**Git Basic Commands**

**⦁ What is Git ?**

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

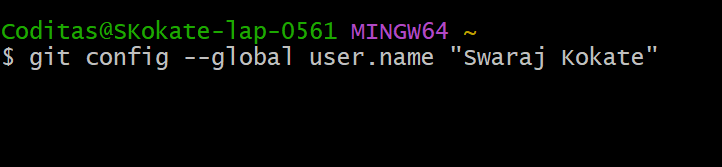
Git relies on the basis of distributed development of software where more than one developer may have access to the source code of a specific application and can modify changes to it that may be seen by other developers..

* **Basic Flow:**

It is a good idea to introduce yourself to Git with your name and public email address before doing any operation. The easiest way to do so is:

**1) For configuring User Name :**

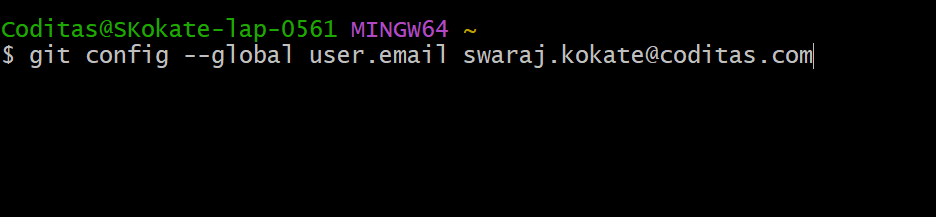
**git config --global user.name "Your Name Comes Here"**



Here, I have set the name as Swaraj Kokate.

**2) For configuring User Email :**

**git config --global user.email** [**you@yourdomain.example.com**](mailto:you@yourdomain.example.com)



Here, I have set email as swaraj.kokate@coditas.com

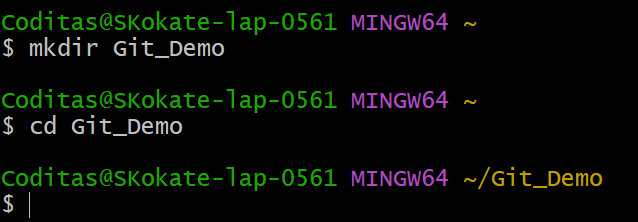
**3) Create and open a Folder :**

**a) Create Folder**

**mkdir Folder\_Name**

**b) Open Folder**

**cd Folder\_Name**

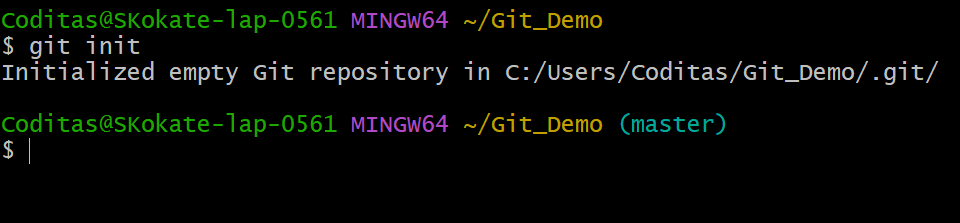


Here, folder name is Git\_Demo

**4) Initialize working directory to repository :**

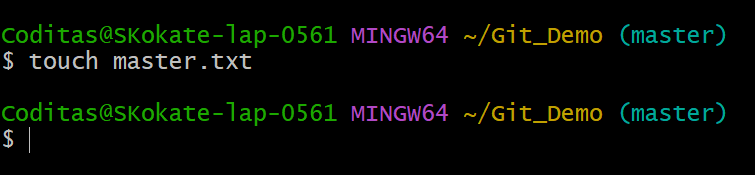
**git init**

Note : you may notice a new directory created, named ".git".



**5) Create a text file :**

**touch File\_Name (for example,touch F1.txt)**

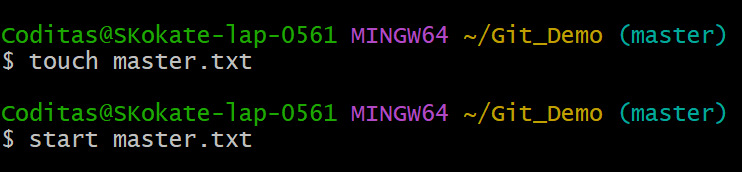


Here, the file name is master.txt

**6) Open the created fie in notepad :**

**notepad File\_Name**

note : If the command is giving an error then use Start File\_Name



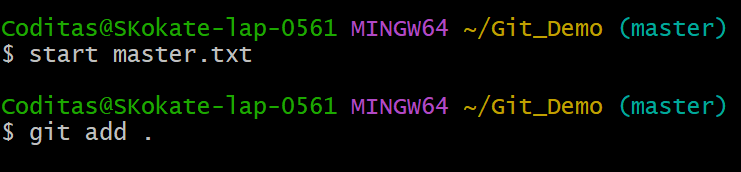
**7) To know the state of working directory :**

**git status**

**8) Next, tell Git to take a snapshot of the contents of all files under the current directory (note the .)**

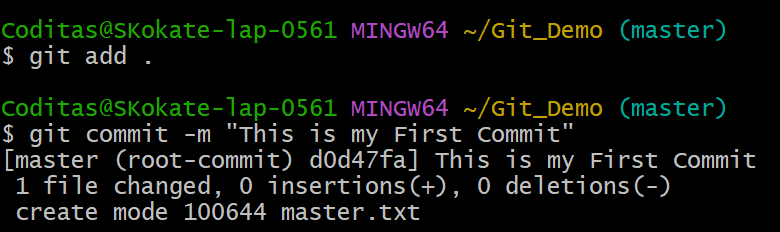
**git add . (for current file)**

Note : For specific file use git add File\_Name



**9) This snapshot is now stored in a temporary staging area which Git calls the "index". You can permanently store the contents of the index in the repository with :**

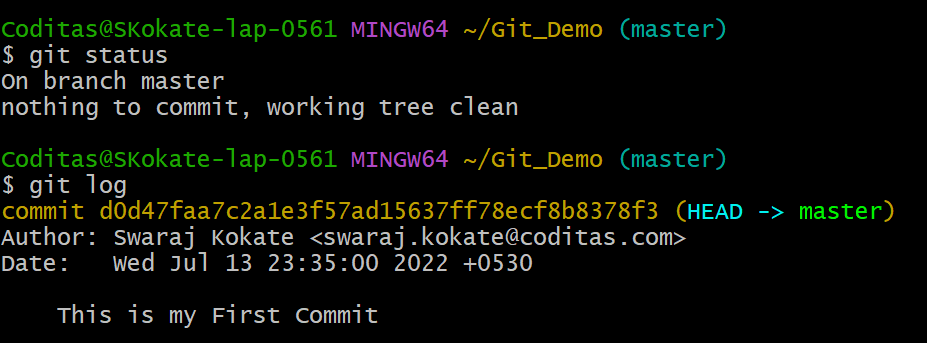
**git commit -m "Message" (-m in here is for the message)**



You’ve now stored the first version of your project in Git.

**10) Check the state of working directory :**

**git status**



**11) In order to check all the commits in a repository's history :**

git log

* **How to avoid conflict in Git ?**

### **Standardize formatting rules:**

Many times conflicts occur because of formatting discrepancies. For example, a technologist inadvertently added extra white space on the same line where another developer added new code. Also, people have different coding styles.

### **Make small commits and frequently review pull requests**

I'm going to admit something super embarrassing. Back in the day (~3 years ago), I would make pull requests changing over 50 files, and then I would get annoyed that I had so many merge conflicts. In hindsight, I was wrong. Changing over 50 files in less than two weeks increased the chances that other developers also made updates to those files.

1. **Rebasing:**

The git rebase command reapplies changes from one branch into another, which is very similar to the git merge command. However, in this case, git rebase rewrites the commit history to produce a straight, linear succession of commits.

### **Pay attention and communicate**

No git command or software tool can replace the need for communication in engineering teams. Being a good software developer and collaborator goes beyond writing code. Good software developers communicate with teammates. Keep your team aware of what files you will be touching and coordinate with your Product Manager and SCRUM Master to avoid working on features that conflict with other features.