Managing multiple accounts

In this article

If you use one workstation to contribute to projects for more than one account on GitHub.com, you can modify your Git configuration to simplify the contribution process.

Mac Windows Linux

About management of multiple accounts

In some cases, you may need to use multiple accounts on GitHub.com. For example, you may have a personal account for open source contributions, and your employer may also create and manage a user account for you within an enterprise.

You cannot use your managed user account to contribute to public projects on GitHub.com, so you must contribute to those resources using your personal account. For more information, see "About Enterprise Managed Users" in the GitHub Enterprise Cloud documentation.

If you want to use one workstation to contribute from both accounts, you can simplify contribution with Git by using a mixture of protocols to access repository data, or by using credentials on a per-repository basis.

> **Warning**: Be mindful when you use one workstation to contribute to two separate accounts. Management of two or more accounts can increase the chance of mistakenly leaking internal code to the public.

If you aren't required to use a managed user account, GitHub recommends that you use one personal account for all your work on GitHub.com. With a single personal account, you can contribute to a combination of personal, open source, or professional projects using one identity. Other people can invite the account to contribute to both individual repositories and repositories owned by an organization, and the account can be a member of multiple organizations or enterprises.

Contributing to two accounts using HTTPS and SSH

If you contribute with two accounts from one workstation, you can access repositories by using a different protocol and credentials for each account.

Git can use either the HTTPS or SSH protocol to access and update data in repositories on GitHub.com. The protocol you use to clone a repository determines which credentials your workstation will use to authenticate when you access the repository. With this approach to account management, you store the credentials for one account to use for HTTPS connections and upload an SSH key to the other account to use for SSH connections.

You can find both the HTTPS or an SSH URLs for cloning a repository on GitHub. For more information, see "Cloning a repository."

For more information about the use of SSH to access repositories on GitHub, see "Connecting to GitHub with SSH."

Contributing to multiple accounts using HTTPS and personal access tokens

Alternatively, if you want to use the HTTPS protocol for both accounts, you can use different personal access tokens for each account by configuring Git to store different credentials for each repository.

1. Open Terminal.
2. To confirm your use of a credential manager, enter the following command and note the output.

   Shell

   ```
   git config --get credential.helper
   ```

3. If the output confirms that you're using a credential manager, clear the stored credentials for the credential manager.

   - If the output doesn't include the name of a credential manager, there is no credential manager configured, and you can proceed to the next step.
   - If the output is `osxkeychain`, you're using the macOS keychain. To clear the credentials, enter the following command.

     Shell

     ```
     git credential-osxkeychain erase https://github.com
     ```

   - If the output is `manager` (or `manager-core` in previous versions), you're using Git Credential Manager. To clear the credentials, run the following command.

     Shell

     ```
     echo "protocol=https\nhost=github.com" | git credential-manager erase
     ```

4. To configure Git to cache credentials for the full remote URL of each repository you access on GitHub, enter the following command.

   Shell

   ```
   git config --global credential.https://github.com.useHttpPath true
   ```

5. For each of your accounts, create a dedicated personal access token (classic) with

`repo` scope. Or, for each of your accounts and for each organization that you are a member of, create a fine-grained personal access token that can access the desired repositories and that has read and write permissions on repository contents. For more information, see "Managing your personal access tokens."

6. The first time that you use Git to clone a repository or access data in a repository that you've already cloned, Git will request credentials. Provide the personal access token for the account with access to the repository.

   Git will cache the personal access token based on the full remote URL of the repository, and you'll be able to access and write repository data on GitHub.com using the correct account.

1. Open Git Bash.

2. To confirm your use of a credential manager, enter the following command and note the output.

Shell

```
git config --get credential.helper
```

3. If the output confirms that you're using a credential manager, clear the stored credentials for the credential manager.

   - If the output doesn't include the name of a credential manager, there is no credential manager configured, and you can proceed to the next step.
   - If the output is `manager` (or `manager-core` in previous versions), you're using Git Credential Manager. To clear the credentials, run the following command.

Shell

```
echo "protocol=https`nhost=github.com" | git credential-manager erase
```

   - If the output is `wincred`, you're using the Windows Credential Manager. To clear the credentials, enter the following command.

     Shell

     ```
     cmdkey /delete:LegacyGeneric:target=git:https://github.com
     ```

4. To configure Git to cache credentials for the full remote URL of each repository you access on GitHub, enter the following command.

Shell

```
git config --global credential.https://github.com.useHttpPath true
```

5. For each of your accounts, create a dedicated personal access token (classic) with `repo` scope. Or, for each of your accounts and for each organization that you are a member of, create a fine-grained personal access token that can access the desired repositories and that has read and write permissions on repository contents. For more information, see "Managing your personal access tokens."

6. The first time that you use Git to clone a repository or access data in a repository that you've already cloned, Git will request credentials. Provide the personal access token for the account with access to the repository.

   Git will cache the personal access token based on the full remote URL of the repository, and you'll be able to access and write repository data on GitHub.com using the correct account.

1. Open Terminal.

2. To confirm your use of a credential manager, enter the following command and note the output.

   Shell

   ```
   git config --get credential.helper
   ```

3. If the output confirms that you're using a credential manager, clear the stored credentials for the credential manager.

   - If the output doesn't include the name of a credential manager, there is no credential manager configured, and you can proceed to the next step.
   - If the output is `manager` (or `manager-core` in previous versions), you're using Git Credential Manager. To clear the credentials, run the following command.

   Shell

   ```
   echo "protocol=https\nhost=github.com" | git credential-manager erase
   ```

4. To configure Git to cache credentials for the full remote URL of each repository you access on GitHub, enter the following command.

   Shell

   ```
   git config --global credential.https://github.com.useHttpPath true
   ```

5. For each of your accounts, create a dedicated personal access token (classic) with `repo` scope. Or, for each of your accounts and for each organization that you are a member of, create a fine-grained personal access token that can access the desired repositories and that has read and write permissions on repository contents. For more information, see "Managing your personal access tokens."

6. The first time that you use Git to clone a repository or access data in a repository that you've already cloned, Git will request credentials. Provide the personal access token for the account with access to the repository.

   Git will cache the personal access token based on the full remote URL of the repository, and you'll be able to access and write repository data on GitHub.com using the correct account.

Contributing to multiple accounts using SSH and `GIT_SSH_COMMAND`

If you want to use the SSH protocol for both accounts, you can use different SSH keys for each account. For more information about using SSH, see "Connecting to GitHub with SSH."

To use a different SSH key for different repositories that you clone to your workstation, you must write a shell wrapper function for Git operations. The function should perform the following steps.

1. Determine the repository's full name with owner, using a command such as `git config --get remote.origin.url`.
2. Choose the correct SSH key for authentication.
3. Modify `GIT_SSH_COMMAND` accordingly. For more information about `GIT_SSH_COMMAND`, see Environment Variables in the Git documentation.

For example, the following command sets the `GIT_SSH_COMMAND` environment variable to specify an SSH command that uses the private key file at ***PATH/TO/KEY/FILE*** for authentication to clone the repository named OWNER/REPOSITORY on GitHub.com.

```
GIT_SSH_COMMAND='ssh -i PATH/TO/KEY/FILE -o IdentitiesOnly=yes' git clone git@githu
```

Legal

Terms Privacy Status Pricing Expert services Blog