Quick Tip: How to Work with GitHub and Multiple Accounts

So you have a personal GitHub account; everything is working perfectly. But then, you get a new job, and now need to have the ability to push and pull to multiple accounts. How do you do that? I'll show you how!

Prefer a Screencast?

Choose **720p** for the best picture.

Envato Elements membership. Get unlimited access to a growing library to millions of creative and code assets.

Step 1 - Create a New SSH Key

We need to generate a unique SSH key for our second GitHub account.

```
ssh-keygen -t rsa -C "your-email-address"
```

Be careful that you don't over-write your existing key for your personal account. Instead, when prompted, save the file as `id_rsa_COMPANY`. In my case, I've saved the file to `~/.ssh/id_rsa_nettuts`.

Step 2 - Attach the New Key

Next, login to your second GitHub account, browse to "Account Overview," and attach the new key, within the "SSH Public Keys" section. To retrieve the value of the key that you just created, return to the Terminal, and type: `vim ~/.ssh/id_rsa_COMPANY.pub`. Copy the entire string that is displayed, and paste this into the GitHub textarea. Feel free to give it any title you wish.

Next, because we saved our key with a unique name, we need to tell SSH about it. Within the Terminal, type: `ssh-add ~/.ssh/id_rsa_COMPANY`. If successful, you'll see a response of "Identity Added."

Step 3 - Create a Config File

We've done the bulk of the workload; but now we need a way to specify when we wish to push to our personal account, and when we should instead push to our company account. To do so, let's create a `config` file.

```
touch ~/.ssh/config
vim config
```

If you're not comfortable with Vim, feel free to open it within any editor of your choice. Paste in the following snippet.

```

```

```
#Default GitHub
Host github.com
  HostName github.com
  User git
  IdentityFile ~/.ssh/id_rsa
```

This is the default setup for pushing to our personal GitHub account. Notice that we're able to attach an identity file to the host. Let's add another one for the company account. Directly below the code above, add:

```
Host github-COMPANY
  HostName github.com
  User git
  IdentityFile ~/.ssh/id_rsa_COMPANY
```

This time, rather than setting the host to `github.com`, we've named it as `github-COMPANY`. The difference is that we're now attaching the new identity file that we created previously: `id_rsa_COMPANY`. Save the page and exit!

Step 4 - Try it Out

It's time to see if our efforts were successful. Create a test directory, initialize git, and create your first commit.

```
git init
git commit -am "first commit'
```

Login to your company account, create a new repository, give it a name of "Test," and then return to the Terminal and push your git repo to GitHub.

```
git remote add origin git@github-COMPANY:Company/testir
git push origin master
```

Note that, this time, rather than pushing to `git@github.com`, we're using the custom host that we create in the config file: `git@github-COMPANY`.

Return to GitHub, and you should now see your repository. Remember:

- When pushing to your personal account, proceed as you always have.
- For your company account, make sure that you use `git!github-COMPANY` as the host.

Be sure to refer to the screencast if you need a more visual overview of the steps above!