Using Git to sync different computers

25 June 2015

The previous posts have introduced the basics of using Git to create a local repository, upload it to an online repository, and ways of reverting to previous snapshots.

Currently, I have a desktop in my office which I use to do most of my work. However, sometimes while travelling I would like to work on the train/plane on my laptop. Until recently, transferring files between the two machines was somewhat primitive, and I would simply email a zip folder from one computer to the other. Of course, using a shared folder in Dropbox was another option, but probably not much better.

If you are using an online repository, then Git offers a great solution: the `clone` and `pull` commands.

For this post we will assume you have two computers on hand, which we'll call Computer 1 and Computer 2, but you could also follow the same steps on a single computer by using separate folders.

Step 1: Create a repo on Computer 1

We'll start by quickly creating a local repo called `sharedrepo` on Computer 1 with a single file:

```
$ mkdir ./sharedrepo
$ cd ./sharedrepo
$ echo 'standing in line to see the show tonight' >> file1.txt
$ git init
$ git add .
$ git commit -m 'first commit for shared repo on Computer 1'
```

We next need to create an online repo, preferably also with the name `sharedrepo` on our Github/Bitbucket account. See Steps 2 and 3 in this post for info on how to do this.

After the online repo is created, `add` it to your local repo on Computer 1 with:

```
$ git remote add origin https://username@bitbucket.org/username/sharedre
```

(replacing `username` with your own username twice)

Finally, `push` the local repo to the online repo:

```
$ git push -u origin master
```

We have just created a very basic repo, and in the next step we shall `clone` it to Computer 2.

Step 2: Clone the repo to Computer 2

On Computer 2, move to a folder where you want to replicate the `sharedrepo`

```
$ cd /path/to/
```

You do not need to make the folder. The `clone` command now lets us download an exact copy of the `sharedrepo`:

```
$ git clone https://username@bitbucket.org/username/sharedrepo.git
```

(as usual, replacing username twice above. You may be asked for your Bitbucket password.)

Step 3: Modify on Computer 2 and push

On Computer 2, modify `file1.txt` by:

```
$ cd ./sharedrepo
$ echo "and there's a light on" >> file1.txt
```

and now do the usual `add`, `commit`, `push`:

```
$ git add .
$ git commit -m 'second commit from Computer 2'
$ git push -u origin master
```

which will update both the local and online versions of the repo.

Step 4: Pull modified repo to Computer 1

Now, suppose we want to work on Computer 1 again. But, the version on Computer 1 is older than the version on the online repo and Computer 2. No problem, the `pull` command will sort this out. On Computer 1, in the `sharedrepo` folder:

```
$ git pull origin master
```

or (if you have already used `push -u origin master` - the key is the `-u` flag)

```
$ git pull
```

And it's as simple as that, the local `sharedrepo` repo has been updated to the latest version.

Rinse and repeat

As you can probably guess, you now just cycle through Steps 3 and 4:

- modify files
- `add`, `commit`, `push`
- `pull` repo to other machine

The `clone` command is only used to download a repo when no previous version exists on that machine.

Conclusion

This post discussed the `clone` and `pull` commands which can be used (along with `push`) to share a repo between computers. Although the explanation may have been a bit long, the process is actually very quick - even faster than zipping/emailing files.

Note: Step 2 alternative

When cloning a repo, you could also use:

```
$ git clone https://bitbucket.org/username/sharedrepo/
```

where the string is just the url from the `sharedrepo` page on the Bitbucket site.

Either version of the command will work.