AutoSSH - a Reverse Proxy Alternative

This document is best read printed out on paper.

1 Overview

I recently added another apache server to an existing infrastructure, and I wanted it to be accessible under a similar IP as another server. Due to the complexity of the website, it was not possible to simply do a reverse proxy without knowing the correct settings (e.g. X-Forwarded for). Instead, AutoSSH was used.

2 Work Log

Ok, I'm going to get right to the configs that I used. You want the tool, you don't need to know all the details.

2.1 Crontab

Here is the crontab script I used. I put this in /etc/crontab, so it has root after the times. I only use /etc/crontab, as it's easier to manage.

```
* * * * * root  pgrep autossh > /dev/null || \
 /usr/local/bin/autosshzm/autosshzm.sh
```

A few notes about this. Pgrep will search for autossh. If it doesn't find it, then it will try the next command. (—— is an OR). Put the bash script wherever you want.

2.2 Bash Script

This script is obviously what the crontab calls.

```
#!/bin/bash
logger " /usr/local/bin/autosshzm script started."
#source $HOME/.bash_profile #not needed.
source $HOME/.keychain/$HOSTNAME-sh
logger " /usr/local/bin/autosshzm sourced."

autossh   -L 0.0.0.0:2:localhost:80 -f user@ipaddress sleep 31536000
 &> /var/log/autosshzm/autosshzm.log
```

```
#autossh  -M 0 -o "ServerAliveInterval 30" -o "ServerAliveCountMax 3"
 -L 0.0.0.0:2:localhost:80 user@ipaddress &>
 /var/log/autosshzm/autosshzm.log
logger "auto ssh ran"
```

Note that the second autossh does not work, as it's missing the sleep and the -f command. [1] In order for this to work, you'll also need the following commands:

```
apt-get install keychain autossh
```

There were some more setup steps required for keychain... From stackexchange:

```
25
keychain
solves this in a painless way. It's in the repos for Debian/Ubuntu:

sudo apt-get install keychain

and perhaps for many other distros (it looks like it originated
from Gentoo).

This program will start an ssh-agent if none is running, and
 provide shell scripts that can be sourced and connect the current
 shell to this particular ssh-agent.

For bash, with a private key named id_rsa, add the following to
 your .profile:

keychain --nogui id_rsa

This will start an ssh-agent and add the id_rsa key on the first
 login after reboot. If the key is passphrase-protected, it will
 also ask for the passphrase. No need to use unprotected keys
 anymore! For subsequent logins, it will recognize the agent
 and not ask for a passphrase again.

Also, add the following as a last line of your .bashrc:
```

[1] Figuring this kind of stuff out can take about an hour.

```
. ~/.keychain/$HOSTNAME-sh

This will let the shell know where to reach the SSH agent managed
 by keychain. Make sure that .bashrc is sourced from .profile.

However, it seems that cron jobs still don't see this. As a
 remedy, include the line above in the crontab, just before
 your actual command:

* * * * * . ~/.keychain/$HOSTNAME-sh; your-actual-command
```

The only thing that I needed to do here was
keychain –nogui id_rsa
The rest of it (notes about crontab) was not required.

3 What Did NOT Work

Here's some things I tried that did not work.

- https://github.com/obfusk/autossh-init - This init script, didn't do much for me. Remember, I'm stuck with systemd in Ubuntu 19.04...[2]
- Reverse proxy with Apache - As I said, my website [3] was too complex, and I didn't want to go down that rabbit hole.
- Starting AutoSSH in rc.local. Didn't work.

[2]The scourge of deleting software history. Keep backwards compatibility at ALL COSTS, developers.

[3]Some people might call it a web application. I will not.