Github Repo Works

***Git Download zrur krein***

https://git-scm.com/downloads/win  
  
GitHub par repository bana kar team ke sath project par kaam karne ka process seekhna kaafi asaan hai. Main aapko step-by-step guide deta hoon jo aapko samajhne mein madad karegi.

### **1. GitHub Account Banaye**

Sabse pehle, agar aapka GitHub account nahi hai, to [GitHub](https://github.com/?utm_source=chatgpt.com) par jakar ek account bana lein.

### **2. Repository Create Karna**

* GitHub account mein login karne ke baad, aapko apne profile page pe jaana hoga.
* Wahan "Repositories" tab par click karein aur fir "New" button pe click karein.
* Repository ka naam (Example: project-name), description (optional) aur visibility (public ya private) set karein.
* Agar aapko ek README.md file chahiye ho, to is option ko check kar sakte hain.
* "Create Repository" pe click kar den.

### **3. GitHub par Repository Ko Clone Karna (Apne Local Machine Par)**

Repository create karne ke baad, aapko apne local machine par us repository ko clone karna padega, taaki aap code likh sakein.

* GitHub repository page par jaakar "Code" button pe click karein.
* HTTPS link ko copy karen.

Apne terminal/command prompt mein type karein:  
  
 git clone https://github.com/username/repository-name.git

* Isse repository aapke local machine par aa jayegi.

### **4. Code Edit Karna Aur Changes Commit Karna**

* Ab aap apne local machine par repository mein changes kar sakte hain.

Jab bhi aap code edit karenge, to git add, git commit aur git push commands ka use karna padega.  
  
 Example commands:  
  
 git add .

git commit -m "Description of changes"

git push origin main

### **5. Team Members Ko Invite Karna**

* Aap apne GitHub repository ke "Settings" section mein jakar "Manage access" tab pe click kar sakte hain.
* Wahan aap team members ko invite kar sakte hain email ke zariye.

### **6. Branches Ka Use Karna (Teamwork ke liye)**

Teamwork mein code ko organize karna important hota hai. Aap branches ka use kar sakte hain, taaki har team member apna kaam alag branch par kare aur master branch ko safe rakhe.

**New Branch Create Karna:** git checkout -b branch-name

**Branch Ko Push Karna:** git push origin branch-name

* **Branch Merge Karna (Pull Request ke zariye):** Jab aapka kaam complete ho, to GitHub par jaakar "Pull Request" create karein, taaki aapka kaam main branch mein merge ho sake.

### **7. Pull Requests Ka Use Karna**

* Jab aap ek new feature ya fix karte hain, to apne changes ko pull request ke through main branch mein merge kar sakte hain.
* Team members ko pull request par feedback de sakte hain aur agar sab kuch sahi ho to usko merge kar sakte hain.

### **8. Collaboration Tools**

* **Issues:** Agar koi bug ya task hai, to aap "Issues" section mein issue create kar sakte hain. Isse team ko pata rahega ki kya kaam pending hai.
* **Projects:** Aap GitHub par project boards bhi create kar sakte hain, jahan aap tasks ko organize kar sakte hain.
* **Actions:** Aap GitHub Actions ka use karke CI/CD (Continuous Integration/Continuous Deployment) set kar sakte hain, jisse har baar code push hone par automatic testing ya deployment ho sake.

### **9. GitHub Desktop (Optional)**

Agar aap terminal ke bajaye graphical interface chahte hain, to GitHub Desktop install kar sakte hain. Isse aap easily commit, push, aur pull kar sakte hain bina terminal ke.

### **Summary:**

1. GitHub par repository banaye.
2. Local machine par clone karein.
3. Code edit karne ke baad git add, git commit, git push ka use karein.
4. Team members ko invite karein.
5. Branches aur Pull Requests ka use karke collaboration ko smooth banayein.

Agar aapko koi bhi specific step mein help chahiye ho to bataiye, main aapko aur detail mein samjha sakta hoon!

Commands

**# 1. Check git version**

git --version

**# 2. Configure git (one-time setup)**

git config --global user.name "Aditya"

git config --global user.email "adityaprakash.tm@gmail.com"

**# 3. Clone a repository**

git clone<https://github.com/chauhan07/HTML-CSS-Javascript-Jquery-Projects-for-Beginners.git>

**# 4. Navigate into the repository**

cd HTML-CSS-Javascript-Jquery-Projects-for-Beginners

**# 5. Check the status of the repository**

git status

**# 6. Create and switch to a new branch (do this BEFORE editing)**

git checkout -b my-feature

**# ---- Make changes to files ----**

**# 7. Stage changes for commit**

git add .

**# 8. Commit changes with a message**

git commit -m "your message"

**# 9. Push changes to the remote repository (new branch)**

git push origin my-feature

**# 10. Delete a branch from remote (permanent)**

git push origin --delete my-feature

**# 11. Delete a branch locally**

git branch -d my-feature

**📥 Keeping Code Updated**

**# Pull latest changes from main branch**

git pull origin main

**# Fetch all remote branches without merging**

git fetch origin

**🔍 Checking Branches**

**# List all local branches**

git branch

**# List all remote branches**

git branch -r

**📝 Commit History**

**# Show commit history (simple)**

git log --oneline

**# Show commit history with graph**

git log --oneline --graph --all

**🔀 Branch Management**

**# Switch to an existing branch**

git checkout branch-name

**# Rename current branch**

git branch -m new-branch-name

**🔄 Undo / Fix Mistakes**

**# Unstage a file (remove from git add)**

git reset HEAD filename

**# Undo last commit but keep changes**

git reset --soft HEAD~1

**# Undo last commit and delete changes**

git reset --hard HEAD~1

**🧹 Cleanup**

# Delete all local branches already merged into main

git branch --merged main | grep -v "main" | xargs git branch -d

GitHub - chauhan07/HTML-CSS-Javascript-Jquery-Projects-for-Beginners: Multiple Projects

Multiple Projects. Contribute to chauhan07/HTML-CSS-Javascript-Jquery-Projects-for-Beginners development by creating an account on GitHub.

**Vikas**

git pull --rebase origin main  
git push origin main  
git push origin main --force

Revert Last Commit

### **1. Revert Last Commit (Without Removing Changes)**

If you want to undo the last commit but **keep the changes** in your working directory (so you can modify or re-commit them):

git reset --soft HEAD~1

This will undo the last commit but leave all changes in your working directory staged for commit.

### **2. Revert Last Commit (And Discard Changes)**

If you want to undo the last commit **and discard the changes** (so they’re not in your working directory anymore):

git reset --hard HEAD~1

This will remove the last commit completely and also discard any changes you made in that commit.

### **3. Push Changes to GitHub**

Once you’ve reset the commit locally, you'll need to push the changes to GitHub. If you've used git reset --hard, you may need to force-push to overwrite the remote commit:

git push origin HEAD --force

### **4. Reverting Commit With Git Revert (Safe Option)**

If you'd rather not mess with the commit history and want to create a new commit that undoes the last commit:

git revert HEAD

This will create a new commit that undoes the changes of the previous one. Then you can push it as usual:

git push origin main