CloudSavvy IT

What is SSH Agent Forwarding and How Do You Use It?

ANTHONY HEDDINGS
MAY 7, 2020, 11:30AM EDT

Funtap / Shutterstock

SSH agent forwarding allows you to use your private, local SSH key remotely without worrying about leaving confidential data on the server you're working with. It's built into `ssh`, and is easy to set up and use.

What Is an SSH Agent?

Your public SSH key is like your username or identity, and you can share it with everybody. Your private SSH key is like a password, and is saved locally on your computer. But, this is like storing your passwords on a sticky note—anyone can view them if they have access to it. So, for security, SSH will ask you for a passphrase when you generate your keys (hopefully you didn't skip that step) and it will use that passphrase to encrypt and decrypt your private key.

However, this means you'll have to enter your passphrase every time you need to use your private key, which will get annoying. To manage this, most SSH implementations will use an *agent*, which keeps your decrypted key in memory. This means you'll only need to unlock it once, and it will persist until you restart, letting you log into your servers securely without a passphrase prompt.

What is SSH Agent Forwarding?

SSH agent *forwarding* is like going another layer deeper. For example, imagine you're connecting to a remote server, and you want to `git pull` some code that you're storing on Github. You want to use SSH authentication for Github, but you don't want your private keys on that remote server, only on your machine.

To solve this problem, you can open your local SSH agent to the remote server, allowing it to act as you while you're connected. This doesn't send your private keys over the internet, not even while they're encrypted; it just lets a remote server access your local SSH agent and verify your identity.

It works like this: you ask your remote server to pull some code from Github, and Github says "who are you?" to the server. Usually the server would consult its own `id_rsa` files to answer, but instead it will forward the question to your local machine. Your local machine answers the question and sends the response (which does not include your private key) to the server, which forwards it back to Github. Github doesn't care that your local machine answered the question, it just sees that it's been answered, and lets you connect.

How to Enable SSH Agent Forwarding

On Mac and Linux, SSH agent forwarding is built into `ssh`, and the `ssh-agent` process is launched automatically. All you'll have to do is make sure your keys are added to `ssh-agent` and configure `ssh` to use forwarding.

Add Keys to ssh-agent

You can use the utility `ssh-add` to add keys to your local agent. Assuming your private key is stored in `id_rsa`, you can run:

```
ssh-add ~/.ssh/id_rsa
```

You can also manually paste in the key rather than using `id_rsa`. Check that the key is added properly with:

```
ssh-add -L
```

If it is, it should spit out your key.

Add Keys on macOS

On macOS, you will instead need to run:

```
ssh-add -K ~/.ssh/id_rsa
```

The `-K` flag will store the key in the macOS Keychain, which is necessary for it to remember your keys through reboots.

Allow Forwarding in Your Client's Config

Open up your `~/.ssh/config` file on your local machine, or make a new one if it's empty. We'll set a new rule to make sure agent forwarding is enabled for this server's domain:

```
Host example
```

```
    ForwardAgent yes
```

You should replace `example` with your servers domain name or IP address. You can use the wildcard * for the host, but then you'll be forwarding access to your private keys to every server you connect to, which is probably not what you want.

Depending on your operating system, you may also have config files at `/etc/ssh/ssh_config` for macOS or `/etc/ssh_config` for Ubuntu. These files may override the user config file at `~/.ssh/config`, so make sure nothing is conflicting. Lines that start with # are commented out, and have no effect.

You can also manually enable agent forwarding for any domain by using `ssh -A user@host`, which will bypass all config files. If you want an easy method for forwarding without touching config, you can add `alias ssh="ssh -A"` to your bash settings, but this is the same as using a wildcard host, so we don't recommend it for anything security-focused.

Test SSH Forwarding

If you don't have two servers on hand, the easiest way to test if SSH forwarding is working is to add your public key from your local machine to your Github profile and try to SSH from a remote server:

```
ssh git@github.com
```

If it worked, you should see your username, and you should be able to push and pull code from a repo without ever putting private keys on the server.

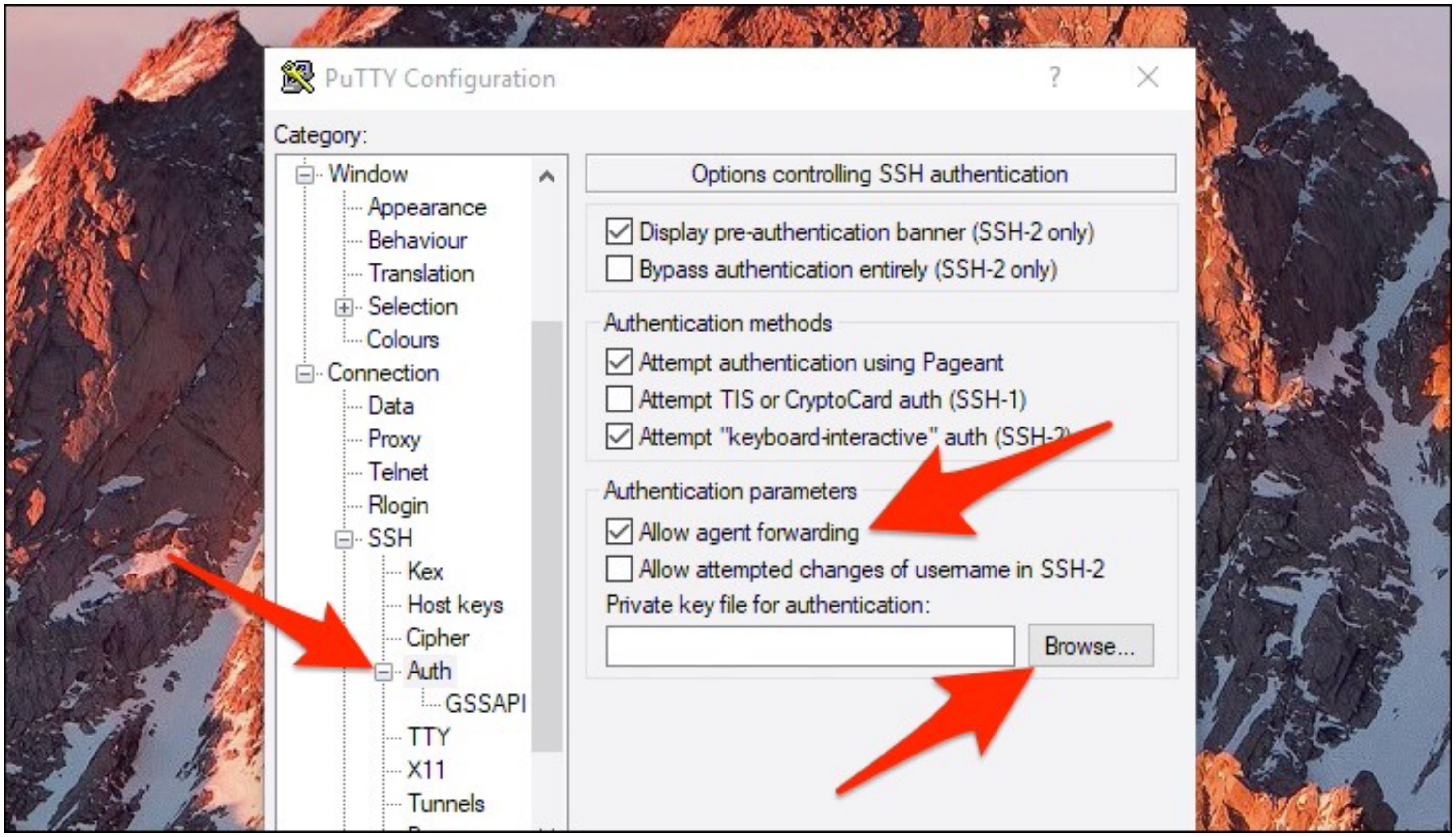
Setup SSH Forwarding for Windows Clients

Since Windows isn't a Unix operating system, setup will vary depending on how exactly you're running `ssh` in the first place.

If you're using the Linux Subsystem for Windows, which lets you run bash on Windows, the setup will be the same as on Linux or macOS, since it's fully virtualizing a Linux distro to run the command line.

If you're using Git Bash, the setup is the same as on Linux, but you'll need to manually start `ssh-agent` when you launch the shell, which you can do with a startup script in `.bashrc`.

If you're using PuTTY, setup is quite simple. From the configuration, go to Connection > SSH > Auth and enable "Allow agent forwarding."



Options controlling SSH authentication

- [✓] Display pre-authentication banner (SSH-2 only)
- [ ] Bypass authentication entirely (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

- [✓] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:

You can also add your private key file from the same pane. PuTTY will handle the SSH agent for you, so you don't have to mess around with any config files.

What to Do if SSH Forwarding Isn't Working

Make sure you actually have SSH keys in the first place; if you don't, you can run `ssh-keygen`, which will place your private key in `~/.ssh/id_rsa` and your public key in `~/.ssh/id_rsa.pub`.

Verify that your SSH keys are working properly with regular auth, and add them to `ssh-agent`. You can add keys with `ssh-add`.

The `ssh-agent` process also needs to be running. On macOS and Linux, it should start automatically, but you can verify that it is running with:

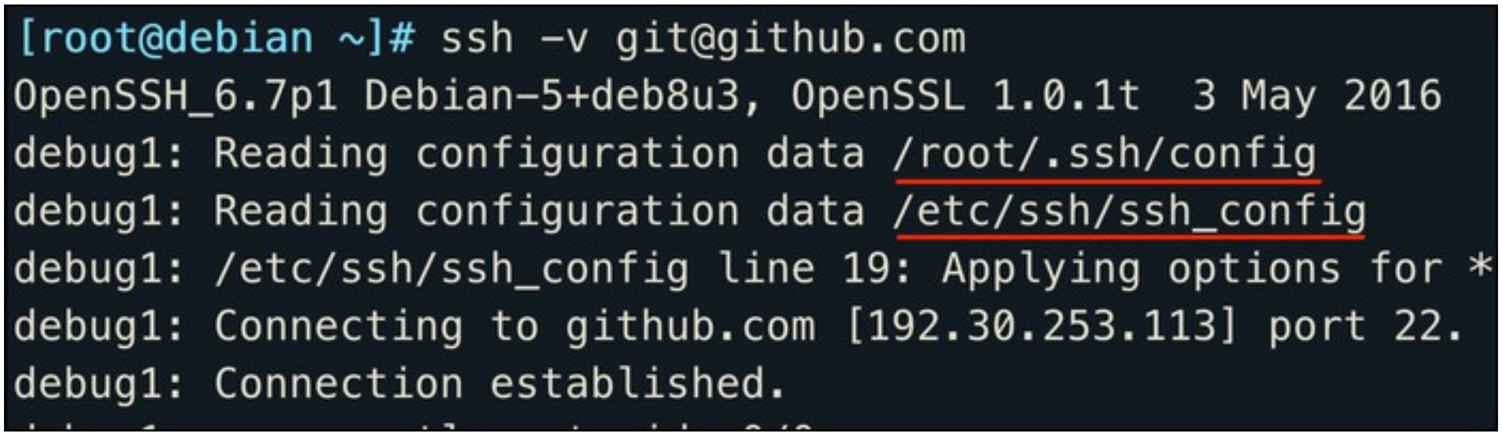
```
echo "$SSH_AUTH_SOCK"
```

If it's correctly set up, you should see a `Listeners` socket returned.

Make sure your config files are set up properly to include `ForwardAgent yes`, and make sure no other config files are overwriting this behaviour. To check which config files SSH is using, you can run `ssh` in verbose mode:

```
ssh -v git@github.com
```

Which should display which config files are being used. Files displayed later in this list take precedence over earlier files.



```
[root@debian ~]# ssh -v git@github.com
OpenSSH_6.7p1 Debian-5+deb8u3, OpenSSL 1.0.1t  3 May 2016
debug1: Reading configuration data /root/.ssh/config
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: Applying options for *
debug1: Connecting to github.com [192.30.253.113] port 22.
debug1: Connection established.
```

And of course, command line options override config files. If agent forwarding isn't working with `ssh -A`, and your keys are properly configured in your agent, then something else is wrong, and you'll need to check your connection to the servers in the chain.

READ NEXT

- › Should You Care About IPv6 Connectivity For Your Web Server?
- › Should You Use a Git Alternative?
- › NVIDIA's New Ampere GPU is a Game Changer for Artificial Intelligence
- › How to Lock Down Your SSH Server
- › The Best Alternatives to Github

ANTHONY HEDDINGS

Anthony Heddings is the resident cloud engineer for LifeSavvy Media, a technical writer, programmer, and an expert at Amazon's AWS platform. He's written hundreds of articles for How-To Geek and CloudSavvy IT that have been read millions of times.

READ FULL BIO »