GitButler CLI Cheat Sheet

Essential commands for working with GitButler

GitButler is a Git client that allows you to work on multiple branches simultaneously. This cheat sheet features the most important and commonly used GitButler CLI commands for easy reference.

Setup and Teardown

Switching to and from GitButler branch management

```
but setup
```

Switch to GitButler in the current repository

```
but teardown
```

Go back to vanilla git branch management

Branch Management

Create and manage parallel and stacked branches

```
but branch
```

List all branches available

```
but branch new [name]
```

Create a new parallel branch

```
but branch new --anchor [branch]
```

Create a new stacked branch from an existing branch

```
but reword [branch]
```

Rename a branch

```
but branch apply [branch]
```

Apply (enable) a branch to your working directory

```
but branch unapply [branch]
```

Unapply (disable) a branch from your working directory

Inspection

View your workspace state and changes

```
but status
```

Show workspace state, applied branches, staged files, and commits

```
but status -f -v
```

Show status with committed files listed and more verbose output

```
but status -u
```

Show detailed list of unintegrated upstream commits

```
but diff
```

Show changes in working directory compared to last commit

```
but diff [branch]
```

Show changes specific to a branch

```
but show [commit|branch]
```

Show details of a specific commit or branch

GitButler CLI Cheat Sheet

```
but branch list [search]
```

List all branches whose names match the search term

```
but branch delete [branch]
```

Delete a branch from the workspace

Remote Operations

Push changes and interact with forges

```
but push
```

Push all branches with unpushed commits

```
but push [branch]
```

Push a specific branch to remote

```
but pull
```

Pull latest upstream and integrate all active branches

```
but pull --check
```

See what a pull would do without making changes

```
but pr
```

Create/update pull requests for active branches

```
but config forge
```

View and modify forge authentications

Operations Log

View and restore workspace history

```
but undo
```

Undo the last operation

Committing Changes

Create commits on your branches

```
but commit -m "[message]"
```

Commit all changes to current branch with a message

```
but commit [branch] -m "[message]"
```

Commit changes to a branch if there is more than one applied

```
but stage [file] [branch]
```

Stage specific file(s) for the next commit

```
but commit --only [branch]
```

Commit only changes already staged to the branch

```
but commit -c [branch]
```

Create a new branch and commit to it

Commit Editing

Modify existing commits

```
but reword [commit]
```

Edit the commit message of a specified commit

```
but commit new --before [commit]
```

Insert a blank commit before the specified commit

```
but pick [commit]
```

Cherry-pick a commit into an applied branch

```
but rub [target] [source]
```

Combine two commits or branches together

GitButler CLI Cheat Sheet

```
but oplog
```

Show operation history of workspace

```
but oplog restore [snapshot-id]
```

Restore workspace to a specific snapshot

```
but oplog snapshot -m "[message]"
```

Create an on-demand snapshot with message

Helper Commands

Other miscellaneous commands including automatic file assignment or committing

```
but gui
```

Open the GUI to the current project

```
but alias
```

Create and manage aliases for commonly used commands

```
but config
```

View and modify GitButler configuration settings

```
but update
```

Check for and install updates to GitButler

```
but skill
```

Get tips and tricks for using GitButler effectively

Conflict Management

Resolve conflicts and manage merges

```
but resolve
```

Go into resolution mode

```
but merge [branch]
```

Merge an active branch into your upstream branch (local mode only)

Auto-Assignment

Configure automatic file assignment or committing

```
but mark [branch|commit]
```

Auto-assign changes to branch or auto-commit to commit

```
but unmark
```

Remove all marks from workspace