About commit signature verification

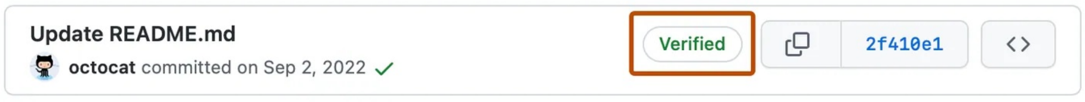
In this article

About commit signature verification
GPG commit signature verification
SSH commit signature verification
S/MIME commit signature verification
Further reading

Using GPG, SSH, or S/MIME, you can sign tags and commits locally. These tags or commits are marked as verified on GitHub Enterprise Server so other people can be confident that the changes come from a trusted source.

About commit signature verification

You can sign commits and tags locally, to give other people confidence about the origin of a change you have made. If a commit or tag has a GPG, SSH, or S/MIME signature that is cryptographically verifiable, GitHub Enterprise Server marks the commit or tag "Verified."


Update README.md
octocat committed on Sep 2, 2022 ✓

If a commit or tag has a signature that can't be verified, GitHub Enterprise Server marks the commit or tag "Unverified."

For most individual users, GPG or SSH will be the best choice for signing commits. S/MIME signatures are usually required in the context of a larger organization. SSH signatures are the simplest to generate. You can even upload your existing authentication key to GitHub Enterprise Server to also use as a signing key. Generating a GPG signing key is more involved than generating an SSH key, but GPG has features that SSH does not. A GPG key can expire or be revoked when no longer used. GitHub Enterprise Server shows commits that were signed with such a key as "Verified" unless the key was marked as compromised. SSH keys don't have this capability.

Signature verification for rebase and merge

When using the **Rebase and Merge** option on a pull request, it's important to note that the commits in the head branch are added to the base branch without commit signature verification. When you use this option, GitHub creates a modified commit, using the data and content of the original commit. This means that GitHub didn't truly create this commit, and can't therefore sign it as a generic system user. GitHub doesn't have access to the committer's private signing keys, so it can't sign the commit on the user's behalf.

A workaround for this is to rebase and merge locally, and then push the changes to the pull request's base branch.

For more information, see "About merge methods on GitHub."

Repository administrators can enforce required commit signing on a branch to block all commits that are not signed and verified. For more information, see "About protected branches."

You can check the verification status of your signed commits or tags on GitHub Enterprise Server and view why your commit signatures might be unverified. For more information, see "Checking your commit and tag signature verification status."

If a site administrator has enabled web commit signing, GitHub Enterprise Server will automatically use GPG to sign commits you make using the web interface. Commits signed by GitHub Enterprise Server will have a verified status. You can verify the signature locally using the public key available at `https://HOSTNAME/web-flow.gpg`. For more information, see "Configuring web commit signing."

GPG commit signature verification

You can use GPG to sign commits with a GPG key that you generate yourself.

GitHub Enterprise Server uses OpenPGP libraries to confirm that your locally signed commits and tags are cryptographically verifiable against a public key you have added to your account on your GitHub Enterprise Server instance.

To sign commits using GPG and have those commits verified on GitHub Enterprise Server, follow these steps:

1. Check for existing GPG keys
2. Generate a new GPG key
3. Add a GPG key to your GitHub account
4. Tell Git about your signing key
5. Sign commits
6. Sign tags

SSH commit signature verification

You can use SSH to sign commits with an SSH key that you generate yourself. For more information, see the Git reference documentation for `user.Signingkey`. If you already use an SSH key to authenticate with GitHub Enterprise Server, you can also upload that same key again for use as a signing key. There's no limit on the number of signing keys you can add to your account.

GitHub Enterprise Server uses ssh_data, an open source Ruby library, to confirm that your locally signed commits and tags are cryptographically verifiable against a public key you have added to your account on your GitHub Enterprise Server instance.

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

To sign commits using SSH and have those commits verified on GitHub Enterprise Server, follow these steps:

1. Check for existing SSH keys

2. Generate a new SSH key
3. Add a SSH signing key to your GitHub account
4. Tell Git about your signing key
5. Sign commits
6. Sign tags

S/MIME commit signature verification

You can use S/MIME to sign commits with an X.509 key issued by your organization.

GitHub Enterprise Server uses the Debian ca-certificates package, the same trust store used by Mozilla browsers, to confirm that your locally signed commits and tags are cryptographically verifiable against a public key in a trusted root certificate.

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

To sign commits using S/MIME and have those commits verified on GitHub Enterprise Server, follow these steps:

1. Tell Git about your signing key
2. Sign commits
3. Sign tags

You don't need to upload your public key to GitHub Enterprise Server.

Further reading

- "Signing commits"
- "Signing tags"
- "Troubleshooting commit signature verification"