SSH Keys and GitHub - Guidance

***Generated by Claude, vetted by Hale**

2/4/2026

Generate the SSH Key

Open a terminal and run:

bash

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

Replace `your_email@example.com` with your actual email (ideally the one associated with your GitHub account). If your system doesn't support ed25519, use RSA instead:

bash

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

When prompted:

- Accept the default file location by pressing Enter (usually `~/.ssh/id_ed25519`)
- Enter a passphrase for added security, or press Enter for no passphrase

Start the SSH Agent and Add Your Key

bash

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
```

Copy Your Public Key

Display and copy your public key:

bash

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

Select and copy the entire output (it starts with `ssh-ed25519` and ends with your email).

Add the Key to GitHub

1. Go to GitHub.com and sign in
2. Click your profile photo (top right) → **Settings**
3. In the left sidebar, click **SSH and GPG keys**
4. Click **New SSH key** or **Add SSH key**
5. Give it a descriptive title (e.g., "Ubuntu Laptop")
6. Paste your public key into the "Key" field
7. Click **Add SSH key**

Test the Connection

bash

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

You should see a message like: "Hi username! You've successfully authenticated..."

GitHub Classroom

Good news: GitHub Classroom uses your GitHub account's SSH keys automatically. Once you've added your SSH key to GitHub, it works for GitHub Classroom assignments too. When you accept an assignment and clone the repository using SSH (not HTTPS), it'll use the key you just added.

Using SSH URLs

When cloning repositories, use the SSH URL format:

bash

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

Instead of HTTPS format (`https://github.com/username/repository.git`).

If you have existing repositories cloned via HTTPS, you can switch them to SSH:

bash

```
git remote set-url origin git@github.com:username/repository.git
```

That should have you set up!

The Complete Workflow

Here's the typical sequence:

bash

```
# 1. Stage your changes
git add .                    # stages all modified files
# or
git add filename.txt         # stages specific file(s)

# 2. Commit locally
git commit -m "Your commit message describing the changes"
```

It may require you to configure your email and name before accepting a commit, in which case, it is:

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"

# 3. Push to GitHub
git push origin main
```

First Time Pushing a New Repository?

If this is a brand new local repository that you haven't connected to GitHub yet, you'll need to:

1. Create the repository on GitHub first (without initializing with README)
2. Connect your local repo to GitHub:

bash

```
git remote add origin git@github.com:username/repository.git
  git branch -M main
  git push -u origin main
```

The `-u` flag sets up tracking, so future pushes just need `git push`.

Common Scenarios

If someone else pushed changes to GitHub since your last pull:

bash

```
git pull origin main     # fetch and merge remote changes first
git push origin main     # then push your changes
```

Check what branch you're on:

bash

```
git branch
```

See the status before pushing:

bash

```
git status
```

That's it! The `git push` command sends your local commits to GitHub.