## First-Time Git Setup

You can view all of your settings and where they are coming from using:

```
git config --list --show-origin
```

### Your Identity

Set your user name and email address. This is important because every Git commit uses this information.

```
git config --global user.name "Mohamed Yehia"
```

```
git config --global user.email "mohamed.yehia.work@gmail.com"
```

### Your default branch name

By default Git will create a branch called **master** when you create a new repository with git init.
you can set a different name for the initial branch

```
git config --global init.defaultBranch main
```

### Generating SSH Keys (Private & Public)

We will generate a **private key** and a **public key**:

- The **private key** stays on your machine.
- The **public key** will be added to your GitHub account.

Why use SSH keys instead of a password?

1. GitHub **no longer supports password authentication**. Now you must either:
   - Use a **Personal Access Token** (if you're using HTTP)
   - Or use **SSH keys** (if you're using SSH)
2. SSH Keys are:
   - **More secure**
   - **Easier to use** after setup (no need to enter password every time)

**How to generate SSH keys?**

**1. Generate SSH keys (only once per machine):**

```
ssh-keygen -t ed25519 -C "mohamed.yehia.work@gmail.com"
```

After generation, your keys will be stored in:

- **Private key**: ~/.ssh/id_ed25519
- **Public key**: ~/.ssh/id_ed25519.pub

**2. Add the public key to GitHub:**

1. Go to your GitHub account and navigate to: **Settings → SSH and GPG Keys → New SSH key.**

or open https://github.com/settings/keys

2. Copy the contents of the public key file:

```
cat ~/.ssh/id_ed25519.pub
```

3. Check if a key already exists under "SSH keys" (you'll see the name and creation date).

If there's no key, you need to **add it manually**

**3. Test your connection:**

```
ssh -T git@github.com
```

If you see the message like:

"Hi, Mohamed Yehia! You've successfully authenticated..."
Then you're all set!

## Git Basics

### Getting a Git Repository

You typically obtain a Git repository in one of two ways:

1. You can take a local directory that is currently not under version control, and turn it into a Git repository, or

2. You can clone an existing Git repository from elsewhere.

In either case, you end up with a Git repository on your local machine, ready for work

### Initializing a Repository in an Existing Directory

If you have a project directory that is currently not under version control and you want to start controlling it with Git, you first need to go to that project's directory.

cd C:/Users/user/my_project

git init

This creates a new subdirectory named .git that contains all of your necessary repository files — a Git repository skeleton. At this point, nothing in your project is tracked yet. See Git Internals for 26 more information about exactly what files are contained in the .git directory you just created.

To add a remote connection:

git remote add origin https://github.com/Mo-yehia/gitBasics.git

nots:

To check if the remote connection was added correctly:

git remote -v

To remove the remote connection:

git remote remove origin

### Cloning an Existing Repository

git clone https://github.com/Mo-yehia/gitBasics

**Daily Workflow After Connecting the Repository (Git Workflow):**

Stage the files:

git add .

Commit the changes:

git commit -m "Describe what you changed"

Push the changes to GitHub:

git push origin main

**note!**

If it's a new project and you're pushing for the **first time**, use: git push -u origin main

(This links the current local branch to origin/main so future pushes can be done with just git push)

Show the status of files in the project:

git status

(Shows which files have changed and which ones still need to be added to staging.)

View commit history:

git log

Restore the last version of a modified file (before committing):

git checkout -- filename.txt

Unstage a file (remove it from staging without deleting changes):

git reset filename.txt

Discard all changes before committing (resets project to the last commit and deletes all changes made after it):

git reset --hard

Create a new branch:

git branch new-branch-name

Switch to an existing branch:

git checkout branch-name

Create a new branch and switch to it immediately:

git checkout -b new-branch

Merge a branch into the current branch:

git merge branch-name

**Working with Files and Folders**

To create a file:

touch filename.txt

To create a folder:

mkdir foldername

To delete a file:

git rm filename.txt

To delete a folder:

git rm -r foldername