Creating a New Git Repository for Button Code Using Termux on Mobile Devices

A Beginner■Friendly, Step■by■Step Guide and Research Notes

Abstract

This document explains, in beginner-friendly detail, how to create and manage a Git repository for a project named 'buttonscode' using Termux on an Android device. It covers necessary prerequisites, step-by-step commands, common errors, their solutions, and recommendations for fully automating a mobile development workflow. The goal is to enable learners — including absolute beginners — to understand each command, why it's used, and how to troubleshoot issues encountered when working with Git and GitHub from a mobile terminal.

1. Introduction

Developers commonly use desktop environments like Windows, macOS, or Linux for version control with Git. This guide demonstrates how you can perform equivalent operations on your Android phone using Termux — a powerful terminal emulator that brings many Linux tools to mobile devices. We'll use a real example: creating and pushing a repository named 'buttonscode' to GitHub.

2. Prerequisites

Before following the steps below, ensure you have the following:
• An Android device with Termux installed.
• A GitHub account (create one at https://github.com if you do not have it).
• Basic familiarity with the command line (we'll explain each command).
• An internet connection to push/pull from GitHub.

3. Step-by-step Walkthrough

3.1. Open Termux and locate your project folder

Start Termux and navigate to the directory containing your project files. On many Android devices, shared storage is mounted under /storage/emulated/0. Example commands (type these into Termux):

```
Commands:
ls storage
cd /storage/emulated/0/Documents/buttonscode
```

Explanation: 'ls storage' shows mounted storage locations. 'cd' changes your current directory to the folder that holds your project.

3.2. Initialize Git in the project folder

```
Command:
git init
```

Explanation: Initializes a new local Git repository. This creates a hidden .git folder that tracks your project history.

3.3. Rename the default branch to 'main'

```
Command:
git branch -M main
```

Explanation: Older versions of Git use 'master' as the default branch name. This command renames the branch to 'main', which is the modern default on GitHub.

3.4. Fixing 'detected dubious ownership' warning

Problem: When working with files on external storage, Git may show a warning similar to: 'detected dubious ownership in repository at ...'. This happens because Git is cautious when the repository owner doesn't match the user running Git.

```
Solution (example):
git config --global --add safe.directory /storage/emulated/0/Documents/buttonscode
```

Explanation: This tells Git to trust that directory even if ownership looks unusual. Use this only for directories you control.

3.5. Add and commit files

```
Commands:
git add .
git commit -m "Initial commit"
```

Explanation: 'git add .' stages all files in the current directory. 'git commit' saves a snapshot of those files with a message.

3.6. Connect to GitHub (remote origin)

```
Common commands:
git remote add origin https://github.com//.git
git remote set-url origin https://github.com/Apolizy/buttonscode.git
```

Explanation: 'git remote add origin' creates a shortcut named 'origin' pointing to the GitHub repository. If you already have 'origin' set and need to correct the URL, use 'git remote set-url' to update it.

3.7. Verify remote URL

```
Command:
git remote -v
```

Explanation: Shows the remote URLs configured for fetching and pushing. Example output:

```
origin https://github.com/Apolizy/buttonscode.git (fetch) origin
https://github.com/Apolizy/buttonscode.git (push)
```

3.8. Push to GitHub

```
Command:
git push -u origin main
```

Explanation: Pushes the local 'main' branch to the remote 'origin' and sets upstream tracking so future 'git push' and 'git pull' work without extra arguments.

4. Troubleshooting and Common Issues

4.1. Repository not found / can't push because repo doesn't exist

Symptom: 'git push' fails with an error indicating the remote repository doesn't exist. Reason: You haven't created the repository on GitHub yet. Options to fix this:

Option A — Create repository via browser (simple):

Go to GitHub in your browser, click 'New repository', set the name (e.g., 'buttonscode'), and create it. If you create a README.md on GitHub when creating the repo, you might need to pull before pushing.

Option B — Create repository using GitHub CLI (recommended for automation):

```
Command (after installing gh CLI and authenticating):
gh repo create Apolizy/buttonscode --public --source=. --remote=origin
```
Explanation: 'gh' is GitHub's official CLI. It can create a remote repository directly from Termux after you authenticate, avoiding the browser.

4.2. Authentication prompts and personal access tokens

When pushing over HTTPS, GitHub asks for credentials. Since password authentication was removed, you must use one of the following:
• Use a Personal Access Token (PAT) instead of your password. Generate a PAT in GitHub settings (give repo scope). When prompted for password, paste the PAT.
• Use SSH keys: create an SSH key pair on your phone and add the public key to your GitHub account. Then use the SSH remote URL (git@github.com:username/repo.git).
• Use GitHub CLI (gh) and authenticate once; it will handle tokens for you.

4.3. Merge conflicts due to README.md or files created on GitHub

Symptom: Push fails because the remote has commits you don't have locally (common when you created README.md on GitHub). Fix by pulling the remote changes first and rebasing or merging:
```
Commands (safe approach):
git pull --rebase origin main
git push -u origin main
```
Explanation: 'git pull --rebase' applies your local commits on top of the updated remote branch for a cleaner history. Avoid 'git push --force' unless you understand the consequences.

4.4. 'detected dubious ownership' explanation (revisited)

Git warns when the filesystem owner doesn't match the user because it could be a security risk. On Android, shared storage may have different ownership. Adding the directory to safe.directory tells Git you trust it.

4.5. When to use 'git push --force' (and when not to):

Force-pushing overwrites the remote branch history. Only use this if you intentionally want to replace the remote contents with your local copy (e.g., during a destructive rewrite). For normal pushes, avoid using --force.

5. Recommendations for Beginners

• Start with HTTPS + PAT if SSH feels hard; it's easier to set up initially.
• Learn to use 'git status' frequently to know what's staged, modified, or untracked.
• Use 'git log --oneline' to see your commit history in a compact form.
• Consider installing GitHub CLI (gh) inside Termux for streamlined repository creation and authentication.
• Back up important work before experimenting with 'git push --force'.

Appendix A — Sample Terminal Transcript and Outputs

```
$ ls storage emulated self $ cd /storage/emulated/0/Documents/buttonscode $ git init
Initialized empty Git repository in /storage/emulated/0/Documents/buttonscode/.git/
$ git branch -M main $ git config --global --add safe.directory
/storage/emulated/0/Documents/buttonscode $ git remote add origin
https://github.com/Apolizy/buttonscode.git fatal: remote origin already exists. $
git remote set-url origin https://github.com/Apolizy/buttonscode.git $ git remote -v
```

```
origin https://github.com/Apolizy/buttonscode.git (fetch) origin
https://github.com/Apolizy/buttonscode.git (push) $ git add . $ git commit -m
"Initial commit" [main (root-commit) abc1234] Initial commit 10 files changed,
created 10 files $ git push -u origin main Username for 'https://github.com':
Apolizy Password for 'https://Apolizy@github.com': Enumerating objects: 15, done.
Counting objects: 100% (15/15), done. Compressing objects: 100% (10/10), done.
Writing objects: 100% (15/15), done. To https://github.com/Apolizy/buttonscode.git *
[new branch] main -> main
```

6. Conclusion and Next Steps

This guide showed how to initialize a repository in Termux, handle ownership warnings, set and verify remotes, and push to GitHub. If you want a fully mobile workflow without opening the browser, the next steps are: configure SSH authentication, or install and authenticate the GitHub CLI (gh) in Termux. Once you are comfortable with these commands, you can record tutorial videos explaining the steps to your audience.

References and Resources

- Git documentation — https://git-scm.com/docs
- GitHub Docs — https://docs.github.com
- Termux project — https://termux.com
- GitHub CLI — https://cli.github.com/