Oh-My-Git

Ashik Salman

Awesome Git tweaks

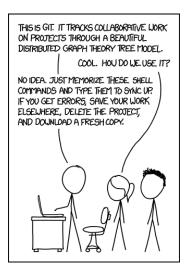
Backend Developer
Chillr, Backwater Technologies
Kochi, Kerala

September 25, 2016

oh-my-git, what?

Generic View

Git Tool



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

Scenario 1

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

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.

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}

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.

Scenario 2

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

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!
```

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.

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 .

Scenario 4

Ooops..! I accidentally committed to the wrong branch!

Example (Solution)

```
# undo the last commit, but leave the changes available
$ git reset HEAD~ --soft
```

- \$ git stash
- # move to the correct branch
- \$ git checkout name-of-the-correct-branch
- \$ git stash pop
- \$ git add . # or add individual files
- \$ git commit -m "your message here"
- # now your changes are on the correct branch

Note!

A lot of people have suggested using cherry-pick for this situation too, so take your pick on whatever one makes the most sense to you!

Note!

A lot of people have suggested using cherry-pick for this situation too, so take your pick on whatever one makes the most sense to you!

Example (Solution)

- \$ git checkout name-of-the-correct-branch
- # grab the last commit to master
- \$ git cherry-pick master
- # delete it from master
- \$ git checkout master
- \$ git reset HEAD~ --hard

Git Merge (Conflicts!!!)

Git rebase - interactive (Conflicts !!!)

Git Stash - Easy work save/restore

Thank You

Questions?

