | second commit [devop]
7d8bd35 2018-04-30 | first commit [devop]

Alternately you can also clone the entire repository to your local machine:

[devops@centserver ~]$ git clone https://github.com/prakaram/devopdemo.git
Cloning into 'devopdemo'...
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 9 (delta 1), reused 9 (delta 1), pack-reused 0
Unpacking objects: 100% (9/9), done.

[devops@centserver ~]$ ls
devopdemo  gitlab  gitrepo

[devops@centserver ~]$ cd devopdemo/

[devops@centserver devopdemo]$ ls
hello.py

[devops@centserver devopdemo]$ ls -a
.  ..  .git  hello.py

[devops@centserver devopdemo]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')

[devops@centserver devopdemo]$ git log --pretty=format:"%h %ad | %s%d [%an]" --
date=short
d39b206 2018-04-30 | third commit (HEAD, origin/master, origin/HEAD, master) [de
3853290 2018-04-30 | second commit [devop]
7d8bd35 2018-04-30 | first commit [devop]
```

Branching and Merging:

```
[devops@centserver ~]$ cd devopdemo/

[devops@centserver devopdemo]$ ls
hello.py

[devops@centserver devopdemo]$ ls -a
. .. .git  hello.py

[devops@centserver devopdemo]$ git branch
* master

[devops@centserver devopdemo]$ git branch newfeature

[devops@centserver devopdemo]$ git branch
```

```
* master
  newfeature

[devops@centserver devopdemo]$ git checkout newfeature
Switched to branch 'newfeature'

[devops@centserver devopdemo]$ git branch
  master
* newfeature

[devops@centserver devopdemo]$ git config --global user.name devop
[devops@centserver devopdemo]$ git config --global user.email devop@devop.com

[devops@centserver ~]$ cat .gitconfig
[user]
	name = devop
	email = devop@devop.com

[devops@centserver devopdemo]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
print ('Adding another feature')

[devops@centserver devopdemo]$ git commit -a -m "Adding new feature"
[newfeature 82618a8] Adding new feature
 1 file changed, 1 insertion(+)

[devops@centserver devopdemo]$ git log --pretty=format:"%h %cd | %s%d [%an]" --date=short
82618a8 2018-04-30 | Adding new feature (HEAD, newfeature) [devop]
d39b206 2018-04-30 | third commit (origin/master, origin/HEAD, master) [devop]
3853290 2018-04-30 | second commit [devop]
7d8bd35 2018-04-30 | first commit [devop]

[devops@centserver devopdemo]$ git branch
  master
* newfeature

[devops@centserver devopdemo]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
print ('Adding another feature')
```

```
[devops@centserver devopdemo]$ git checkout master
Switched to branch 'master'

[devops@centserver devopdemo]$ git branch
* master
  newfeature

[devops@centserver devopdemo]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')

[devops@centserver devopdemo]$ git merge newfeature
Updating d39b206..82618a8
Fast-forward
 hello.py | 1 +
 1 file changed, 1 insertion(+)

[devops@centserver devopdemo]$ git checkout newfeature
Switched to branch 'newfeature'

[devops@centserver devopdemo]$ git branch
  master
* newfeature

[devops@centserver devopdemo]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
print ('Adding another feature')

[devops@centserver devopdemo]$ git commit -a -m "fifth line newfeature"
[newfeature 1611201] fifth line newfeature
 1 file changed, 1 insertion(+)

[devops@centserver devopdemo]$ git branch
  master
* newfeature

[devops@centserver devopdemo]$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
```

```
[devops@centserver devopdemo]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
print ('Adding another feature')
print ('fifth line of print')

[devops@centserver devopdemo]$ git commit -a -m "fifth line master"
[master 8d41b2c] fifth line master
 1 file changed, 1 insertion(+)

[devops@centserver devopdemo]$ git merge newfeature
Auto-merging hello.py
CONFLICT (content): Merge conflict in hello.py
Automatic merge failed; fix conflicts and then commit the result.

[devops@centserver devopdemo]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
print ('Adding another feature')
<<<<<<< HEAD
print ('fifth line of print')
=======
print ('new feature on line 5')
>>>>>>> newfeature

[devops@centserver devopdemo]$ git checkout newfeature
hello.py: needs merge
error: you need to resolve your current index first

[devops@centserver devopdemo]$ git branch
* master
  Newfeature

[devops@centserver devopdemo]$ git merge --abort

[devops@centserver devopdemo]$ git branch
* master
  newfeature

[devops@centserver devopdemo]$ cat hello.py
print ('Hello World')
print ('second line of code')
```

```
print ('third line of code')
print ('Adding another feature')
print ('fifth line of print')

[devops@centserver devopdemo]$ git checkout newfeature
Switched to branch 'newfeature'

[devops@centserver devopdemo]$ git branch
  master
* newfeature

[devops@centserver devopdemo]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
print ('Adding another feature')
print ('new feature on line 5')

[devops@centserver devopdemo]$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

[devops@centserver devopdemo]$ git branch
* master
  newfeature

[devops@centserver devopdemo]$ git merge -s ours newfeature
Merge made by the 'ours' strategy.
```

Note: ours for current branch and theirs for the other branch

```
[devops@centserver devopdemo]$ cat hello.py
print ('Hello World')
print ('second line of code')
print ('third line of code')
print ('Adding another feature')
print ('fifth line of print')
```

How to avoid conflicts:

```
[devops@centserver devopdemo]$ git branch
* master
  newfeature
```

```
[devops@centserver devopdemo]$ ls -a
.  ..  .git  hello.py

[devops@centserver devopdemo]$ cat file_on_master
This is file on master

[devops@centserver devopdemo]$ git add file_on_master

[devops@centserver devopdemo]$ git commit -m "file on master"
[master 6946f24] file on master
 1 file changed, 1 insertion(+)
 create mode 100644 file_on_master

[devops@centserver devopdemo]$ ls
file_on_master  hello.py

[devops@centserver devopdemo]$ git checkout newfeature
Switched to branch 'newfeature'

[devops@centserver devopdemo]$ git branch
  master
* newfeature

[devops@centserver devopdemo]$ vi added_feature

[devops@centserver devopdemo]$ cat added_feature
This will have added feature

[devops@centserver devopdemo]$ git add added_feature

[devops@centserver devopdemo]$ git commit -m "added feature"
[newfeature 085614b] added feature
 1 file changed, 1 insertion(+)
 create mode 100644 added_feature

[devops@centserver devopdemo]$ ls
added_feature  hello.py


[devops@centserver devopdemo]$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 5 commits.
  (use "git push" to publish your local commits)
```

```
[devops@centserver devopdemo]$

[devops@centserver devopdemo]$ ls
file_on_master  hello.py

[devops@centserver devopdemo]$ git merge newfeature
Merge made by the 'recursive' strategy.
 added_feature | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 added_feature

[devops@centserver devopdemo]$ ls
added_feature  file_on_master  hello.py
```

Last Step:

Execute the “exit” command to get the script done to generate log file. Push the file to remote repository “DevOpsTools” in your account.

```
osgdev@TG-DevOps-OS004:~$ exit
exit
Script done, file is Day2_wl.log
```