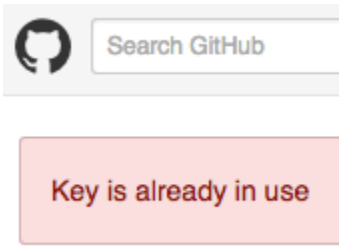
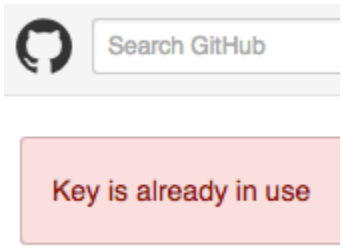
How To Use SSH With Multiple GitHub Accounts

By Itamar Ostricher Wednesday, June 3, 2015 3 Software Engineering git, howto Permalink0



All I wanted is to have two GitHub accounts, and use them both from the same computer, using SSH. Is this too much to ask?

I've been using my personal GitHub account for some time, and associated my default SSH keys with that account. When I created a work account on GitHub, I expected I'd be able to associate the same SSH keys with the work account, so I can `git pull/push` as easily.
This proved to be a problem. GitHub does not allow reusing SSH keys across accounts. Here is the message I got when trying to do that:



GitHub error message when trying to reuse SSH keys across accounts

I managed to overcome this problem by creating new SSH keys for the work account. Read on for the details.

Create new SSH keys

Start with creating new SSH key-pair. Assuming you already have default SSH keys under `~/.ssh/id_rsa`, make sure you specify a different path for the new keys.

```
1  itamar@legolas ~ $ ssh-keygen -C "email@work.com"
2  Generating public/private rsa key pair.
3  Enter file in which to save the key (/Users/itamar/.ssh/id_rsa): /Users/itamar/.ssh/id
4  Enter passphrase (empty for no passphrase):
5  Enter same passphrase again:
6  Your identification has been saved in /Users/itamar/.ssh/id_work_rsa.
7  Your public key has been saved in /Users/itamar/.ssh/id_work_rsa.pub.
8  The key fingerprint is:
9  f6:5c:d2:d8:08:17:23:98:d1:aa:4d:1f:80:24:24:d3 email@work.com
10 The key's randomart image is:
11 +--[ RSA 2048]----+
12 |ooo....=. o      |
13 | oE.. + .. o     |
14 |       o. .      |
15 |      o .o =     |
16 |     + .S.+ +    |
17 |    . ...o o     |
18 |          o      |
```

```
|                  |
|                  |
+------------------+
```

Associate the new SSH keys with a GitHub alias

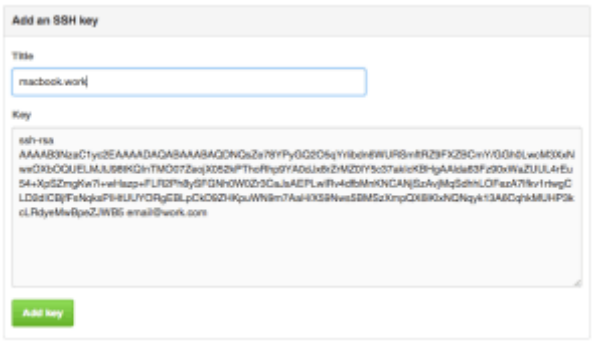
This is needed to “help” Git use the correct keys. Add a *Host* entry to your SSH config file, usually under `~/.ssh/config` (create it if one doesn’t already exist):

```
Host github.work
    HostName github.com
    User git
    IdentityFile ~/.ssh/id_work_rsa
    IdentitiesOnly yes
```

Upload the new SSH keys

Copy the contents of the new SSH public key, and paste it as a new SSH key on the GitHub account.

```
itamar@legolas ~ $ cat /Users/itamar/.ssh/id_work_rsa.pub | pbcopy
```



Use the GitHub.work alias as the remote URL

The final step is to tell the Git client to use the new SSH keys by using the alias:

```
itamar@legolas work $ git clone git@github.work:work/the-work-repo.git
Cloning into 'the-work-repo'...
remote: Counting objects: 130, done.
remote: Compressing objects: 100% (124/124), done.
remote: Total 130 (delta 60), reused 0 (delta 0)
Receiving objects: 100% (130/130), 403.23 KiB | 276.00 KiB/s, done.
Resolving deltas: 100% (60/60), done.
Checking connectivity... done.
```

That’s it. You can see that the repository origin is using the work alias:

```
itamar@legolas the-work-repo (master) $ git remote -v
origin    git@github.work:work/the-work-repo.git (fetch)
origin    git@github.work:work/the-work-repo.git (push)
itamar@legolas the-work-repo (master) $ cat .git/config
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
```

```
9       logallrefupdates = true
10      ignorecase = true
11      precomposeunicode = true
12  [remote "origin"]
13      url = git@github.work:work/the-work-repo.git
14      fetch = +refs/heads/*:refs/remotes/origin/*
15  [branch "master"]
16      remote = origin
17      merge = refs/heads/master
```

Notes

- While I used GitHub for the post, this works just as well with Bitbucket, or any other Git hosting that supports SSH for that matter.
- Tested on a MacBook, should work pretty much the same on any Linux box. YMMV.
- I saw others write about this subject using ssh-agent. Not sure why it's needed. If you know – please share!