

EdgeXFoundry GitHub Signing, Keys and Commits Setup

In this guide you will have the following information and will assist you in setting up your working PC or Mac or Nix box with your EdgeXFoundry, EdgeXFoundry-Holding are GitHub accounts. You will need to setup at least the first 4 sections involving *Two factor authentication and GPG key creation, GPG signing and GPG key association with your GitHub account.*

- Setting up 2 factor authentication (2FA) with your GitHub account.
- Creating a GPG key for Signing your GitHub commits to your repository.
- Committing to your repository using GPG signatures.
- *Creating an SSH key for SSH repositories. (optional)*
- *Committing to your repository using GPG signatures and an SSH repository. (optional)*

Setting up 2 factor authentication

EdgeXFoundry currently requires that all website interaction, mailing etc.. With the EdgeXFoundry GitHub be setup using 2 factor authentication or 2FA. 2FA is a means of enhanced gathering and ensuring who say you are and whom is doing the work is in fact you. 2FA provides GitHub enhanced logging on your connections and the work you do with GitHub. 2FA will prevent a malicious user from using your login credentials *username and password*, from logging in and corrupting or doing anything malicious on your behalf.

Access using 2FA are currently enforced via the GitHub website only. I'm not aware of any reliance or compliance regarding:

The GitHub API or the GitHub Desktop.

Currently I'm using the (SMS) text message setup.

Inside your Security Settings for your GitHub access account. "This should have been setup during onboarding. Usually you will get an email address associated with your access and you will provide a user name. My information is ecotter@gmail.com and my user name is xmlviking.

You will be prompted to enable your 2 factor via email or programmatically when you first attempt to login.

You will need to setup your 2FA under "personal settings". This should be in the upper right of your logged in access to GitHub.

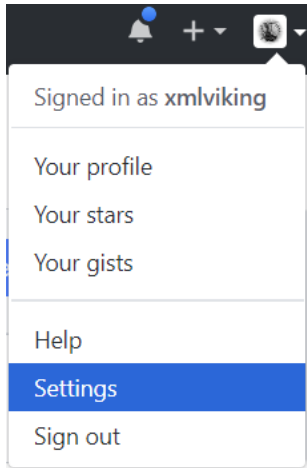


Figure 1 GitHub website authenticated user

Next enable the two-factor authentication and edit the settings

Change your developer options or select the method you want to use. I have selected my primary as SMS message. You will need to enter your phone or device number which will be receiving the 2FA code via SMS.

After you done you should see the afore green check mark in the GitHub > Settings > Security area.

Two-factor authentication



[Save your recovery codes](#) in a safe place. They will allow you to access your account if you ever lose your phone.

Figure 2GitHub User Security settings for 2FA

When I log into EdgeXFoundry online via the GitHub website I'm prompted for my 2FA (SMS) text code.

Example:

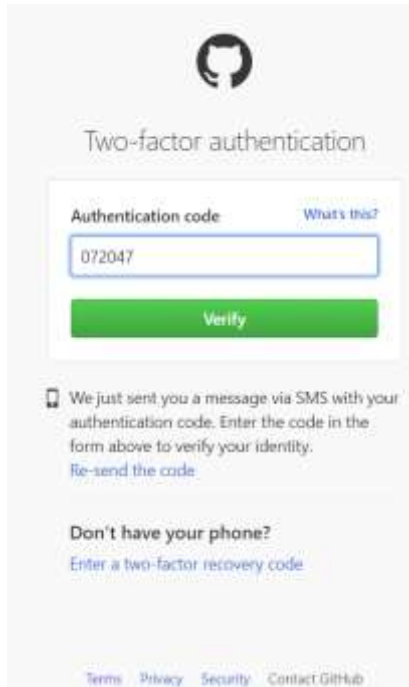


Figure 3 SMS (Text) 2FA token entry

After you enter your code you will be logged into the GitHub website 😊

Note the Security history will do the enhanced logging I mentioned reveals your activity etc.

Security history

This is a security log of important events involving your account.

user.login – Originated from [redacted]	12 minutes ago
user.two_factor_challenge_success – Originated from [redacted]	12 minutes ago
user.two_factor_requested – Originated from [redacted]	13 minutes ago

Figure 4 Security history log for user activity

Note this has nothing to do with your access to uploading code via the github.com client on your pc. This is specifically for logging into the website. We will be using SSH or GPG for our check-ins to determine we are whom we say we are and the code is coming from a reliable source. This will be accomplished in the next section. Setting up SSH for use with your GitHub account on your development machine.

Setting up Signed Commits with GitHub and GPG

EdgeXFoundry currently is requiring signed commits. We are using the GPG key for signing commits and tags¹. In this section I will cover setting up signing with the Mac. The referenced content will allow you to see the Windows or Linux flavors as well. Signing requires you to create a new GPG key associated with your machine. You should have a flavor of the GPG command line tool installed already. You need version 2.1.7 or greater to work correctly with the present release of GitHub signing.

Open at terminal on your MacBook prob. “[Command] + [T]”.

Now enter or paste the following text at the prompt.

```
$ gpg --full-generate-key
```

Note: if this does not work you can use a derivation of the key like `gpg --default-new-key-algo rsa4096 --gen-key`.

When prompted, specify the key type you want to use. I used RSA. If you don't select one and just hit [Enter] you will select the RSA key type by default.

Now enter the desired key length. I used 4096 as this is what GitHub uses predominately.

Now you will be asked how long the key will last or when it will expire. I just hit enter which is 0 or default. This will set your key to never expire.

Next at the prompt it asks you confirm your selection. If this is correctly enter (y/N). I selected yes because I'm good at hitting the [Enter] key.

Next you should see something like the following:

```
GnuPG needs to construct a user ID to identify your key.  
Real name:
```

I used Eric Cotter but you could use any user Id you want, and then hit [Enter]

You will then be prompted to enter the email address you want to associate your sign with.

```
(Real name: Eric Cotter)  
Email address:
```

I used my GitHub email address which I set my account up with ecotter@gmail.com .

And then asks you to enter a comment. I selected “MacBook signing”

```
(Real name: Eric Cotter)  
(Email address: ecotter@gmail.com)  
Comment:
```

¹ GitHub Help - <https://help.github.com/articles/generating-a-new-gpg-key/#platform-mac>

You are then prompted to verify you're information is correct and if so hit the "O" ok and [Enter].

```
You selected this USER-ID
"Eric Cotter (MacBook signing) <ecotter@gmail.com>"
Change (N)ame, (C)omment, (E)mail, or (O)kay/(Q)uit?
```

You will then be prompted to enter a passphrase:

A MacPro window will popup. You need to enter a passphrase or just [Enter] and then repeat [Enter].

Now we will need to print out to the terminal the secret-keys.

```
$ gpg --list-secret-keys --keyid-format LONG
```

From the terminal list of text...select something like the following. Highlighted in yellow.

```
$gpg --list-secret-keys --keyid-format LONG
/Users/hubot/.gnupg/secring.gpg
-----
sec  4096R/3AA5C34371567BD2 2016-03-10 [expires: 2017-03-10]
uid                               Hubot
ssb  4096R/42B317FD4BA89E7A 2016-03-10
```

Now after copying the 3AA5C34371567BD2 (example of your sec key).

Paste that portion after the `$gpg --armor --export <past your key portion here>` over the example number.

```
$ gpg --armor --export 3AA5C34371567BD2
```

Copy your GPG key, start where you see the following

```
-----BEGIN PGP PUBLIC KEY BLOCK----- and ending with -----END PGP PUBLIC KEY
BLOCK-----.
```

Adding your GPG key into your GitHub Account

Now you will need to log into your GitHub account associated with your EdgeXFoundry, and EdgeXFoundry-Holding account on GitHub. Note if you have previously setup 2FA (Two Factor Authentication) you will need to enter your SMS (Text) authentication code.

You should then select your profile dropdown and select the “Settings” ²

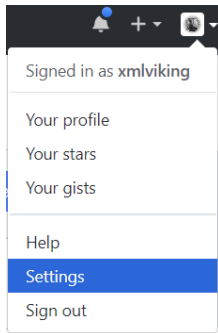


Figure 5 Authenticated User -> Settings

Now in the Personal Setting screen select the “SSH and GPG keys” option on the left side.

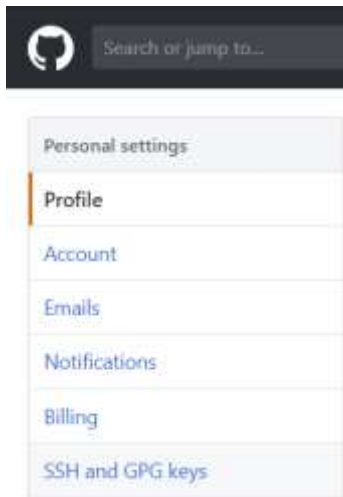


Figure 6 Personal Settings, SSH and GPG keys


Now you should see a screen which has the following section:

² Adding GPG key w/GitHub Account - <https://help.github.com/articles/adding-a-new-gpg-key-to-your-github-account/>

GPG keys

New GPG key

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.



Email address: ecotter@gmail.com
Key ID: 08437
Subkeys: 56D5
Added on Jun 15, 2018

GPG

Delete

Yours might be empty so just click the [New GPG key] green button.

GPG keys / Add new

Key

Begins with "-----BEGIN PGP PUBLIC KEY BLOCK-----"

Add GPG key

Now paste the copied GPG rsa public key string you had from your MacBook terminal and paste it into the "Key" section. Note the grayed-out text "Begins with ---- BEGIN".

After clicking the [Add GPG key] you should be prompted to enter your GitHub password.

You should now see your GPG key associated with your account.

Signing commits from your repository using GPG

Now this is how you sign your commits using your new associated GPG key in your GitHub account.

In your GitHub repository directory open up a MacBook terminal [command] + [T].

At the prompt you can add your changes like normal.

```
$ git add mychangedfile.go
```

Now you can sign your commit to your local repository like the following.

```
$ git commit origin -S -m "my cool GPG signed commit"
```

After you hit the [Enter] key you will be prompted to enter your passphrase which you entered when you setup your GPG key previously. If you elected to not use one and just hit the [Enter] key that is all you must do. Hit [Enter].

A git push is the same just use something like

```
$ git push
```

You should be able to see your commit on your branch of your repository.

And that is now you sign your commits using your GPG key.

NOTE: if you implemented your repository using SSH you will need to add the additional -s or "little s" to your signature. So it would look like this for your commits.

```
$ git commit origin -S -s -m "my cool GPG signed commit"
```

This will use the SSH key and your GPG signing. NOTE You do not need to do this if you did not implement SSH repositories locally with and GitHub SSH key.

Setting up SSH for use with Github.com and your machine

Configuring your GitHub account to use SSH keys will require you to do the following.

This example is for the Mac Pro.

Generating a new SSH key (Mac Pro)

1. Open Git Bash
2. Paste the text below, substituting your GitHub email address. This should be the email address you setup with your GitHub account not necessarily your Dell email account.

```
ssh-keygen -t rsa -b 4096 -C "ecotter@gmail.com"
```

Here the RSA is the key type and the number 4096 is the length of the key. The -C option followed by your email address is the certificate email address this SSH will be associated with.

3. When prompted by your system “Enter a file in which to save the key”, just press [Enter]. This will dump the key in the default location which associated with your user space.

Example: (/Users/ericcotton/.ssh/id_rsa) NOTE: this is directly associated with your users account so if you screw up the email address you can overwrite your rsa cert.

4. At the prompt, type a secure passphrase. Note: you don’t have to enter one but it’s more secure. Using one will ensure it’s you are typing in those GitHub commits and pushes to EdgeXFoundry! If you don’t want to use one you can simply hit [Enter].

```
Enter a passphrase (empty for no passphrase): [Type a passphrase]
Enter same passphrase again: [Repeat the same passphrase]
```

Adding Your SSH key to the ssh-agent.

Here we need to ensure our SSH Agent is running currently on your mac.

Open a terminal window. **Use the Command key + T.**

At the prompt do the following:

```
~$ eval $(ssh-agent -s)
Agent pid 44550
```

Your pid “process id” will be different but you should see one 🤖

Now add your SSH Private key to the ssh-agent. Again, if you want to create a new RSA key do it like afore prior to this setup. Otherwise do the following at the terminal prompt.

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

Now this is the certificate that we created previously using the ssh-keygen utility. And again, it is referring the file location relative to your Users directory.

Add the SSH key to your GitHub account

Now that you have “generated your ssh key” and “associated your ssh key with your ssh-agent” we will not export and copy your id_rsa.pub or “rsa public key” into your GitHub account.

Open a terminal in your Mac. “[Command] + [T]”

```
$ clip < ~/.ssh/id_rsa.pub
```

```
# This will copy the contents of the id_rsa.pub file into your mac pro clipboard
```

Log into your GitHub website account. Follow the above instruction and enter your SMS (token) into the prompt by GitHub if you have 2FA (two factor authentication) enabled. Now go into the Settings section once logged in.

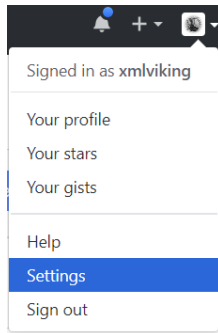


Figure 7 GitHub website Signed in User Settings

In the user settings select SSH and GPG keys.

Click the New SSH key or Add SSH key if you have not created one previously.



Here I have mine already imported from my Macbook Pro but you would click the [New SSH key.]

You will see a large space. You will need to enter a title for your SSH key and then paste the id_rsa.pub file contents we previously put in the clipboard on your macbook pro.

SSH keys / Add new

Title

Key

Begins with 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Add SSH key

The contents should be quite large and your view after you save it should look similar to the above version of my SSH keys section.

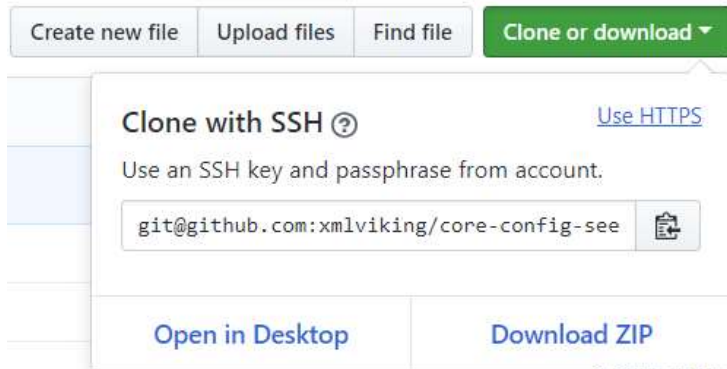
You are now setup for SSH. And now will have to use the “-S” option when committing your code to your local repo.

```
git commit -s -S -m "Signed commit w/GPG and using SSH"
```

This command will commit your changes which you previously added. It uses the -S or signed option see *Signing our Commits with GPG*. And now uses the -s or ssh option when we commit our code to our local repository. This will add our key info to our local repo so when we do a git push origin later it will contain our key information. We will be prompted by GitHub to enter our passphrase we created our RSA key with earlier. Again, if you did not use a passphrase we can simply hit [Enter].

Setting up your local GitHub Repository using SSH

When we clone a repository from EdgeXFoundry or EdgeXFoundry-Holding we will need to use the SSH clone link and not the HTTPS clone link.



Here I'm using core-config-seed-go repository as an example. You can see I selected the "Use SSH" and we can see the "Clone with SSH" title and the clone link below.

Assuming you're in the correct file location for your new SSH enabled repo do the following. In your terminal window you would use the following.

```
$ git clone git@github.com:xmlviking/core-config-seed-go.git
```

You will be prompted to enter your passphrase. Enter it if you have elected to use one. Otherwise just hit the [Enter] key.

Your repo is now setup with SSH enabled and when you do anything with this repository your ssh-agent will be involved in "committing, push etc.." with GitHub.