Generating a new GPG key

In this article

If you don't have an existing GPG key, you can generate a new GPG key to use for signing commits and tags.

Mac Windows Linux

Supported GPG key algorithms

GitHub Enterprise Server supports several GPG key algorithms. If you try to add a key generated with an unsupported algorithm, you may encounter an error.

- RSA
- ElGamal
- DSA
- ECDH
- ECDSA
- EdDSA

Generating a GPG key

Note: Before generating a new GPG key, make sure you've verified your email address. If you haven't verified your email address, you won't be able to sign commits and tags with GPG.

1. Download and install the GPG command line tools for your operating system. We generally recommend installing the latest version for your operating system.
2. Open TerminalTerminalGit Bash.
3. Generate a GPG key pair. Since there are multiple versions of GPG, you may need to consult the relevant *man page* to find the appropriate key generation command.
   - If you are on version 2.1.17 or greater, paste the text below to generate a GPG key pair.

     Shell

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

   - If you are not on version 2.1.17 or greater, the `gpg --full-generate-key` command doesn't work. Paste the text below and skip to step 6.

     Shell

```
gpg --default-new-key-algo rsa4096 --gen-key
```

4. At the prompt, specify the kind of key you want, or press `Enter` to accept the default.
5. At the prompt, specify the key size you want, or press `Enter` to accept the default.
6. Enter the length of time the key should be valid. Press `Enter` to specify the default selection, indicating that the key doesn't expire. Unless you require an expiration date, we recommend accepting this default.
7. Verify that your selections are correct.
8. Enter your user ID information.

   Note: When asked to enter your email address, ensure that you enter the verified email address for your GitHub account.

9. Type a secure passphrase.
10. 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`.

11. 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
    ```

12. Paste the text below, substituting in the GPG key ID you'd like to use. In this example, the GPG key ID is `3AA5C34371567BD2`:

    ```
    gpg --armor --export 3AA5C34371567BD2
    # Prints the GPG key ID, in ASCII armor format
    ```

13. Copy your GPG key, beginning with `-----BEGIN PGP PUBLIC KEY BLOCK-----` and ending with `-----END PGP PUBLIC KEY BLOCK-----`.

14. Add the GPG key to your GitHub account.

Further reading

- "Checking for existing GPG keys"
- "Adding a GPG key to your GitHub account"
- "Telling Git about your signing key"
- "Associating an email with your GPG key"
- "Signing commits"
- "Signing tags"