Telling Git about your signing key

In this article

To sign commits locally, you need to inform Git that there's a GPG, SSH, or X.509 key you'd like to use.

Mac Windows Linux

Telling Git about your GPG key

If you're using a GPG key that matches your committer identity and your verified email address associated with your account on your GitHub Enterprise Server instance, then you can begin signing commits and signing tags.

> If you don't have a GPG key that matches your committer identity, you need to associate an email with an existing key. For more information, see "Associating an email with your GPG key".

If you have multiple GPG keys, you need to tell Git which one to use.

1. Open TerminalTerminalGit Bash.
2. If you have previously configured Git to use a different key format when signing with `--gpg-sign`, unset this configuration so the default format of `openpgp` will be used.

   ```
   git config --global --unset gpg.format
   ```

3. Use the `gpg --list-secret-keys --keyid-format=long` command to list the long form of the GPG keys for which you have both a public and private key. A private key is required for signing commits or tags.

   Shell

   ```
   gpg --list-secret-keys --keyid-format=long
   ```

   > **Note:** Some GPG installations on Linux may require you to use `gpg2 --list-keys --keyid-format LONG` to view a list of your existing keys instead. In this case you will also need to configure Git to use `gpg2` by running `git config --global gpg.program gpg2`.

4. From the list of GPG keys, copy the long form of the GPG key ID you'd like to use. In

this example, the GPG key ID is `3AA5C34371567BD2`:

Shell

```
$ gpg --list-secret-keys --keyid-format=long
/Users/hubot/.gnupg/secring.gpg
------------------------------------
sec   4096R/3AA5C34371567BD2 2016-03-10 [expires: 2017-03-10]
uid                          Hubot <hubot@example.com>
ssb   4096R/4BB6D45482678BE3 2016-03-10
```

5. To set your primary GPG signing key in Git, paste the text below, substituting in the GPG primary key ID you'd like to use. In this example, the GPG key ID is `3AA5C34371567BD2`:

   ```
   git config --global user.signingkey 3AA5C34371567BD2
   ```

   Alternatively, when setting a subkey include the `!` suffix. In this example, the GPG subkey ID is `4BB6D45482678BE3`:

   ```
   git config --global user.signingkey 4BB6D45482678BE3!
   ```

6. Optionally, to configure Git to sign all commits by default, enter the following command:

   ```
   git config --global commit.gpgsign true
   ```

   For more information, see "Signing commits."

7. If you aren't using the GPG suite, run the following command in the `zsh` shell to add the GPG key to your `.zshrc` file, if it exists, or your `.zprofile` file:

   ```
   $ if [ -r ~/.zshrc ]; then echo -e '\nexport GPG_TTY=$(tty)' >> ~/.zshrc; \
     else echo -e '\nexport GPG_TTY=$(tty)' >> ~/.zprofile; fi
   ```

   Alternatively, if you use the `bash` shell, run this command:

   ```
   $ if [ -r ~/.bash_profile ]; then echo -e '\nexport GPG_TTY=$(tty)' >>
   ~/.bash_profile; \
     else echo -e '\nexport GPG_TTY=$(tty)' >> ~/.profile; fi
   ```

8. Optionally, to prompt you to enter a PIN or passphrase when required, install `pinentry-mac`. For example, using Homebrew:

   ```
   brew install pinentry-mac
   echo "pinentry-program $(which pinentry-mac)" >> ~/.gnupg/gpg-agent.conf
   killall gpg-agent
   ```

If you're using a GPG key that matches your committer identity and your verified email address associated with your account on your GitHub Enterprise Server instance, then you can begin signing commits and signing tags.

> If you don't have a GPG key that matches your committer identity, you need to associate an email with an existing key. For more information, see "Associating an email with your GPG key".

If you have multiple GPG keys, you need to tell Git which one to use.

1. Open TerminalTerminalGit Bash.
2. If you have previously configured Git to use a different key format when signing with `--gpg-sign`, unset this configuration so the default format of `openpgp` will be used.

   ```
   git config --global --unset gpg.format
   ```

3. Use the `gpg --list-secret-keys --keyid-format=long` command to list the long form of the GPG keys for which you have both a public and private key. A private key is required for signing commits or tags.

   Shell

   ```
   gpg --list-secret-keys --keyid-format=long
   ```

   Note: Some GPG installations on Linux may require you to use `gpg2 --list-keys --keyid-format LONG` to view a list of your existing keys instead. In this case you will also need to configure Git to use `gpg2` by running `git config --global gpg.program gpg2`.

4. From the list of GPG keys, copy the long form of the GPG key ID you'd like to use. In this example, the GPG key ID is `3AA5C34371567BD2`:

   Shell

   ```
   $ gpg --list-secret-keys --keyid-format=long
   /Users/hubot/.gnupg/secring.gpg
   ------------------------------------
   sec   4096R/3AA5C34371567BD2 2016-03-10 [expires: 2017-03-10]
   uid                          Hubot <hubot@example.com>
   ssb   4096R/4BB6D45482678BE3 2016-03-10
   ```

5. To set your primary GPG signing key in Git, paste the text below, substituting in the GPG primary key ID you'd like to use. In this example, the GPG key ID is `3AA5C34371567BD2`:

   ```
   git config --global user.signingkey 3AA5C34371567BD2
   ```

   Alternatively, when setting a subkey include the `!` suffix. In this example, the GPG subkey ID is `4BB6D45482678BE3`:

   ```
   git config --global user.signingkey 4BB6D45482678BE3!
   ```

6. Optionally, to configure Git to sign all commits by default, enter the following command:

   ```
   git config --global commit.gpgsign true
   ```

   For more information, see "Signing commits."

If you're using a GPG key that matches your committer identity and your verified email address associated with your account on your GitHub Enterprise Server instance, then you can begin signing commits and signing tags.

> If you don't have a GPG key that matches your committer identity, you need to associate an email with an existing key. For more information, see "Associating an email with your GPG key".

If you have multiple GPG keys, you need to tell Git which one to use.

1. Open TerminalTerminalGit Bash.
2. If you have previously configured Git to use a different key format when signing with `--gpg-sign`, unset this configuration so the default format of `openpgp` will be used.

   ```
   git config --global --unset gpg.format
   ```

3. Use the `gpg --list-secret-keys --keyid-format=long` command to list the long form of the GPG keys for which you have both a public and private key. A private key is required for signing commits or tags.

   Shell

   ```
   gpg --list-secret-keys --keyid-format=long
   ```

   > **Note:** Some GPG installations on Linux may require you to use `gpg2 --list-keys --keyid-format LONG` to view a list of your existing keys instead. In this case you will also need to configure Git to use `gpg2` by running `git config --global gpg.program gpg2`.

4. From the list of GPG keys, copy the long form of the GPG key ID you'd like to use. In this example, the GPG key ID is `3AA5C34371567BD2`:

   Shell

   ```
   $ gpg --list-secret-keys --keyid-format=long
   /Users/hubot/.gnupg/secring.gpg
   ------------------------------------
   sec   4096R/3AA5C34371567BD2 2016-03-10 [expires: 2017-03-10]
   uid                          Hubot <hubot@example.com>
   ssb   4096R/4BB6D45482678BE3 2016-03-10
   ```

5. To set your primary GPG signing key in Git, paste the text below, substituting in the GPG primary key ID you'd like to use. In this example, the GPG key ID is `3AA5C34371567BD2`:

   ```
   git config --global user.signingkey 3AA5C34371567BD2
   ```

   Alternatively, when setting a subkey include the `!` suffix. In this example, the GPG subkey ID is `4BB6D45482678BE3`:

   ```
   git config --global user.signingkey 4BB6D45482678BE3!
   ```

6. Optionally, to configure Git to sign all commits by default, enter the following command:

   ```
   git config --global commit.gpgsign true
   ```

   For more information, see "Signing commits."

7. To add your GPG key to your `.bashrc` startup file, run the following command:

   ```
   [ -f ~/.bashrc ] && echo -e '\nexport GPG_TTY=$(tty)' >> ~/.bashrc
   ```

Telling Git about your SSH key

You can use an existing SSH key to sign commits and tags, or generate a new one specifically for signing. For more information, see "Generating a new SSH key and adding it to the ssh-agent."

> **Note:** SSH signature verification is available in Git 2.34 or later. To update your version of Git, see the Git website.

1. Open TerminalTerminalGit Bash.
2. Configure Git to use SSH to sign commits and tags:

   ```
   git config --global gpg.format ssh
   ```

3. To set your SSH signing key in Git, paste the text below, substituting **/PATH/TO/KEY.PUB** with the path to the public key you'd like to use.

   ```
   $ git config --global user.signingkey /PATH/TO/.SSH/KEY.PUB
   ```

Telling Git about your X.509 key

You can use smimesign to sign commits and tags using S/MIME.

> **Note:** S/MIME signature verification is available in Git 2.19 or later. To update your version of Git, see the Git website.

1. Install smimesign.
2. Open TerminalTerminalGit Bash.
3. Configure Git to use S/MIME to sign commits and tags. In Git 2.19 or later, use the `git config gpg.x509.program` and `git config gpg.format` commands:
   - To use S/MIME to sign for all repositories:

     ```
     git config --global gpg.x509.program smimesign
     git config --global gpg.format x509
     ```

- To use S/MIME to sign for a single repository:

```
cd PATH-TO-REPOSITORY
git config --local gpg.x509.program smimesign
git config --local gpg.format x509
```

In Git 2.18 or earlier, use the `git config gpg.program` command:

- To use S/MIME to sign for all repositories:

```
git config --global gpg.program smimesign
```

- To use S/MIME to sign for a single repository:

```
cd  PATH-TO-REPOSITORY
git config --local gpg.program smimesign
```

If you're using an X.509 key that matches your committer identity, you can begin signing commits and tags.

4. If you're not using an X.509 key that matches your committer identity, list X.509 keys for which you have both a certificate and private key using the `smimesign --list-keys` command.

```
smimesign --list-keys
```

5. From the list of X.509 keys, copy the certificate ID of the X.509 key you'd like to use. In this example, the certificate ID is `0ff455a2708394633e4bb2f88002e3cd80cbd76f`:

```
$ smimesign --list-keys
             ID: 0ff455a2708394633e4bb2f88002e3cd80cbd76f
            S/N: a2dfa7e8c9c4d1616f1009c988bb70f
      Algorithm: SHA256-RSA
       Validity: 2017-11-22 00:00:00 +0000 UTC - 2020-11-22 12:00:00 +0000
UTC
         Issuer: CN=DigiCert SHA2 Assured ID
CA,OU=www.digicert.com,O=DigiCert Inc,C=US
        Subject: CN=Octocat,O=GitHub\, Inc.,L=San
Francisco,ST=California,C=US
         Emails: octocat@github.com
```

6. To set your X.509 signing key in Git, paste the text below, substituting in the certificate ID you copied earlier.

- To use your X.509 key to sign for all repositories:

```
git config --global user.signingkey
0ff455a2708394633e4bb2f88002e3cd80cbd76f
```

- To use your X.509 key to sign for a single repository:

```
cd  PATH-TO-REPOSITORY
git config --local user.signingkey
0ff455a2708394633e4bb2f88002e3cd80cbd76f
```

Further reading

- "Adding a new SSH key to your GitHub account."
- "Signing commits"
- "Signing tags"