CLOUD TRAIN

ACCELERATE YOUR GROWTH

GIT RUNBOOK

DevOps Workshop

Contact us

TO ACCELERATE YOUR CAREER GROWTH

For questions and more details:

please call @ +91 98712 72900, or

visit https://www.thecloudtrain.com/, or

email at support@thecloudtrain.com, or

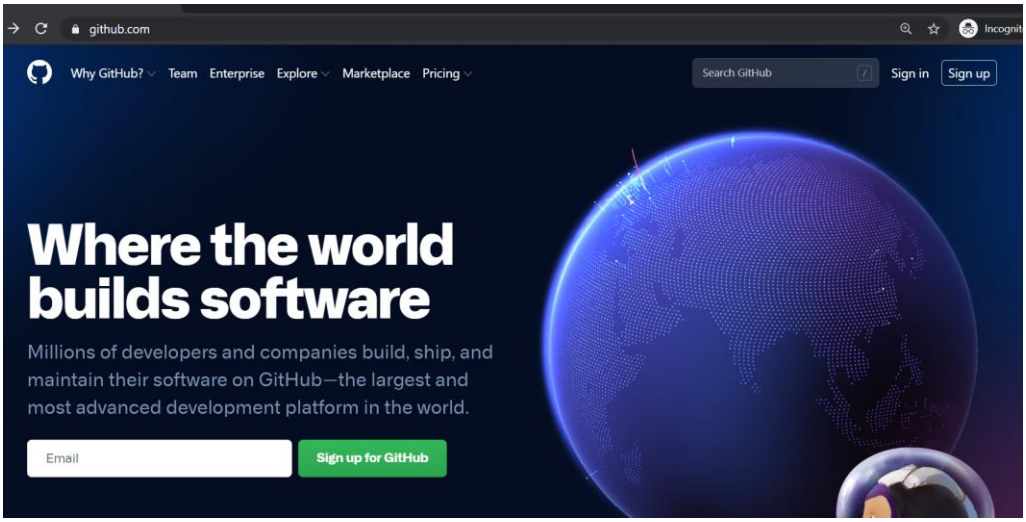
WhatsApp us @ +91 98712 72900

Exercise 1: Complete below tasks as part of this exercise:

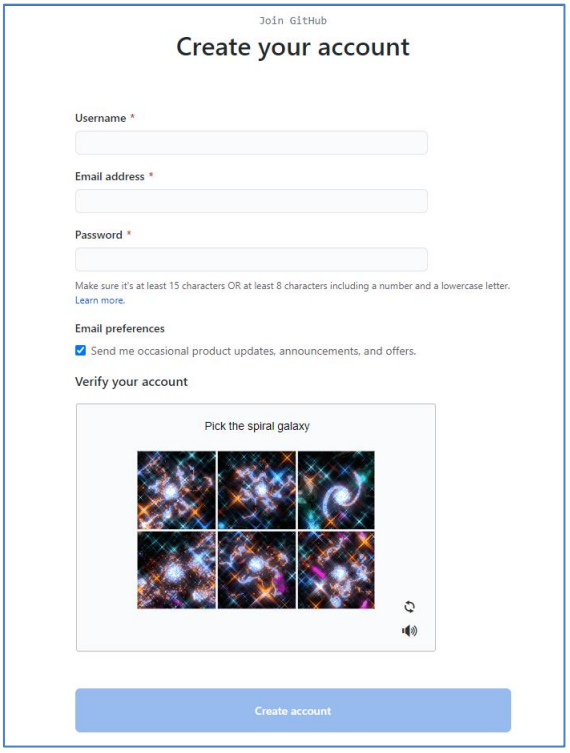
a) Setup a GitHub account and create one repository inside it named '**devopsdemo**'

Solution:

i. Open https://github.com/ in browser and click on signup:



ii. Fill the form by choosing one username, entering email and password for the Github account. Solve the puzzle to verify the account



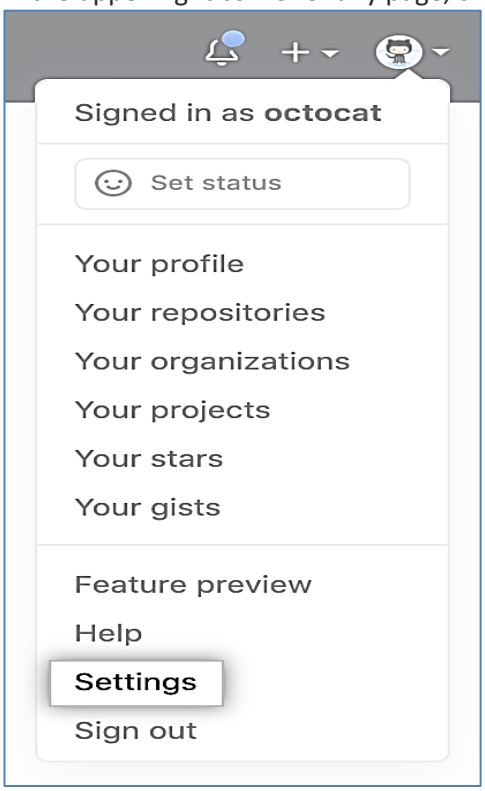
iii. Once all done, click on create account

iv. You must receive email from Github for account verification with verification link. Complete the verification to complete registration process.

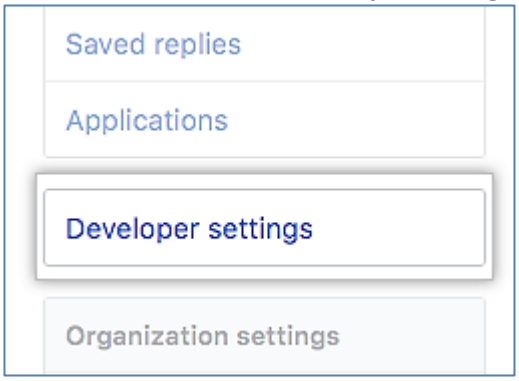
v. Once registration is complete, login to the Github account and generate Personal Access Token next:

Generate Personal Access Token for authentication(Step vi to xiii):

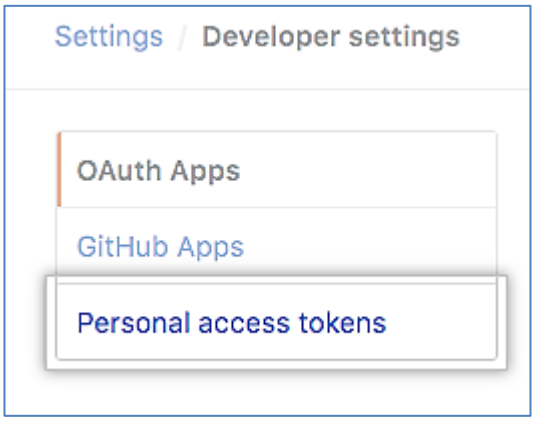
vi. In the upper-right corner of any page, click your profile photo, then click **Settings**.



vii. In the left sidebar, click **Developer settings**.

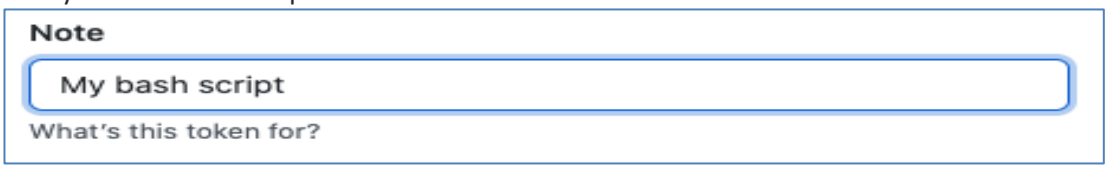


viii. In the left sidebar, click **Personal access tokens**.

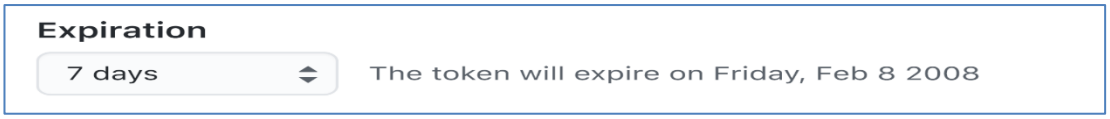


ix. Click **Generate new token**.

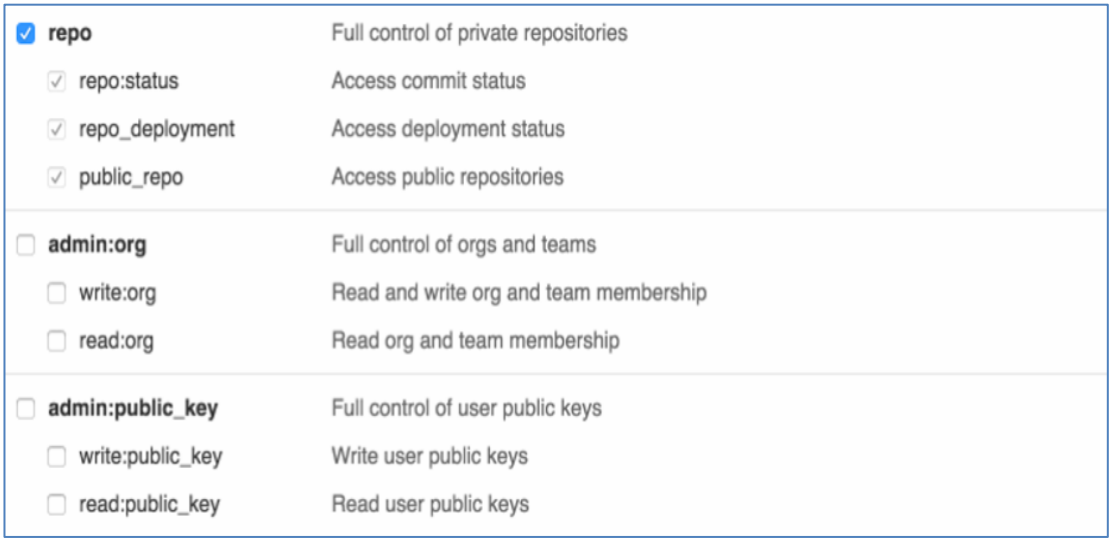
x. Give your token a descriptive name.



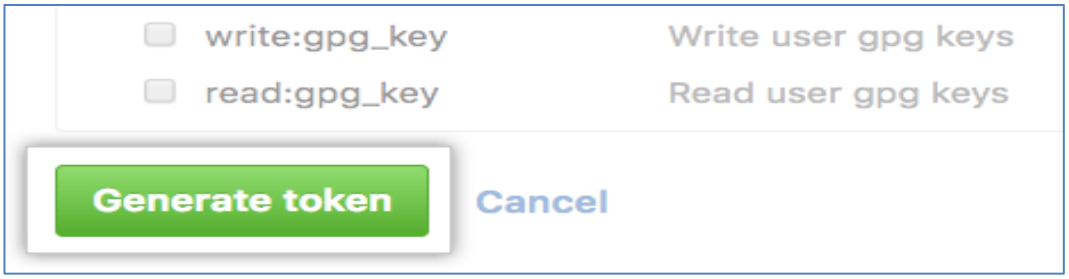
xi. To give your token an expiration, select the **Expiration** drop-down menu, then click a default or use the calendar picker.

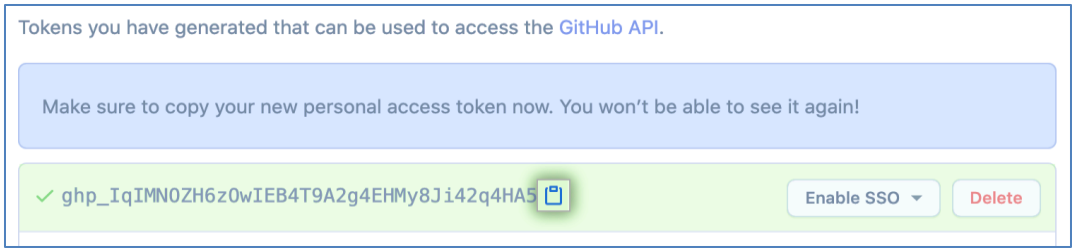


xii. Select the scopes, or permissions, you'd like to grant this token. To use your token to access repositories from the command line, select **repo**.



xiii. Click **Generate token**.



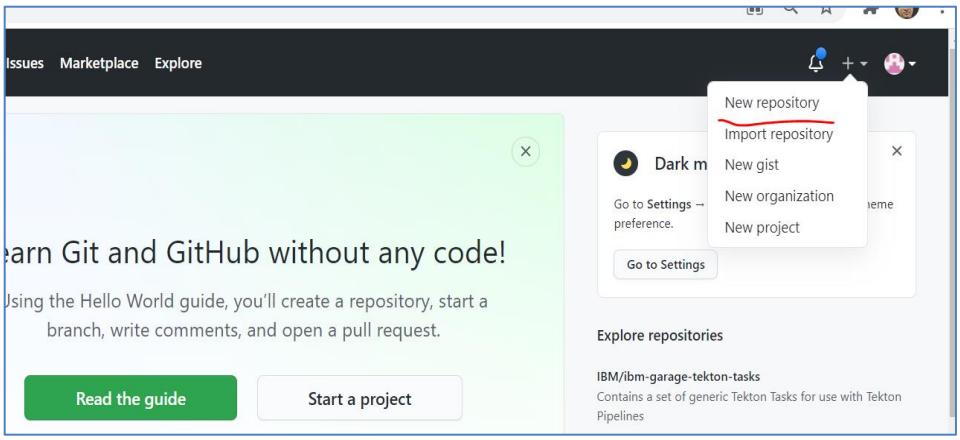


Warning: **Copy the token generated** and save it somewhere. Treat your tokens like passwords and keep them secret. When working with the API, use tokens as environment variables instead of hardcoding them into your programs.

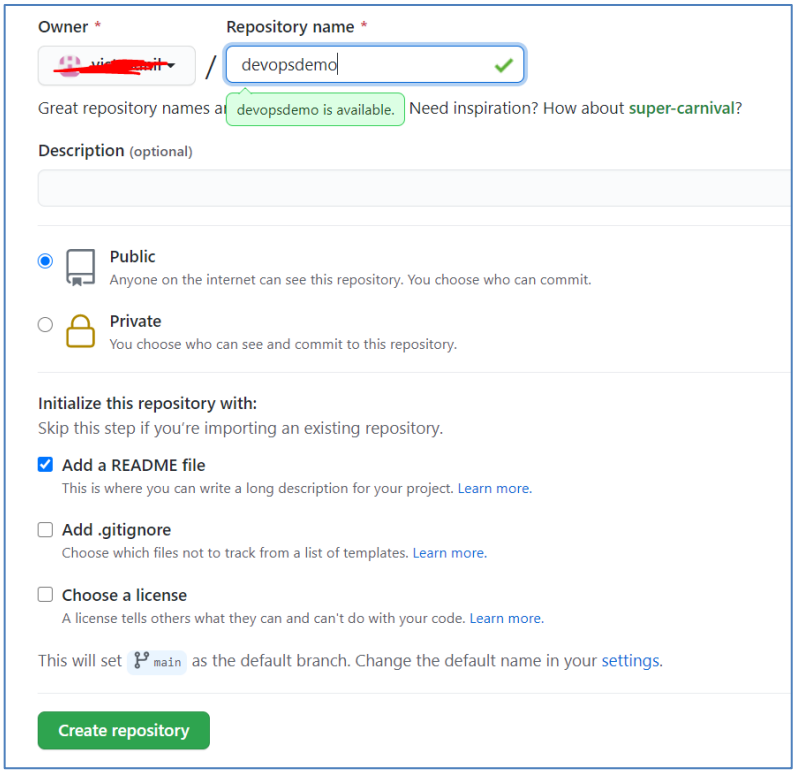
Example of using your access token instead of password, because password authentication is deprecated:

```
$ git push origin master
Username: your_username
Password: your_token
```

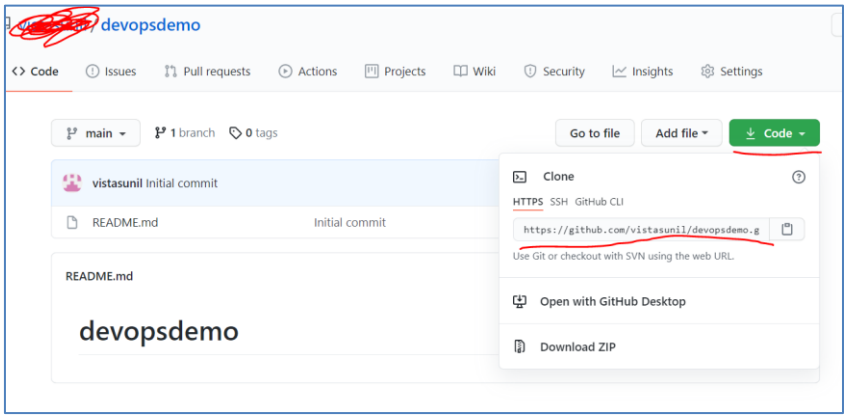
xiv. Create repository now by clicking on 'New repository':



xv. Fill the repo name as **devopsdemo** and click create repo as below:



xvi. Once repo is created, you will see page like below:



xvii. Copy the repo clone URL under code and go to **task b**.

b) Clone this repo '**devopsdemo'** to your GCP compute instance

Solution:

Login to your Ubuntu GCP instance with ubuntu user and run below command to clone:

```
sudo su - ubuntu
git clone https://github.com/vistasunil/devopsdemo.git
```

```
$ git clone https://github.com/vistasunil/devopsdemo.git
Cloning into 'devopsdemo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

c) cd to the repo directory you just cloned

Solution:

```
cd devopsdemo
```

```
$ cd devopsdemo/
```

d) Check the branch name you are checked out currently. It should be **master** by default.

Solution:

```
git branch
```

```
$ git branch
* main
```

e) Add two files using vim editor as below:

i. File1.txt

ii. File2.txt

Solution:

```
vim File1.txt
vim File2.txt
```

```
sunil.s.kumar@IN-2F7RX33 MINGW64 ~/devopsdemo (main)
$ vim File1.txt

sunil.s.kumar@IN-2F7RX33 MINGW64 ~/devopsdemo (main)
$ vim File2.txt

sunil.s.kumar@IN-2F7RX33 MINGW64 ~/devopsdemo (main)
$ cat File1.txt
file1

sunil.s.kumar@IN-2F7RX33 MINGW64 ~/devopsdemo (main)
$ cat File2.txt
file2
```

f) Check the git status

Solution:

```
git status
```

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        File1.txt
        File2.txt

nothing added to commit but untracked files present (use "git add" to track)
```

g) Add and Commit the changes to the repo

Solution:

```
git add .
```

```
$ git add .
```

```
git commit -m 'My first commit'
```

```
$ git commit -m 'My first commit'
[main 024f652] My first commit
 2 files changed, 2 insertions(+)
 create mode 100644 File1.txt
 create mode 100644 File2.txt
```

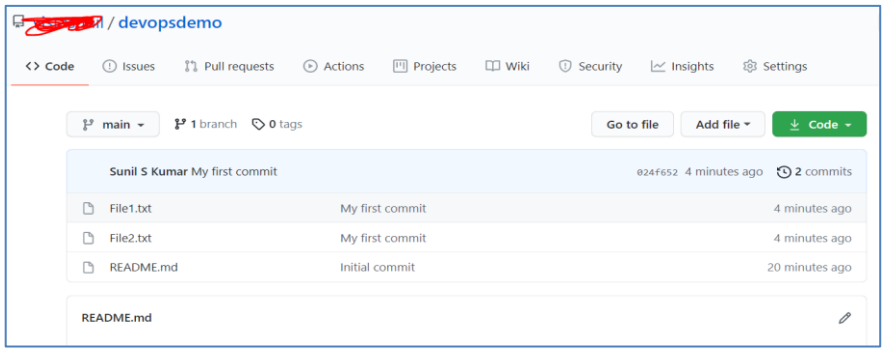
h) Push the changes to the repo 'devopsdemo' to github account

Solution:

```
git push
```

```
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 332 bytes | 83.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/vistasunil/devopsdemo.git
   c4904b8..024f652  main -> main
```

You will see files successfully pushed to Github account in the devopsdemo repo as below:



Exercise 2: Complete below tasks as part of this exercise:

a) cd to the repo directory '**devopsdemo**' on the server

Solution:

```
cd devopsdemo
```

```
$ cd devopsdemo/
```

b) create a new branch with name **feature1**

Solution:

```
git branch feature1
```

```
$ git branch feature1
```

c) Checkout to feature1 branch

Solution:

```
Git checkout feature1
```

```
$ Git checkout feature1
Switched to branch 'feature1'
```

d) Add two files using vim editor as below:
   i. File3.txt
   ii. File4.txt

Solution:

```
vim File1.txt

vim File2.txt
```

```
$ vim File3.txt
$ vim File4.txt
```

You see four files now in feature1 branch

```
$ ls -ltr
total 5
-rw-r--r-- 1 sunil.s.kumar 1049089 12 Apr 19 23:51 README.md
-rw-r--r-- 1 sunil.s.kumar 1049089  6 Apr 19 23:58 File1.txt
-rw-r--r-- 1 sunil.s.kumar 1049089  6 Apr 19 23:58 File2.txt
-rw-r--r-- 1 sunil.s.kumar 1049089  6 Apr 20 00:14 File3.txt
-rw-r--r-- 1 sunil.s.kumar 1049089  6 Apr 20 00:14 File4.txt
```

e) Check status, add files, commit and push to github account

Solution:

```
git add .
```

```
$ git add .
```

```
git commit -m 'My feature1 commit'
```

```
$ git commit -m 'My feature1 commit'
[feature1 80e4310] My feature1 commit
 2 files changed, 2 insertions(+)
 create mode 100644 File3.txt
 create mode 100644 File4.txt
```

f) Merge the changes in **feature1** branch to master branch

Solution:

```
git checkout main
```

```
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Before git merge you will see only old two files in main branch as below:

```
$ ls -ltr
total 3
-rw-r--r-- 1 sunil.s.kumar 1049089 12 Apr 19 23:51 README.md
-rw-r--r-- 1 sunil.s.kumar 1049089  6 Apr 19 23:58 File1.txt
-rw-r--r-- 1 sunil.s.kumar 1049089  6 Apr 19 23:58 File2.txt
```

Now merge feature1 to main branch

```
git merge feature1
```

```
$ git merge feature1
Updating 024f652..80e4310
Fast-forward
 File3.txt | 1 +
 File4.txt | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 File3.txt
 create mode 100644 File4.txt
```

g) Check if all changes from **feature1** branch are available under master branch after merge.

Solution:

After merge you will all files available from feature1 branch in main branch too.

```
ls -ltr
```

```
$ ls -ltr
total 5
-rw-r--r-- 1 sunil.s.kumar 1049089 12 Apr 19 23:51 README.md
-rw-r--r-- 1 sunil.s.kumar 1049089  6 Apr 19 23:58 File1.txt
-rw-r--r-- 1 sunil.s.kumar 1049089  6 Apr 19 23:58 File2.txt
-rw-r--r-- 1 sunil.s.kumar 1049089  7 Apr 20 00:16 File3.txt
-rw-r--r-- 1 sunil.s.kumar 1049089  7 Apr 20 00:16 File4.txt
```