**Git - GitHub**

# **Introduction To Git & GitHub**

**What is Git?**

Git is a **distributed version control system (VCS)** that helps developers track changes in their code, collaborate with others, and manage different versions of a project efficiently.

**Key Features of Git:**

1. **Version Control** – Tracks changes to files over time.
2. **Branching & Merging** – Allows working on different features independently and merging them when ready.
3. **Distributed System** – Every developer has a complete history of the repository.
4. **Staging Area** – Changes can be reviewed before committing them to the repository.
5. **Speed & Efficiency** – Fast operations with minimal storage requirements.

**What is GitHub?**

GitHub is a **cloud-based platform** that provides hosting for Git repositories.   
It enables collaboration, code sharing, and project management among developers.

**Git vs. GitHub**

|  |  |  |
| --- | --- | --- |
| Feature | Git | GitHub |
| Type | Version control system | Cloud-based hosting service |
| Purpose | Tracks changes in code | Stores and shares repositories |
| Accessibility | Local and remote | Online via GitHub.com |
| Collaboration | Local branches & merges | Pull requests & issues |

**Basic Git Commands**

1. **Initialize a Repository**

git init

1. **Clone a Repository**

git clone <repository\_url>

1. **Check Repository Status**

git status

1. **Add Changes to Staging Area**

git add <file\_name> # Add specific file

git add . # Add all files

1. **Commit Changes**

git commit -m "Commit message"

1. **Push Changes to Remote Repository**

git push origin <branch\_name>

1. **Pull Changes from Remote Repository**

git pull origin <branch\_name>

1. **Create a New Branch**

git branch <new\_branch>

1. **Switch to a Branch**

git checkout <branch\_name>

1. **Merge a Branch**

git merge <branch\_name>

Would you like a hands-on example or a GitHub workflow explanation? 🚀

# **Basic Git Configuration**

Before using Git, it's essential to configure your identity and preferences. Here’s how you can set up Git after installation:

**1. Set Your Identity (Name & Email)**

Git uses your name and email in commit messages. Configure them using the following commands:

git config --global user.name "Your Name"

git config --global user.email "your-email@example.com"

To verify your configuration:

git config --global --list

**2. Set Default Editor (Optional)**

Git uses a text editor for commit messages. You can set your preferred editor:

* **VS Code**:

git config --global core.editor "code --wait"

* **Vim**:

git config --global core.editor "vim"

**3. Set Default Branch Name (Optional)**

By default, Git names the first branch master. To change it to main, run:

git config --global init.defaultBranch main

**4. Enable Colored Output (Improves Readability)**

git config --global color.ui auto

**5. Cache Login Credentials (Avoid Re-entering Passwords)**

To store your credentials for a session:

git config --global credential.helper cache

Or, store them permanently:

git config --global credential.helper store

**6. Configure Line Endings (For Windows Users)**

To prevent issues when working on different OS:

* **Windows:**

git config --global core.autocrlf true

* **Mac/Linux:**

git config --global core.autocrlf input

**7. Verify Your Configuration**

Check your current Git settings with:

git config --list

**8. Unset a Configuration (If Needed)**

If you need to remove a setting, use:

git config --global --unset user.name

git config --global --unset user.email

🔹 **Now you’re all set!** Do you need help with initializing a repository or connecting Git with GitHub? 🚀

# **Secure Connection in Git (HTTPS vs. SSH)**

When working with Git and GitHub, you need a secure way to authenticate and push/pull code.

There are two primary methods:

1. **HTTPS (Secure but requires authentication for every push/pull)**
2. **SSH (More secure and recommended for frequent use)**

**1️- Using HTTPS (with Personal Access Token)**

GitHub removed password authentication for Git over HTTPS. Instead, you must use a **Personal Access Token (PAT).**

**Step 1: Generate a Personal Access Token (PAT)**

1. Go to [GitHub → Settings → Developer Settings → Personal Access Tokens](https://github.com/settings/tokens).
2. Click **"Generate new token (classic)"**.
3. Select the necessary scopes (e.g., repo for full repository access).
4. Click **"Generate Token"** and **copy** it (you won’t see it again).

**Step 2: Use PAT Instead of Password**

When pushing or pulling:

git clone https://github.com/username/repository.git

Git will ask for your **GitHub username** and instead of your password, paste the **Personal Access Token**.

**Step 3: Store Credentials to Avoid Re-entering**

git config --global credential.helper store

Next time, Git will remember your token.

**2️- Using SSH (Recommended for Secure Authentication)**

Using SSH keys eliminates the need to enter credentials frequently.

**Step 1: Check for Existing SSH Key**

Run the following command to check if you already have an SSH key:

ls ~/.ssh

If you see files like id\_rsa.pub, you already have an SSH key.

**Step 2: Generate a New SSH Key (If Needed)**

If no SSH key exists, generate one:

ssh-keygen -t rsa -b 4096 -C "your-email@example.com"

* Press **Enter** to save it in the default location (~/.ssh/id\_rsa).
* Optionally, set a passphrase for extra security.

**Step 3: Start the SSH Agent**

Run:

eval "$(ssh-agent -s)"

ssh-add ~/.ssh/id\_rsa

**Step 4: Add SSH Key to GitHub**

1. Copy your SSH key:

cat ~/.ssh/id\_rsa.pub

1. Go to **GitHub → Settings → SSH and GPG Keys**.
2. Click **"New SSH Key"**, paste the key, and save.

**Step 5: Test the SSH Connection**

Run:

ssh -T git@github.com

If successful, you'll see:

Hi username! You've successfully authenticated, but GitHub does not provide shell access.

**Step 6: Use SSH for Git Operations**

Instead of HTTPS, clone using SSH:

git clone git@github.com:username/repository.git

Now, you can push/pull without entering credentials.

**Which One Should You Use?**

|  |  |  |
| --- | --- | --- |
| Feature | HTTPS (PAT) | SSH |
| Security | Secure but requires frequent authentication | More secure with key-based access |
| Ease of Use | Requires token for every session | One-time setup, then seamless use |
| Best For | Occasional contributors | Frequent developers |

💡 **Recommendation**: Use **SSH** for a seamless, secure workflow.

Would you like help setting up SSH in Windows (Git Bash) or Linux/Mac? 🚀