Mastering Git Part 2: Branching, GitHub, and Advanced Commands

---

Introduction

Welcome back to the second part of our series on mastering Git. In this installment, we'll delve into the crucial concepts of branching, GitHub integration, and some advanced commands that will empower you in your version control journey.

Branching in Git

1. Concept of Branching

Branching is a fundamental concept in Git that facilitates efficient code management within a team. It allows developers to work on different functionalities or enhancements without directly affecting the main codebase.

2. Commands for Branching

- `git branch`: Lists all branches in the repository, highlighting the current branch.
- `git branch --show-current`: Displays the name of the current branch.
- `git branch <branch_name>`: Creates a new branch with the specified name, referencing the current commit.
- `git log --oneline`: Shows commits and highlights the branch they belong to.
- `git checkout <branch_name>`: Switches to the specified branch for further modifications.

3. Modifying Files in a Branch

While inside a branch (e.g., `dev1`), any changes made, such as creating, modifying, or deleting files, only impact that specific branch. Commits in one branch do not affect the content of another.

```
# Example: Creating a file in the dev1 branch
vim web.html
git add web.html
git commit -m "Added web.html to dev1 branch"
```

4. Merging Branches

- `git merge <branch_name>`: Merges the specified branch into the current branch, updating the codebase.

5. Deleting Branches

- `git branch -d <branch_name>`: Deletes the specified branch after merging.

GitHub Integration

1. GitHub as a Centralized Version Control System (CVCS)

GitHub serves as a centralized location for collaborative development, allowing multiple developers to work together seamlessly.

2. README.md and Repository Initialization

The `README.md` file, written in Markdown, contains essential information about the repository. Initializing a repository with a `README.md` file creates the working area, staging area, and commit area.

3. Authentication with GitHub

GitHub authentication can be achieved through HTTPS (username and password) or SSH (private and public keys).

ssh-keygen.exe
- It will create a public key and
private key combinations using
RSA algorithm for us, Now we can go to GitHub->setting->" SSH And GPG keys",
and there we can paste our public key which we have just created.
ssh -Tgit@github.com
-> Now it will authenticate
our git with GitHub successfully.

4. Linking Local and Remote Repositories

# Example: Adding a remote repository

```
git remote add origin https://github.com/username/repo.git
git remote -v
git push --set-upstream origin master
```

5. Checking Repository Status

- `git remote show origin`: Displays information about the remote repository.
- `git status`: Shows the status of the local repository and its relation to the remote repository.

Conclusion

In this part of our Git mastery series, we've explored branching, GitHub integration, and advanced commands. Understanding these concepts will empower you to manage code efficiently, collaborate seamlessly, and maintain a clean version control history.

Stay tuned for the next part, where we'll dive into more advanced Git functionalities to enhance your development workflow.

Link to the Full Blog - https://medium.com/@srivastavayushmaan1347/a-deep-dive-into-git-mastery-part-2-branching-github-integration-and-advanced-commands-fdfe71f372f9