Musa Al-hassy August 16, 2018

Reference Sheet for Git

Workflow ⟨

–Getting things done–

1. Create a branch, say off of Master.
2. Work :: Make branches, make commits, push and pull changes, review history.
3. Merge it back to the parent, e.g., Master, when done.

Getting started

Insist that our Git related actions will be stamped by our identifiying information, so that it's clear what we did.

```
git config --global user.name "⟨My Name⟩"
git config --global user.email ⟨myEmail@example.com⟩

# Add our GitHub username to our configuration, this is case sensative.
git config --global user.username ⟨userName⟩

# Open the global configuration file in a text editor for manual editing.
git config --global --edit
```

Branches

–Group changes by naming a series of commits and combining completed efforts–

When contributing, work on a branch: A copy of the original project, with its own history and its changes are isolated from the original, until it's merged back into the original branch –which is commonly called "master".

Do so to avoiding stepping on another contributor's efforts, or having them step on yours, and also an excellent idea for experimenting: When unsure what's best, experiment on separate branches then compare your approaches. –we can even branch off of any branch!

git branch ⟨branch⟩ Create a new branch which is a copy of the current branch.
- ◇ Omitting any name yields the branches available; `git branch -a` shows all branches including remote ones. The current branch is marked with an asterisk.

git checkout ⟨branch⟩ Switch to a given branch.
- ◇ Create and check out a new branch with `git checkout -b ⟨name⟩`.

git merge We apply the changes of the current branch to its parent branch.

git branch -d ⟨branch⟩ When a branch is no longer needed, and has been merged, we delete it.

git branch -m ⟨newBranchName⟩ Rename the branch we're currently on.

Making Changes

Review edits and craf a commit transaction

A commit represents the state of our repository at a given point in time. It's like a snapshot, which we can go back to and see how thing were when we took it.

git add ⟨file⟩ Snapshots the file in preparation for versioning

git diff Shows file differences not yet staged.

git diff –staged Shows file differences between staging and the last file version

git status Show information about the current state of the repository: What is the current branch, is everything up to date, what's new, what's changed.

git commit -m "⟨descriptive message⟩" Records file snapshots permanently in version history

git show ⟨commit⟩ To see what was new in a commit we can run `git show commit-id-here`.

Undoing Changes

Git allows us to return any selected file back to the way it was in a certain commit.

git checkout ⟨commit⟩ ⟨file⟩ Reverse everything done to a file to since the given commit.

Erase mistakes and craf replacement history in *local* work.

git reset –hard ⟨commit⟩ Discards all history and changes back to the specified commit

git reset ⟨commit⟩ Undoes all commits afer ⟨commit⟩, preserving changes locally

git reset ⟨file⟩ Unstages the file, but preserve its contents

For changes already pushed, we use `git revert` which undoes the changes of a given commit.

git revert HEAD The newest commit can be accessed by the `HEAD` alias.
- ◇ For other commits it's best to use an id; e.g., `git revert b10cc123.`

Refactoring Filenames

Relocate and remove versioned files

git rm ⟨file⟩ Deletes the file from the working directory and stages the deletion

git rm –cached ⟨file⟩ Removes the file from version control but preserves the file locally

git mv ⟨file-original⟩ ⟨file-renamed⟩ Changes the file name and prepares it for commit

What changed? –Reviewing our history

Browse and inspect the evolution of project files

git whatchanged Display the entire commit history along with which files were altered in each commit.

git log –oneline Lists version history for the current branch in one line for each commit.

git log –author="⟨pattern⟩" Search for commits by a particular author.

git log -p Display the full diff of each commit.

git log –grep="⟨pattern⟩" Search for commits with a commit message that matches ⟨pattern⟩.

git log – ⟨file⟩ Only display commits that have the specified file.