This guide will show you how you can configure your development setup to automatically sign git commits using GPG and link your GPG key with GitHub.

Prerequisites

Make sure you follow the Prerequisites, How To Contribute (microsoft/vscode) guide.

Install Tools

Windows 10

Install Gpg4win and make sure Git uses that GPG version:

```
git config --global gpg.program "C:\Program Files (x86)\GnuPG\bin\gpg.exe"
```

macOS Big Sur

Install the necessary tools and configure GPG:

```
brew install gpg2 gnupg pinentry-mac
mkdir -p ~/.gnupg
echo "pinentry-program $(brew --prefix)/bin/pinentry-mac" >> ~/.gnupg/gpg-agent.conf
echo "use-agent" >> ~/.gnupg/gpg.conf
echo 'export GPG_TTY=$(tty)' >> .bash_profile # replace with .zprofile if using ZSH
```

Restart your machine. Yes, really.

Create Signing Key

There are two options: generate or copy an existing key.

Generate Key

> ***Windows:*** *make sure you're using a Command Prompt or PowerShell instead of the Git Bash shell. Make sure the right GPG is being used:*
>
> ```
> C:\Users\User> where gpg
> C:\Program Files (x86)\GnuPG\bin\gpg.exe
> ```

Run:

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

With the following options:

- Kind of key: `RSA and RSA (default)`
- Keysize: `4096`
- Expiration: `0` (does not expire)
- Real Name: use your real name

- Email address: use your Microsoft email address
- Comment: `Key for signing commits for Microsoft`
- **Passphrase**: Pick a long, secure passphrase, which you'll easily remember.

In the following example, `DF536B632D7967F9` is the **key ID**:

```
$ gpg --list-secret-keys --keyid-format LONG
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
/home/joao/.gnupg/pubring.kbx
-----------------------------
sec   rsa4096/DF536B632D7967F9 2021-04-07 [SC]
      1D0FC7C0350BB570143C934FDF536B632D7967F9
uid                 [ultimate] Joao Moreno (Key for signing commits for Microsoft)
<joao.moreno@microsoft.com>
ssb   rsa4096/41FC60C87D442095 2021-04-07 [E]
```

You can use your key ID to get your **public key**:

```
$ gpg --armor --export DF536B632D7967F9
-----BEGIN PGP PUBLIC KEY BLOCK-----
...
-----END PGP PUBLIC KEY BLOCK-----
```

Copy Key

Alternatively, you can export a existing key from another machine, by replacing `KEYID` with your key ID:

```
gpg --export-secret-keys -a KEYID > my-key.asc
```

Then, copy it into your machine and import it:

```
gpg --import my-key.asc
```

Finally, set its trust level to `ultimate`:

```
$ gpg --edit-key KEYID
gpg> trust
Your decision? 5
```

Configure GitHub

Follow the Adding a new GPG key to your GitHub account guide to give GitHub your **public key**.

Configure Git

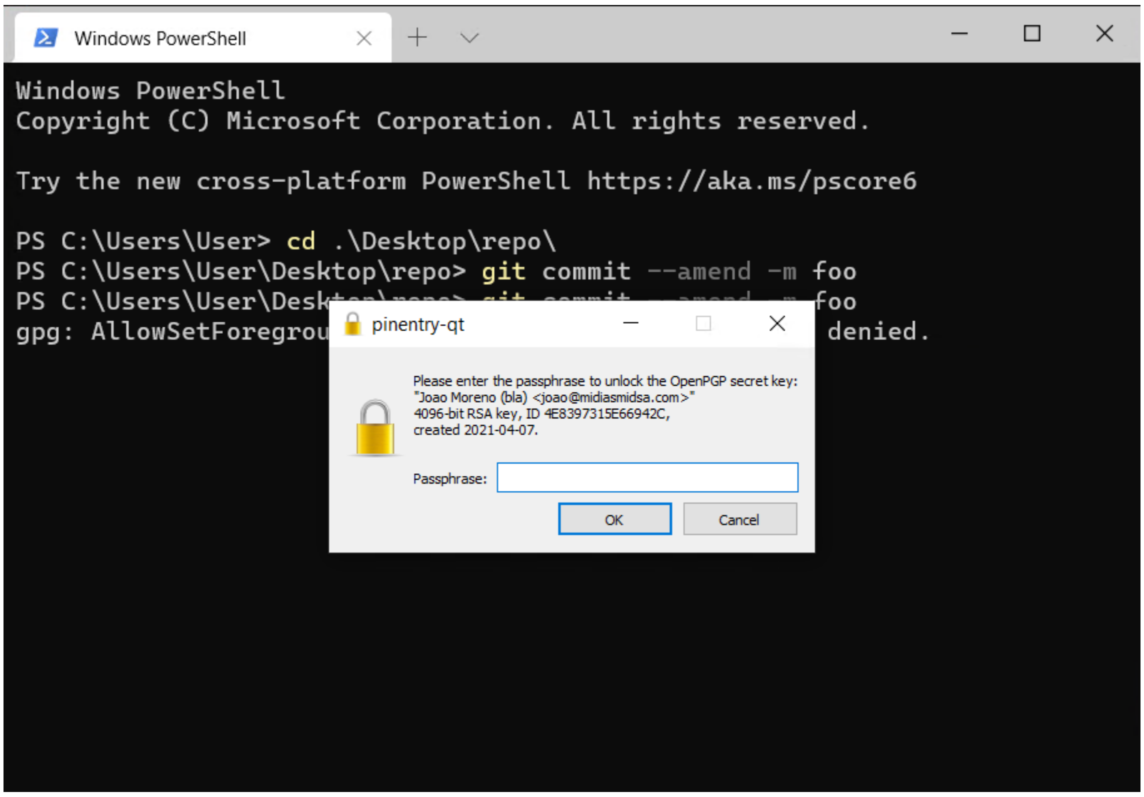
Run the following, replacing `KEYID` with your key ID:

```
git config --global user.signingkey KEYID
git config --global commit.gpgsign true
```

Git will now sign all commits by default. Signing requires access to your GPG key, which requires the passphrase. Follow the respective platform specific steps below to decrease pain.

Windows

Create a dummy commit on a sample local repository. You should see a prompt for your key's passphrase:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\User> cd .\Desktop\repo\
PS C:\Users\User\Desktop\repo> git commit --amend -m foo
PS C:\Users\User\Desk                                    foo
gpg: AllowSetForegrou                                    denied.
```

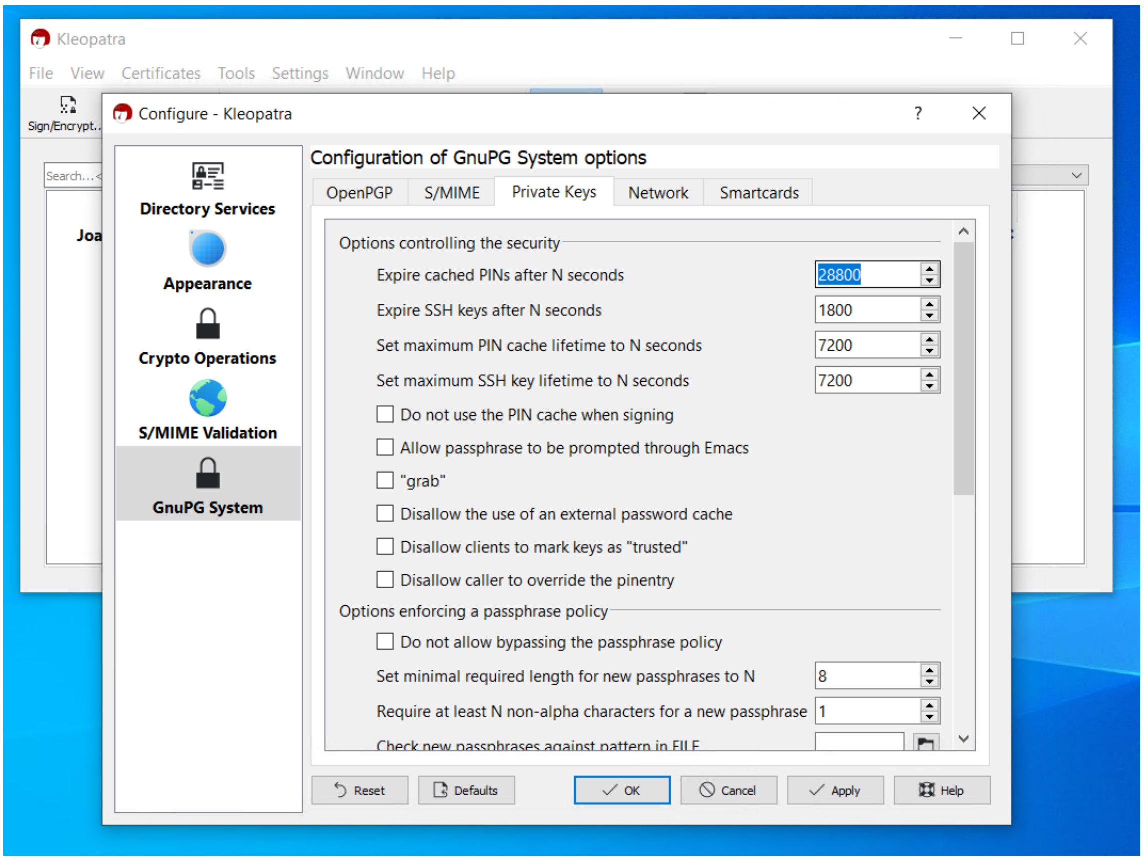
pinentry-qt

Please enter the passphrase to unlock the OpenPGP secret key:
"Joao Moreno (bla) <joao@midiasmidsa.com>"
4096-bit RSA key, ID 4E8397315E66942C,
created 2021-04-07.

Passphrase:

Enter your GPG key **passphrase** and hit OK. In order to avoid typing the passphrase on every commit, you can make GPG remember it for 8 hours or until the next reboot:

1. Launch Kleopatra
2. Click `Settings > Configure Kleopatra...`
3. Navigate to `GnuPG System > Private Keys`
4. Change `Expire cached PINs after N seconds` to `28800`



Configuration of GnuPG System options

Options controlling the security

Expire cached PINs after N seconds 28800

Expire SSH keys after N seconds 1800

Set maximum PIN cache lifetime to N seconds 7200

Set maximum SSH key lifetime to N seconds 7200

☐ Do not use the PIN cache when signing

☐ Allow passphrase to be prompted through Emacs

☐ "grab"

☐ Disallow the use of an external password cache

☐ Disallow clients to mark keys as "trusted"

☐ Disallow caller to override the pinentry

Options enforcing a passphrase policy

☐ Do not allow bypassing the passphrase policy

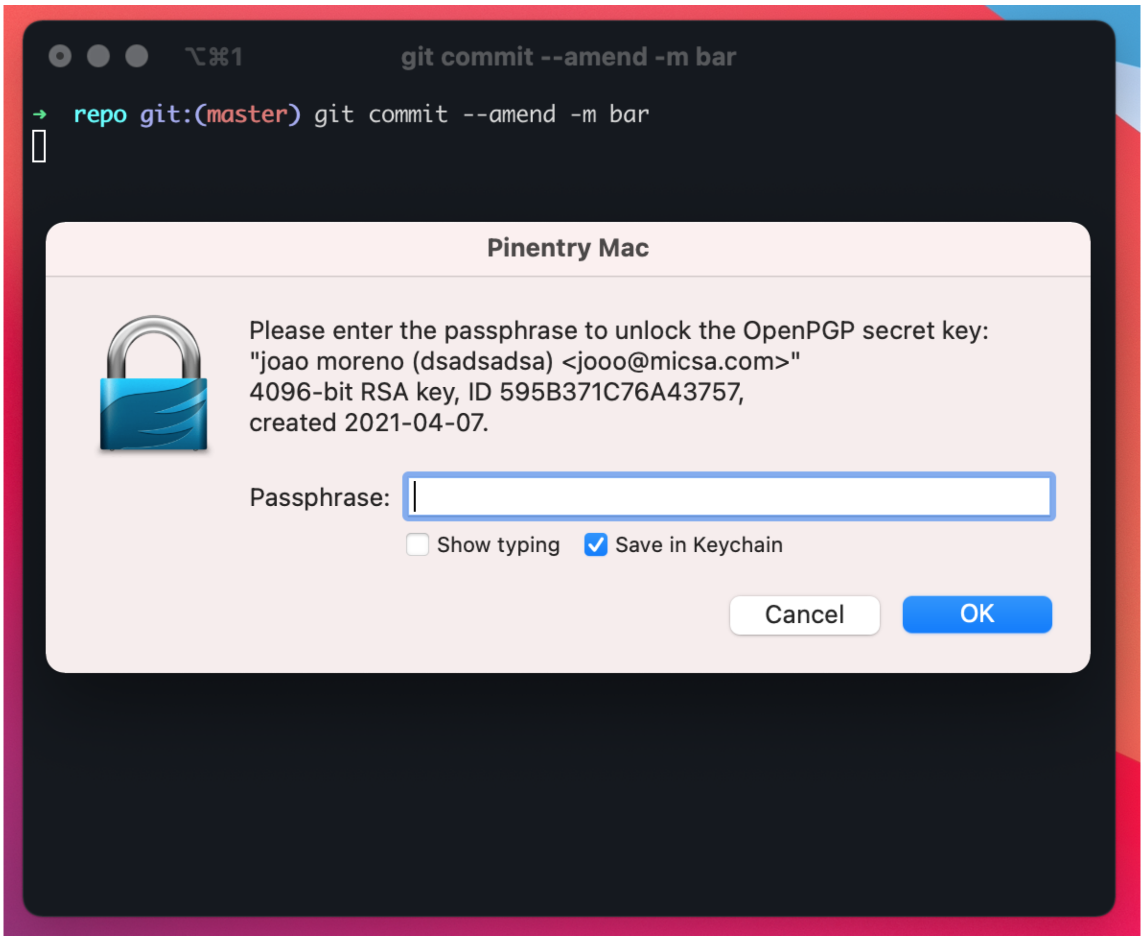
Set minimal required length for new passphrases to N 8

Require at least N non-alpha characters for a new passphrase 1

Check new passphrases against pattern in FILE

macOS

Create a dummy commit on a sample local repository. You should see a prompt for your key's passphrase:



```
⌥⌘1                git commit --amend -m bar
➜ repo git:(master) git commit --amend -m bar
```

Pinentry Mac

Please enter the passphrase to unlock the OpenPGP secret key:
"joao moreno (dsadsadsa) <jooo@micsa.com>"
4096-bit RSA key, ID 595B371C76A43757,
created 2021-04-07.

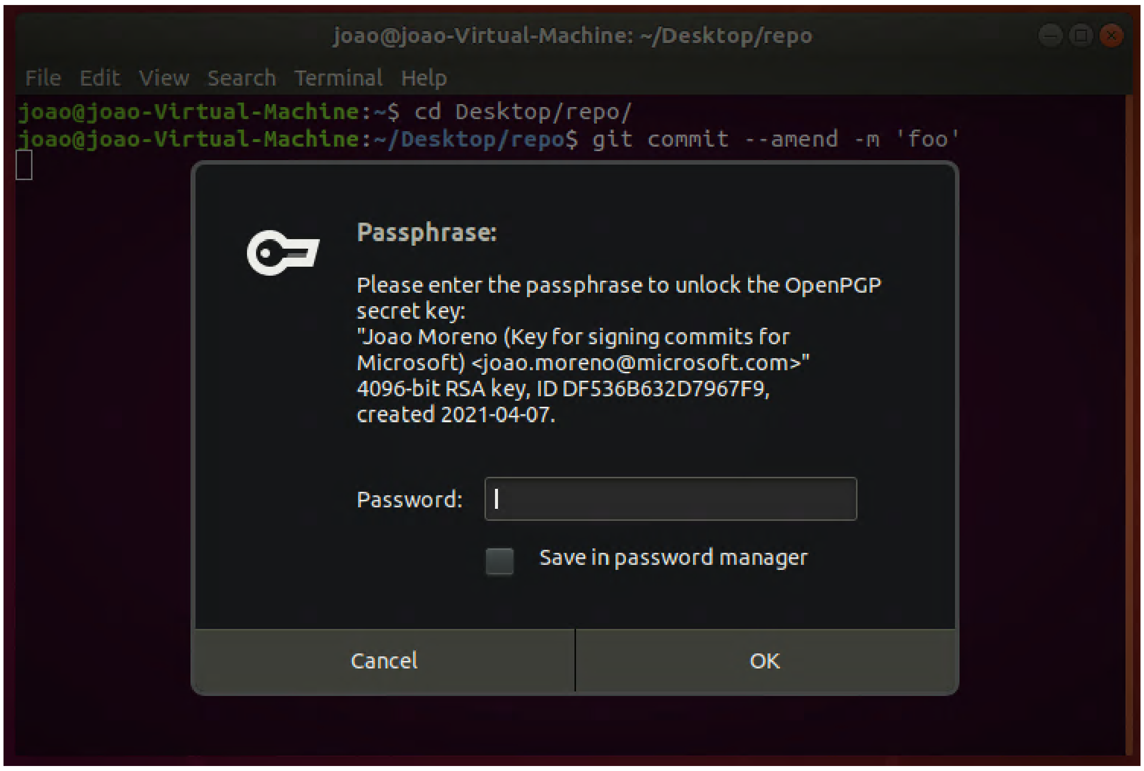
Passphrase:

[ ] Show typing [✓] Save in Keychain

In order to avoid typing the passphrase on every commit, just select `Save in Keychain`. That should've been the last time you typed the passphrase.

Ubuntu 18.04

Create a dummy commit on a sample local repository. You should see a prompt for your key's passphrase:


```
joao@joao-Virtual-Machine:~$ cd Desktop/repo/
joao@joao-Virtual-Machine:~/Desktop/repo$ git commit --amend -m 'foo'
```

Passphrase:

Please enter the passphrase to unlock the OpenPGP secret key:
"Joao Moreno (Key for signing commits for Microsoft) <joao.moreno@microsoft.com>"
4096-bit RSA key, ID DF536B632D7967F9,
created 2021-04-07.

Password:

Save in password manager

Cancel OK

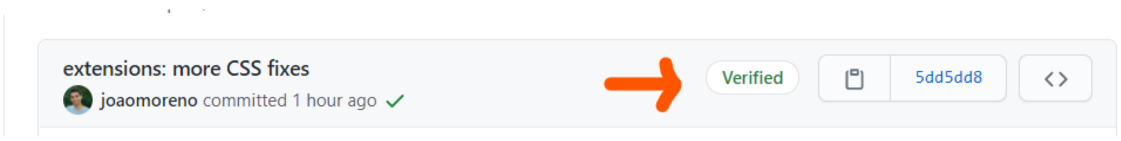
Enter your GPG key **passphrase** and hit OK. In order to avoid typing the passphrase on every commit, you can make GPG remember it for 8 hours or until the next reboot:

```
mkdir -p ~/.gnupg
echo "default-cache-ttl 28800" >> ~/.gnupg/gpg-agent.conf
```

> *Note: Apparently, it's a moot point to select `Save in password manager`, because problems 🙄. Every time you reboot your machine you'll always be asked for your passphrase. If you select `Save in password manager`, you'll also be asked for your Keyring password every time you reboot, so it's best not to do it.*

Verify Setup

Make sure that if you push a signed commit to GitHub it appears as `Verified`:


extensions: more CSS fixes
joaomoreno committed 1 hour ago ✓
Verified 5dd5dd8

FAQ

How important is this key?

You should guard this key as well as you guard your SSH key, maybe even better. While an SSH key can easily be replaced by another, since it only affects the login to Github, a GPG key will forever be associated with your commits.

This means that as soon as you remove the GPG key from GitHub, because say you replaced it with another, the commits won't appear `Verified` any longer.

How can I best store this key?

One of the best ways to store a backup or even to use the key directly is to use a YubiKey. You can store a copy of your key in a YubiKey, serving as a backup. Or you can even store the only copy of your key in it and always require your YubiKey when creating commits. This definitely increases security but with the obvious added cost of always needing your YubiKey around. Learn more.

What about Codespaces?

Codespaces will use a different GPG key, but it will indeed sign all your commits automatically, so they appear as verified on GitHub. Learn more.

Reference

- GitHub: About commit signature verification
- GitHub: Signing commits
- GnuPG: GnomeKeyring
- macOS: Methods of Signing with GPG
- How (and why) to sign Git commits