# Git

Patrick Bucher

## Contents



## Git Basics

### Creating a Repository

Create a new Git repository from an existing working directory (move into the working directory first):

```
git init
```

Cloning an existing Git repository from GitHub:

```
git clone https://github.com/[username]/[repository].git
```

## Making Changes

Display changes in the working directory to be staged:

```
git add --dry-run .
git add -n .
```

Add a file to the staging area:

```
git add [file]
```

Commit changes to a repository (with a message):

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

Add and commit changes at the same time:

```
git commit --all
git commit -a
```

### Writing Proper Commit Log Messages

From the discussion section of `git log commit`:

```
"Though not required, it's a good idea to begin the commit
message with a single short (less than 50-character) line
summarizing the change, followed by a blank line and then a
more thorough description."
```

Example:

```
Adding printf.

This is to make the output a little more human readable.

printf is part of BASH, and it works just like C's printf()
function.
```

The first line shows *what* has been done, the second line shows *why* it has been done. The third line gives additional (technical) *details*. `git log --oneline` only shows the first line of the commit message (*what*).

The most recent commit message can be improved:

```
git commit --amend
```

## Getting Information

Show a repository's state:

```
git status
```

List a repository's—not the working directory's!—files:

```
git ls-files
```

Show revisions of a file:

```
git blame [file]
```

### Git's Log

Show the repository's commit history (also in one line, with statistics and a combination of those with short statistics):

```
git log
git log --oneline
git log --stat
git log --shortstat --oneline
```

With abbreviated SHA1 IDs (only the first eight characters):

```
git log --abbrev-commit
```

With every commit's parent commit (also with abbreviated SHA1 IDs):

```
git log --parents
```

Show the log in patch and statistics view (and combined):

```
git log --patch
git log --stat
git log --patch-with-stat
```

Show the commit history of a certain file:

```
git log [filename]
```

### Showing Differences

Show changes between files in working directory and the repository—or the staging area, if changes have been staged already:

```
git diff
```

Show changes between files in staging area and in the repository:

```
git diff --staged
git diff --cached
```

### The Staging Area

These commands not only make changes to the working directory, but to the staging area at the same time.

Remove a file from the staging area:

```
git rm [file]
```

Rename a file in the staging area:

```
git move [file]
```

Stage parts of a file:

```
git add -p
```

Undo staging area changes for a file:

```
git reset [file]
```

Check out a file (replace file in the working directory with the version of its latest commit):

```
git checkout -- [file]
```

## Configuration (`git config`)

Set global configuration (name and email):

```
git config --global [option] [value]
```

```
git config --global user.name "Patrick Bucher"
git config --global user.email "patrick.bucher@stud.hslu.ch"
```

Show all configuration:

```
git config --list
```

Show a specific configuration item (name and email):

```
git config [option]
```

```
git config user.name
git config user.email
```

## Help (`git help`)

Show the help page (most important commands):

```
git help
```

Show all commands (with pager):

```
git -p help -a
```

Show all available guides:

```
git help -g
```

Getting help on a specific command or read a guide (help itself, the glossary and the tutorial guide):

```
git help [command/subject]

git help help
git help glossary
git help tutorial
```

## Miscellaneous

Starting Git GUI (the package `tk` is required under Linux):

```
git gui
```

Starting Git GUI to commit changes (`citool`):

```
git citool
```

Starting the Git log viewer (`gitk`):

```
gitk
```

### Switches

Display the installed version of git:

```
git --version
```

Use a pager (usually `less`) for the output:

```
git -p [command]
git --paginate [command]
```