Git remote branches

Reuven M. Lerner, PhD
reuven@lerner.co.il • http://lerner.co.il/

1

Cloning

- So we have a repository!
- Let’s say we want to duplicate it
- How do we do that? We clone it!

“git clone”

```
$ git clone simple new-simple
Cloning into new-simple...
done.

$ cd new-simple

$ git log --pretty=oneline  -5
d942bd054e726d64bf2485e1720f42528e668c66 Added a comment
516c684ca66d28b8e2be74a1f60a0918f73ac134 Fixed
1bff9773fcd762afd75008e654054467a6b29361 Added comment in master
0105700f3cd79322343e96d44ccf6b0a4def5ee7 Added a comment
f6a023d6bf108639d75e7f7f79f033d56bd582bc cherry
```

Truly a clone!

- We have duplicated our repository
- It contains not only the same files, but the same history, including all commits
- It is a clone in every sense ...
- ... but it also remembers where it came from!

origin

```
$ cat simple/.git/config
[core]
  repositoryformatversion = 0
  filemode = true
  bare = false
  logallrefupdates = true
  ignorecase = true

$ cat new-simple/.git/config
[core]
  repositoryformatversion = 0
  filemode = true
  bare = false
  logallrefupdates = true
  ignorecase = true
[remote "origin"]
  fetch = +refs/heads/*:refs/remotes/origin/*
  url = /Users/reuven/Consulting/Courses/Git/Programs/simple
```

Update the clone

```
$ git checkout master
Previous HEAD position was d942bd0... Added a comment
Branch master set up to track remote branch master from
origin.
Switched to a new branch 'master'
$ cat >> hello.rb
# Added a line in the clone
$ git commit -a -m 'Added line'
[master e86308d] Added line
 1 files changed, 1 insertions(+), 0 deletions(-)
```

Examine our status

```
$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
nothing to commit (working directory clean)
```

Remote branches

- Git normally works with local branches
- But we can work with remote branches, as well — branches in other repositories
- “git status” tries to tell us where we stand relative to a remote branch

Remote repositories

- Remember before, I told you that the most important thing in Git is a commit?
- When you have two repositories, the main thing you're doing is exchanging commits.
- Each repository can say to the other, "Hey, send me all of the commits that you're missing."
- Sounds like a merge, right? Exactly! Working with a remote repository is nothing more than a merge — but the branch from which we merge is elsewhere

Remote protocols

- One complicating factor with remote branches is that you need to somehow access the remote server
- Git uses a number of protocols to do this. Two of the most common are HTTPS (i.e., secure HTTP) and SSH (i.e., same as used under Unix)

Remote HTTPS

- If the remote server uses HTTPS, then you'll use a URL to connect to it

```
https://reuven@bitbucket.org/reuven/foo
```

- In order to log in, you'll need to enter a username and password
- You can specify the password in the URL, although that's not very secure:

```
https://reuven:password@bitbucket.org/reuven/foo
```

Credential caches

- Tired of re-typing your password each time you use an HTTP/HTTPS repository?
- Use the Git credential cache!

```
git config --global credential.helper cache
```

- This stores your password for 15 minutes in memory
- You can also set up other credentials that stick around longer, depending on OS

SSH protocol

- SSH is both secure and standard
- You're more likely to use SSH. But to do so, you'll need an ssh key pair. The pair contains one private key and one public key:
  - private key: Never, *ever* share this with anyone.
  - public key: Share this with whomever you want

Generating keys: Unix

- Under Linux and OS X, you can generate a keypair with the "ssh-keygen" program, included in your operating system
- Just invoke:

```
ssh-keygen
```

- By default, the files will be installed as ~/.ssh/id_rsa and ~/.ssh/id_rsa.pub. You may change the names, if you like.
- You may also wish to use a passphrase, to restrict usage of your key. This is almost certainly a good idea.
- Your keys will be located under ~/.ssh/. This directory is only readable (and writable) by your user. Don't change those permissions!

Generating keys: Windows

- Windows doesn't have any ssh key-generating software by default
- If you installed Git, then you have the ssh-keygen program (available via git-bash)
- If you installed Git-GUI, then you should go to the "help" menu and click on "show SSH key." You can click on "generate key" to have it create a new SSH key for you.
- This will show the public key, and let you copy it (e.g., to e-mail it to a Git administrator)

Windows via git-bash

- From the git-bash prompt, you can also write:

```
ssh-keygen -b 4096
```

- https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/

What now?

- Your Git administrator will need your public key (*not your private key!*) in order to give you permission to use the Git server.
- Whether your Git administrator has you e-mail the public key, or enter it in a Web site, depends on the site
- You should never be afraid to share your public key. **You should be *very* afraid to share your private key.**

Use a new key?

- Some people use the same ssh key for everything they do.
- It's probably a good idea to use a new, separate key pair for working with a Git repository.
- That way, if someone steals your private key, you don't need to lock down *everything* you use.

Remote repositories

- Git can handle any number of remote repositories
- Each is known as a "remote", and has a "remote" section in .git/config, giving it a unique name
- The main remote is traditionally known as "origin", but can be given any name
- The remote has a URL (specifying a protocol) and an indication of the refs (i.e., branches) on that remote

Cloning

- So we have a repository!
- Let’s say we want to duplicate it
- How do we do that? We clone it!

Remote example

```
[remote "origin"]

  url = ssh://reuven@lerner.co.il:8080/git-class

  fetch = +refs/heads/*:refs/remotes/class/*
```

Cloning a remote

- If you’re using ssh, just give the user@hostname:/pathname of the remote

```
git clone git@example.com:/home/git/foo
```

- Once you do this, your local version will have the remote's origin set correctly

Adding a remote

- You can connect to a remote with “git remote add”, giving the nickname ("foo") and the URL:

```
git remote add foo reuven@github.com:/foo
```

Adding a remote branch

- If you're in branch "develop", you can do

```
git push origin develop
```

- and that will create a remote branch by the same name ("develop")!
- When someone does a "git pull", they'll then get the name of the remote branch. This doesn't mean you have a local branch "develop". But if you create such a branch, it'll track the remote automatically.

Checking your remotes

```
$ git remote show origin
* remote origin
  Fetch URL: ssh://reuven@new.lerner.co.il:29418/testrml
  Push  URL: ssh://reuven@new.lerner.co.il:29418/testrml
  HEAD branch: master
  Remote branch:
    master tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (fast-forwardable)
```

Remote commands

- "fetch" tells Git to retrieve all of the commits from a remote branch that it is missing:

```
git fetch origin master
```

- However, this doesn't perform a merge! It merely ensures that the remote's commits exist locally
- "pull" and "push" are the main Git commands that have to do with remotes, and what you'll use each day

fetch and pull

- "git fetch" tells Git to go to the remote repository and grab all of the commits we're missing
- But it doesn't then merge them in!
- "git pull" fetches them, but it also merges them.
- You will likely "git pull" very often
- You can also use "git pull --rebase" to merge using rebase, rather than a simple merge

What if you just fetch?

- You get additional commits
- These commits extend your current tree
- Now the "origin/master" pointer points to a commit past your local "master" pointer. In theory, you can then just fast-forward to get there

"git pull" warning

- If you "git pull", Git tries to make life easier for you — by merging the commits from the remote branch, without asking you first!
- It will merge into the current branch, which might (or might not) be what you want

push

- By contrast, "git push" tells Git that you want to send to the remote server all of the commits that you have, and which it lacks.
- It might feel like you're "committing to a central server," but you're not! You're sharing your commits with a central repository.
- You cannot push if the remote has commits that you're missing. In other words: pull, then push

How to push

- The usual syntax is:

```
git push origin master
```

- That says, "Take all of my commits, and merge them into the 'master' branch on the remote host known as 'origin'."

- Sometimes, you can get away with saying "git push" without specifying an origin or branch. If Git complains, specify them!

Keep things shorter

- Or, you can tell Git that when you say "git push", it should infer that the current branch should be used, and the current remote:

```
[push]

    default = current
```

Push our changes!

```
$ git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 309 bytes, done.
Total 3 (delta 2), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /Users/reuven/Consulting/Courses/Git/Programs/simple
   ce0e1f3..e86308d  HEAD -> master
```

This is important!

- Pull means, "get all of the commits from the remote branch, and merge them into my current branch"
- Push means, "take all of the commits from my current branch, and merge them to the remote branch."
- You must always pull before you push! Git will warn you if you try to do otherwise. (And don't use the —force option to avoid this warning…)

Check our origin

```
$ git log --pretty=oneline  -5
e86308d1a2135e46932d68d26150bec1088037ee Added line
ce0e1f3d123f27c72083b5b1f887f3f7ba8f3f5f Added a print statement
326d7237bc52cf2a0d047e8c48e675f2b4d757f9 Added a bad comment
d942bd054e726d64bf2485e1720f42528e668c66 Added a comment
516c684ca66d28b8e2be74a1f60a0918f73ac134 Fixed
```

Add to the origin

```
$ git commit new-file.txt -m 'Added new-file'
[master 6dc1dc7] Added new-file
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 new-file.txt
```

Now the clone can pull

```
$ git pull
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From /Users/reuven/Consulting/Courses/Git/Programs/simple
   e86308d..6dc1dc7  master     -> origin/master
Updating e86308d..6dc1dc7
Fast-forward
 new-file.txt |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 new-file.txt
```

Pushing and pulling

- Pulling is the equivalent of retrieving the commits and merging them into your current branch
- Pushing does the same in the other direction — but you can only push if you’re up to date with the master
  - Otherwise, the push will be refused

Pushing and pulling

- So:
  - Only push after you have done a “git pull” (or “git pull --g”) and run your tests
  - Only pull after you have committed your latest changes — otherwise, it can be messy

Pulling

- You pull from an origin and a branch:

```
git pull ORIGIN BRANCH
```

- For example:

```
git pull origin master

git pull myserver development
```

Pulling with rebase

- Normally, “git pull” fetches the remote’s commits, and merges them into your current branch.
- But it’s often a good idea to use rebase, such that you can fast-forward on your next push. To do that, say

```
git pull --rebase
```

Pushing

- You can push to an origin (default branch):

```
git push origin master
```

- You can push to an origin (other branch):

```
git push myserver development
```

Tracking branches

- When you say just "git pull" or "git push", Git tries to find and use the "tracking branch".
- That is, a remote branch that's associated with your current local branch

Creating a tracking branch

- Use the —tracking option when creating a new branch:

```
git branch --track feature1 origin/feature1
```

- Now, when you use "git checkout feature1", Git will know the source you should use

Tracking an existing branch

- If you have a local branch, and want it to track a remote one, the syntax is somewhat different:

```
git branch -u upstream/foo
```

- This tells Git that the current branch should track the branch "foo" on the remote "upstream"

- If you want another branch to track, add its name:

```
git branch -u upstream/foo foo
```

Multiple remotes

- You can pull from, or push to, multiple remotes
- Typically, you'll be pulling from one remote (the tracking branch for the current branch) and pushing to multiple remotes
- Just make sure to say "git push REMOTE BRANCH", and you'll be fine

Why multiple remotes?

- Multiple "central" servers
- Different servers to which you're deploying
- Production vs. development
- The "lieutenants" way of doing things (a la Linux)

Listing remotes

```
$ git branch -r       # show remote branches

  origin/foo

  origin/master



$ git branch -a       # show all branches, including remote

* master

  remotes/origin/foo

  remotes/origin/master
```

Tags and remotes

- "git push" doesn't normally push tags!
- If you want to push a tag, use the same syntax you would use with remote branches:

```
git push origin v1.4
```

- You can also say

```
git push origin --tags
```

- which will push all tags to the server

Not a central server!

- Remember that every repository is separate — so we’re not talking here about a “server” and “client”
- However, if everyone agrees to use a central repository, that helps things
- Pushing to (and pulling from) an agreed-upon repository is very common

GitHub

- You can put together your own central Git server and repository
- But many people say: Why do so?
- GitHub offers such services, for pay (or free for open-source projects)
- Many, many features above and beyond that

Pull requests

- GitHub allows you to work in the regular way, with pushing and pulling
- You can also work with "pull requests," in which each user has his or her own repository. The user then sends a "pull request" to the manager, asking that one or more commits be pulled into the main repository.
- Many open-source projects use GitHub (and pull requests)

Other options

- BitBucket: A competitor to GitHub; much smaller, but also lower prices and a different pricing model
- Gitlab: An open-source project that has many of GitHub's features, but can be installed on your own server. A hosted, paid version (with additional features) is also available
- Gerrit: Open-source system in which pushing to the server doesn't perform a merge, but rather puts the commit on the side, awaiting approval