SSH to GitHub.com Through a Corporate Network

by **Paul Spears**, Oasis Digital Angular Boot Camp instructor

A large portion of modern web application development involves downloading and exploring the capabilities of third party tools and libraries. At the center of this tooling is github.com. With over 4 million users GitHub is the most popular resource for publicly shared code. GitHub has been integrated into the heart of many tools and access to such a resource is invaluable. Unfortunately, security measures on some network environments can prevent this.

I use NPM to manage my application's tooling dependencies. NPM looks at the dependencies of various node modules and resolves them automatically. This is done by cloning them from remote repositories, often, GitHub. If this process is performed on a corporate network with strict access policies a user may encounter any number of possible errors. The most common error on Windows is

```
connect to host github.com port 22: Bad file number.
```

On a *nix environment this same error is displayed as:

```
github.com[0: 207.97.227.239]: errno=Connection timed out fatal: unable to connect a socket (Connection timed out)
```

This is usually an indication that SSH traffic is not allowed on the current network.

Lets spend a minute analyzing what is going on when we encounter this error using NPM so that we know what to do about it. When we attempt to install a tool or library using NPM a list of dependencies and their corresponding repositories are calculated. In this situation one of the dependencies supplied a repository URL that indicates Git should use SSH to fetch the code. When Git attempts to access this repository the network firewall/proxy denies the request.

So what do we do to resolve this issue? Normally we would simply reissue the request using the project's HTTPS address. Unfortunately, we do not always have access to the dependency list of the module we are trying to install and have to accept the URL we are given. That leaves us with one of two choices. The first option is to use URL rewriting in Git. This is by far the easiest approach to implement. The premise is simple. We can instruct Git to match URLs and redirect the request to another URL of our choice. For our purposes, the command to achieve this is as follows:

```
git config --global url."https://github".insteadOf git://github
```

The previous command adds an entry to our git configuration that will send all Git traffic intended for git://github to the HTTPS equivalent. Now, if a dependency specifies an SSH or Git

protocol URL for its source code, the fetch will be performed using a less often blocked HTTPS protocol.

If Git is properly configured for HTTPS you should now be able to use tools like NPM and Bower to obtain code from repos on GitHub.

But what about the second choice? This is a far less elegant and much more involved process. If it is possible to use the first solution you should do so. If for some reason URL rewriting does not solve this problem for you it is possible to setup SSH over HTTPS.

This solution involves setting up SSH to make requests to GitHub over HTTPS. To start, sign up for a github.com. Yes, it really is necessary for this solution. Github only allows SSH traffic with a properly configured SSH key, even when using the default git@github.com account. Once you have an account you will need to generate an SSH key and associate it with your account.

Using the git bash prompt (in Windows) issue the following commands:

```
ssh-keygen -t rsa -C "your_email@example.com"

ssh-agent -s
//This may not work if the agent is already started


ssh-add ~/.ssh/id_rsa

clip < ~/.ssh/id_rsa.pub
```

Your SSH key is now on the clipboard and is ready to be added to your Github account.

Now that we have our account credentials associated with GitHub we need to inform SSH to tunnel through HTTPS. Find or create the following file ~/.ssh/config. This location must be relative to the home of the same bash environment used by your SSH agent. Once the file is located add the following line:
(For Windows)

```
ProxyCommand /bin/connect.exe -H proxy.server.name:proxyport %h %p

(For *nix)

ProxyCommand nc -x MY_PROXY_HOST:MY_PROXY_PORT %h %p

Followed by

Host github.com
  Hostname ssh.github.com
  Port 443
```

The ProxyCommand tells SSH what to run in order to redirect to a proxy. The “Host” entry redirects any requests for Github to its HTTPS port enabled host. In the purposes, if a username and password is required, the user should be specified but the password should be entered when prompted. (This may depend upon the proxy in use).

With these steps completed you will be prompted for your proxy password and SSH key password as necessary when requests are tunneled through HTTPS to Github.com.

Congratulations, you now have access to Github!

https://help.github.com/articles/generating-ssh-keys
https://help.github.com/articles/using-ssh-over-the-https-port
http://stackoverflow.com/questions/5103083/problem-of-testing-ssh-in-git-behind-proxy-on-window-7