**:** Basics about Git and GitHub

: History of Git and GitHub

: Downloading git, creating account

: Doing first commits

: Making branches, working on different branches

:git push and pull between branches

:knew about git conflict and how to reduce conflicts

: Learned about .gitignore file

:

**Git:** Imagine Git as a time machine for your code. It saves snapshots of your work at different points, so you can go back and undo changes if something goes wrong. It's like having a backup of your code that you can access whenever you need it.

**GitHub:** Think of GitHub as a social network for developers. It's a platform where you can share your code with others, collaborate on projects, and learn from the work of other developers. It's like a library for code, where you can find and contribute to different projects.

**What is version control system? [VCS]**

**Version control system** is a software tool that helps you track and manage changes to your files over time.

It's like a time machine for your work, allowing you to go back to previous versions of your files if you need to.

Here are some key benefits of using a version control system:

* **Undo changes:** If you make a mistake, you can easily revert back to a previous version of your work.
* **Collaborate with others:** Multiple people can work on the same project simultaneously without overwriting each other's changes.
* **Track changes:** You can see who made changes to a file and when, making it easier to identify the source of errors.
* **Create backups:** Version control systems automatically create backups of your work, so you don't have to worry about losing your data.

Popular version control systems include Git, SVN, and Mercurial.

echo "# Git-and-GitHub" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/bishaldsrija08/Git-and-GitHub.git

git push -u origin main

**Branching**

**Branches** in GitHub are like parallel universes of your project. They allow you to work on different features, bug fixes, or experimental ideas without affecting the main, stable version of your code. Each branch is a separate copy of the project, so you can make changes to it without worrying about breaking the main version.

**Key benefits of using branches:**

* **Isolation:** Work on new features or bug fixes without affecting the main codebase.
* **Collaboration:** Multiple developers can work on different branches simultaneously.
* **Experimentation:** Test new ideas without risk to the main project.
* **Rollback:** If a change goes wrong, you can easily switch back to a previous branch.

**Common branch types:**

* **Main branch:** The primary branch of your project, typically representing the stable version.
* **Feature branches:** Created for developing new features or enhancements.
* **Bugfix branches:** Created to address specific bugs or issues.
* **Release branches:** Created to prepare a new version of the project for release.

By effectively using branches, you can manage your project more efficiently, reduce the risk of errors, and collaborate more effectively with your team.

**Command to create new branch:**

Git branch {branch name}

**Command to switch branch:**

git checkout {branch name}

**Command to pull from branches**

git pull origin <branch-name>

**Command to merge from branches**

git merge <branch-name>

**Command to ignore already pushed file**

git rm -r –cached

git rm -r –cached <file name>