Using passphrase-protected SSH keys with SSH Git URLs

When you use Git to access a private repository over SSH, Git uses an SSH client to establish a secure connection with the server. While establishing the connection, Git uses your configured SSH key during the SSH handshaking phase. During this phase, the SSH client needs to be able to read your key. However, if you encrypted your key with a passphrase, the SSH client can't use the key directly. In this case, you're prompted to type in the passphrase in the terminal. After you enter the correct passphrase, the SSH connection completes and the Git command runs using that connection.

When the Unity Package Manager fetches packages using Git URLs, there's no interface for you to enter credentials requested by the SSH client. As a result, if you protected your SSH key file with a passphrase, the SSH client fails to establish the connection and Git reports an error. To solve this, an authentication agent for SSH must be running and loaded with the SSH key, so that the SSH client can use it without requiring its passphrase.

The method varies, depending on your operating system and the SSH client that you use:

- If you use Windows 10 or later and its built-in OpenSSH client, refer to Loading SSH keys automatically on Windows (OpenSSH).
- If you use Windows and PuTTY and its authentication agent (Pageant), refer to Loading SSH keys automatically on Windows (PuTTY).
- If you use macOS, refer to Loading SSH keys automatically on macOS.

Additional resources

- Using private repositories with HTTPS Git URLs