SE489 DevOps Engineering

Lab 2

# Lab 2: More Git Operations

Objective: After completion of this lab session successfully, students will be able to use some of most widely used Git Operations of creating branches and merging them together.

In last lab we have used git commands from windows command prompt, now we will customize git bash shell to the local repository we have created in previous lab session.

1. From start menu, start Git by writing git bash   
   Graphical user interface, application, Teams

   Description automatically generated
2. Git bash shell will appear, showing default directory of the installation

Graphical user interface, text, application

Description automatically generated

1. Git follows DOS commands, use dos commands to change the working directory to the local repository we have created at d:\DevOps Tools\Lab Manual  
   (Hint: because folder name consist of spaces, we can’t write them directly, use tab after writing part of the name before the space e.g. for *DevOps Tools*, write DevOps and press tab, ***DevOps\ Tools*** will appear)

Text

Description automatically generated  
  
master at the end of the path confirms that we are now at the original branch known as **master** in the terminology of Git

1. Before proceeding further, if we want to check the log,

Text

Description automatically generated

The string shown after commit are sha keys allotted to the commit operations, author shows who has made these commit operations, last strings are messages used with the commit operations, now it helps us to identify the commit operations.

1. Now if we want to experiment with some new features, while keeping original intact, we create branches, a branch inherits all the files from the original branch (“master” in our case)

Let’s create two branches with the help of git branch command  
  
Text

Description automatically generated

We have created two branches viz. test0 and test1

1. To list all the local as well as remote branches, -a switch is used  
   Text

   Description automatically generated  
     
   \*shows, we are at master now
2. To delete a branch, -d switch is used, delete branch test1 and then verify it through listing  
   Text

   Description automatically generated  
     
   now it is showing only one branch, obviously another one has been deleted.
3. Now let’s switch to this branch **test0**, use checkout   
   Text

   Description automatically generated with low confidence  
     
   last word shows that we are in test0 branch
4. Check contents of the new branch with **ls/dir** command  
   Text

   Description automatically generated
5. Now create a copy of demo.java file using CP command, verify it with the help of ls command  
   Text

   Description automatically generated

A copy of the demo.java with name demo111.java has been created.

1. Edit this file with the default editor we have selected while installation of the git, Notepad, with following command  
   **$notepad demo111.java**, moment you will press enter, *a notepad window will open demo111.java file*  
   Text

   Description automatically generated
2. To make some changes, delete last print statement then save & close it.
3. Since we have created and modified a file, we can check, if it is being tracked or not by invoking git status  
   Text

   Description automatically generated
4. Now let’s add this to stage area and then check the status again  
   Text

   Description automatically generated  
     
   **-A**, when used with add, adds all tracked and tracked files to staging area (file will be tracked)
5. Commit this change with message tag, *with two print statements.*  
   Text

   Description automatically generated
6. We can also create a branch with the help of checkout command by using switch **-b** and branch name, and check the contents with **ls**

Text

Description automatically generated

1. Repeat steps from 10 -12 to create and edit the copy, rename the file **demo222.java**, time delete the second print statement as well so as only one print statement remains in the file.  
   Text

   Description automatically generated
2. Same as previously, lets check status, add this file and check status again.  
   Text

   Description automatically generated
3. Commit this file with message tag, “with one print statement”.  
   Text

   Description automatically generated
4. Use successive checkout and ls, to look into various branches and their contents.  
   Text

   Description automatically generated
5. We can use **merge** to merge files from various branches, point to note that, we must be at the destination branch of the merge operation.  
     
   being at master branch, we will merge contents from branch test0 into branch master

Text

Description automatically generated

1. Rebase command is used to rebase the project, rebasing merges all files into one and produces much clear structure of project tree.  
   Text

   Description automatically generated

As obvious from the output produced, all files from the branch test1 have been merged into master.