# **Table of Contents**

[Table of Contents 1](#_Toc185492731)

[Install Git on Windows 2](#_Toc185492732)

[**Useful Git commands** 2](#_Toc185492733)

[**Install Git on Windows** 6](#_Toc185492734)

[**Create a New Folder and Initialize Git** 7](#_Toc185492735)

[**Run the Following Commands in Git Bash** 8](#_Toc185492736)

[**Create a GitHub Account** 12](#_Toc185492737)

[**Create a Repository on GitHub** 12](#_Toc185492738)

[**HTTPS Method** 13](#_Toc185492739)

[**Clone Repository** 13](#_Toc185492740)

[**Steps to push a new file to your GitHub repository** 14](#_Toc185492741)

[**Steps to pull a new file to your GitHub repository** 17](#_Toc185492742)

[**SSH Method** 19](#_Toc185492743)

[**Personal Access Token (PAT) Method** 22](#_Toc185492744)

[**Merge case** 24](#_Toc185492745)

[**Understanding git pull Options with Examples and Diagrams** 31](#_Toc185492746)

[**git pull --no-rebase (Default Behavior)** 31](#_Toc185492747)

[**git pull --rebase** 32](#_Toc185492748)

[**git pull --ff-only** 32](#_Toc185492749)

# Install Git on Windows

**Diagram from Claas**

A diagram of a diagram of a product

Description automatically generated with medium confidence

### **Useful Git commands**

# **Initialize a new Git repository in the current directory**

git init

# **Check the current status of the repository (which files are staged, modified, etc.)**

git status

# **Add files to the staging area (use '.' to add all files)**

git add .

# **Commit changes with a message describing the change**

git commit -m "Your commit message"

# **View the commit history in a concise format (shows commit hashes and messages)**

git log --oneline

# **View the commit history in a detailed format**

git log

# **Create a new branch**

git branch <branch\_name>

# **List all local branches**

git branch

# **Switch to an existing branch**

git checkout <branch\_name>

# Alternatively, you can use:

git switch <branch\_name>

# **Create and switch to a new branch in one command**

git checkout -b <branch\_name>

# Or alternatively:

git switch -c <branch\_name>

# **Merge a branch into the current branch**

git merge <branch\_name>

# **Check which branch you are currently on**

git branch --show-current

# **View remote repositories**

git remote -v

# **Add a remote repository (useful when cloning or connecting your local repo to a remote repo)**

git remote add origin <repo\_url>

# **Change the URL of a remote repository (useful when the remote URL needs to be updated)**

git remote set-url origin <new\_repo\_url>

# **Fetch updates from the remote repository (without merging changes)**

git fetch

# **Pull updates from the remote repository (fetches and merges changes)**

git pull

# **Pull updates from the remote repository and disable rebase (force a merge commit)**

git pull --no-rebase

# **Pull updates from the remote repository, but only if a fast-forward merge is possible**

git pull --ff-only

# **Push your local commits to the remote repository**

git push

# **Delete a local branch**

git branch -d <branch\_name>

# **Delete a remote branch**

git push origin --delete <branch\_name>

# **Rename a branch locally**

git branch -m <old\_name> <new\_name>

# **Rename the current branch**

git branch -M <new\_name>

# **Show the status of all branches (local and remote)**

git branch -a

# **View changes in your working directory compared to the last commit**

git diff

# **View changes in a specific commit**

git show <commit\_id>

# **Reset your repository to a previous commit (will unstage or remove changes)**

git reset --hard <commit\_id>

# **Check the status of files in the staging area**

git diff --staged

# **Revert a commit (undo the changes made by a previous commit)**

git revert <commit\_id>

# **See a graphical representation of your branch history**

git log --oneline --graph --all

# **View the differences between your current branch and the remote repository**

git diff origin/<branch\_name>

# **View a specific file's history**

git log -- <file\_name>

# **Compare two branches**

git diff <branch1>..<branch2>

# **Stage a specific file**

git add <file\_name>

# **Track a new file**

git add <file\_name>

# **Clone a repository from GitHub or another remote source**

git clone <repo\_url>

# **Check the remote repository's URL**

git remote show origin

# **Undo the last commit (but keep the changes in the working directory)**

git reset --soft HEAD~1

# **View the differences between the last two commits**

git diff HEAD~2..HEAD

# **Check out a specific commit (detached HEAD state)**

git checkout <commit\_id>

# **Create a tag for a specific commit (useful for marking releases)**

git tag <tag\_name> <commit\_id>

# **Push tags to the remote repository**

git push origin --tags

# **Fetch all branches from the remote repository**

git fetch --all

# **Pull only a specific branch from the remote repository**

git pull origin <branch\_name>

# **List all remotes configured for the repository**

git remote -v

# **Check for untracked files (files that are not being tracked by Git)**

git ls-files --others --exclude-standard

# **Discard changes in a file (reset file to last commit state)**

git checkout -- <file\_name>

# **Show detailed statistics of changes in commits**

git log --oneline --stat

# **Show a summary of changes in each commit**

git log --oneline --patch

# **Check out and create a new branch from an existing commit**

git checkout -b <new\_branch> <commit\_id>

# **Clean untracked files and directories**

git clean -fd

# **Check the current branch's remote tracking branch**

git rev-parse --abbrev-ref <branch\_name>@{u}

# **Fetch changes from the remote repository without merging (only downloading)**

git fetch origin <branch\_name>

# **Change the default branch name locally (e.g., from master to main)**

git branch -M main

# **Create an alias for a common git command (e.g., to shorten commands)**

git config --global alias.st status

# **View configuration settings for your git repo**

git config --list

# **Set your username and email for commits (global settings)**

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

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

# **See the status of the local repository in a more human-friendly format**

git status -s

### **Install Git on Windows**

Open your browser and visit the official Git website: <https://git-scm.com/downloads/win>

Click the **"Download for Windows"** button.

A screenshot of a computer

Description automatically generated

The download will start automatically for the latest version

Locate the downloaded .exe file (usually in the Downloads folder).

Double-click the installer to launch it.

A screenshot of a computer

Description automatically generated

Click **Next** until the installation begins.

Click **Finish** once the installation is complete.

### **Create a New Folder and Initialize Git**

Navigate to the location where you want the new folder. **Right-click** → **New** → **Folder**.

A screenshot of a computer

Description automatically generated

Name the folder (e.g., my-git-project).

A screenshot of a computer

Description automatically generated

Right-click inside the new folder. Select **Git Bash Here**.

A screenshot of a computer

Description automatically generated

### **Run the Following Commands in Git Bash**

# Configure User Details (Global Settings)

git config --global user.name <name>

A screenshot of a computer

Description automatically generated

git config --global user.email <email>

A screen shot of a computer

Description automatically generated

Initialize the Git Repository

git init

A screen shot of a computer program

Description automatically generated

**Create a File for Testing:** Use this command to create a sample file:

touch example.txt

A black background with yellow text

Description automatically generated

**Add Files to Staging Area:** Add your file(s) to the staging area:

git add example.txt

A screen shot of a computer

Description automatically generated

**Check Git Status:** Verify the status of the repository:

git status

A screenshot of a computer screen

Description automatically generated

**See the files in the staging area**:

git diff --cached # Show staged changes (details of changes)

git diff --name-only --cached # Show file names in the staging area

A screen shot of a computer program

Description automatically generated

**Commit the Changes:** Commit the changes to the local repository

git commit -m "Initial commit"

A screenshot of a computer program

Description automatically generated

**See files in the local repository:**

git ls-files # List all tracked files in the repository

git log --stat # Show commits with files changed in each

A computer screen with text and numbers

Description automatically generated

### **Create a GitHub Account**

If you don’t have a GitHub account, follow these steps to create one:

Visit [GitHub](https://github.com/).

Click on **Sign up** in the top right corner.

Enter your email address, create a password, and choose a username.

Verify your email and complete the CAPTCHA to confirm you're not a robot.

Complete the setup by following the on-screen instructions.

### **Create a Repository on GitHub**

**Login to GitHub**: Go to [GitHub](https://github.com/) and log in with your credentials.

In the top right corner of the GitHub homepage, click on the **+** icon and select **New repository**.

A screenshot of a computer

Description automatically generated

**Repository name**: Enter a unique name for your repository

A screenshot of a computer

Description automatically generated

**Create Repository**: Click on the **Create repository** button to finalize the creation

A screenshot of a computer

Description automatically generated

### **HTTPS Method**

**create a new folder on your local machine** and **access Git Bash** by right-clicking on it. Run the below command.

#### **Clone Repository**

**Clone Repository**: git clone [https://github.com/<USERNAME>/<REPO\_NAME>.git](https://github.com/%3cUSERNAME%3e/%3cREPO_NAME%3e.git)

Ex: git clone <https://github.com/tprabhakarreddy/Devops.git>

A screen shot of a computer

Description automatically generated

**cd devops**

A black screen with yellow text

Description automatically generated

**Verify Remote Repository:**  git remote -v

A screen shot of a computer

Description automatically generated

**Update Remote URL to HTTPS**

git remote set-url origin [https://github.com/<USERNAME>/<REPO\_NAME>.git](https://github.com/%3cUSERNAME%3e/%3cREPO_NAME%3e.git)

ex: git remote set-url origin https://github.com/tprabhakarreddy/Devops.git

A computer screen with text

Description automatically generated

#### **Steps to push a new file to your GitHub repository**

Create a new file

touch file1

A black screen with yellow text

Description automatically generated

**Stage the New File:** Add the new file to Git's staging area

git add file1

A black screen with yellow text

Description automatically generated

Shows the number of commits and details.

git log

A computer screen shot of a computer code

Description automatically generated

View files added/modified in a particular commit.

A screen shot of a computer

Description automatically generated

commit your changes to local repository

git commit -m "Add newfile.txt"

A black screen with yellow text

Description automatically generated

Push the changes to your remote GitHub repository

git push

A screenshot of a computer

Description automatically generated

Observer that the file has been pushed into Github repository

A screenshot of a computer

Description automatically generated

#### **Steps to pull a new file to your GitHub repository**

In your repository, click on the **Add file** button and select **Create new file**

A screenshot of a chat

Description automatically generated

Enter File Name and Content and Commit the New File

A screenshot of a computer

Description automatically generated

Click on Commit changes

A screenshot of a computer

Description automatically generated

Once the file is created on GitHub, you can **pull** it to your local machine

A computer screen with white text

Description automatically generated

### **SSH Method**

Generate SSH Key using below command

ssh-keygen

Hit enter in all others steps

A computer screen shot of a black screen

Description automatically generated

This generates two files:

* **Private Key**: ~/.ssh/id\_rsa
* **Public Key**: ~/.ssh/id\_rsa.pub

A computer screen shot of a code

Description automatically generated

Copy the public key to the clipboard

A black background with colorful letters

Description automatically generated

Go to **GitHub > Settings**

A screenshot of a computer

Description automatically generated

**SSH and GPG keys > Add New SSH Key**

A screenshot of a computer

Description automatically generated

Add key details

A screenshot of a computer

Description automatically generated

Enter your password to confirm your details.

A screenshot of a login form

Description automatically generated

**We can push the file using the following sequence of commands: git add \*, git commit -m "my new file", and git push, similar to the HTTPS method**

### **Personal Access Token (PAT) Method**

Go to **GitHub > Settings**

A screenshot of a computer

Description automatically generated

Choose Developer Settings

A screenshot of a computer

Description automatically generated

**Personal Access Token > Token (classic)**.

Click on Generate a new token

A screenshot of a computer

Description automatically generated

Enter name of the key

A screenshot of a computer

Description automatically generated

Choose scope options

A screenshot of a computer

Description automatically generated

Click on Generate token

A screenshot of a computer

Description automatically generated

Copy the token and paste in notepad

A screenshot of a computer

Description automatically generated

Clone Repository Using PAT

git clone [https://<TOKEN>@github.com/<USERNAME>/<REPO\_NAME>.git](https://%3cTOKEN%3e@github.com/%3cUSERNAME%3e/%3cREPO_NAME%3e.git)

ex: git clone <https://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@github.com/tprabhakarreddy/Devops.git>

A computer screen with white text

Description automatically generated

**We can push the file using the following sequence of commands: git add \*, git commit -m "my new file", and git push, similar to the HTTPS method**

#### **Merge case**

When the **local repository** and **remote repository** are **not in sync**, and the local repository is trying to add a new file but the last commits (either local or remote) are out of sync, Git will attempt to merge the changes. This situation typically leads to a **merge conflict** if both the local and remote repositories have changes in the same file or the same section of the file.

##### **Create a new file from your machine**

**Clone Repository into local on your windows machine** :

git clone [https://github.com/<USERNAME>/<REPO\_NAME>.git](https://github.com/%3cUSERNAME%3e/%3cREPO_NAME%3e.git)

Ex: git clone <https://github.com/tprabhakarreddy/mergepractice.git>

A black screen with yellow text

Description automatically generated

**Create and commit an initial file**:

cd mergepractice

echo "Hello, World!" > file.txt

git add file.txt

git commit -m "Initial commit"

A screen shot of a computer

Description automatically generated

**Push the initial commit to the remote repository**:

git push

A screen shot of a computer program

Description automatically generated

At this point, both the local and remote repositories are in sync, and the main branch has the same commit history.

git log

A screenshot of a computer screen

Description automatically generated

##### **Create new file from ubuntu EC2 instance**

Connect to ubuntu EC2 instance

ssh-keygen

A screenshot of a computer

Description automatically generated

Add below public key in Github

A screenshot of a computer

Description automatically generated

**Clone Repository into local on EC2 instance** :

git clone [git@github.com:tprabhakarreddy/mergepractice.git](mailto:git@github.com:tprabhakarreddy/mergepractice.git) # change to your repository

A screen shot of a computer program

Description automatically generated

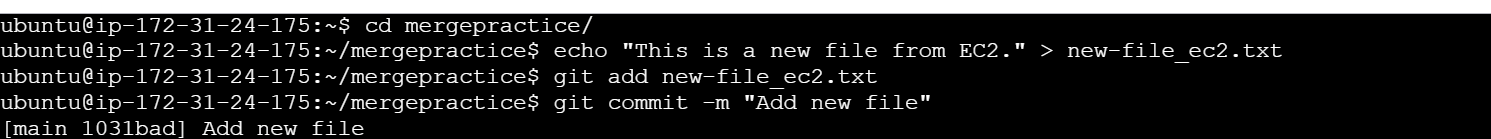
**On your EC2 machine (in the main branch), add a new file**:

cd mergepractice/

echo "This is a new file from EC2." > new-file\_ec2.txt

git add new-file\_ec2.txt

git commit -m "Add new file"



**Push the local commit to the remote repository**:

git push

A screenshot of a computer screen

Description automatically generated

##### **Create another new file from your machine**

Now, local repository on your local machine and Remote Repository are Out of Sync

echo "This is a new file." > new-file.txt

git add new-file.txt

git commit -m "Add new file"

A screen shot of a computer

Description automatically generated

Observe that local and remote repositories are out of sync

A screen shot of a computer screen

Description automatically generated

**Push the local commit to the remote repository**

git push

However, since the remote branch has changes that are not present locally (the changes to file.txt), Git will reject the push and ask you to first pull the changes

A screen shot of a computer screen

Description automatically generated

##### **Merge Conflict Scenario: Local and Remote Are Out of Sync**

To resolve the conflict, you need to **pull the changes** from the remote first. This will bring in the remote commits and try to merge them with your local changes.

git pull

esc then :wq

A screenshot of a computer program

Description automatically generated

Git will attempt to merge the changes.

A screen shot of a computer

Description automatically generated

Push the Changes to Remote

git push

A screenshot of a computer screen

Description automatically generated

Observe the commit on remote repository

A close-up of a computer screen

Description automatically generated

### **Understanding git pull Options with Examples and Diagrams**

The git pull command is used to fetch and integrate changes from a remote repository into your local branch. Depending on the scenario, different pull options can be used to manage how changes are merged or rebased. This guide explains three common options:

1. **--no-rebase**: Performs a merge, preserving branch history with a merge commit.
2. **--rebase**: Re-applies local commits on top of the remote branch, creating a clean, linear history.
3. **--ff-only**: Ensures fast-forward merges only, aborting if the branch cannot be fast-forwarded.

Each option is illustrated with examples and diagrams to clarify their behavior and usage.

#### **git pull --no-rebase (Default Behavior)**

This merges the fetched branch into the current branch, creating a **merge commit**. Both histories are preserved.

**Diagram**

**Before git pull --no-rebase:**

Local: A - D - E

Remote: A - B – C

**After git pull --no-rebase:**

Merged: A - B - C

\

D - E

\

Merge Commit (M)

**Command:**

git pull --no-rebase

**When to Use:**

When you want to preserve the commit history and see the exact branching and merging of work.

#### **git pull --rebase**

This re-applies your local commits on top of the fetched branch, making the history linear. **No merge commit** is created.

**Diagram**

**Before git pull --rebase:**

Local: A - D - E

Remote: A - B - C

**After git pull --rebase:**

Rebased: A - B - C - D - E

**Command:**

git pull –rebase

**When to Use:**

* When you want a clean, linear commit history without merge commits.
* Ideal for collaborative workflows where clean history is important.

#### **git pull --ff-only**

This attempts a **fast-forward merge**. It moves the branch pointer to the latest commit if there are no local changes. If fast-forwarding isn’t possible, the pull is aborted.

**Diagram**

**Before git pull --ff-only:**

Local: A

Remote: A - B - C

**After git pull --ff-only:**

Fast-Forwarded: A - B - C

**Command:**

git pull --ff-only

**When to Use:**

* When you want to avoid merge commits entirely and ensure that the branch history remains unchanged.
* Use when you’re certain there are no local changes or divergences.