

Troubleshooting GPG verification for GitHub Codespaces

In this article

Errors after disabling GPG verification

Errors caused by conflicting Git configuration

Errors in the VS Code "Source Control" view

Further reading

This article provides troubleshooting advice for errors related to signing your commits in codespaces.

If you enable GPG verification, GitHub Codespaces automatically signs your commits in codespaces that you create from selected repositories. For more information, see "[Managing GPG verification for GitHub Codespaces](#)."

Once you enable GPG verification, it will automatically take effect in any new codespaces you create from the relevant repositories. To have GPG verification take effect in an existing active codespace, you will need to stop and restart the codespace. For more information, see "[Stopping and starting a codespace](#)."

If GitHub Codespaces fails to sign a commit, you may see the error message `gpg failed to sign the data` in the command line or in a Visual Studio Code pop-up window.

The following sections of this article provide troubleshooting advice for common causes of this error.

- If GPG verification has previously been enabled in your settings for GitHub Codespaces, and you have recently disabled GPG verification or removed a repository from your list of trusted repositories, Git may still be trying to sign your commits. For more information, see "[Errors after disabling GPG verification](#)."
- If GPG verification is enabled for the codespace, you may have overridden the Git configuration required to sign your commits. For more information, see "[Errors caused by conflicting Git configuration](#)."
- If GPG verification is disabled for the codespace, and you're encountering the error when trying to commit from the "Source Control" view in VS Code, this may be because of your VS Code settings. For more information, see "[Errors in the VS Code "Source Control" view](#)."

Errors after disabling GPG verification [↗](#)

When you enable GPG verification, GitHub Codespaces signs all the commits you make in codespaces by default. It does this by setting the `commit.gpgsign` Git configuration value to `true`.

If you have disabled GPG verification, and are working in an existing codespace, then this value will still be set to `true`. This means that GitHub Codespaces will try to sign your commits, but will be unable to do so, because you have disabled the GPG verification setting.

To keep making regular, unsigned commits in your codespace, reset `commit.gpgsign` to the default value of `false` by entering the following command in the terminal.

Shell

```
git config --unset commit.gpgsign
```

To check that the value has been correctly removed from your configuration, you can enter `git config --list`. You should not see a value for `commit.gpgsign` in the list.

Errors caused by conflicting Git configuration [↗](#)

To automatically sign your commits, GitHub Codespaces sets certain Git configuration values in your codespace. If you override the values set by GitHub Codespaces, you may be unable to sign your commits.

You may be inadvertently overriding these values if you have linked GitHub Codespaces with a dotfiles repository that contains Git configuration files. For more information about using dotfiles with GitHub Codespaces, see "[Personalizing GitHub Codespaces for your account](#)."

Checking for conflicting configuration [↗](#)

To sign your commits with GPG, GitHub Codespaces automatically sets the following Git configuration values at the system level.

Configuration setting	Required value
<code>user.name</code>	Must match the full name set on your GitHub profile
<code>credential.helper</code>	Must be set to <code>/.codespaces/bin/gitcredential_github.sh</code>
<code>gpg.program</code>	Must be set to <code>/.codespaces/bin/gh-gpgsign</code>

To check that these values are set correctly in a codespace, you can use the `git config --list --show-origin` command. Because GitHub Codespaces sets this configuration at the system level, the required configuration settings should come from `/usr/local/etc/gitconfig`.

```
$ git config --list --show-origin
file:/usr/local/etc/gitconfig
credential.helper=/.codespaces/bin/gitcredential_github.sh
file:/usr/local/etc/gitconfig user.name=Mona Lisa
file:/usr/local/etc/gitconfig gpg.program=/.codespaces/bin/gh-gpgsign
```

In addition to the values listed above, you may run into errors if the dotfiles used in your codespaces contain any of the following values.

- The `user.signingkey` Git config value
- The `commit.gpgsign` Git config value
- A manually set `GITHUB_TOKEN`

Removing conflicting configuration [↗](#)

If you want to keep automatic GPG verification for GitHub Codespaces enabled, you will

need to remove any conflicting configuration from the dotfiles used in your codespaces.

For example, if the global `.gitconfig` file on your local machine contains a `gpg.program` value, and you have pushed this file to a dotfiles repository that is linked with GitHub Codespaces, then you may want to remove `gpg.program` from this file and set it at the system level on your local machine instead.

Note: Any changes to your dotfiles repository will apply to new codespaces you create, but not to your existing codespaces.

- 1 On your local machine, open a terminal.
- 2 To remove the conflicting value from `~/.gitconfig` (Mac/Linux) or `C:\Users\YOUR-USER\.gitconfig` (Windows), use the `git config --global --unset` command.

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

- 3 Push the change to your dotfiles repository on GitHub.
- 4 Optionally, to keep your local configuration, set the value again in a Git configuration file that you do not push to your dotfiles repository.

For example, you can use the `--system` flag to set the configuration in the system-level file at `PATH/etc/gitconfig`, where `PATH` is the directory in which Git is installed on your system.

```
git config --system gpg.program gpg2
```

Alternatively, if your dotfiles repository contains an installation script in a recognized file such as `install.sh`, you can use the `$CODESPACES` environment variable to add conditional logic, such as only setting `gpg.program` when you are not in a codespace. In the following example, `-z "$CODESPACES"` returns `true` if you are not in a codespace.


Shell

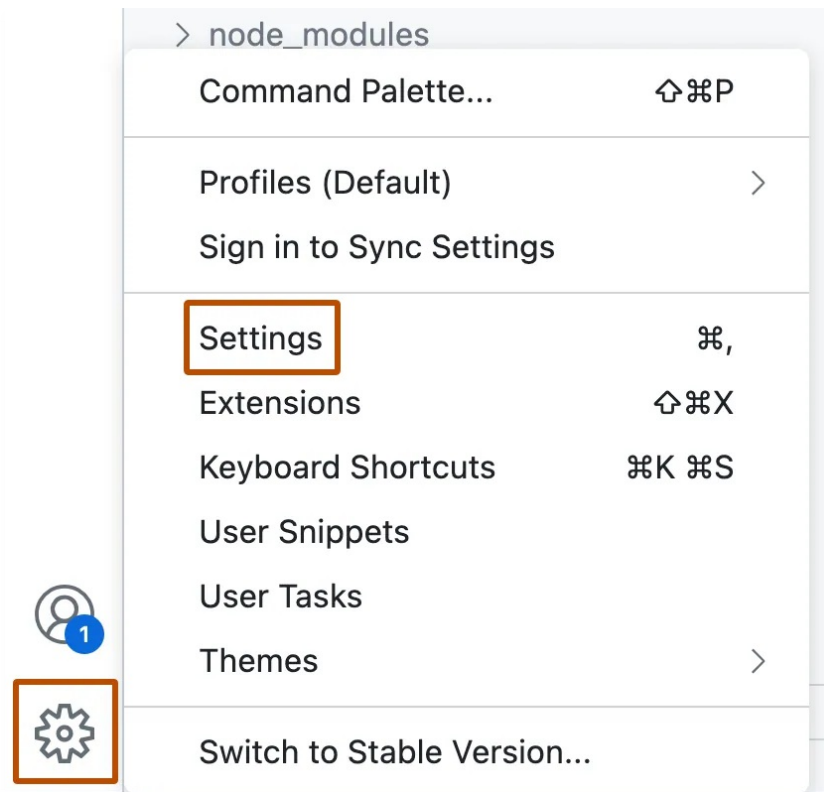


```
if [ -z "$CODESPACES" ]; then
  git config --global gpg.program gpg2
fi
```

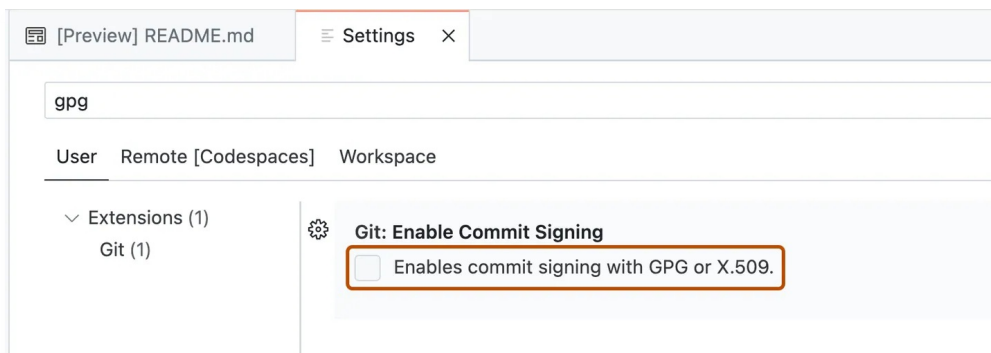
Errors in the VS Code "Source Control" view [↗](#)

If GPG verification is disabled in your settings for GitHub Codespaces, or the repository you created the codespace from isn't in your list of trusted repositories, then Git should not attempt to sign your commits. If you encounter a signing error when trying to commit from the "Source Control" view in VS Code, you should check the VS Code settings in your codespace.

- 1 In the lower-left corner of the window, select , then click **Settings**.



- 2 On the "User" tab, in the search bar, search for "gpg".
- 3 Verify that the "Enables commit signing with GPG or X.509" setting is deselected.



If you find this setting is enabled, you should either deselect the checkbox to stop VS Code trying to sign your commits, or you should enable GPG verification for the repository you're working in so your commits can be signed successfully.

If you change your VS Code settings, you must ensure Settings Sync is enabled if you want to share your changes with other codespaces you create. You should only turn on Settings Sync in a codespace created from a repository you trust. For more information, see "[Personalizing GitHub Codespaces for your account](#)."

Further reading

- "[About commit signature verification](#)"
- [git config](#) in the official Git documentation

Legal

© 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Status](#) [Pricing](#) [Expert services](#) [Blog](#)

