**Experiment No. 9**

**Aim:** Study of Configuration Management using GitHub

**Theory:**

Git is a distributed revision control and source code management system with an emphasis on speed. Git was initially designed and developed by Linus Torvalds for Linux kernel development. Git is a free software distributed under the terms of the GNU General Public License version 2.

Git Life Cycle

General workflow is as follows −

1. Clone the Git repository as a working copy.
2. Modify the working copy by adding/editing files.
3. If necessary, update the working copy by taking other developer's changes.
4. Review the changes before commit.
5. Commit changes. If everything is fine, then push the changes to the repository.
6. After committing, if something is wrong, then correct the last commit and push the changes to the repository.



Git Life Cycle

1. Creating Git Repository

Initialize a new repository by using **init** command followed by **--bare** option. It initializes the repository without a working directory. By convention, the bare repository must be named as **.git**.

|  |
| --- |
| [gituser@CentOS ~]$ pwd  /home/gituser  [gituser@CentOS ~]$ mkdir project.git  [gituser@CentOS ~]$ cd project.git/  [gituser@CentOS project.git]$ ls  [gituser@CentOS project.git]$ git --bare init  Initialized empty Git repository in /home/gituser-m/project.git/  [gituser@CentOS project.git]$ ls  branches config description HEAD hooks info objects refs |

1. Generate Public-Private RSA Key Pair

|  |
| --- |
| User1@CentOS ~]$ pwd  /home/user1  [user1@CentOS ~]$ ssh-keygen |

1. Adding keys to authorized keys

Suppose there are two developers working on a project. Both users have generated public keys.

Both add their public key to the server by using ssh-copy-id command as given below

|  |
| --- |
| [user1@CentOS ~]$ pwd  /home/user1  [user2@CentOS ~]$ ssh-copy-id -i ~/.ssh/id\_rsa.pub gituser@git.server.com |

1. Push changes to the repository

We have created a bare repository on the server and allowed access for two users. Both users can push their changes to the repository by adding it as a remote.

Git init command creates **.git** directory to store metadata about the repository every time it reads the configuration from the .**git/config** file.

User1 creates a new directory, adds README file, and commits his change as initial commit. After commit, he verifies the commit message by running the **git log** command.

|  |
| --- |
| [user1@CentOS ~]$ pwd  /home/user1  [user1@CentOS ~]$ mkdir user1\_repo  [user1@CentOS ~]$ cd user1\_repo/  [user1@CentOS user1\_repo]$ git init  Initialized empty Git repository in /home/user1/user1\_repo/.git/  [user1@CentOS user1\_repo]$ echo 'TODO: Add contents for README' > README  [user1@CentOS user1\_repo]$ git status -s  ?? README  [user1@CentOS user1\_repo]$ git add .  [user1@CentOS user1\_repo]$ git status -s  A README  [user1@CentOS user1\_repo]$ git commit -m 'Initial commit' |

1. Checking log message by executing the git log command.

|  |
| --- |
| [user1@CentOS user1\_repo]$ git log |

1. Commit changes

To commit the changes, he used the git commit command followed by –m option. If we omit –m option. Git will open a text editor where we can write multiline commit message

|  |
| --- |
| [user2@CentOS project]$ git commit -m 'Implemented my\_strlen function' |

**Performance:**

1. Perform all the commands using Git
2. Take screenshots of each of the command and respective output
3. Explore the commands for merging the documents and show the screenshots.

**Conclusion:**

<<Write conclusion in your own words>>