

Michael Herman

Software Developer



<input type="text" value="Search"/>
<input type="button" value="Navigate..."/>

- [Blog](#)
- [About](#)

Managing Multiple Github Accounts

Sep 16th, 2013 7:43 am

Let's look at how to manage multiple Github accounts from one computer. In essence, it's simply a matter of balancing both your git and ssh configurations - which actually is not as bad as it might seem.

Note: This tutorial is meant for Unix users.

Set up SSH Keys

Let's assume your two Github accounts are named *githubPersonal* and *githubWork*, respectively.

Create two SSH keys, saving each to a separate file:

```
1 $ cd ~/.ssh
2 $ ssh-keygen -t rsa -C "your_email@associated_with_githubPersonal.com"
3 # save it as id_rsa_personal when prompted
4 $ ssh-keygen -t rsa -C "your_email@associated_with_githubWork.com"
5 # save it as id_rsa_work when prompted
```

The above commands setup the following files:

- *id_rsa_personal*
- *id_rsa_personal.pub*
- *id_rsa_work*
- *id_rsa_work.pub*

Add the keys to your Github accounts:

Copy the key to your clipboard:

```
1 $ pbcopy < ~/.ssh/id_rsa_personal.pub
```

Add the key to your account:

- Go to your Account Settings
- Click "SSH Keys" then "Add SSH key"
- Paste your key into the "Key" field and add a relevant title
- Click "Add key" then enter your Github password to confirm

Repeat the process for your *githubWork* account.

Create a configuration file to manage the separate keys

Create a config file in `~/.ssh/`

```
1 $ touch config
```

Edit the file using the text editor of your choice. I used vim - \$ vim config:

```
1 # githubPersonal
2 Host personal
3     HostName github.com
4     User git
5     IdentityFile ~/.ssh/id_rsa_personal
6
7 # githubWork
8 Host work
9     HostName github.com
10    User git
11    IdentityFile ~/.ssh/id_rsa_work
```

Update stored identities

Clear currently stored identities:

```
1 $ ssh-add -D
```

Add new keys:

```
1 $ ssh-add id_rsa_personal
2 $ ssh-add id_rsa_work
```

Test to make sure new keys are stored:

```
1 $ ssh-add -l
```

Test to make sure Github recognizes the keys:

```
1 $ ssh -T personal
2 Hi githubPersonal! You've successfully authenticated, but GitHub does not provide shell access.
3 $ ssh -T work
4 Hi githubWork! You've successfully authenticated, but GitHub does not provide shell access.
```

Test PUSH

On Github, create a new repo in your personal account, *githubPersonal*, called *test-personal*.

Back on your local machine, create a test directory:

```
1 $ cd ~/documents
2 $ mkdir test-personal
3 $ cd test-personal
```

Add a blank “readme.md” file and PUSH to Github:

```
1 $ touch readme.md
2 $ git init
3 $ git add .
4 $ git commit -am "first commit"
5 $ git remote add origin git@personal:githubPersonal/test-personal.git
6 $ git push origin master
```

Notice how we're using the custom account, `git@personal`, instead of `git@github.com`.

Repeat the process for your *githubWork* account.

Test PULL

Add some text to the *readme.md* file in your personal account on Github.

Now PULL and merge the changes by running the following command within the *test-personal* directory:

```
1 $ git pull origin master
```

Again, repeat this for your *githubWork* account.


Questions? Comments? Did I miss something? Comment below.


Authored by Michael Herman Sep 16th, 2013 7:43 am [github](#)

92

[« What's the Difference Between a Pre-Order and an MVP? Basic Git Commands »](#)

Comments

15 Comments Michael Herman  Login ▾

♥ Recommend  Share Sort by Best ▾



Join the discussion...



sigumagwa · a year ago

Works awesomely. Bookmarked

2 ^ | ▾ · Reply · Share ›



michaelherman Mod → sigumagwa · a year ago

Cheers!

^ | ▾ · Reply · Share ›



Subidubam · 9 months ago

Has been looking for this! Simple and works well. Thank you!

1 ^ | ▾ · Reply · Share ›



David Condrey · a year ago

If your on OSX there is a simpler solution.. You can configure git to associate your login credentials specific to the repository path rather than the domain (which is its default behavior).

In terminal enter the command

```
| git config --global --edit
```

This will open a configuration file. If you haven't already, you may want to set your default editor so the file opens in your preferred application.. For example, to set Sublime Text as your default editor: `git config --global core.editor "subl -n -w"`

With the config file opened, search for `useHttpPath` (or define it if it doesn't exist). And set it's value to `true`. It