*DEPARTMENT OF INFORMATION TECHNOLOGY* Experiment No 2

|  |  |  |
| --- | --- | --- |
| **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** | 2 | |
| **Experiment**  **Title** | To perform Version Control using GitHub | |
| **Resources / Apparatus Required** | Hardware:  Compatible Computer System | Git, Github |
| **Objectives** | Learn and Implement Version Control using GitHub | |

|  |  |
| --- | --- |
| **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?**  Make sure that you have a [Github](https://github.com/) account and that you have created a [Github](https://help.github.com/articles/create-a-repo/) [repository](https://help.github.com/articles/create-a-repo/) for your project. You can then [push a local repo](https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/) to this repository. Alternatively, you can [clone](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-clone) an existing repository, which essentially creates a copy of the public repository onto your machine.  **git clone filepath**  When you clone, a remote connection (known as ‘origin’) to the original repository is automatically created. This allows you to push and pull changes (to and from the original repository) going forward.  The primary usefulness of Github is because it expedites collaborative work. Every contributor to a project can have their own local private repository for the project, make changes to their local repository, and push changes to the public repository later. They can do all this without having to keep up with changes made to the public repository.  A centralized workflow depends on a single public repository.   1. Contributors clone the central repository from Github, thereby creating copies of the repository onto their local machines.   **git clone repository**   1. They can then create their own branch on their local repository, where they can make, stage, and commit their changes locally.   **git checkout -b branch**   1. Then they can push those (committed) changes to the remote version of their branch on Github (still separate from the master remote branch).   #ensure local version of master is up-to-date  **git pull --rebase origin master#push to remote branch git push origin branch**   1. Once a contributor is ready, they can submit a [pull request](https://help.github.com/articles/creating-a-pull-request/) for their remote branch. Other collaborators can review the changes and once the changes are   approved, those changes can be merged to the master remote branch. |

|  |  |
| --- | --- |
|  |  |
| **Output** | **Pushing the following structure to GitHub:**  Screenshot (198)  Screenshot (255) |

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

|  |  |
| --- | --- |
|  | Screenshot (258) |

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

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

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

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

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

|  |  |
| --- | --- |
|  | Screenshot (269)  **Changes made to file 7 in master branch:** |

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

|  |  |
| --- | --- |
|  | **After pushing, reflected on remote repository i.e GitHub**  Screenshot (272)  **A list of changes made:**  Screenshot (273) |

|  |  |
| --- | --- |
|  | **Clone Request:** |

|  |  |
| --- | --- |
|  | **Pull Request(f1 branch saved as master branch on local repository):** |
| **Conclusion** | Thus we have implemented Version Control using GitHub |