

How to Configure Multiple GitHub Accounts in Windows 11.

I have been using git for months now and it's been very useful in version controlling my projects. I created a GitHub Account then used my cmd to communicate to the account. Working with Git and GitHub has been very smooth not until I created another GitHub Account for my freelance projects. It says that Git is not permitting me to make changes to this new account because I do not have permission to do so.

This is where I get into digging for solutions and found Truth Seekers YouTube Video <https://www.youtube.com/watch?v=ap56ivm0dhw> which tackles exactly what I needed. I decided I want to write about it just to share another perspective on how I managed to configure two GitHub Accounts in my machine. For context, I have a personal GitHub account which is logged in to my Chrome Browser and another secondary account logged in to my Opera Browser I use in freelancing.

Goals of this article:

1. Tackle how to use SSH to manage multiple accounts.
2. Talk about the format of the configuration file git will use to tell apart from the different accounts.
3. Test the connection of each account.

A. Key Generation

SSH is an encryption that, when used with Git and GitHub, lets us transfer data in a secured way from a local machine to a remote repository. SSH is amazing, it is an abbreviation for Secured Shell. We need a key pair for this to work, a public key and a private key. The public key will be added to a GitHub account then the private key will be stored in our local machine. Once we're making a connection, the machine will check if the private key and the public key matches. If so, we do the data transfer.

To do this, first cd to C:\\Users\\Dexter

(Note: I will use my computer for these examples, you just need to change your path according to your machine and you should be good).

```
cd C:\\Users\\Dexter
ssh-keygen -t rsa -C "testEmail1@gmail.com"
```

-t is the type of encryption

-C is the comment, we want to comment the GitHub Account we want to connect to

Running these two commands will create a private-public key pair but first, you need to specify which folder these files will be saved. It's going to suggest

```
C:\Users\Dexter\.ssh/id_rsa
```

but be cautious because it will override an existing id_rsa file. This time though, we will let this override the file. Next, we will set a passphrase for increased security. This is not mandatory you can set it as blank for no passphrase. What's the difference? Every time you perform git push it will prompt you to enter your passphrase, it will be annoying in the long run but it's still your call. For me, I want extra protection so I will set mine to

```
*****
```

After this, your keys as testEmail1@gmail.com is now saved in the .ssh folder.

```
id_rsa -> private key
```

```
id_rsa.pub -> public key
```

B. Add an SSH key on testEmail1@gmail.com GitHub Account.

GitHub > Settings > SSH and GPG Keys > new SSH Key

Title

Key

For the title, name it the same as your machine. Im using Intel NUC M15 so I named mine that.

For the key, its going to be the text you're going to copy from id_rsa.pub.

C. Add the key to SSH Agent

You need to run this command to activate ssh-agent. Activating ssh-agent will let you run the ssh-add command.

```
start-ssh-agent
```

You can now add the private key by running:



```
ssh-add C:\\Users\\Dexter\\.ssh\\id_rsa
```

A message should be displayed telling you that your identity as testEmail1@gmail.com has been added.

D. Testing first account.

- i. Create a Repo in GitHub
- ii. Create a directory in your local machine.
- iii. Configure the directory. (init, add, commit)
 - a. `git init`
 - b. `git config --global user.name "yourUsername"`
 - c. `git config --global user.email yourEmail`
 - d. `git config --global init.defaultBranch main`
 - e. `git add -all`
 - f. `git commit -m "Initial Commit."`
- iv. Go back to GitHub then copy the SSH URL. Make sure to select SSH instead of HTTPS URL.
- v. In the terminal, push the directory to GitHub.
 - a. `git remote add origin githubSshUrl`
 - b. `git push -u origin main`
 - c. enter passphrase

If this works, you're now ready for the second account. We will just repeat most of the steps earlier.

E. Generate Key

```
ssh-keygen -t rsa -C "secondTest@gmail.com"
```

Then, create the other id_rsa file but instead of id_rsa, we will name it id_rsa_secondary

```
C:\\Users\\Dexter\\.ssh\\id_rsa_secondary
```

I am adding a passphrase for this account as well.

```
*****
```

```
id_rsa_secondary -> private key
```

```
id_rsa_secondary.pub -> public key
```

- F. Add a new SSH key in secondTest@gmail.com GitHub account.
GitHub > Settings > SSH and GPG Keys > new SSH Key



Title

Key

For the title, name it the same as your machine. Im using Intel NUC M15 so I named mine that.

For the key, its going to be the text you're going to copy from id_rsa_secondary.pub.

G. Add key to ssh-agent

If the prompt says: cannot connect to agent when you run the ssh-add command, follow these steps. It's from this stack overflow answer here <https://stackoverflow.com/questions/18683092/how-to-run-ssh-add-on-windows>

Search for "Services" in the start menu.

Find the "Open SSH Authentication Agent" then double click.

Change th StartUp Type from disabled to Automatic Delayed Start.

To confirm that the top-listed path is System32, open cmd then type "where ssh"

You should now be able to use ssh-agent. If not, try restarting the PC and then reopen cmd.

Now, run the command:

```
ssh-add C:\\Users\\Dexter\\.ssh\\id_rsa_secondary
```

The identity as secondTest@gmail.com should now be added.

Now that we have two accounts registered, the computer will be confused when using GitHub, so we need to create a configuration file.

H. Create a configuration file.

The file name is "config" without any extension. It must be created inside the .ssh folder.

```
# MAIN ACCOUNT
```

```
Host github.com
```

```
    HostName github.com
```

```
    User git
```

```
    IdentityFile ~/.ssh/id_rsa
```








```
# SECONDARY ACCOUNT
```

```
Host github-secondary
```

```
    HostName github.com
```

```
    User git
```

```
    IdentityFile ~/.ssh/id_rsa_secondary
```

Name	Date modified	Type	Size
 config	10/14/2023 2:45 PM	File	1 KB
 id_rsa	10/14/2023 12:44 PM	File	3 KB
 id_rsa.pub	10/14/2023 12:44 PM	Microsoft Publishe...	1 KB
 id_rsa_secondary	10/14/2023 1:29 PM	File	3 KB
 id_rsa_secondary.pub	10/14/2023 1:29 PM	Microsoft Publishe...	1 KB
 known_hosts	8/11/2023 12:09 AM	File	1 KB
 known_hosts.old	9/27/2022 8:44 PM	OLD File	1 KB

Now, the computer will be able to tell apart which has access to which account. You should now be able to push and pull repositories on both accounts but with some minor differences.

I. Test the second GitHub Account

- vi. Create a Repo in GitHub
- vii. Create a directory in your local machine.
- viii. Configure the directory. (init, add, commit)
 - a. `git init`
 - b. `git config --global user.name "yourUsername"`



- c. `git config --global user.email yourEmail`
- d. `git config --global init.defaultBranch main`
- e. `git add -all`
- f. `git commit -m "Initial Commit."`
- ix. Go back to GitHub then copy the SSH URL. Make sure to select SSH instead of HTTPS URL. Here, instead of github.com, you will change it to github-secondary.
Ex:
[Dexdev@github.com:testrepo.git](ssh://Dexdev@github.com:testrepo.git)
You will change it to:
[Dexdev@github-secondary:testrepo.git](ssh://Dexdev@github-secondary:testrepo.git)
- x. In the terminal, push the directory to GitHub.
 - a. `git remote add origin githubSshUrl`
 - b. `git push -u origin main`

enter passphrase

Everything should work fine now.

<dev>

</dev>

Blog



Managing Multiple GitHub Accounts from One Computer.