=====================================Git & GitHut=============================================

GitHut Login detail: **Url: https://github.com/**

ID: gmail: \*\*\*\*\*/ Password: **Santosh!@123**

1. Loaded project Dotnetcore [**N\_TierCrud\_DotNetCore]:** <https://github.com/SantoshGupta111/MyDotNetCoreApp>

### **Quick Summary (Shortcut) – If any changes of existing in VS code then execute command one by one. Means update from git to githut online**

git status

git add .

git commit -m "Updated UI"

git push origin main

### **GitHub और Git को Connect करने के लिए (Git Bash में Login)**

अब अगर आप GitHub को अपने कंप्यूटर से कनेक्ट करना चाहते हैं, तो Git Bash (या Terminal) में यह कमांड यूज़ करें:

git config --global user.name "YourGitHubUsername"

git config --global user.email "YourGitHubEmail"

👉 इसमें "YourGitHubUsername" और "YourGitHubEmail" को अपने GitHub अकाउंट के अनुसार बदलें।

**Check करने के लिए:**

git config --global user.name

git config --global user.email

यह आपके सेट किए हुए नाम और ईमेल को दिखा देगा।

**I want convert existing all project from folder drvie of dotnet and dot net or any things into git and githut then process command below one by one**

open Folder in VS code:

**git init** (Git Initialize करें (अगर पहले से नहीं किया है))

**git add .** (सभी Files को Git में Add करें)

git commit -m "Initial commit" (Initial Commit करें) save karna

GitHub पर New Repository Create करें (Manually या GitHub CLI से) with choose Private ( or public)

फिर GitHub Repository का URL Copy करें (Example: https://github.com/YourUsername/YourRepo.git)

6️ GitHub Repository को Local Repository से Connect करें

git remote add origin https://github.com/YourUsername/YourRepo.git

7️ Code को GitHub पर Push करें

**git branch -M main** ( only first time in new create)

**git push -u origin main**

अब आपका .NET Project GitHub पर Upload हो चुका है! 🚀

Git Commands की पूरी जानकारी हिंदी में (With Explanation)

**git init**

📌 यह क्या करता है?

👉 यह आपके प्रोजेक्ट फ़ोल्डर को Git Repository में बदल देता है और .git नाम का एक Hidden Folder बनाता है।

👉 यह Command सिर्फ एक बार चलानी होती है, जब आप किसी नई Repository को Git में जोड़ रहे होते हैं।

**🔹 2️⃣ किसी Git Repository को Clone करना (GitHub से कॉपी करना)**

**git clone <Repository\_URL>**

📌 **यह क्या करता है?**  
👉 यह **GitHub पर मौजूद Repository** को आपके Local System में **डाउनलोड (copy)** करता है।  
👉 Example:

**git clone https://github.com/YourUsername/YourRepository.git**

👉 अब आपका पूरा प्रोजेक्ट आपके System में आ जाएगा।

**🔹 3️ Git में Files को Add करना (Staging में डालना)**

**git add . (dot with space must for all files/folders)**

📌 **यह क्या करता है?**  
👉 यह **सभी Modified Files** को Git में **Stage** कर देता है, ताकि हम उन्हें **Commit** कर सकें।  
👉 अगर सिर्फ एक File को Add करना हो:

**git add filename.ext**

**🔹 4️ Changes को Commit करना (Save करना)**

**git commit -m "Your Commit Message"**

📌 **यह क्या करता है?**  
👉 यह आपके Changes को **Git में Save (Commit)** कर देता है।  
👉 "Your Commit Message" में आप बता सकते हैं कि इस Change में क्या किया गया।

✍ **Example:**

**git commit -m "Login Page Updated with New Validation"**

👉 यह Changes को **Permanent रूप से Save** कर देता है, लेकिन अभी GitHub पर **Push नहीं** करता।

**🔹 5️⃣ Changes को GitHub पर Push करना (Upload करना)**

**git push origin main**

📌 **यह क्या करता है?**  
👉 यह आपके **Local Repository** से Changes को **GitHub Repository** पर Upload करता है।  
👉 यहाँ main आपका **Branch Name** है।  
👉 अगर पहली बार Push कर रहे हैं:

**git push -u origin main**

**🔹 6️ GitHub से Latest Code Pull करना (Update करना)**

**git pull origin main**

📌 **यह क्या करता है?**  
👉 अगर कोई और Team Member ने Code Update किया है, तो यह Command **Latest Code** को आपके System में लाएगी।  
👉 यह **Merge Conflicts** से बचने के लिए ज़रूरी होती है।

**🔹 7 Branch Create और Switch करना**

✅ **नई Branch बनाना:**

**git branch new-feature**

📌 **यह क्या करता है?**  
👉 यह एक **नई Branch (new-feature)** बनाएगा।

✅ **Branch पर स्विच करना:**

**git checkout new-feature**

📌 **यह क्या करता है?**  
👉 यह आपको new-feature Branch पर ले जाएगा।

✅ **Shortcut (Create & Switch Together)**

**git checkout -b new-feature**

✅ **Branch को Delete करना:**

**git branch -d new-feature**

**🔹 8️ Changes को Undo करना (वापस लाना)**

✅ **Staged Changes को हटाना (Undo Staging)**

**git reset HEAD filename.ext**

✅ **Last Commit को Undo करना (Commit हटाना)**

**git reset --soft HEAD~1**

✅ **Hard Reset (Commit और Changes दोनों हटाने के लिए)**

**git reset --hard HEAD~1**

**🔹 9️ Status और Logs देखना**

✅ **Git Status Check करना** (कौनसी फ़ाइलें Modified हैं?)

**git status**

✅ **Commit History देखना**

**git log**

**🔹 1️0 Remote Repository को Change करना**

✅ **अगर आपने गलती से गलत Remote URL सेट कर दिया, तो इसे Update कर सकते हैं:**

**git remote set-url origin <New\_Repository\_URL>**

**🎯 Summary Table (Quick Reference)**

| **Git Command** | **क्या करता है?** |  |
| --- | --- | --- |
| git init | New Repository शुरू करता है |  |
| git clone <URL> | GitHub से Code Download करता है |  |
| git add . | Changes को Stage करता है |  |
| git commit -m "Message" | Changes को Save करता है |  |
| git push origin main | GitHub पर Upload करता है |  |
| git pull origin main | Latest Code को Update करता है |  |
| git branch new-feature | नई Branch बनाता है |  |
| git checkout new-feature | नई Branch पर Switch करता है |  |
| git merge new-feature | Branch को Merge करता है |  |
| git reset --hard HEAD~1 | Last Commit को Undo करता है |  |
| git status | Modified Files दिखाता है |  |
| git log | Commit History दिखाता है |  |

**🚀 Bonus: Git के बारे में और सीखने के लिए क्या करें?**

✅ **GitHub पर Practice करें**  
✅ **Real Projects में Git का इस्तेमाल करें**  
✅ **GitHub पर Open Source Projects में Contribute करें**

**====================================file see ================**

### **1. सभी फाइलों के बदलाव और उनका पूरा Path देखने के लिए:**

git log --name-status --pretty=format:"%h - %an, %ar : %s"

✅ **Output Example:**

a1b2c3d - Raj, 2 days ago : Updated login feature

M src/controllers/UserController.cs

A src/models/User.cs

D src/views/UserView.cshtml

🔹 **M** = Modified (बदला गया)  
🔹 **A** = Added (नई फ़ाइल जोड़ी गई)  
🔹 **D** = Deleted (हटा दी गई)

### 🔹 **2. फ़ाइलों के Full Path के साथ Detailed History देखने के लिए:**

git log --name-only --pretty=format:"%h - %an, %ar : %s" --full-diff

🔹 इससे यह दिखेगा कि **कौन सी फाइल, किसने बदली, कब बदली, और उनका पूरा पथ क्या है।**

### 🔹 **3. किसी Specific Folder में सभी बदलाव देखने के लिए:**

git log --name-status -- folder\_name/

**Example:**

git log --name-status -- src/

🔹 यह src फ़ोल्डर में हुए सभी बदलावों को दिखाएगा।

### 🔹 **4. किसने कौन सी फाइल बदली और उसका पूरा Path:**

git log --format="%h %an %ad" --name-only --date=short

🔹 यह **commit ID, Author, Date और File का पूरा Path** दिखाएगा।

### 🔹 **5. किसी File में Detailed Change History देखने के लिए:**

git log -p -- filename.ext

**Example:**

git log -p -- src/config/appsettings.json

🔹 यह appsettings.json फ़ाइल में **क्या बदलाव हुआ था, वह भी line-by-line** दिखाएगा।

### 🔹 **6. किसी ने कौन-कौन सी फाइलें बदली (Summary Format):**

git log --stat

🔹 यह दिखाएगा कि **कौन-कौन सी फाइलें बदली गईं और उनमें कितनी लाइनें जोड़ी या हटाई गईं।**

### 🔹 **7. Git Blame: किसने कौन सी लाइन बदली किसी File में**

git blame filename.ext

**Example:**

git blame src/controllers/UserController.cs

=============================================Maximum uses in company team environment================

### **🚀 Company में सबसे ज़्यादा Use होने वाले Git Commands (Teamwork के लिए)**

#### ✅ **Repository Clone (पहली बार Repository को Local में लाने के लिए)**

sh

CopyEdit

git clone https://github.com/YourUsername/YourRepo.git

#### ✅ **Branch Create & Switch (नया Feature या Bug Fixing के लिए)**

sh

CopyEdit

git checkout -b feature-branch-name

#### ✅ **Latest Changes Fetch करें (Team के Changes लाने के लिए)**

sh

CopyEdit

git fetch origin

#### ✅ **Pull Latest Changes (Team के Latest Changes को Merge करने के लिए)**

sh

CopyEdit

git pull origin main

#### ✅ **Code को Stage करने के लिए (सिर्फ Changed Files)**

sh

CopyEdit

git add .

#### ✅ **Commit करने के लिए (अपने Changes को Save करने के लिए)**

sh

CopyEdit

git commit -m "Your message about the changes"

#### ✅ **Push करने के लिए (अपने Changes को GitHub पर भेजने के लिए)**

sh

CopyEdit

git push origin feature-branch-name

#### ✅ **Main Branch पर Merge करने के लिए (Pull Request के बाद)**

sh

CopyEdit

git checkout main

git merge feature-branch-name

git push origin main

#### ✅ **Branch Delete करने के लिए (Merge के बाद)**

sh

CopyEdit

git branch -d feature-branch-name

git push origin --delete feature-branch-name

#### ✅ **Status Check करने के लिए (कौन-कौन से Changes हैं)**

sh

CopyEdit

git status

#### ✅ **कौन-कौन से Changes हुए हैं देखने के लिए**

sh

CopyEdit

git diff

#### ✅ **Last 10 Commits Log देखने के लिए**

sh

CopyEdit

git log --oneline -n 10

#### ✅ **Specific File की History देखने के लिए**

sh

CopyEdit

git log --follow -- filename.txt

#### ✅ **अगर गलती से Code Stage कर दिया हो तो Unstage करने के लिए**

sh

CopyEdit

git reset HEAD filename.txt

#### ✅ **अगर गलती से Last Commit गलत हो गया हो तो Undo करने के लिए**

sh

CopyEdit

git reset --soft HEAD~1

#### ✅ **Local Branch List देखने के लिए**

sh

CopyEdit

git branch

#### ✅ **Remote Branch List देखने के लिए**

sh

CopyEdit

git branch -r

#### ✅ **अगर किसी Specific Commit पर वापस जाना हो**

sh

CopyEdit

git checkout <commit-hash>

#### ✅ **Hard Reset करने के लिए (Local Changes हटाने के लिए)**

sh

CopyEdit

git reset --hard origin/main

#### ✅ **Stash (Temporary Changes को Save करने के लिए)**

sh

CopyEdit

git stash

फिर वापस लाने के लिए:

sh

CopyEdit

git stash pop

### **🎯 यह Git Commands आपके Team Environment में सबसे ज़्यादा Use होंगे! 🚀**

===================================================Branch/Tag============

## ****Git में Branch और Tags क्या होते हैं?****

### **1️⃣ Git Branch क्या होता है?**

Git में **Branch** एक अलग Workspace की तरह होता है जहाँ आप नए Features या Bug Fixes पर काम कर सकते हैं बिना Main Codebase को प्रभावित किए।

👉 **Example:**

* **main Branch** → Production Code (Stable Version)
* **dev Branch** → Development Code (New Features Work)
* **feature-login Branch** → सिर्फ़ Login Feature पर काम करने के लिए

#### ✅ **Branch Create & Use करने के Commands:**

sh

CopyEdit

git branch feature-branch-name # नया Branch बनाना

git checkout feature-branch-name # उस Branch पर जाना

git checkout -b feature-branch-name # Create और Switch दोनों एक साथ

git branch # Available Branches देखना

git branch -d feature-branch-name # Local Branch Delete करना

git push origin --delete feature-branch-name # Remote से Branch हटाना

### **2️⃣ Git Tags क्या होता है?**

Git में **Tag** किसी Particular Commit को Mark करने के लिए होता है। यह ज़्यादातर **Version Releases** के लिए Use होता है, जैसे v1.0, v2.1.5 इत्यादि।

👉 **Example:**

* v1.0 → First Release
* v2.0 → Major Update
* v2.1.5 → Minor Bug Fix Update

#### ✅ **Tag Create & Use करने के Commands:**

sh

CopyEdit

git tag v1.0 # Simple Tag Create करना

git tag -a v1.0 -m "Version 1.0 release" # Annotated Tag Create करना

git show v1.0 # Tag Details देखना

git push origin v1.0 # Remote पर Tag भेजना

git push origin --tags # सभी Tags Push करना

git tag -d v1.0 # Local Tag Delete करना

git push origin --delete v1.0 # Remote से Tag Delete करना

### **🔍 Branch vs Tags में क्या अंतर है?**

| **Feature** | **Branch** | **Tag** |
| --- | --- | --- |
| Purpose | नए Feature या Development के लिए | किसी Specific Version को Mark करने के लिए |
| Editable | **हाँ**, आप Changes कर सकते हैं | ❌ **नहीं**, यह सिर्फ़ एक Snapshot होता है |
| Movement | **हाँ**, Branch आगे बढ़ सकता है | ❌ **नहीं**, Tag Fixed रहता है |
| Merge | **हाँ**, Merge कर सकते हैं | ❌ **नहीं**, Merging Possible नहीं |

### **🎯 Conclusion (Final Thought)**

* **Branch** का Use **Development & New Features** के लिए होता है।
* **Tags** का Use **Versioning & Release Marking** के लिए होता है।
* **Branch को Merge & Delete कर सकते हैं, लेकिन Tags Static होते हैं।**

💡 **अब आप Git में Branch और Tag दोनों का Use कर सकते हैं! 🚀**