Power Learn Project

Name: Khuliso Junior Ravhuravhu

Module: Software Engineering | Day 2

Topic: Git & GitHub

1. Fundamental concepts of version control & GitHub

Version control is a system that allows you to track changes to files over time. It's essentially a way to save snapshots of your work at different points, so you can revert to a previous version if necessary. This is particularly useful for software development, where projects can involve multiple contributors working on different parts of the codebase simultaneously. Version control helps maintain the integrity of a project by tracking every modification and allowing developers to revert to previous versions if something goes wrong.

GitHub is a popular platform for hosting version control repositories, primarily using the Git version control system. GitHub's popularity stems from its ease of use, wide adoption, and additional features like pull requests, issues, and project boards, which facilitate collaborative development and project management.

2. Setting up repository on GitHub

Log into GitHub account, if you don't have account then create one for free. Once logged in, navigate to repository page by clicking the "+" icon in the top right corner of the page and select new repository.

On the new repository page, you have to give your repository a unique name and description details and choose if you want it to be accessible to other by choosing 'public' or choose 'private' to hide it from everyone, which requires subscription.

Initialize your repository with a README file to document your project. This one is optional.

Click commit changes to save your new repository.

3. Importance of README

The README file serves as the digital storefront of your project on GitHub. It's the first thing potential contributors, users, or collaborators will see when they visit your repository. It also

facilitates collaboration as it consists of well-written guidelines for contribution. It also provides clarity and understanding of the project, making it easier to grasp the purpose and the functionality of the project.

4. Difference between public and private repository

Public repositories are accessible to anyone on the internet. This means that anyone can view, download, and contribute to the code. This can be a great way to attract new contributors and users, but it also means that sensitive information could be exposed to unauthorized individuals.

Private repositories are only accessible to authorized users. This means that sensitive information can be kept confidential. However, it also means that the project's visibility and potential user base is limited.

5. Steps to commit to GitHub repository

Clone the Repository:

- Using the following command
  git clone <repository_url>
- Create a Branch:
  - It's often recommended to create a new branch for your changes to isolate them from the main branch. This allows you to experiment and make changes without affecting the main codebase.
  - Use the following command to create a new branch:

  git branch <branch_name>
- Switch to the newly created branch:
  git checkout <branch_name>

- Stage Changes:
  - Use the git add command to stage the changes you've made. This indicates that you want to include these changes in the next commit.
  - git add <filename>
- To stage all changes in the current directory:

git add .

- Commit Changes:
  - Use the git commit command to create a commit and record the changes you've staged.
  - Provide a clear and concise commit message that describes the changes you've made: git commit -m
- Push Changes to GitHub:
  - Once you're satisfied with your commit, push it to your remote repository on GitHub: git push origin <branch_name>

6. How does branching work in GitHub

Branching allows you to create a separate version of your project where you can work on new features, bug fixes, or experiments without affecting the main codebase.

Process of Branching start with creating a branch by using 'git branch <name>', Use git checkout <branch-name> to switch to the new branch. Develop and commit changes in the branch. Once changes are complete, merge the branch into the main branch using 'git merge <branch-name>'.

7. Pull requests on GitHub workflow

Pull requests (PRs) are a fundamental mechanism in GitHub for proposing changes to a repository. They serve as a central hub for code review, discussion, and collaboration, ensuring that code quality and consistency are maintained.

Steps involved are:

- Create a New Branch: Start by creating a new branch from the main branch (usually main -or master) to isolate your changes.
- Make Changes: Develop your feature or fix the bug on the new branch.
- Commit Changes: Commit your changes regularly to save your progress.
- Push to Remote Repository: Push your branch to a remote repository (like GitHub).
- Create a Pull Request: From the GitHub web interface, create a pull request from your branch to the main branch.
- Add Reviewers: Assign reviewers to your pull request who can provide feedback and approve the changes.

- Address Comments: Respond to any comments or suggestions from reviewers. Make necessary changes to your code and push updates to the pull request.
- Merge: Once the code has been reviewed and approved, the pull request can be merged into the main branch.

8. Discuss the concept of "Forking" a repository

Forking creates a personal copy of someone else's repository under your GitHub account. Unlike cloning, which creates a local copy, forking links your copy to the original repository, allowing you to submit changes via pull requests.

It is useful when you want to contribute to an open-source project, and also when you need to customize a project for your own needs without affecting the original project.

9. Importance of issues and project boards

Issues on GitHub allow you to track bugs, suggest features, or discuss topics related to the project. Project boards provide a visual way to organize and prioritize tasks. You can create issues for bugs, enhancements, or questions. While Project Boards categorizes issues, track progress, and manage sprints.

Both of these tools improve project organization and collaboration by clearly defining tasks and tracking their progress.

10. Common challenges and best practices with GitHub

The common challenges with GitHub include accidentally overwriting changes made by other team members can lead to conflicts and lost work. When two or more developers make changes to the same file, merge conflicts can arise, requiring manual resolution. Using branches incorrectly or not understanding their purpose can lead to confusion and difficulties in managing changes. Or accidentally committing sensitive information to a public repository can pose a security risk.

While on the other hand, the best practices with GitHub include:

- Committing your changes frequently and use clear, descriptive commit messages. This makes it easier to track changes and revert to previous versions if necessary.
- Adopt a consistent branching strategy (e.g., Gitflow, GitHub Flow) to manage different features, bug fixes, and releases.

- Implement a code review process using pull requests to ensure code quality and catch potential issues before merging changes.
- Use a `.gitignore` file to specify files or directories that should be ignored by Git, such as temporary files or build artifacts.
- Use remote repositories to back up your code and collaborate with other developers.
- Familiarize yourself with essential Git commands to efficiently manage your repository.
- Consider using a Git GUI client like GitHub Desktop or GitKraken to simplify the workflow for beginners.
- Keep up-to-date with the latest Git features and best practices.