Programming for Data Science

Version Control Systems

Say OL

say_ol@yahoo.com

September 27, 2024

What is Version Control?

- A system to track changes in files over time.
- Essential for collaboration and managing project histories.

Why Use Version Control?

- Collaborate seamlessly.
- Maintain project history.
- Easily revert to previous versions.
- Manage multiple development branches.

Types of Version Control

- **Local Version Control:** Simple file backup.
- **Centralized Version Control:** Single central repository.
- **Distributed Version Control:** Each user has a full repository (e.g., Git).

Introduction to Git

- A distributed version control system.
- Tracks changes and allows multiple users.
- Commands are executed in the terminal.

Setting Up Git

- Install Git: Download from `https://git-scm.com/`.
- Configure Git:

```
git config --global user.name "Your Name"
git config --global user.email "address@domain.com"
```

Creating a New Repository

- Command: `git init`
- Example:

```
mkdir my_project
cd my_project
git init
```

Cloning a Repository

- Command: `git clone [url]`
- Example:

```
git clone https://github.com/username/repo.git
```

Staging Changes

- Command: `git add [file]`
- Example:

```
git add my_file.py
```

Committing Changes

- Command: `git commit -m "message"`
- Example:

```
git commit -m "Initial commit"
```

Viewing Commit History

- Command: `git log`
- Explanation: Displays commit history with hashes and messages.

Branching

- Command: `git branch [branch-name]`
- Example:

```
git branch new_feature
```

Switching Branches

- Command: `git checkout [branch-name]`
- Example:

```
git checkout new_feature
```

Merging Branches

- Command: `git merge [branch-name]`
- Example:

```
git checkout main
git merge new_feature
```

Resolving Merge Conflicts

- Explanation: Occurs when changes in two branches clash.
- Steps:
  - Identify conflict markers in the files.
  - Edit files to resolve conflicts.
  - Stage and commit the changes.

Working with Remote Repositories

- Command: `git remote add origin [url]`
- Example:

```
git remote add origin https://github.com/username/repo.git
```

Pushing Changes

- Command: `git push origin [branch-name]`
- Example:

```
git push origin main
```

Pulling Changes

- Command: `git pull origin [branch-name]`
- Example:

```
git pull origin main
```

Creating Releases

- Tagging a release:
- Command: `git tag -a v1.0 -m "Version 1.0"`

GitHub Overview

- A platform for hosting Git repositories.
- Features include pull requests, issues, and project management.

Creating a GitHub Repository

- Go to GitHub and click "New Repository."
- Follow prompts to create a new repo.

Forking a Repository

- Click "Fork" on a GitHub repository to create a personal copy.
- Enables changes without affecting the original project.

Pull Requests

- A request to merge changes from one branch to another.
- Steps:
  - Create a pull request on GitHub after pushing changes.
  - Collaborators can review and comment.

Best Practices for Git and GitHub

- Commit often with meaningful messages.
- Use branches for new features or fixes.
- Regularly sync with the remote repository.
- Review pull requests thoroughly.

Common Git Commands Summary

- `git init`, `git clone`, `git add`, `git commit`
- `git push`, `git pull`, `git merge`, `git branch`

Conclusion

- Mastering Git and GitHub enhances collaboration and project management.
- Essential skills for data science workflows.

Questions?

Thank you for your attention!
Feel free to ask any questions.