Git Tutorial

A Distributed Version-Control System

- Chapter 2 -

Git Basics

- Git Basics -

Getting a Git Repository

- There is 2 ways to obtain a Git repository :
  - Turn a local directory to a Git repository.
  - Clone an existing Git repository.

1. Go to your project directory:

- Linux:

```
$ cd /home/user/my_project
```

- MacOS:

```
$ cd /Users/user/my_project
```

- Windows:

```
$ cd C:/Users/user/my_project
```

2. Make it a Git repository:

```
$ git init
```

At this point, nothing in your project is tracked yet.
To start version-controlling existing files:

```
$ git add *.c
$ git add LICENSE
$ git commit -m 'Initial project version'
```

To get a copy of an existing Git repository:

- The command format is like this: `git clone <url>`

```
$ git clone https://github.com/libgit2/libgit2
```

- To clone it with another directory:
  `git clone <url> <new-directory>`

```
$ git clone https://github.com/libgit2/libgit2 mylibgit
```

Git's transfer protocols:

- HTTPS (the url is like this:)

```
https://github.com/ ...
```

- SSH (the url is like this:)

```
git@github.com: ...
```

To see pros and cons of each,
read "Getting Git on a Server" in page 111 of the Book.

- Git Basics -

Recording Changes to the Repository

Each file in your working directory can be in one of two states:

- Tracked
  - Unmodified
  - Modified
  - Staged
- Untracked

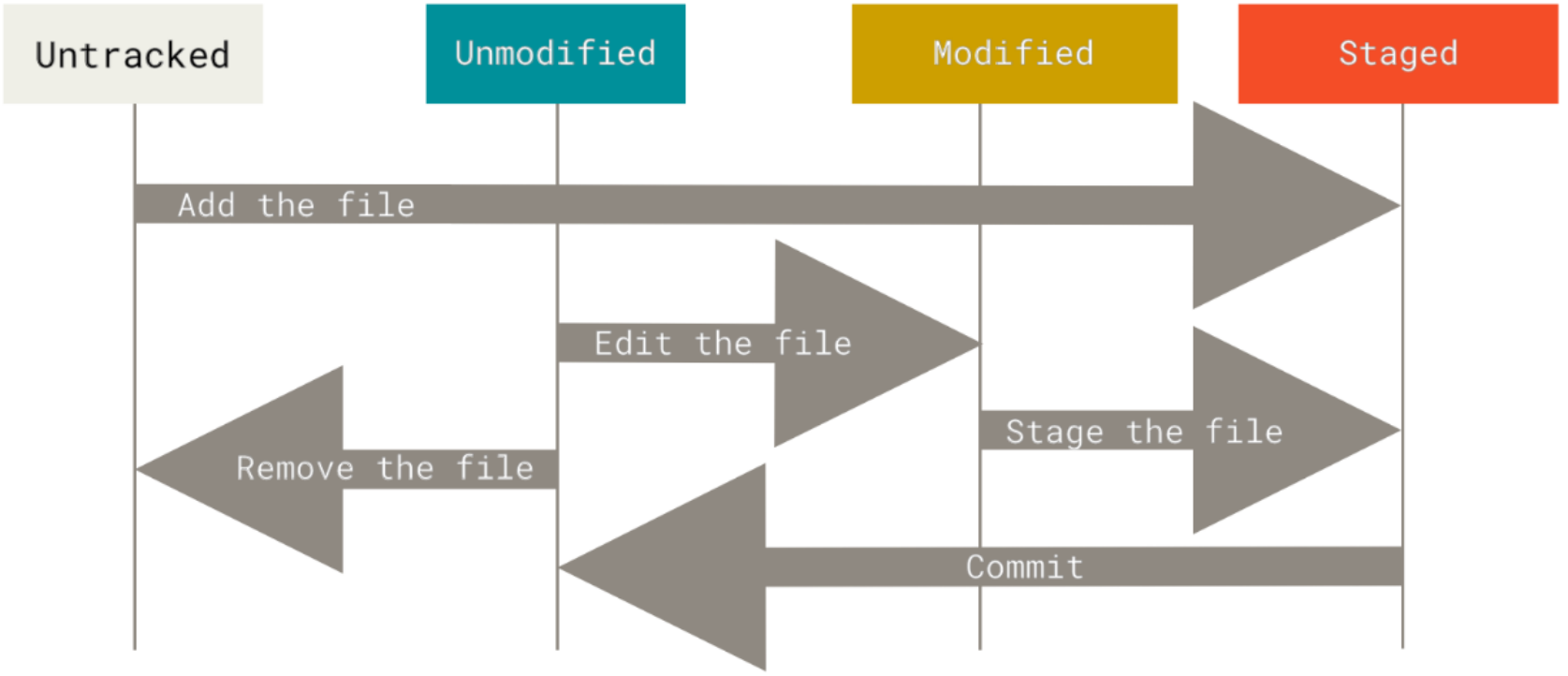


Figure 8. The lifecycle of the status of your files

Use `git status` to determine which files are in which state is the.

- Right after a clone:

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

Let's add a new file to your project, a simple `README` file:

```
$ echo 'My Project' > README
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README

nothing added to commit but untracked files present
(use "git add" to track)
```

Use `git add` to begin tracking the `README` file:

```
$ git add README
```

Now check the status:

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)

    new file:   README
```

Some sort of tracking files:

Considering this form: `git add <files>`

- Use a path to add a file from subdirectories:

```
$ git add code/v2/file.py
```

- Add multiple file at once (you also can add multiple path):

```
$ git add file_1.py file_2.py file_3.py
```

Let's change a file that was already tracked:

```
$ echo 'My Project' > README
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CONTRIBUTING.md
```

`git add` is a multipurpose command. Now we use it to stage a file:

```
$ git add CONTRIBUTING.md
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README
    modified:   CONTRIBUTING.md
```

Suppose you remember one little change that you want to make in `CONTRIBUTING.md` before you commit it.

So lets change the `CONTRIBUTING.md`:

```
$ vim CONTRIBUTING.md
```

Then run `git status` one more time:

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README
    modified:   CONTRIBUTING.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CONTRIBUTING.md
```

What happend?

The `CONTRIBUTING.md` is listed as both staged and unstaged.

So if you modify a file after you run `git add`,
you have to run `git add` again to stage the latest version of the file:

To get a simplified status, use one of the following:

- `git status -s`
- `git status --short`

```
$ git status -s
 M README
MM Rakefile
A  lib/git.rb
M  lib/simplegit.rb
?? LICENSE.txt
```

- `M` → Modified
- `MM` → Was modified, staged and then modified again
- `A` → New staged file
- `M` → Modified and staged
- `??` → New untracked file

If you have a class of files that you want Git to ignore them, create a `.gitignore` file and add the patterns to it:

```
$ cat .gitignore
*.[oa]
*~
```

- First line: ignore any files ending in ".o" or ".a".
- Second line: ignore all files whose names end with a tilde (~).

The rules of these patterns are given on page 31 of the book.

Here is another example `.gitignore` file:

```
# ignore all .a files
*.a

# but do track lib.a, even though you're ignoring .a files above
!lib.a

# only ignore the TODO file in the current directory, not subdir/TODO
/TODO

# ignore all files in any directory named build
Build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

# ignore all .pdf files in the doc/ directory and any of its subdirectories
doc/**/*.pdf
```

If you want to know exactly what you changed, not just which files were changed. You'll probably use it most often to answer these two questions:

- What have you changed but not yet staged?
- What have you staged that you are about to commit?

Assume that we edit and stage `README` file and then edit the `CONTRIBUTING.md` file without staging it.

So `git status` shows this:

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   README

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CONTRIBUTING.md
```

To see what you've changed but not yet staged:

```
$ git diff
diff --git a/CONTRIBUTING.md b/CONTRIBUTING.md
index 8ebb991..643e24f 100644
--- a/CONTRIBUTING.md
+++ b/CONTRIBUTING.md
@@ -65,7 +65,8 @@ branch directly, things can get messy.
 Please include a nice description of your changes when you submit your PR;
 if we have to read the whole diff to figure out why you're contributing
 in the first place, you're less likely to get feedback and have your change
-merged in.
+merged in. Also, split your changes into comprehensive chunks if your patch is
+longer than a dozen lines.

 If you are starting to work on a particular area, feel free to submit a PR
 that highlights your work in progress (and note in the PR title that it's
```

To see what you've staged that will go into your next commit,

```
$ git diff –staged
diff --git a/README b/README
new file mode 100644
index 0000000..03902a1
--- /dev/null
+++ b/README
@@ -0,0 +1 @@
+My Project
```

For another example, if you stage the `CONTRIBUTING.md` file and then edit it, you can use `git diff` tosee the changes in the file that are staged and the changes that are unstaged.

```
$ git add CONTRIBUTING.md
$ echo '# test line' >> CONTRIBUTING.md
```

If our environment looks like this:

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   CONTRIBUTING.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   CONTRIBUTING.md
```

Now you can use `git diff` to see what is still unstaged

```
$ git diff
diff --git a/CONTRIBUTING.md b/CONTRIBUTING.md
index 643e24f..87f08c8 100644
--- a/CONTRIBUTING.md
+++ b/CONTRIBUTING.md
@@ -119,3 +119,4 @@ at the
## Starter Projects

 See our [projects
list](https://github.com/libgit2/libgit2/blob/development/PROJECTS.md).
+# test line
```

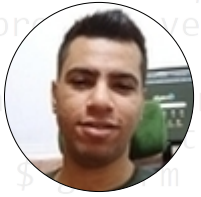
And `git diff –cached` to see what you've staged so far (`--staged` and `--cached` are synonyms):

```
$ git diff --cached
diff --git a/CONTRIBUTING.md b/CONTRIBUTING.md
index 8ebb991..643e24f 100644
--- a/CONTRIBUTING.md
+++ b/CONTRIBUTING.md
@@ -65,7 +65,8 @@ branch directly, things can get messy.
 Please include a nice description of your changes when you submit your PR;
 if we have to read the whole diff to figure out why you're contributing
 in the first place, you're less likely to get feedback and have your change
-merged in.
+merged in. Also, split your changes into comprehensive chunks if your patch is
+longer than a dozen lines.

 If you are starting to work on a particular area, feel free to submit a PR
 that highlights your work in progress (and note in the PR title that it's
```

...

Group Members



Mohammad A. S. Minabi
bigm00bnd@gmail.com

Mohammad H. Bahrampour
bahrampour@pm.me

Hamid R. K. Pishghadam
kaveh@riseup.net