Evans Mutwiri

4 April 2022

**Version Control Report**

**Git** is a version control system that help keep track of different versions of code or files, maintaining a history of the versions. For collaboration git is essential in tracking changes made to a project by showing the details of the change ie the name of author, time of change and the message attached to the version(commit message).

You can make copies of files, for example when writing code you can modify each copy independently without the risk of messing up the other copy, then optionally merge the changes to the main copy if they satisfy the conditions and are safe.

**Github** is an online store for git repositories as the name suggests. Github extends the collaboration functionality of git by providing a remote copy of git that is in sync for all computers that make a copy. With GitHub developers or git users can connect, upload and download resources.

**Important** **Git commands**

|  |  |
| --- | --- |
| Command | * Definition |
| git config --global user.name “[firstname lastname]”  git config --global user.email “[valid-email]” | - Set globally the name and email of the attributed author. |
| Git init | - Initialise a folder as a git repository. |
| Git clone [url] | * Create a local copy of a repository in GitHub |
| git status | - Check the state of the repository and check for changes that are unstated. |
| git add . /git add [file] | - Add files as ready for commit. |
| Git commit -m “message” | - Commit changes to repository |
| Git branch | - Shows the current working branch |
| Git branch [branch-name] | - Create a branch with a unique name |
| Git checkout [branch-name] | - Navigate to an already existing branch |
| Git checkout -b [branch-name] | - Create a branch with a unique name and checkout to it |
| Git log | - Show all commit history |
| Git remote add [username][url] | - Add a GitHub url |
| Git fetch | - Fetch down all the branches in a remote repository |
| Git merge [alias]/[branch] | - Merge an author’s branch into current branch to update it. |
| Git push origin main | - Upload files to the remote main branch |
| Git pull | - Fetch and merge commits |
| Git log -oneline | - Log history in short hand more terminal friendly way ie one commit per line. |

**Git workflow utilising some of the commands.**

1. Edit files normally for instance using a text editor.
2. Git init to create a repository
3. \* Create a git ignore file that contains files that should not be tracked by git.
4. Create git keep file(optional) in empty folders. Git does not keep track of empty directories. Use if you want it tracked for some reason.
5. Git status - check among the files that are being tracked for changes. By default un-staged changes appear in red colour on the terminal.
6. Git add - make the changes ready for commit ie add files to the staging area.
7. **Git status** after this will show green colour file and folder names indicating they are ready
8. **Make a commit** - git commit -m “message” : save the project version to git.
9. **Git log** - this will display a timeline list of all commits. For each commit in the timeline there is a commit hash, author name and email address, date of commit and the commit message. Alternatively adding **—online** flag prints a list with **a shorter commit hash and commit message only**.
10. Using branches - typically the default branch is the **main** branch. Branches enable collaboration when using git. Feature branches are used to make changes for various contributors. The changes can then be pooled together(git merge) in the main branch once approved.
11. Pull request: On GitHub one can create pull requests to alert a repo owner that you want to make some changes to their code.
12. Merge a pull request: Also preferably on GitHub due to user interface. This shows that the changes have been accepted and are now included into the main branch. All this happens in the remote repo in GitHub.
13. Git pull - once the changes in GitHub are adopted this command with bring the local branch to the same level.

An alternative workflow starts with creating an empty repo in GitHub, then cloning it using your GitHub api key to authenticate all operations on the repo without need to enter credentials for every commit. **git clone https://[api-key-here]@github.com/[username]/[repositony-name]**

**Reference**

<https://www.simplilearn.com/tutorials/git-tutorial/git-vs-github>

<https://education.github.com/git-cheat-sheet-education.pdf>

https://www.freecodecamp.org/news/what-is-git-learn-git-version-control/