SE489 DevOps Engineering

Lab 3

# Lab 3: GitHub

## Objective: After successful completion of this lab, students will understand concepts and commands of GitHub and would be able to develop code collaboratively.

## Open URL <https://github.com/new> use your credentials if you have already registered, if not, sign on, and then it will bring you to this interface, create a new repository

## Graphical user interface, text, application, email, website Description automatically generated

Graphical user interface, text, application

Description automatically generated

With this you have created a public repository in your account

1. Let’s make a SSH secure connection with this repository, this is necessary to secure your public repository from malicious or erroneous pushes. To achieve this, do followings
   1. Open git bash terminal
   2. Navigate to local repository
   3. On $ prompt, write **ssh-keygen**
   4. It will ask for **file name** to which key will be stored, don’t write anything and when it asks for passphrase also don’t provide anything, by default, key will be saved into id\_rsa.pub(public key) and id\_rsa(private key)

Text

Description automatically generated

1. The key shown here is masked, to see the complete key, let’s open the *sshkey* file, use command **cat file\_name** to print the contents of the file at terminal
   1. Text

      Description automatically generatedTo see the list of files in the repository, use **ls**.
   2. second last file contains private key, we don’t need it, the last file with .pub extension is the sought file containing public key, use cat command to open it.

Text

Description automatically generated

copy this key

Alternatively

**$cat id\_rsa.pub | clip,**  will copy the key into clipboard without displaying it

1. Now go back to GitHub website🡪Settings🡪SSH and GPG keys🡪New SSH key

Graphical user interface, text, application

Description automatically generated

1. Paste the copied key into the space given, assign it a name and then finally click on Add SSH key  
     
   Graphical user interface, text, application

   Description automatically generated
2. Following screen appears, showing the key added  
   Graphical user interface, text, application, email

   Description automatically generated
3. Let’s authenticate this key, go back to the git terminal, and run following command  
     
   Text

   Description automatically generated

Means we have authenticate the pair of keys we have generated. This can be verified from the website as well.

1. Let’s go back to the website, go to the repository we have created viz. DevOps, it will be showing only one branch as we didn’t do anything here

Graphical user interface, text, application, email

Description automatically generated

1. Graphical user interface, text, application, email

   Description automatically generatedCopying the url of the repository, click on the code button at the right side, click on SSH, and then click on the copy icon to copy the url of the repository. This is url of the remote repository, in which we will be doing remote operations.
2. Adding remote repository, **git remote add origin** command is used to add a remote repository to the local repository

Text

Description automatically generatedPS: “origin” in the command refers to the current repository, it may be any repository, it just adds a current repository into remote location, specified by url.

1. Now let’s push files from local repository to remote repository

Text

Description automatically generated

1. To make a local working copy of the remote repository, clone the repository with **git clone** command
   1. Write **ls** to check the contents of the master repo
   2. Use **git clone** command to get the local working repo of the remote repo
   3. Text

      Description automatically generatedCheck contents again by **ls**
2. Graphical user interface, text, application, email

   Description automatically generatedThese changes can be observed at GitHub website

Change branch from *main to master*

Graphical user interface, text, application, email

Description automatically generated

Clearly the commits are visible

Graphical user interface, text, application, email

Description automatically generated

1. Let’s make some changes into the remote repository
   1. Click at the **Add a README**
   2. Graphical user interface, application

      Description automatically generated with medium confidenceMake some changes into the shown readme file, e.g.

* 1. Commit new file

Graphical user interface, text, application, email

Description automatically generated

1. Now since we had made some changes in the remote repository (added README file), let’s make a pull request, to reflect those changes into the local repository

Text

Description automatically generatedClearly, we can see, README.md is now part of local repository.

1. Let’s make some changes into demo222.java and subsequently commit and push it.
   1. At $ prompt, write notepad demo222.java
   2. Add a simple for loop

A screenshot of a computer

Description automatically generated with medium confidence

* 1. Save the file

1. Add this file to the staging area, -u can be used to update only file that has changed  
   Text

   Description automatically generated
2. Now commit this file with message, “for loop added”  
   Text

   Description automatically generated
3. Let’s suppose after committing this change, user give their feedbacks, and we realize that we need to revert back to previous version of the code, then do the following steps
   1. Write **cat demo222.java**, it will print contents of the demo222.java file on the console
   2. Run the code, **$ git log**
   3. Use arrow keys to navigate to the end of the file
   4. Find the commit message “with one print statement”
   5. Text

      Description automatically generatedCopy first 8 characters from the long string written just next to the commit
4. **Text

   Description automatically generated**Write **git checkout <8 characters> <filemame>** and subsequently **cat demo222.java** to verify the changes

Clearly, there is no more for loop.

**This is known as version restore.**