# Oh-My-Git

Ashik Salman

Awesome Git tweaks

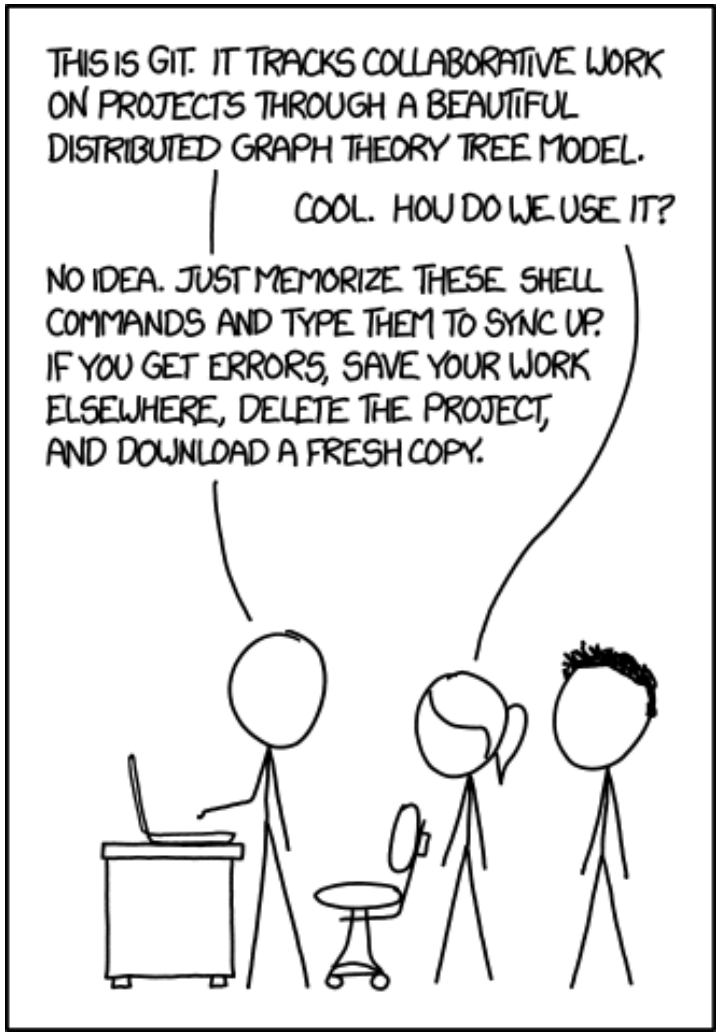
.....

Backend Developer
Chillr, Backwater Technologies
Kochi, Kerala

September 25, 2016

# oh-my-git, what?

## Git Tool


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

# Versioning by Patch changes

# Git with Patches

## Add changes by patches

```
$ git add -p
$ git add -i (interactive menu)
```

# Git with Patches

## Add changes by patches

```
$ git add -p
$ git add -i (interactive menu)
```

## Checkout changes by patches(Remove un-staged changes)

```
$ git checkout -p
```

# Git with Patches

## Add changes by patches

```
$ git add -p
$ git add -i (interactive menu)
```

## Checkout changes by patches(Remove un-staged changes)

```
$ git checkout -p
```

## Reset changes by patches(Remove staged changes)

```
$ git reset -p
```

# oh-my-git !!!

## Scenario 1

I did something terribly wrong, I want to got to a stage where everything worked fine.

# oh-my-git !!!

## Scenario 1

I did something terribly wrong, I want to got to a stage where everything worked fine.

## Why... ?

You can use this to get back stuff you accidentally deleted, or just to remove some stuff you tried that broke the repo, or to recover after a bad merge, or just to go back to a time when things actually worked.

# oh-my-git !!!

## Example (Solution)

```
$ git reflog
# you will see a list of every thing you've done in git, a
# each one has an index HEAD@{index}
# find the one before you broke everything
$ git reset HEAD@{index}
```

# oh-my-git !!!

## Example (Solution)

```
$ git reflog
# you will see a list of every thing you've done in git, a
# each one has an index HEAD@{index}
# find the one before you broke everything
$ git reset HEAD@{index}
```

## Note !

If the work was committed at any point, then it can be recovered from the reflog. By default all commits stay alive in the reflog for at least 2 weeks.

# oh-my-git !!!

## Scenario 2

I just committed some changes and immediately realized I need to make one small change.

# oh-my-git !!!

## Example (Solution)

```
# make your change
$ git add .  # or add individual files
$ git commit --amend
# follow prompts to change or keep the commit message
# now your last commit contains that change!
```

# oh-my-git !!!

## Example (Solution)

```
# make your change
$ git add .  # or add individual files
$ git commit --amend
# follow prompts to change or keep the commit message
# now your last commit contains that change!
```

## Note !

You could also make the change as a new commit and then do rebase -i in order to squash them both together, but this is about a million times faster.

# oh-my-git !!!

## Scenario 3

I accidentally committed something to master that should have been on a brand new branch!

## Example (Solution)

```
# create a new branch from the current state of master
$ git branch some-new-branch-name
# remove the commit from the master branch
$ git reset HEAD~ --hard
$ git checkout some-new-branch-name
# your commit lives in this branch now :)
```

## Example (Solution)

```
# create a new branch from the current state of master
$ git branch some-new-branch-name
# remove the commit from the master branch
$ git reset HEAD~ --hard
$ git checkout some-new-branch-name
# your commit lives in this branch now :)
```

## Note !

This doesn't work if you've already pushed to origin, and if you tried other things first, you might need to git reset HEAD@number instead of HEAD .

# Git Merge (Conflicts!!!)

# Git rebase - interactive (Conflicts !!!)

# Git Stash - Easy work save/restore

# Thank You

# Questions?