**Git Workflow Guidelines:**

To ensure a smooth and conflict-free development process, please follow the Git workflow guidelines outlined below:

**🔹 Git Workflow Guidelines**

Consider that **Koushik, Anirban, and Tanmay are developers (team members)** working on the same project.

**1. Individual Development Branches**

* **koushik** will push all commits to the **koushik** branch.
* **anirban** will push all commits to the **anirban** branch.
* **tanmay** will push all commits to the **tanmay** branch.

👉 Please make sure to use **only your assigned branch** for all development work and local testing.

**2. Merge to Development Branch (Shared dev Branch)**

* Once your changes are successfully tested locally, push the stable code to the **dev** branch.
* The other collaborators should then **pull the latest changes from dev**, integrate them locally, and resolve any conflicts if needed.
* This ensures everyone is working with the latest stable and tested code.

**3. Merge to Main Branch (Production Release)**

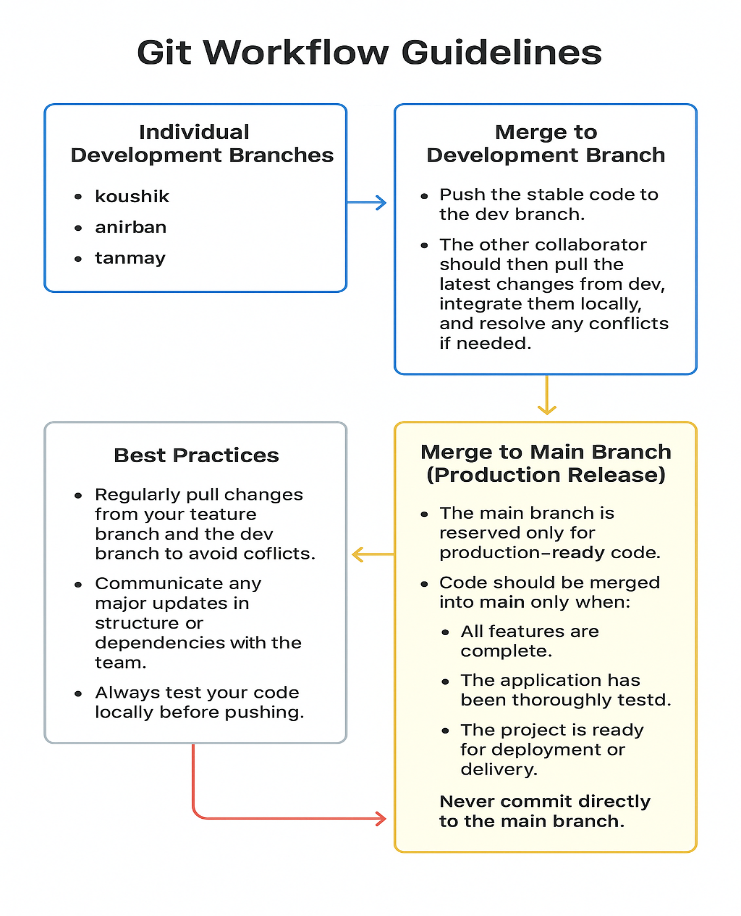
* The **main** branch is **reserved only for production-ready code**.
* Code should be merged into **main** only when:
  + All features are complete.
  + The application has been thoroughly tested.
  + The project is ready for deployment or delivery.

⚠️ **Never commit directly to the main branch.**  
All changes must go through the **dev** branch before being merged into **main**.

**4. Best Practices**

* Regularly **pull changes** from your feature branch and the dev branch to avoid conflicts.
* Communicate any **major updates** in structure or dependencies with the team.
* Always **test your code locally** before pushing.

✅ Request: I kindly ask everyone to follow this workflow consistently to keep our development process clean and efficient.



**🔄 Git Workflow Practice (with Different Accounts)**

**🔹 Setup**

1. **Each developer clones the repo** with their own GitHub account:

git init

git clone https://github.com/tanmayMt/benda-project-hiring-process.git

cd benda-project-hiring-process

1. **Set your username & email** (so commits are tracked correctly):

git config user.name "koushik"

git config user.email [koushik@example.com](mailto:koushik@example.com)

*(Each developer replaces with their own details: anirban, tanmay.)*

**✅ 2. Set up your branch (first time only)**

1. **Make sure you’re up to date with remote:**

**git fetch origin**

1. **Switch to the dev branch:**

**git checkout dev**

**git pull origin dev**

1. **Create and switch to your personal branch:**

**For Koushik:**

**git checkout -b koushik**

**For Anirban:**

**git checkout -b anirban**

**For Tanmay:**

**git checkout -b Tanmay**

**🔹 Round 1 – Personal Branch Work**

**Each developer will work in their own branch (koushik, anirban, tanmay).**

 **Koushik**

git checkout -b koushik

# make some changes

git add .

git commit -m "koushik: added feature A"

git push -u origin koushik

 **Anirban**

git checkout -b anirban

# make some changes

git add .

git commit -m "anirban: fixed bug B"

git push -u origin anirban

 **Tanmay**

git checkout -b tanmay

# make some changes

git add .

git commit -m "tanmay: updated UI"

git push -u origin tanmay

**🔹 Round 2 – Merge into dev**

1. **Each developer switches to dev:**

**git checkout dev**

**git pull origin dev**

**git merge koushik # (or anirban / tanmay depending on who is merging)**

**git push origin dev**

1. **Other developers pull latest dev and rebase their branch:**

**git checkout dev**

**git pull origin dev**

**git checkout your-branch**

**git rebase dev**

**🔹 Round 3 – Merge dev → main**

**After all features are tested in dev:**

**git checkout main**

**git pull origin main**

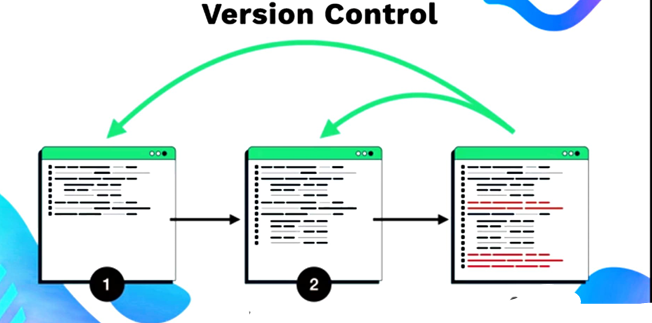
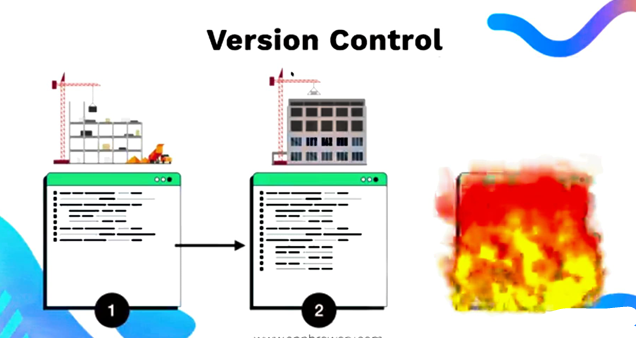
**git merge dev**

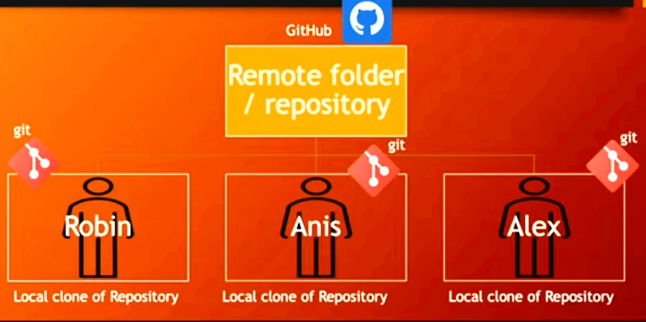
**git push origin main**

**✅ After completing these steps, you will have:**

* **Each developer’s feature in their branch.**
* **All features integrated into dev.**
* **A stable, production-ready main.**

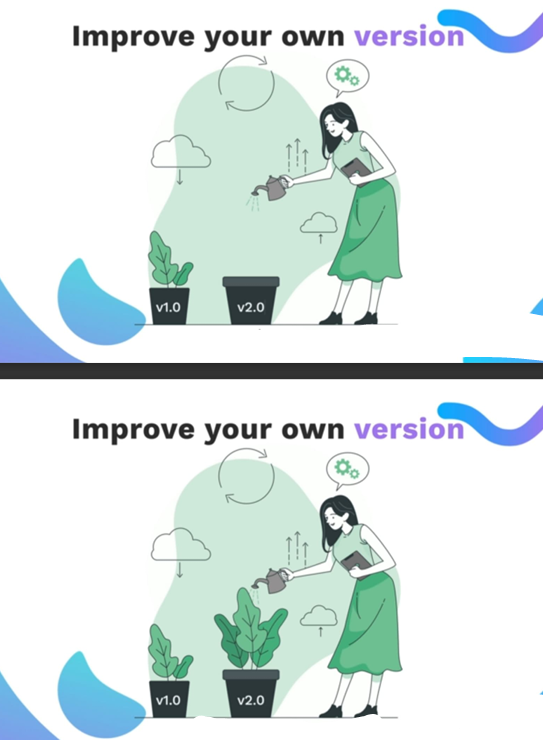
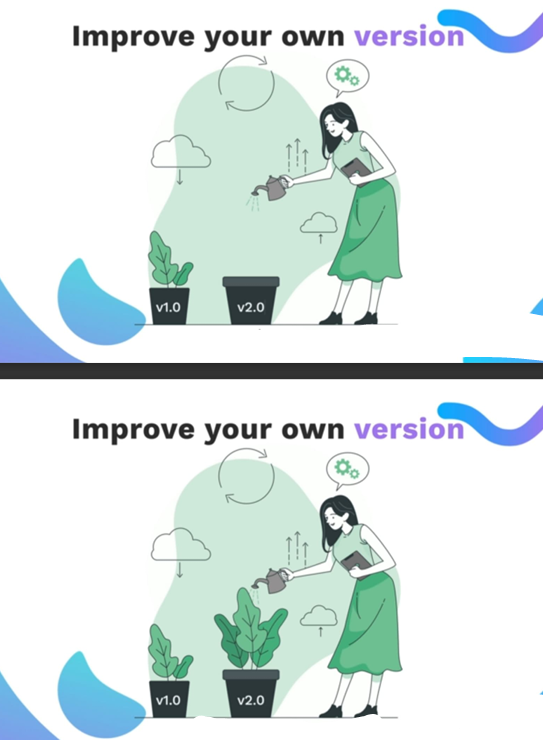
**Git & GitHub:**

** **

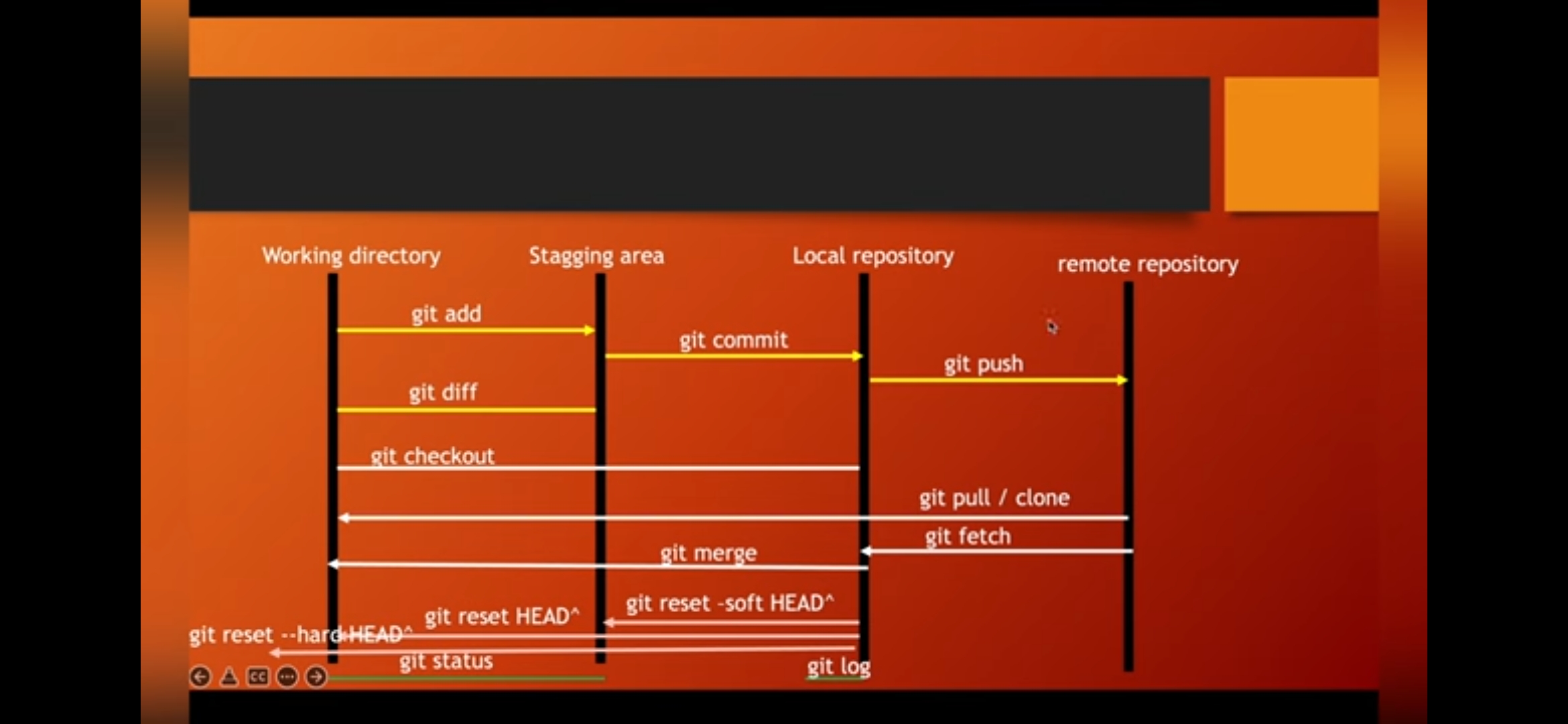
****

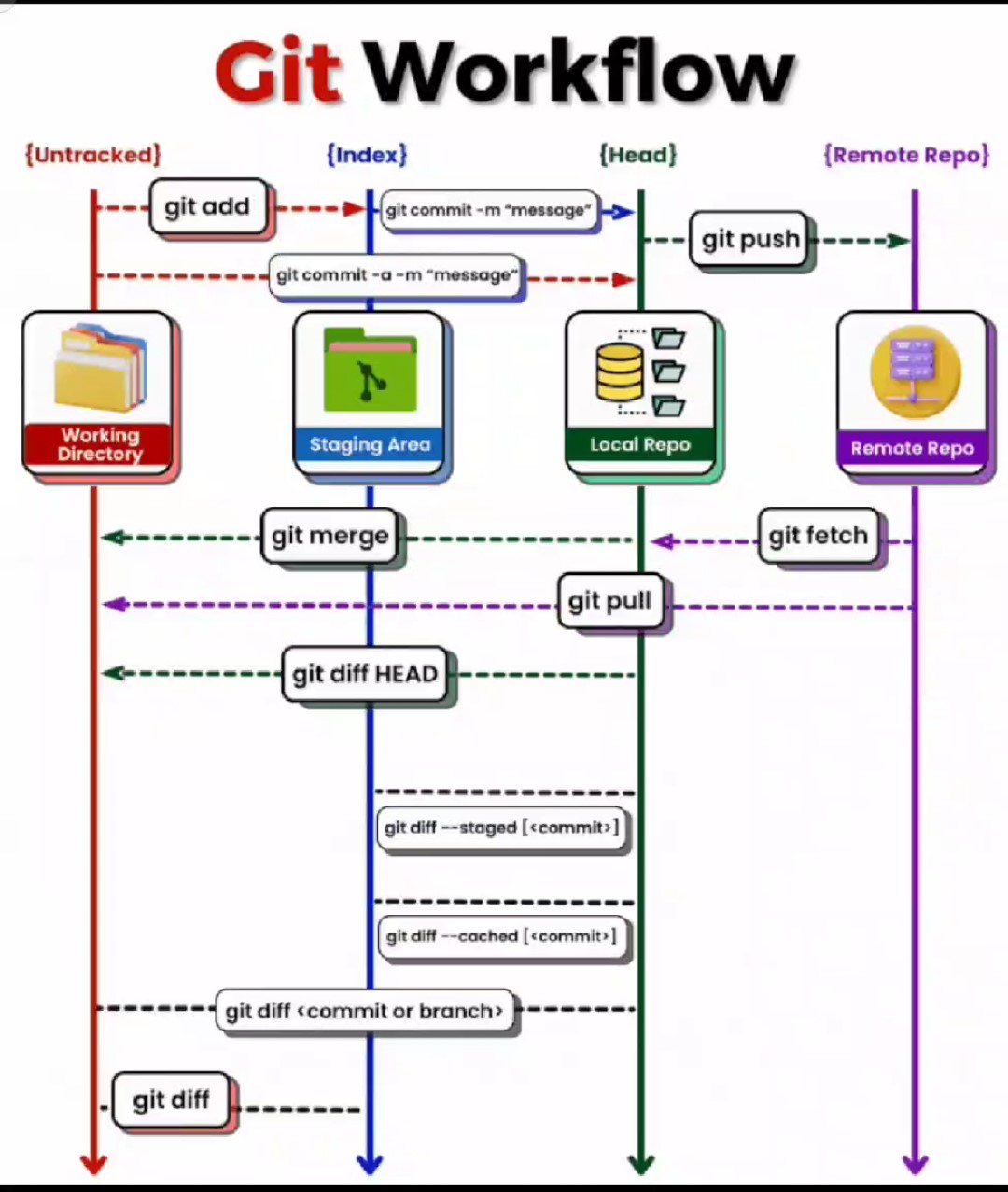
* **Git** tracks changes to files on your local machine. It records historical versions or snapshots of files as they exist at specific points in time.
* Git is a **version control system** that helps keep track of changes and supports collaboration in a project.
* **GitHub** is a **hosting platform** where you can upload your project directory, so that you can share with anyone from anywhere.
* When you and your colleagues work on the project from different local machines and make individual changes, GitHub acts as a **centralized platform** where everyone can **push their changes**. This allows all contributions to be merged and managed effectively in one place.

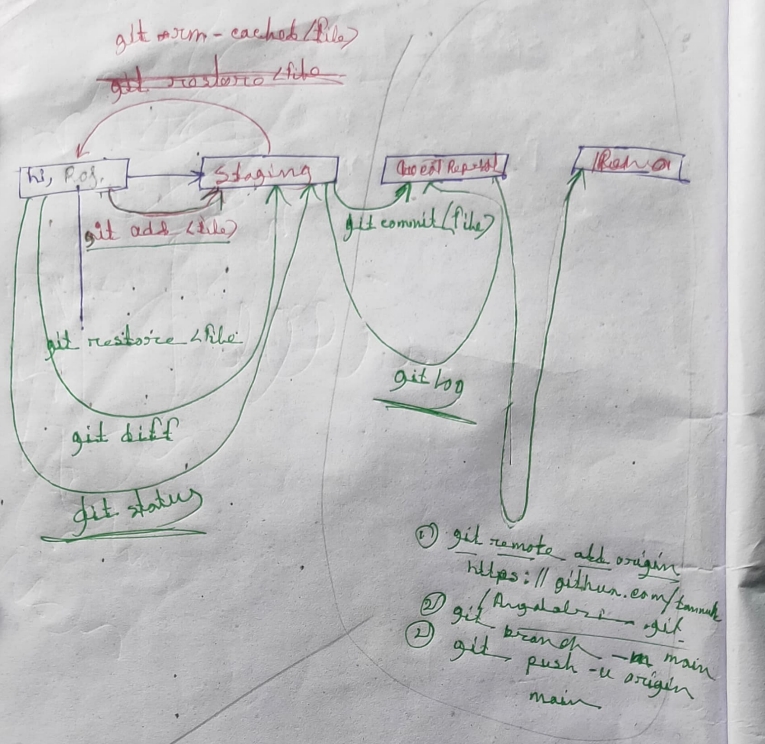
# Git is working locally and GitHub is working globally.

 -

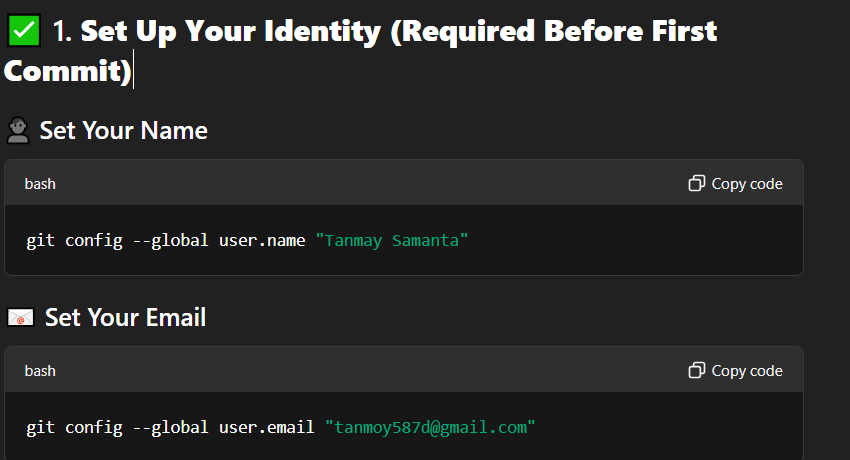


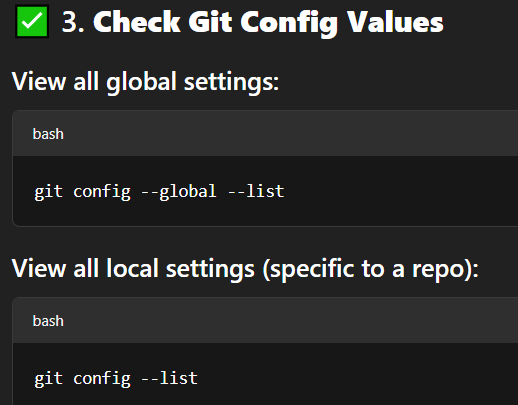


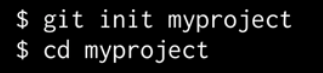




**Git:**







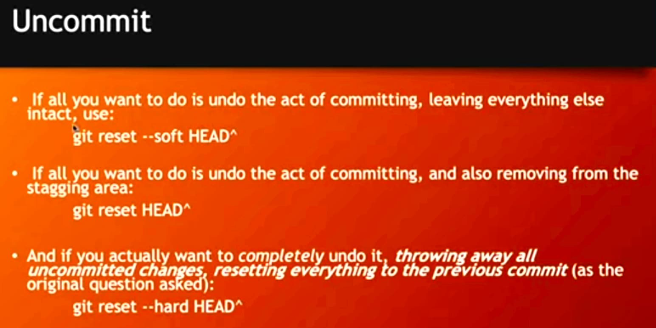
This creates a directory that can contain the project files as will as control files that store the historical elements, the history snapshots of yours documents, images, source code if you’re working with program.

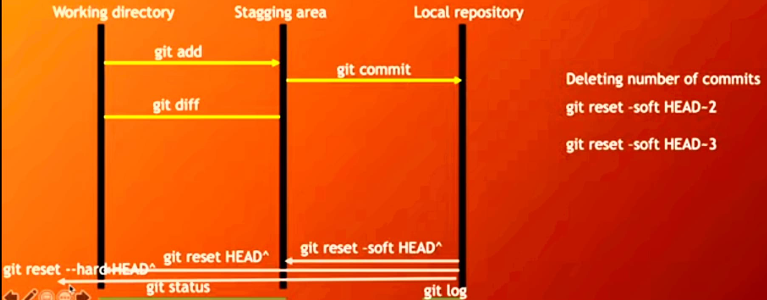


This commend actually notice the files and puts them into a kind of holding zone, ready to committed.



Using this we permanently records a historical version or snapshot of the files as they exist at a given point in time.







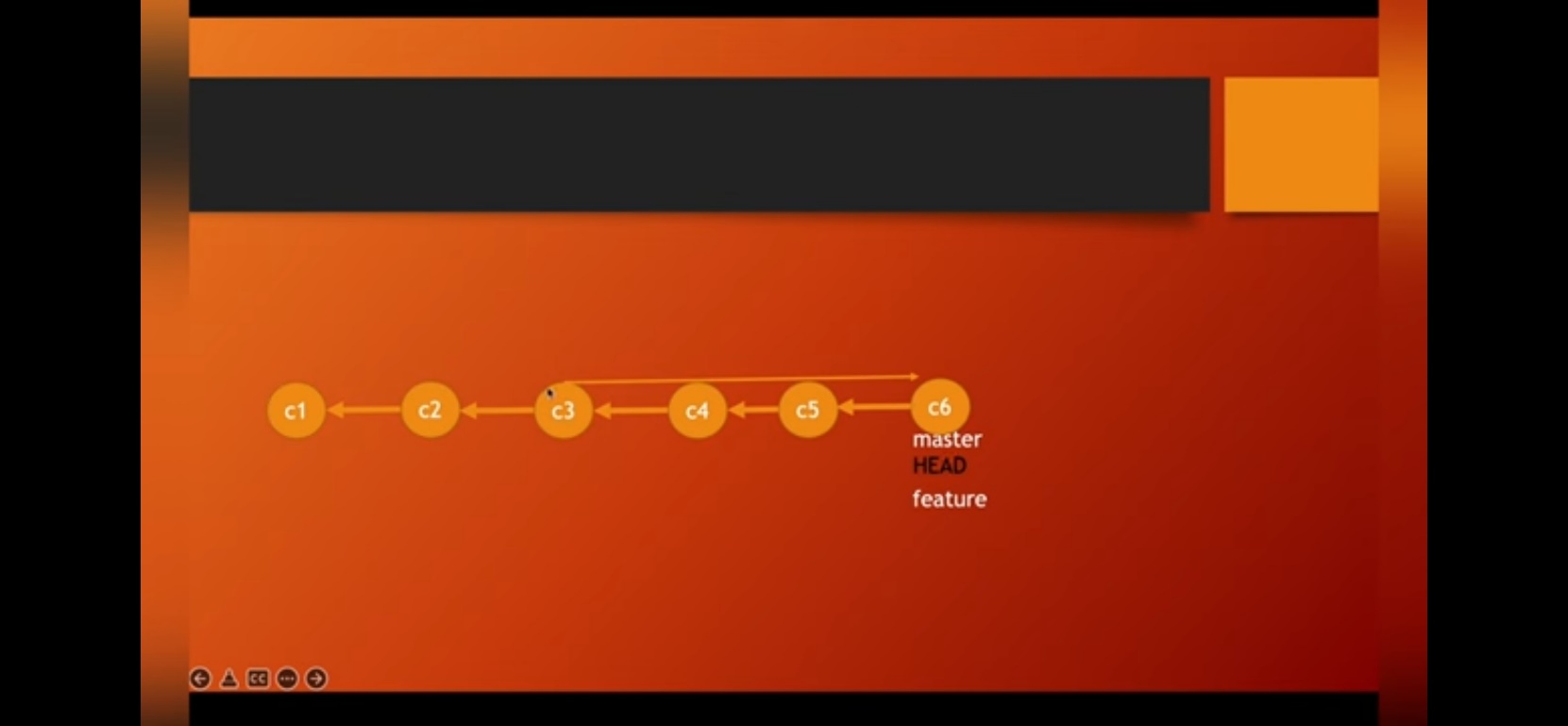
**Branch:**

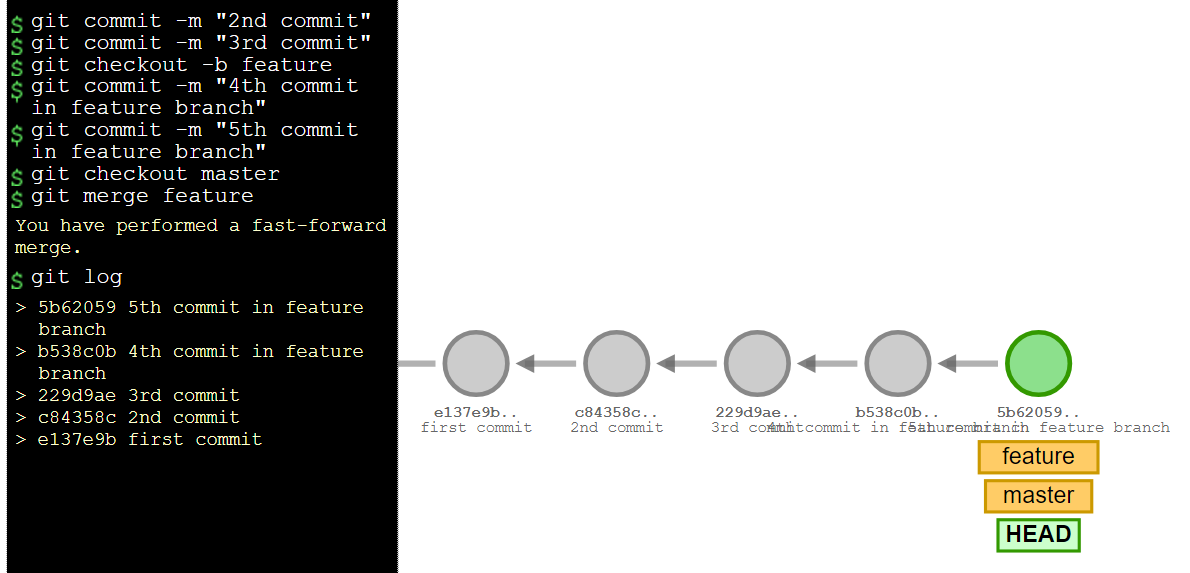
**Merging:**

****

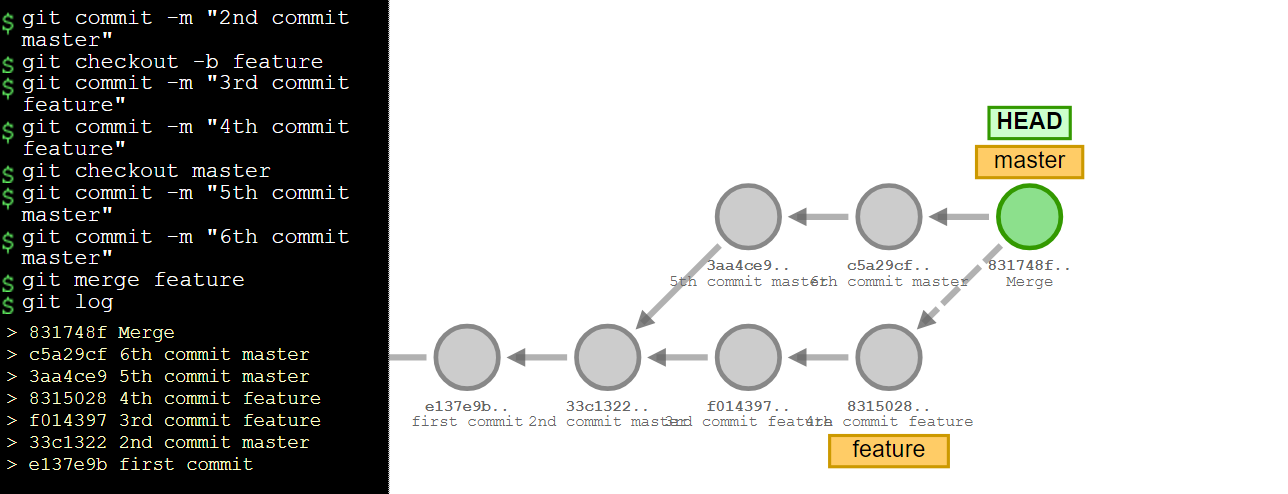
[Visualizing Git](https://git-school.github.io/visualizing-git/)

2-way Merging Forward Merging:

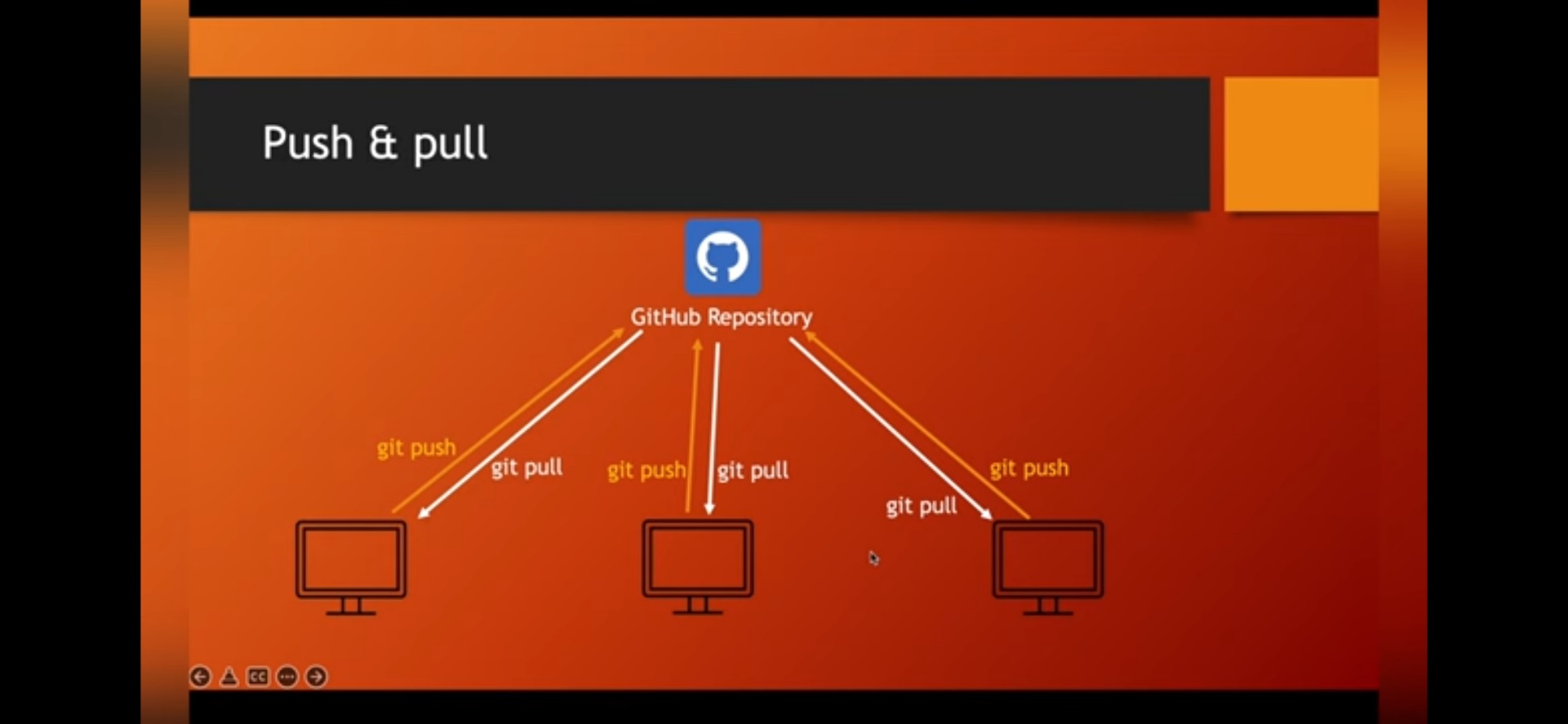




3-way Merging:



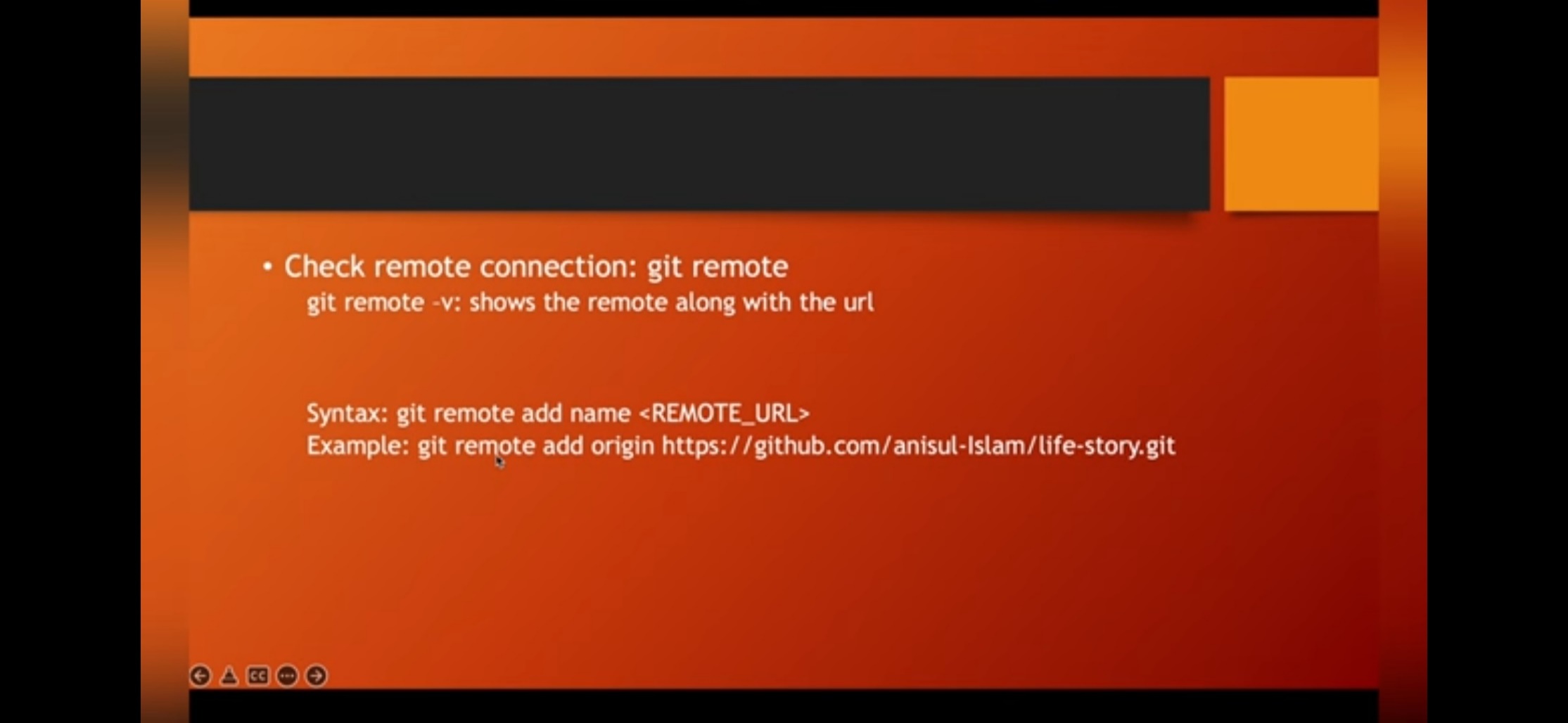
Pull & Push:

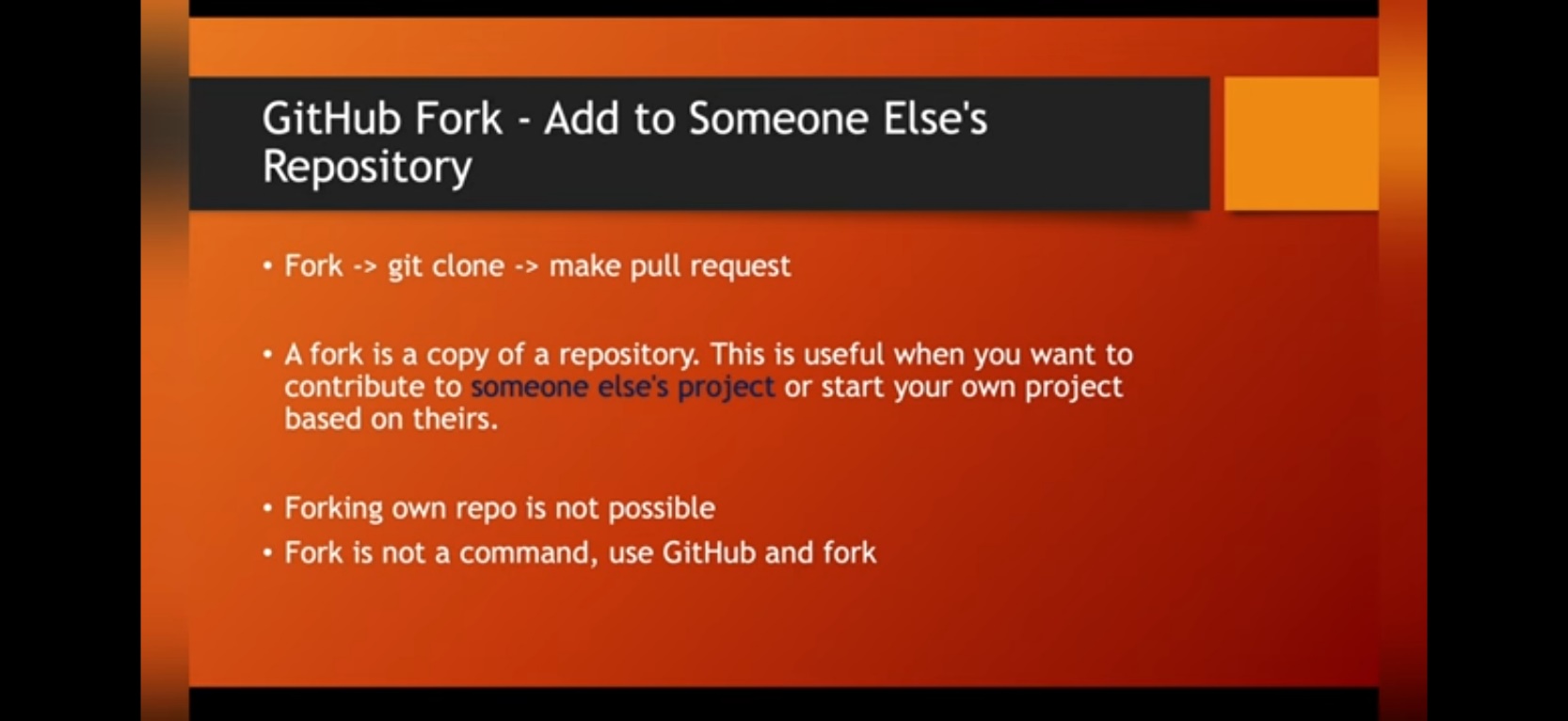


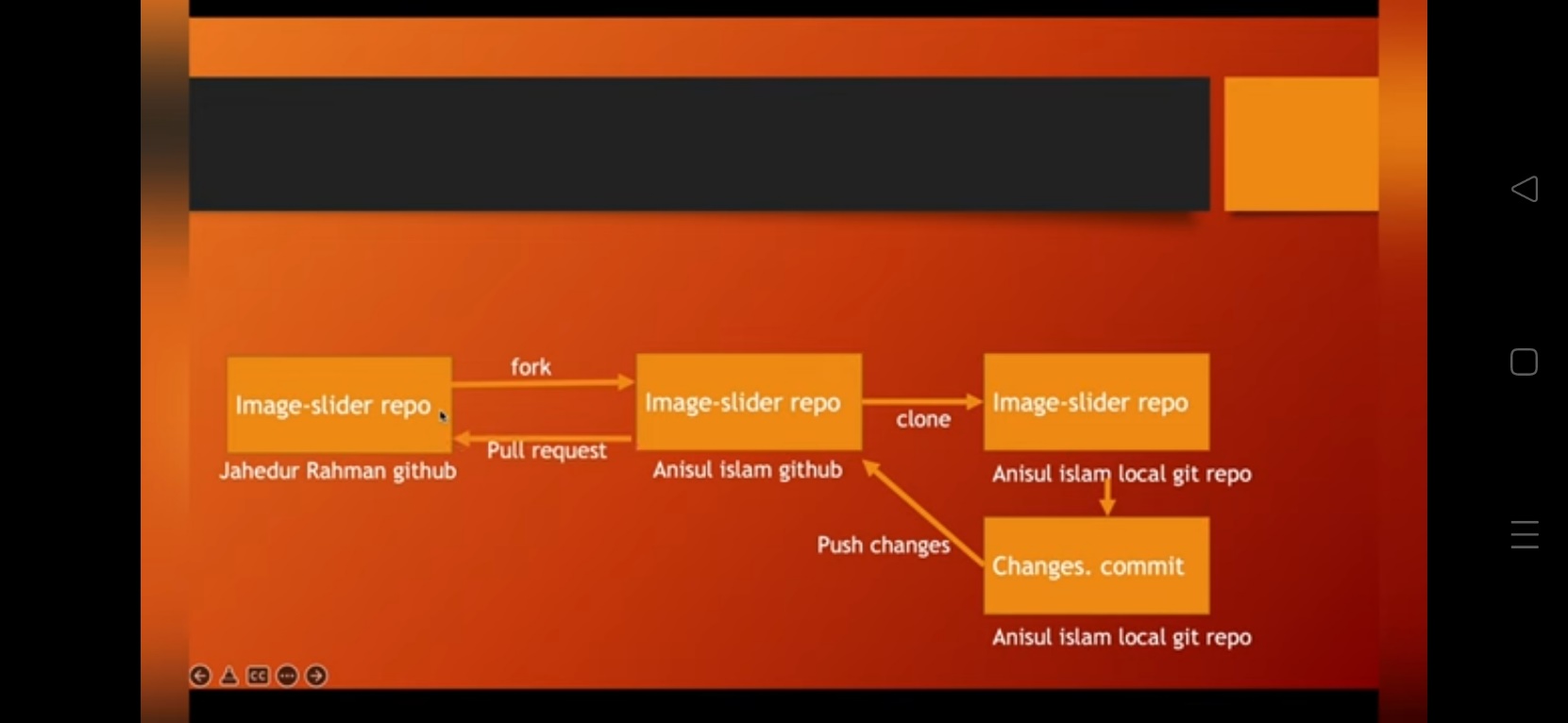
**GitHub:**



****

****

****

****

**Local repository & a Remote repository**

The important thing to note is that you can have a local repository completely in parallel with a remote repository check the differences between them, but you can also sync them or push things from your local repository to your remote local repository completely in parallel with a remote repository.

when we performed the command git push, then that effectively pushed all of those commits, all of those various versions and changes and code pieces to our remote repository on GitHub. So that's what "git push" does.