Mastering Git and GitHub

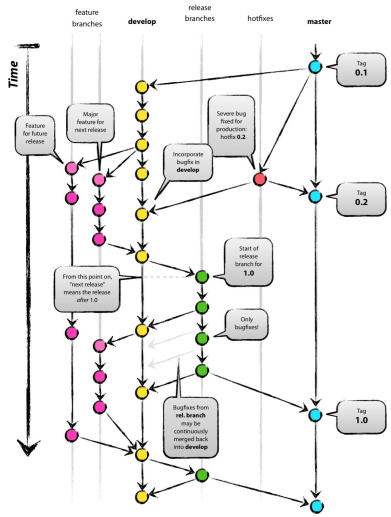
Version Control and Collaborative Development

What is Git?

- **Definition**: Git is a distributed version control system for tracking changes in source code.
- **Purpose**: Facilitates collaboration and maintains a history of code changes.

Why Use Git?

- **Collaboration**: Enables multiple developers to work simultaneously.
- **History**: Provides a detailed history of project changes.
- **Branching**: Supports branching and merging for feature development.



Introduction to GitHub

- **Definition**: GitHub is a web-based platform for hosting Git repositories.
- **Role**: Enhances Git with features like issue tracking, code review, and project management.

Git vs. GitHub

	Git	GitHub
Type	Version Control System	Hosting Service for Git Repos
Purpose	Track code changes	Collaboration and sharing

Setting Up Git

- **Installation**:
  - Download Git for your operating system.
- **Configuration**:

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

- **Verify Configuration**:

```
git config --list
```

Initializing a Repository

- **Command**: `git init`
- **Explanation**: Initializes a new Git repository in the current directory.
- **Example**:

```
cd my-project
git init
```

Checking Repository Status

- **Command**: `git status`
- **Explanation**: Displays the state of the working directory and staging area.
- **Example**:

```
git status
```

Adding Changes

- **Command**: `git add [file]` or `git add .`
- **Explanation**: Adds changes to the staging area.
- **Example**:

```
git add index.html
# or
git add .
```

Committing Changes

- **Command**: `git commit -m "commit message"`
- **Explanation**: Records staged changes to the repository.
- **Example**:

```
git commit -m "Add homepage"
```

Viewing Commit History

- **Command**: `git log`
- **Explanation**: Shows a list of all commits.
- **Example**:

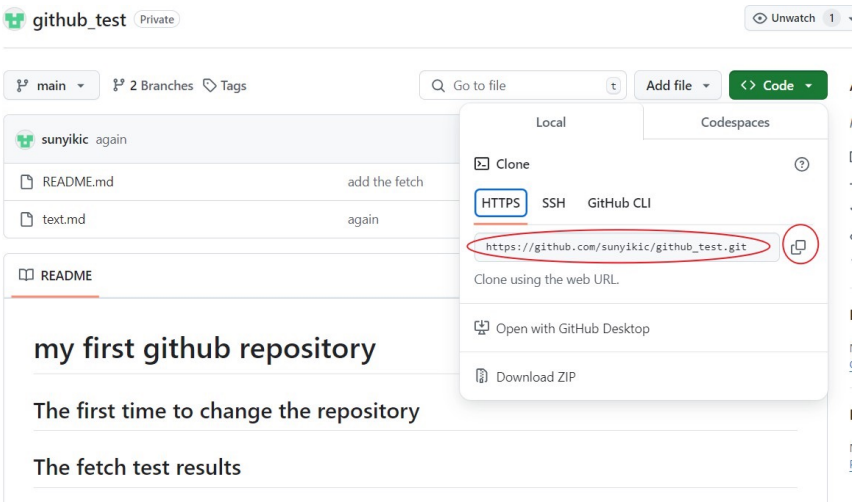
```
git log
```

- **Tip**: Write clear and descriptive commit messages.

Cloning a Repository from GitHub

- **Command**: `git clone [repository URL]`
- **Explanation**: Creates a local copy of a remote repository.
- **Example**:

```
git clone https://github.com/user/repo.git
```



Pushing Changes to Remote

- **Command**: `git push origin [branch]`
- **Explanation**: Uploads local commits to a remote repository.
- **Example**:

```
git push origin main
```

Pulling Updates from Remote

- **Command**: `git pull`
- **Explanation**: Fetches and merges changes from the remote repository.
- **Example**:

```
git pull
```

Branching in Git

- **Work on different parts of a project simultaneously.**
- **Creating and Switching Branches**
- **Create a Branch:**

```
git branch feature-login
```

- **Switch to a Branch:**

```
git checkout feature-login
```

- **Combined Command:**

```
git checkout -b feature-login
```

Merging Branches

- **Command**: `git merge [branch-name]`
- **Explanation**: Combines changes from one branch into another.
- **Example**:

```
git checkout main
git merge feature-login
```

Resolving Merge Conflicts

- **When Conflicts Occur:**
  - **Git can't automatically merge changes.**
- **Steps to Resolve:**
  1. Open conflicting files.
  2. Manually edit to fix conflicts.
  3. Stage and commit resolved files.

Stashing Changes

- **Command**: `git stash`
- **Explanation**: Temporarily saves changes not ready to commit.
- **Example**:

```
git stash
git checkout main
```

- **To Apply Stashed Changes**:

```
git stash apply
```

Reverting and Resetting

- **Revert a Commit:**

```
git revert [commit]
```

- **Reset to a Commit:**

```
git reset [commit]
```

- **Explanation**:
  - **Revert:** Undoes changes by creating a new commit.
  - **Reset:** Moves branch pointer to a specific commit.

Git Tags

- **Command:**

```
git tag -a v1.0 -m "Version 1.0"
```

- **Explanation**: Marks specific points in history as important.
- **Example:**

```
git tag -a v1.0 -m "Release version 1.0"
git push origin v1.0
```

Best Practices

- **Commit Often**: Keep commits small and logical.
- **Clear Messages**: Write descriptive commit messages.
- **Use Branches**: Separate features and fixes.
- **Regular Sync**: Pull and push changes frequently.

Practice 1: Local to GitHub

1. Create Local Repository

```
mkdir rwanda-cuisine
cd rwanda-cuisine
git init
```

2. Create and Edit Files

```
touch index.html styles.css
```

Practice 1: File Contents

index.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>Rwanda Cuisine</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Traditional Rwanda Dishes</h1>
    <div class="dish">
        <h2>Ugali</h2>
        <p>A traditional cornmeal dish served with stews.</p>
    </div>
</body>
</html>
```

Practice 1: CSS File

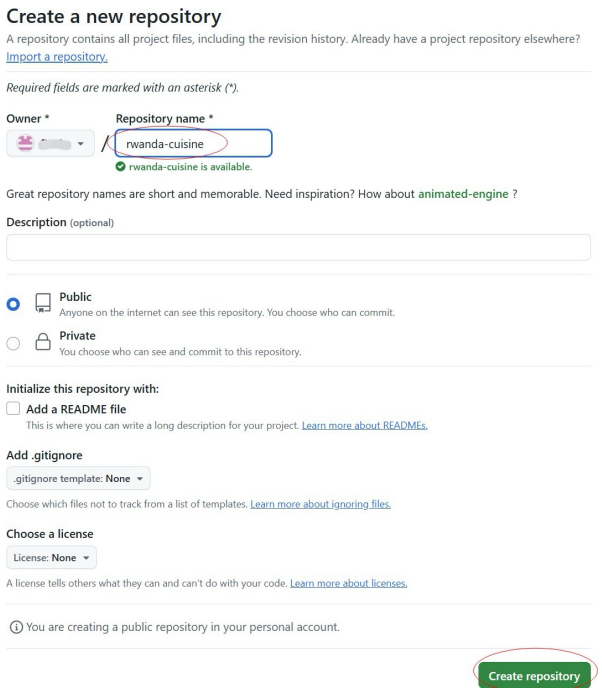
styles.css:

```css
body {
    font-family: Arial, sans-serif;
    margin: 20px;
    background-color: #f0f0f0;
}

.dish {
    padding: 15px;
    background-color: white;
    border-radius: 5px;
}
```

Practice 1: Create a new repository in GitHub

- At the repository dashboard screen click the `add` button.
- setup the repository name as `rwanda-cuisine` and click the `Create repository` button



Practice 1: Push to GitHub

```
git add .
git commit -m "Initial commit: Rwanda cuisine website"
git remote add origin https://github.com/your-username/rwanda-cuisine.git
git push -u origin master
```

Add the SSH key and adding it to the ssh-agent

```
# generate the SSH key
ssh-keygen -t ed25519 -C "your email@example.com"

# Start the ssh-agent in the background.
eval "$(ssh-agent -s)"

# Add your SSH private key to the ssh-agent. (id_ed25519 is the name of your private key file.)
ssh-add ~/.ssh/id_ed25519
```

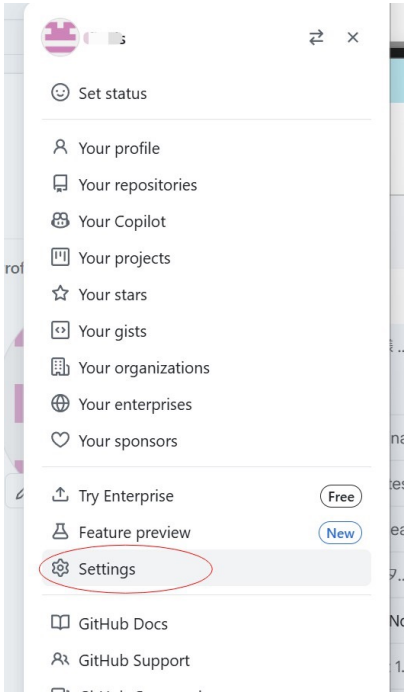
Reference

Adding a new SSH key to your GitHub account

- Copy the SSH public key to your clipboard.

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

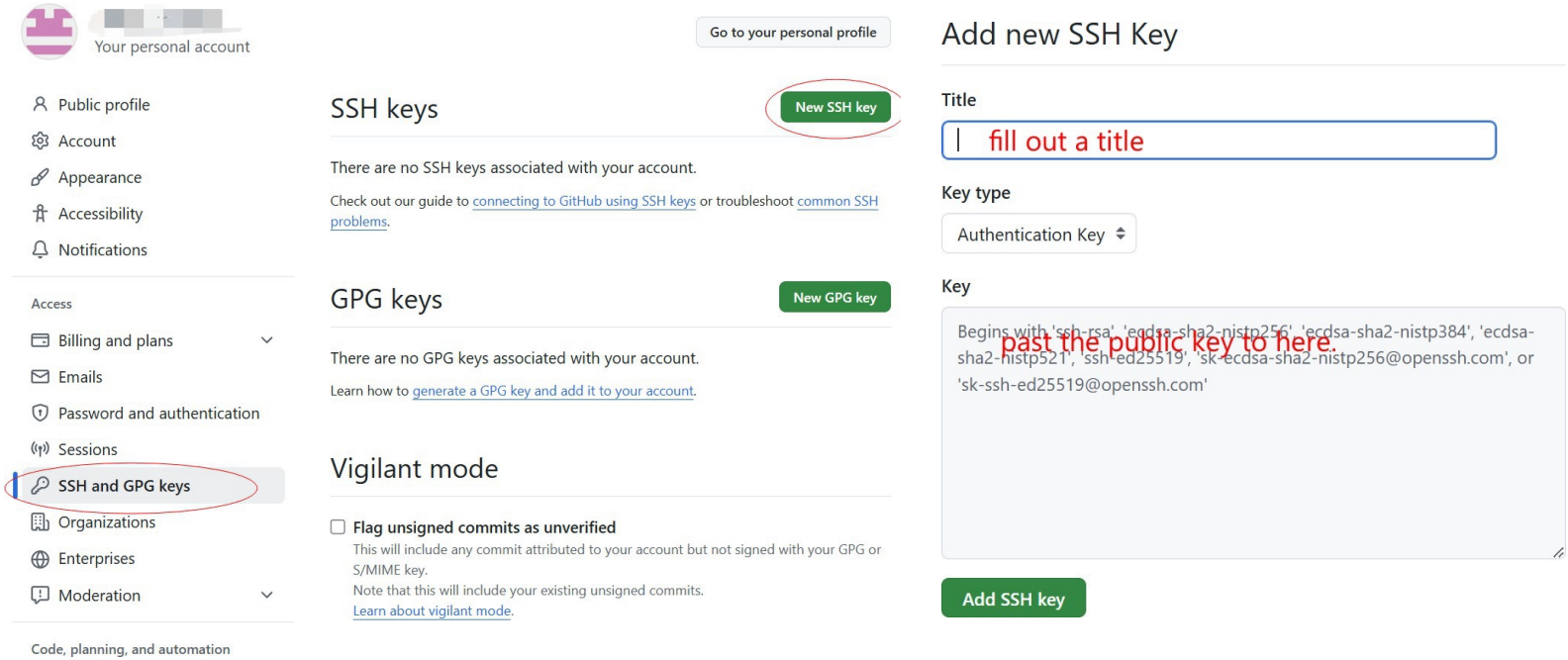
- In the upper-right corner of any page on GitHub, click your profile photo, then click Settings.



Setup the key to the GitHub account

- In the "Access" section of the sidebar, click SSH and GPG keys.
- Click New SSH key or Add SSH key.
- In the "Title" field, add a descriptive label for the new key. For example, if you're using a personal laptop, you might call this key "Personal laptop".
- In the "Key" field, paste your public key.
- Click Add SSH key.

Reference


Your personal account
Go to your personal profile
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and plans
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Enterprises
Moderation
Code, planning, and automation
SSH keys
New SSH key
There are no SSH keys associated with your account.
Check out our guide to connecting to GitHub using SSH keys or troubleshoot common SSH problems.
GPG keys
New GPG key
There are no GPG keys associated with your account.
Learn how to generate a GPG key and add it to your account.
Vigilant mode
Flag unsigned commits as unverified
This will include any commit attributed to your account but not signed with your GPG or S/MIME key.
Note that this will include your existing unsigned commits.
Learn about vigilant mode.
Add new SSH Key
Title
fill out a title
Key type
Authentication Key
Key
Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'
past the public key to here.
Add SSH key

Practice 2: GitHub to Local

1. Create new repository "rwanda-cuisine2" on GitHub
2. Create the same files on GitHub
3. Clone to local

```
git clone git@github.com:firrds/rwanda-cuisine2.git
cd rwanda-cuisine2
```

Practice 2: Branch Management

1. Create and switch to new branch

```
git checkout -b new-colors
```

2. Modify styles.css

```
body {
    font-family: Arial, sans-serif;
    margin: 20px;
    background-color: #fff5e6;  /* Warmer background */
}

.dish {
    padding: 15px;
    background-color: #ffe6cc;  /* Warmer card background */
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}
```

Practice 2: Merge and Push

```
git add styles.css
git commit -m "Update color scheme"
git checkout main
git merge new-colors
git push origin main
```

Key Points to Remember

- Always check status with `git status`
- Write clear commit messages
- Create meaningful branch names
- Test changes before merging
- Keep local repository updated with `git pull`

Practice Complete!

You've learned:

- Creating local and remote repositories
- Basic file management with Git
- Branch creation and management
- Merging changes
- Pushing to GitHub

For more git command, please visit the Git cheat sheet

https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet