

Git Exercises

October 1, 2018

Peter J. Jones

✉ pjones@devalot.com

🐦 @devalot

<http://devalot.com>



Exercise: Cloning a Repository

1. Change to the following directory:
`repos`
2. Clone the `basic.git` repository:
`$ git clone basic.git`
3. This should have created a `basic` directory

Exercise: Viewing the History Log

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Use the following command to see the commit history:
`$ git log --oneline`
3. Pick a commit and run:
`$ git cat-file -p <commit>`
4. Also cat-file the listed tree, then the listed blob

Exercise: Telling Git Who You Are

If you haven't done so previously:

1. Set your name and email address using `git config`
2. Verify your settings:

```
$ git config --list
```

Exercise: Telling Git Which Editor to Use

If you haven't done so previously:

1. Tell Git which text editor to use:

```
$ git config --global core.editor <program>
```

Here are some examples for <program>:

- ▶ emacs
- ▶ vi
- ▶ "'C:/Program Files/Notepad++/notepad++.exe'"
- ▶ "code --wait"

Or search the web for instructions for your preferred editor.

Exercise: Adding a New File

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create a new file called `config.h`
3. Add some content to the file (any text will do)
4. Review the output of `git status`
5. Add the file to the index
6. Check the output of `git status` again
7. Commit the change

Exercise: Editing an Existing File

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Edit the `main.c` file
3. Add some content to the file (any text will do)
4. Review the output of `git status`
5. Review the output of `git diff`
6. Add and commit the change

Exercise: Renaming a File

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Rename the `config.h` file from the previous exercise to `init.h`
3. Review the output of `git status`
4. Notice that Git already staged the rename
5. Commit your change

Exercise: Removing a File

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Remove the `init.h` file
3. Review the output of `git status`
4. Notice that the file deletion was staged for you
5. Commit your changes
6. Restore the file

Exercise: Moving HEAD Around

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. List the names of all branches
3. For each branch:
 1. Check out the branch
 2. Use `git log --oneline` to find the latest commit hash

Exercise: Creating Branches

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Switch to the `master` branch
3. Create a new branch called `m2`
4. Use `git log` to see the latest commit:

`$ git log --oneline -1`
5. Create a branch called `f2` that starts at `merge-start`
6. Compare the latest commits on `m2` and `f2`

Exercise: Merging a Branch

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create and switch to a new branch that starts at v1.0:
`$ git checkout -b NAME v1.0`
3. Merge the `origin/feature` branch into your new branch

Exercise: Merging with Conflicts

1. Change to the directory holding a clone of the following repository:
`repos/conflicts.git`
2. Create and switch to a new branch that starts at v1.0:
`$ git checkout -b NAME v1.0`
3. Merge the `origin/feature` branch into your new branch
4. Resolve conflicts and finish the merge

Exercise: Automatically Resolving Conflicts

1. Change to the directory holding a clone of the following repository:
`repos/conflicts.git`
2. Create a new branch whose ancestor is the commit before HEAD
3. Merge with the `ours` option
4. Repeat the above steps, except this time use the `theirs` option

Exercise: Rebasing a Branch

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create a branch that starts at the commit named `merge-start`
`$ git checkout -b NAME merge-start`
3. Use `git log --oneline` to review the commit history
4. Rebase onto of the `master` branch
5. Review the commit history again and identify the rewritten commits and the merged commits from `master`

Exercise: Rebasing with Conflicts

1. Change to the directory holding a clone of the following repository:
`repos/conflicts.git`
2. Create a branch that starts at the commit named `merge-start`
`$ git checkout -b NAME conflict-start`
3. Use `git log --oneline` to review the commit history
4. Rebase onto of the `master` branch
5. Fix the conflict and use `git rebase --continue` to resume
6. Review the commit history again and identify the rewritten commits and the merged commits from `master`

Exercise: Interactive Rebasing

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create a branch that starts at the commit named `merge-start`
3. Interactively edit the commit which has the message "Add a version number"
4. Amend the commit when the rebase stops on it:

```
$ git commit --amend
```
5. Resume the rebase:

```
$ git rebase --continue
```

Exercise: Soft Reset

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create a branch that starts at the commit named `merge-start`
`$ git checkout -b NAME merge-start`
3. Use `git log --oneline` to review the commit history
4. Use `git reset --soft` to back up 2 commits:
`$ git reset --soft HEAD~2`
5. Use `git log --oneline` to see how the history changed
6. Use `git status` to review the index and working directory
7. Use `git commit` to create a new commit

Exercise: Mixed Reset

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create a branch that starts at the commit named `merge-start`
`$ git checkout -b NAME merge-start`
3. Use `git log --oneline` to review the commit history
4. Use `git reset --mixed` to back up 2 commits:
`$ git reset --mixed HEAD~2`
5. Use `git log --oneline` to see how the history changed
6. Use `git status` to review the index and working directory
7. Use `git commit` to create a new commit

Exercise: Hard Rest

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create a branch that starts at the commit named `merge-start`
`$ git checkout -b NAME merge-start`
3. Use `git log --oneline` to review the commit history
4. Use `git reset --hard` to back up 2 commits:
`$ git reset --hard HEAD~2`
5. Use `git log --oneline` to see how the history changed
6. Use `git status` to review the index and working directory
7. Notice that the working directory was also changed

Exercise: Amending the Last Commit

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create a branch that starts at the commit named `merge-start`
`$ git checkout -b NAME merge-start`
3. Take note of the last commit hash
4. Amend the last commit, changing the commit message
5. Notice that the most recent commit was rewritten

Exercise: Unstaging a File

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create a branch that starts at the commit named `merge-start`
`$ git checkout -b NAME merge-start`
3. Edit the `main.c` file, making a simple change to it
4. Stage (`git add`) and unstage (`git reset`) the file

Exercise: Restoring a File

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. In the last exercise we changed the `main.c` file, restore it back to its previous state

Exercise: Restoring a File

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Look through the commit history and find the commit that added a version number to `main.c`
3. Restore `main.c` to its content before the version number was added

Exercise: Reverting a Commit

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create a branch that starts at the commit named `merge-start`
`$ git checkout -b NAME merge-start`
3. Edit the `main.c` file, making a simple change to it
4. Commit the change
5. Revert the commit you just created

Exercise: Squashing Commits

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create a branch that starts at the commit named `merge-start`
`$ git checkout -b NAME merge-start`
3. Take note of the commit history
4. Start an interactive rebase with the first commit in the history
5. Squash all commits into the first listed commit
6. Review the commit history

Exercise: Staging Patch Hunks

1. Change to the directory holding a clone of the following repository:
`repos/conflicts.git`
2. Create a branch that starts off the `origin/feature` branch
3. Edit `main.c`; add comments to the top and bottom of the file
4. Review the difference between the working directory and the index:

```
$ git diff
```
5. Interactively stage one of the patch hunks by splitting the first hunk:

```
$ git add -p main.c
...
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? s
```
6. Review the updated index:

```
$ git status
$ git diff --cached
```

Exercise: Pushing and Popping the Stash

1. Change to the directory holding a clone of the following repository:
`repos/conflicts.git`
2. Create a branch that starts off the `origin/feature` branch
3. Add a comment to the bottom of `main.c`
4. Create a new stash
5. Review the index and working directory state
6. Restore the stash that was previously created

Exercise: Create Some Aliases

1. Take a moment and create some helpful aliases
2. Pick a repository and test your new aliases

Exercise: Tagging the Current Commit

1. Change to the directory holding a clone of the following repository:
`repos/basic.git`
2. Create a tag called `v1.1`
3. Use `git tag --list` to list your tags
4. Try using `git rev-parse` to see how the tag name resolves
5. Create a branch starting from the tag
6. Push the tag back to the `origin` remote
7. Use `git ls-remote` to confirm the tags were pushed

Exercise: Experimenting with Submodules

For each of the following steps you should create a unique branch off of master so you can easily jump back to the previous state.

1. Change to the directory holding a clone of the following repository:
`repos/submodules.git`
2. Go into the other submodule and see what branch it is on
3. Update the submodule to its latest upstream commit
4. Commit and push the project
5. Make a change to the submodule and commit it
6. Commit to the project and push everything
7. Add another submodule

Exercise: Experimenting with Subtrees

1. Change to the directory holding a clone of the following repository:
`repos/subtrees.git`
2. Alter the `other/main.c` file and commit the change
3. Notice that all files are local to the repository
4. Read the `notes.txt` file and then push the subtree

Exercise: Using the Blame Command

(Instructions forthcoming.)

Exercise: Bisecting a Project

(Instructions forthcoming.)