*DEPARTMENT OF INFORMATION TECHNOLOGY* Experiment No 1

|  |  |  |
| --- | --- | --- |
| **Semester** | Semester VIII | |
| **Subject** | DevOps Lab | |
| **Subject Professor In- charge** | Prof. Yash Shah | |
| **Laboratory** | L11B | |
| **Student Name** | Ashwini Jadhav | |
| **Roll Number** | 17101B0038 | |
| **Grade and Subject Teacher’s Signature** |  |  |

|  |  |  |
| --- | --- | --- |
| **Experiment**  **Number** | 1 | |
| **Experiment**  **Title** | To Perform Version Control using Version control tool GIT | |
| **Resources / Apparatus Required** | Hardware:  Compatible Computer System | Git, GitHub |
| **Objectives** | Learn and explore Version Control using GIT | |
| **Theory** | **What is Version Control?**  Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.  **Why is Version Control needed?**  As development environments have accelerated, version control systems help software teams work faster and smarter. They are especially useful  for [DevOps](https://www.atlassian.com/devops/what-is-devops) teams since they help them to reduce development time and increase successful deployments.  For most software teams, the source code is a repository of the invaluable knowledge and understanding about the problem domain that the developers have collected and refined through careful effort. Version control protects source code from both catastrophe and the casual degradation of human error and unintended consequences.  Software developers working in teams are continually writing new source code and changing existing source code. The code for a project, app or software component is typically organized in a folder structure or "file tree". One  developer on the team may be working on a new feature while another | |

|  |  |
| --- | --- |
|  | developer fixes an unrelated bug by changing code, each developer may make their changes in several parts of the file tree.  Version control helps teams solve these kinds of problems, tracking every individual change by each contributor and helping prevent concurrent work from conflicting. Changes made in one part of the software can be incompatible with those made by another developer working at the same time. This problem should be discovered and solved in an orderly manner without blocking the work of the rest of the team. Further, in all software development, any change can introduce new bugs on its own and new software can't be trusted until it's tested. So testing and development proceed together until a new version is ready.  **How to carry out Version Control?**   1. **Initialize Git Repository**   To initialize a Git repository in a project directory:  git init  This will create a .git directory in the project directory.   1. **Check File Status**   To check if any files were modified and not yet committed, git status  This will return the current state of the repository   1. **Stage File(s) Changes**   You can selectively stage modified files, adding them to the ‘staging area’ to prepare them to be committed. Modified files that are not added to the staging area will subsequently not be committed. This allows for more refined and specific commits (ex: changes to part A of the project only), which is useful for future reference.  #to stage specific modified files git add filename  #to stage all modified files git add .   1. **Commit File(s) Changes**   All staged files are then committed, essentially creating a ‘screenshot’ of those particular files at that particular moment. This effectively records a new change to the repository.  git commit -m 'describe change(s) made here'  Each commit must be made with a message, describing the change(s) made.   1. **To display a log of all commits made:**   git log  **Branching:**  A [branch](https://www.atlassian.com/git/tutorials/using-branches) is essentially a ‘new’ directory, on which you can work on a specific part or feature of a project, before you merge those changes to the main branch that contains all of your source code.  The default branch that you always start with is always called the master |

|  |  |
| --- | --- |
|  | branch. The master branch contains the most updated, available source code. Always assume that the master branch is ready to be deployed. All the experimentation and changes, big and small, are made on other branches to be merged in later.  **To list all branches in the repository:**  git branch  **To create a new branch:**  git branch new-branch-name  **This will only create the branch. To switch to a specific branch and start working within it:**  git checkout another-branch  **Alternatively, if you want to create a new branch and immediately switch to it:**  git checkout -b new-branch-name |
| **Output** | Screenshot (227) |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |
| **Conclusion** | Thus, we have implemented the above diagram of Version Control using the Git  tool |