Context

This is a quick cheatsheet of common operations PyTorch developers use.

Developing a new feature

```
git checkout upstream viable/strict
git branch my-awesome-feature
git checkout my-awesome-feature
```

Checkout the PyTorch viable/strict branch. All PyTorch tests are guaranteed to be passing on the viable/strict branch (as opposed to the master branch, where they might not pass). viable/strict lags behind PyTorch's master branch and is automatically advanced whenever a new commit to master passes all tests. More details here

Create a new branch that is based on viable/strict and make some changes to PyTorch.

Building and testing PyTorch

When you've made some changes to PyTorch, build and test PyTorch. PyTorch has a lot of build flags; if you turn off some of them your build will go faster. The most important one is USE_CUDA, which defaults to 1. If your machine doesn't have a GPU, you probably want to build without CUDA: `USE_CUDA=0`

```
USE_KINETO=0 BUILD_CAFFE2=0 USE_DISTRIBUTED=0 USE_NCCL=0 BUILD_TEST=0 USE_XNNPACK=0
USE_FBGEMM=0 USE_QNNPACK=0 USE_MKLDNN=0 USE_MIOPEN=0 USE_NNPACK=0 BUILD_CAFFE2_OPS=0
USE_TENSORPIPE=0 python setup.py develop
```

PyTorch has a lot of test files under `test/`. You'll have to pick one (or a couple) that you want to test.

```
python test/test_torch.py
```

More details here

Submitting a change (the traditional way)

`git push` to the branch on your PyTorch fork. Then, open a pull request.

Submitting a change (using ghstack)

Run `ghstack`, if you have ghstack installed.

Rebasing your Pull Request (the traditional way)

Your Pull Request is too old and you want to rebase it onto a newer commit. What do you do?

```
git checkout my-awesome-feature
git pull --rebase upstream viable/strict
git push -f
```

The idea is to pull Pytorch into your branch while rebasing your (uncommited) commits on top of the newer ones. For example, if PyTorch's viable/strict branch looks like `A -> B -> C`, and you've made a new commit D at an older stage in time (`A -> D`), then `git pull --rebase viable/strict` will change your history to look like `A -> B -> C -> D`

Rebasing your Pull Request (if it was submitted via ghstack)

Your ghstack-created Pull Request is too old and you want to rebase it onto a newer commit. What do you do?

You have two options:

```
# Option 1: checkout the pull request using ghstack
ghstack checkout <pull_request_url>
git pull --rebase upstream viable/strict
ghstack
```

```
# Option 2: checkout the branch with your changes where you originally ran `ghstack`:
git checkout my-awesome-feature
git pull --rebase upstream viable/strict
ghstack
```

How to undo (almost) anything with Git

Did you run `git pull upstream viable/strict` instead of `git pull --rebase upstream viable/strict` and regret it? See https://github.blog/2015-06-08-how-to-undo-almost-anything-with-git/