Setting up multiple GitHub accounts with custom config

TIPS & TRICKS / POSTED ON JUNE 20, 2022

With Github, many use different accounts for different tasks, and you may find yourself switching between a personal and work account or toggling between a plethora of individual customer accounts. Trying to work with this setup and knowing that you are using the correct account and utilizing the right SSH key can be a hassle to get to work correctly.

Thankfully, there is a great solution that doesn't require you to clone a repo with a custom host. Let us share how we use custom config files, and profiles for multiple GitHub accounts greatness.

Initial setup of SSH keys

Before diving in, make sure that you have generated your SSH keys and added them to your agent. You can follow the referenced guide on Github for details.

1. Create SSH keys for every account that you plan to use.

2. Your .ssh s

3. Using the screenshot as our reference, we have SSH keys for one personal account, one work account, two customer accounts, and the config file (don't be like me and forget to add your keys to your GitHub accounts).

4. Correct setup of the config file is essential for this to work properly. You need to add all of your accounts to the config file and synch it with the correct SSH key. It can look a little something like this:

```
# Personal Account
Host personal
  Hostname github.com
  AddKeysToAgent yes
  UseKeychain yes
  User git
  IdentityFile ~/.ssh/personal_rsa
  IdentitiesOnly yes

# Work Account
Host work
  Hostname github.com
  AddKeysToAgent yes
  UseKeychain yes
  User git
  IdentityFile ~/.ssh/work_rsa
  IdentitiesOnly yes

# Customer 1
Host customer1
  Hostname github.com
  AddKeysToAgent yes
  UseKeychain yes
  User git
  IdentityFile ~/.ssh/customer1_rsa
  IdentitiesOnly yes

# Customer 2
Host customer2
  Hostname github.com
  AddKeysToAgent yes
  UseKeychain yes
  User git
  IdentityFile ~/.ssh/customer2_rsa
  IdentitiesOnly yes
```

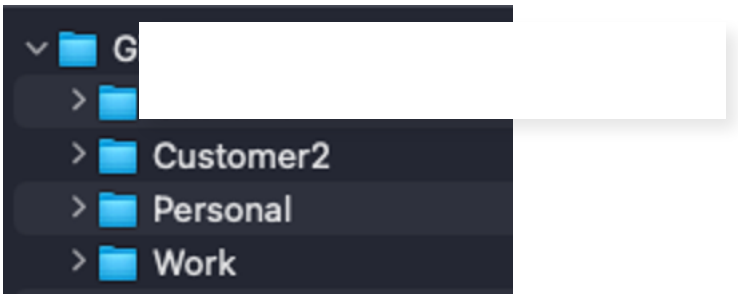
5. Because we don't want to change the host every time we clone a repo, keep in mind that you need to use the following when cloning the GitHub repos:

```
~ git clone git@<host>:<account>/<repo>.git
```

6. Next, you can actually start working.

Custom Configs are the Secret Sauce

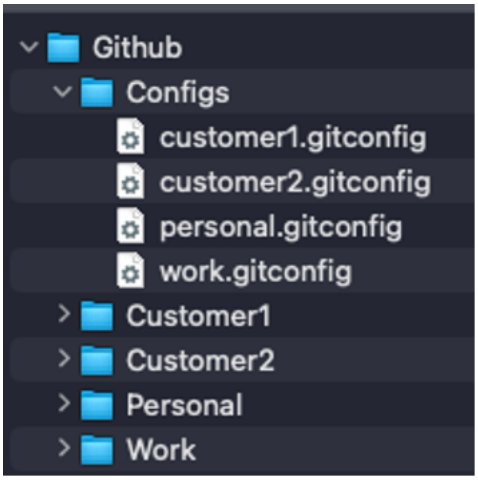
To save time, energy, and money, make custom configurations for each account your secret sauce. The first step in this process is using a logical folder structure for your Github repos. It should look like the screengrab below. This setup will help us point towards the correct config when cloning a repo.



Want to take it a step further? Add a replacement of "git@github.com" every time a repo is cloned based on your current path.

Optimize Organization with .gitconfigs

Speaking of config files; now is the time to create your .gitconfigs for each organization and account. These are simple config files that contain the correct parameters used when cloning a repo, and can be placed in a folder together to optimize it organizationally.



We need to enter some kind of information into each config file. Thankfully, it's only two parts, and doesn't require much time or effort. The first part includes adding the [user] information, and the second part involves replacing the URL when cloning a repo.

```
# gitconfig for personal projects
[user]
  name = Naser Al-Daher
  email = naser@personal.com
  username = naserpersonal

[url "git@personal"]
  insteadOf = git@github.com
```

First, we add user.name, user.email and username to make sure that we are using the correct information. For the replacement portion, make sure that [url "git@<host>"] is correct. <host> is taken from the config file in ~/.ssh/config and checking the name of "Host".

Next, make sure your .gitconfig is setup correctly. We will be removing the user.email from the global config and adding in a pointer towards the correct config using conditional includes.

The .gitconfig should look like this:

As you can see, we have conditional includes that checks your current path and uses a config based on that path. Note that the /i after gitdir makes the path case insensitive.

When using this config file, if you are in the path "~/github/personal", we will use the config "~/github/configs/personal.gitconfig".

Finished setup & Ready to Perform

With all of the initial configurations now out of the way, you are ready to get to work. You will be able to traverse to the correct path and clone a repo without making any changes to the ssh URL. Get the SSH URL from GitHub, and just make sure you are in the correct path before cloning.

WRITTEN BY
Naser Al-Daher

Need an Expert?

Whether you are looking to begin your cloud journey, modernize existing applications, or need help with cloud operations and automations, we can help with every step of your full journey to the cloud.

Contact Us