27 MARCH 2018 / #GITHUB

How to manage multiple GitHub accounts on a single machine with SSH keys

Bivil M Jacob

Read more posts by this author.

The need to manage multiple GitHub accounts on the same machine comes up at some point in time for most developers. Every single time I happen to change my Mac or need to Git push with a new work account, I end up surfing for the how to's of something I have done over half a dozen times.

My laziness in not documenting the process and inability to remember the steps makes me spent a decent amount of time getting the bits and pieces from all over the web and then somehow making it work.

I'm sure there are many of you who have been there, done that and many more of you who are just waiting for the next time the same thing occurs (myself included!). This endeavor is meant to help us all out.

1. Generating the SSH keys

Before generating an SSH key, we can check to see if we have any existing SSH keys: `ls -al ~/.ssh` This will list out all existing public and private key pairs, if any.

If `~/.ssh/id_rsa` is available, we can reuse it, or else we can first generate a key to the default `~/.ssh/id_rsa` by running:

```
ssh-keygen -t rsa
```

When asked for the location to save the keys, accept the default location by pressing enter. A private key and public key `~/.ssh/id_rsa.pub` will be created at the default ssh location `~/.ssh/` .

Let's use this default key pair for our personal account.

For the work accounts, we will create different SSH keys. The below code will generate the SSH keys, and saves the public key with the tag *"email@work_mail.com"* to `~/.ssh/id_rsa_work_user1.pub`

```
$ ssh-keygen -t rsa -C "email@work_mail.com" -f "id_rsa_work_user1"
```

We have two different keys created:

```
~/.ssh/id_rsa
~/.ssh/id_rsa_work_user1
```

2. Adding the new SSH key to the corresponding GitHub account

We already have the SSH public keys ready, and we will ask our GitHub accounts to trust the keys we have created. This is to get rid of the need for typing in the username and password every time you make a Git push.

Copy the public key `pbcopy < ~/.ssh/id_rsa.pub` and then log in to your personal GitHub account:

1. Go to `Settings`
2. Select `SSH and GPG keys` from the menu to the left.
3. Click on `New SSH key`, provide a suitable title, and paste the key in the box below
4. Click `Add key` — and you're done!

> For the work accounts, use the corresponding public keys (`pbcopy < ~/.ssh/id_rsa_work_user1.pub`) and repeat the above steps in your GitHub work accounts.

3 . Registering the new SSH Keys with the ssh-agent

To use the keys, we have to register them with the **ssh-agent** on our machine. Ensure ssh-agent is running using the command `eval "$(ssh-agent -s)"`.
Add the keys to the ssh-agent like so:

```
ssh-add ~/.ssh/id_rsa
ssh-add ~/.ssh/id_rsa_work_user1
```

Make the ssh-agent use the respective SSH keys for the different SSH Hosts.

This is the crucial part, and we have two different approaches:

Using the SSH configuration file (Step 4), and having only one active SSH key in the ssh-agent at a time (Step 5).

4. Creating the SSH config File

Here we are actually adding the SSH configuration rules for different hosts, stating which identity file to use for which domain.

The SSH config file will be available at **~/.ssh/config**. Edit it if it exists, or else we can just create it.

```
$ cd ~/.ssh/
$ touch config           // Creates the file if not exists
$ code config            // Opens the file in VS code, use any editor
```

Make configuration entries for the relevant GitHub accounts similar to the one below in your `~/.ssh/config` file:

```
# Personal account, - the default config
Host github.com
   HostName github.com
   User git
   IdentityFile ~/.ssh/id_rsa

# Work account-1
Host github.com-work_user1
   HostName github.com
   User git
   IdentityFile ~/.ssh/id_rsa_work_user1
```

"**work_user1**" is the GitHub user id for the work account.

"***github.com*-work_user1**" is a notation used to differentiate the multiple Git accounts. You can also use "**work_user1.*github.com*"** notation as well. Make sure you're

repository or when you set the remote origin for a local repository

The above configuration asks ssh-agent to:

- Use **id_rsa** as the key for any Git URL that uses **@github.com**
- Use the **id_rsa_work_user1** key for any Git URL that uses **@github.com-work_user1**

5. One active SSH key in the ssh-agent at a time

This approach doesn't require the SSH config rules. Rather we manually ensure that the ssh-agent has only the relevant key attached at the time of any Git operation.

`ssh-add -l` will list all the SSH keys attached to the ssh-agent. Remove all of them and add the one key you are about to use.

If it's to a personal Git account that you are about to push:

```
$ ssh-add -D            //removes all ssh entries from the ssh-agent
$ ssh-add ~/.ssh/id_rsa                 // Adds the relevant ssh key
```

The ssh-agent now has the key mapped with the personal GitHub account, and we can do a Git push to the personal repository.

To push to your work GitHub account-1, change the SSH key mapped with the ssh-agent by removing the existing key and adding the SSH key mapped with the GitHub work account.

```
$ ssh-add -D
$ ssh-add ~/.ssh/id_rsa_work_user1
```

can do a Git push to the work repository. This requires a bit of manual effort, though.

Setting the git remote Url for the local repositories

Once we have local Git repositories cloned /created, ensure the Git config user name and email is exactly what you want. GitHub identifies the author of any commit from the email id attached with the commit description.

To list the config name and email in the local Git directory, do `git config user.name` and `git config user.email`. If it's not found, update accordingly.

```
git config user.name "User 1"   // Updates git config user name
git config user.email "user1@workMail.com"
```

6. While Cloning Repositories

Note: step 7 will help, if we have the repository already available on local.

Now that the configurations are in place, we can go ahead and clone the corresponding repositories. On cloning, make a note that we use the host names that we used in the SSH config.

Repositories can be cloned using the clone command Git provides:

```
git clone git@github.com:personal_account_name/repo_name.git
```

The work repository will require a change to be made with this command:

```
git clone git@github.com-work_user1:work_user1/repo_name.git
```

between @ and : should match what we have given in the SSH config file.

7. For Locally Existing Repositories

If we have the repository already cloned:

List the Git remote of the repository, `git remote -v`

Check whether the URL matches our GitHub host to be used, or else update the remote origin URL.

```
git remote set-url origin git@github.com-worker_user1:worker_user1/repo_name.git
```

Ensure the string between @ and : matches the Host we have given in the SSH config.

If you are creating a new repository on local:

Initialize Git in the project folder `git init`.

Create the new repository in the GitHub account and then add it as the Git remote to the local repository.

```
git remote add origin git@github.com-work_user1:work_user1/repo_name.git
```

Ensure the string between @ and : matches the Host we have given in the SSH config.

Push the initial commit to the GitHub repository:

```
git add .
```

We are done!

Adding or updating the Git remote of the local Git directory with the proper host will take care of selecting the correct SSH key to verify our identity with GitHub. With all the above in place, our `git operations` should work seamlessly.

If this article was helpful, tweet it or share it.

Happy holidays from freeCodeCamp.org.

Become a monthly supporter of freeCodeCamp.org. Or make a tax-deductible year end gift. Every little bit helps.

Support freeCodeCamp

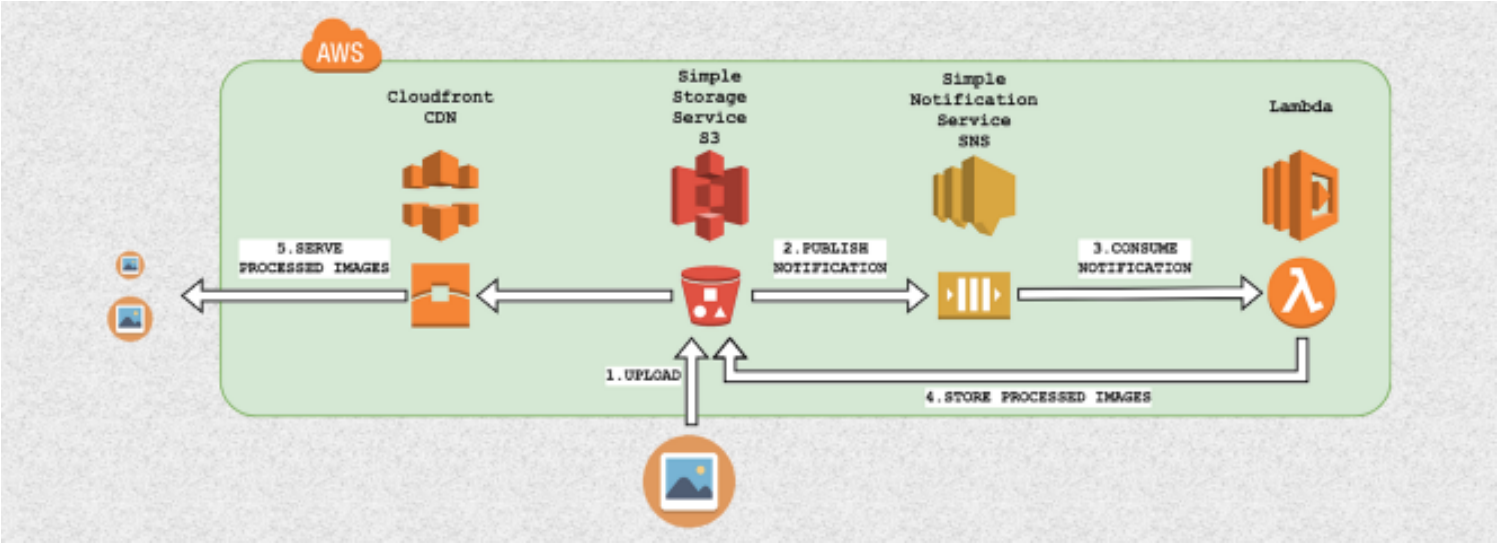
Countinue reading about

Github

How to Write Good Commit Messages: A Practical Git Guide

My list of GitHub tips and third-party apps that help me stay productive

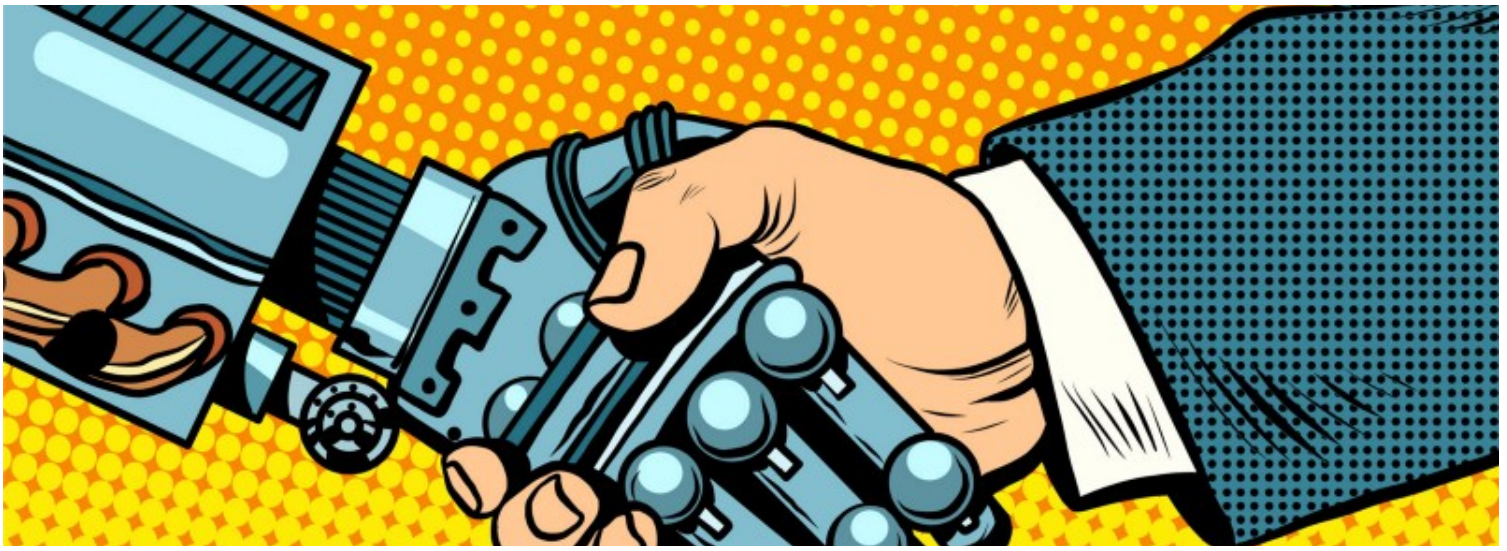
The beginner's guide to Git & GitHub



#AWS

How to boost your performance with serverless architectures

2 YEARS AGO



#ARTIFICIAL INTELLIGENCE

AI is Taking Peoples' Jobs - Are Developers Next?

SAM WILLIAMS 2 YEARS AGO

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) nonprofit organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

You can make a tax-deductible donation here.

Our Nonprofit

About
Alumni Network
Open Source
Shop
Support
Sponsors
Academic Honesty
Code of Conduct
Privacy Policy
Terms of Service
Copyright Policy

Trending Guides

2019 Web Developer Roadmap
Python Tutorial
CSS Flexbox Guide
JavaScript Tutorial
Python Example
HTML Tutorial
Linux Command Line Guide
JavaScript Example
Git Tutorial
React Tutorial
Java Tutorial
Linux Tutorial
CSS Tutorial
jQuery Example
SQL Tutorial
CSS Example
React Example
Angular Tutorial
Bootstrap Example
How to Set Up SSH Keys
WordPress Tutorial
PHP Example