Streamline HTTP traffic tunnel using PuTTyTray on windows boot and chrome proxy switch

by Benny Bottema - Monday, March 28, 2016

http://www.bennybottema.com/2016/03/28/streamline-http-traffic-tunnel-using-puttytray-on-windows-boot-and-chrome-proxy-switch/

From time to time it happens that I'm blocked out from a website by a corporate proxy, or that there's simply a problem with HTTP traffic. Whenever that happens, you have a choice: bring your mobile network-connected phone / tablet and work on that or wait until the proxy is fixed or postpone until you're home. Another option is to redirect the traffic yourself, using a tunnel.

This is a quick guide to setting up a HTTP proxy tunnel.

Requirements:

- A running SSH server out side of the network. For me it's my NAS at home
- The ability to download files or bring files into the system from USB or external storage

The most basic setup is by downloading PuTTY, configure a tunnel and point your browser to it. Today we're going with a slight variation for a better user experience.

Contents

Install the tunnel

Download puTTYTray

puTTYTray is a shell around PuTTY and is on PuTTY's officially sanctioned list. puTTYTray adds tray functionality to PuTTY so that you can simply minimize it. Since we're only interested in the tunnel it provides we don't need the terminal that normal stays open until you close the tunnel.

Go ahead and download then execute puTTYTray.

Set configuration and save session

From here, configuring the runnel is standard PuTTY business. You can change settings and then save the combination of settings into a named session. Each time you want to change the config, load the session, perform the change(s) and again save the session to the same name.

The trick to an SSH tunnel is that you can do this on any port, even ports that are not blocked by the corporate proxy. One of these always unblocked ports is port 443, also known as the HTTPS SSL port. With this port, Putty can communicate to any SSH server to create a tunnel.

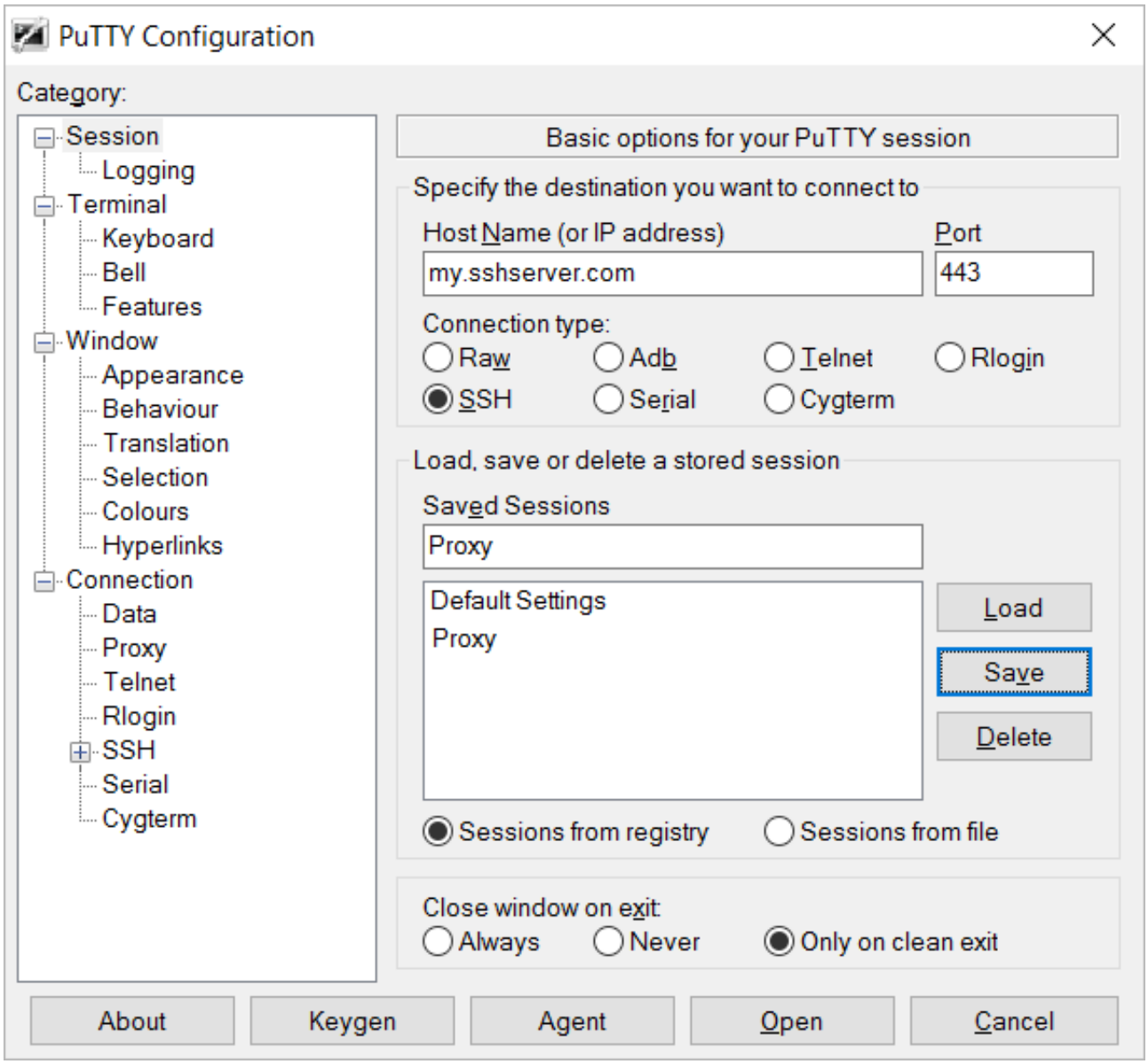
With such a tunnel you can direct any kind of traffic through the local end to the remote end on the SSH server.

You can use this tunnel also to give the remote SSH server access to the local client. This is called a *reverse tunnel*.

- Read more about tunnels with putty

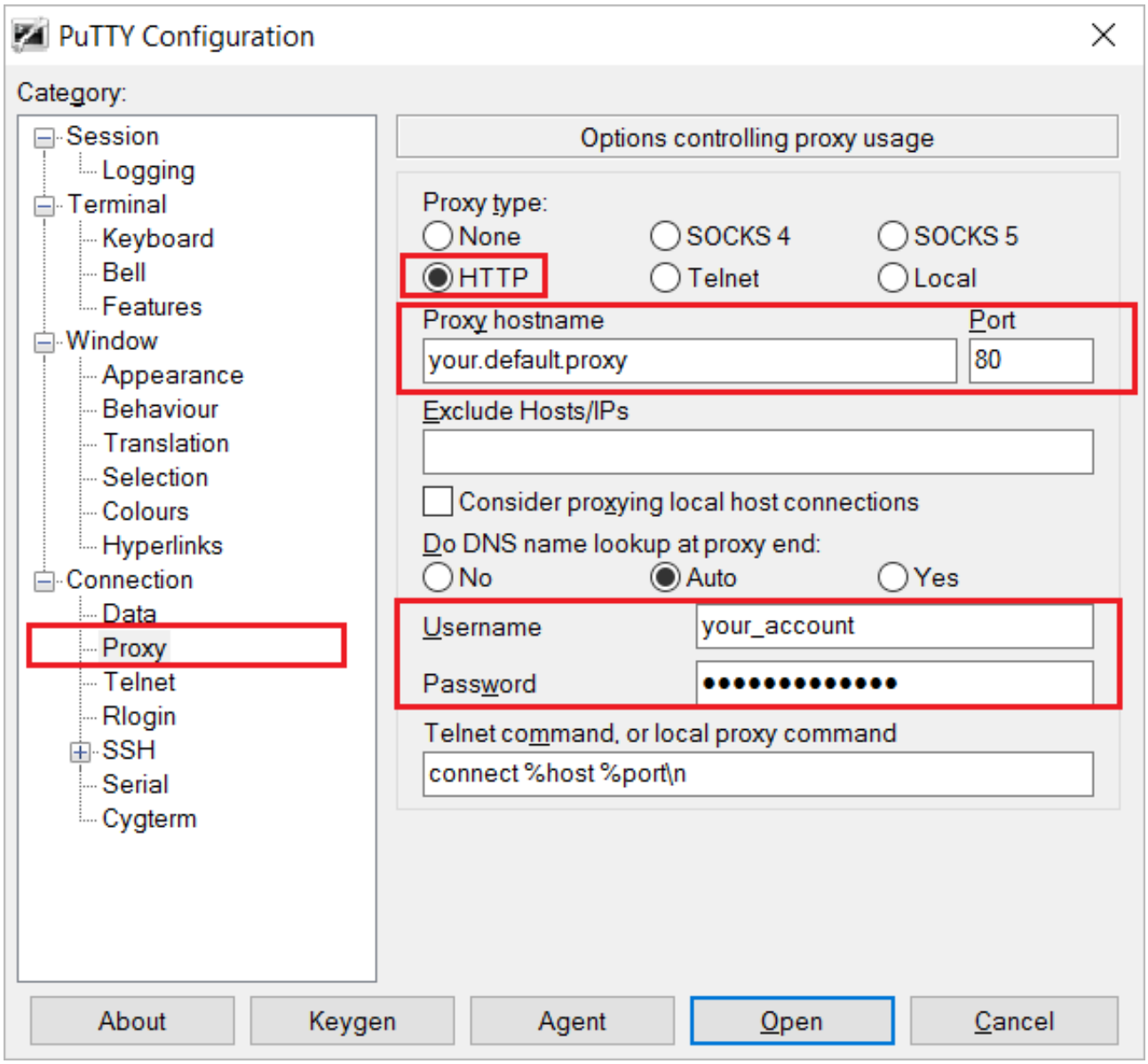
To configure the tunnel, first we need to indicate our SSH server, which is going to do the heavy lifting for us of performing the HTTP request to the outside world.

Got Session (default) -> add your SSH server's ip and make sure you use port 443 and give your session the name "Proxy" (we'll refer to it later):



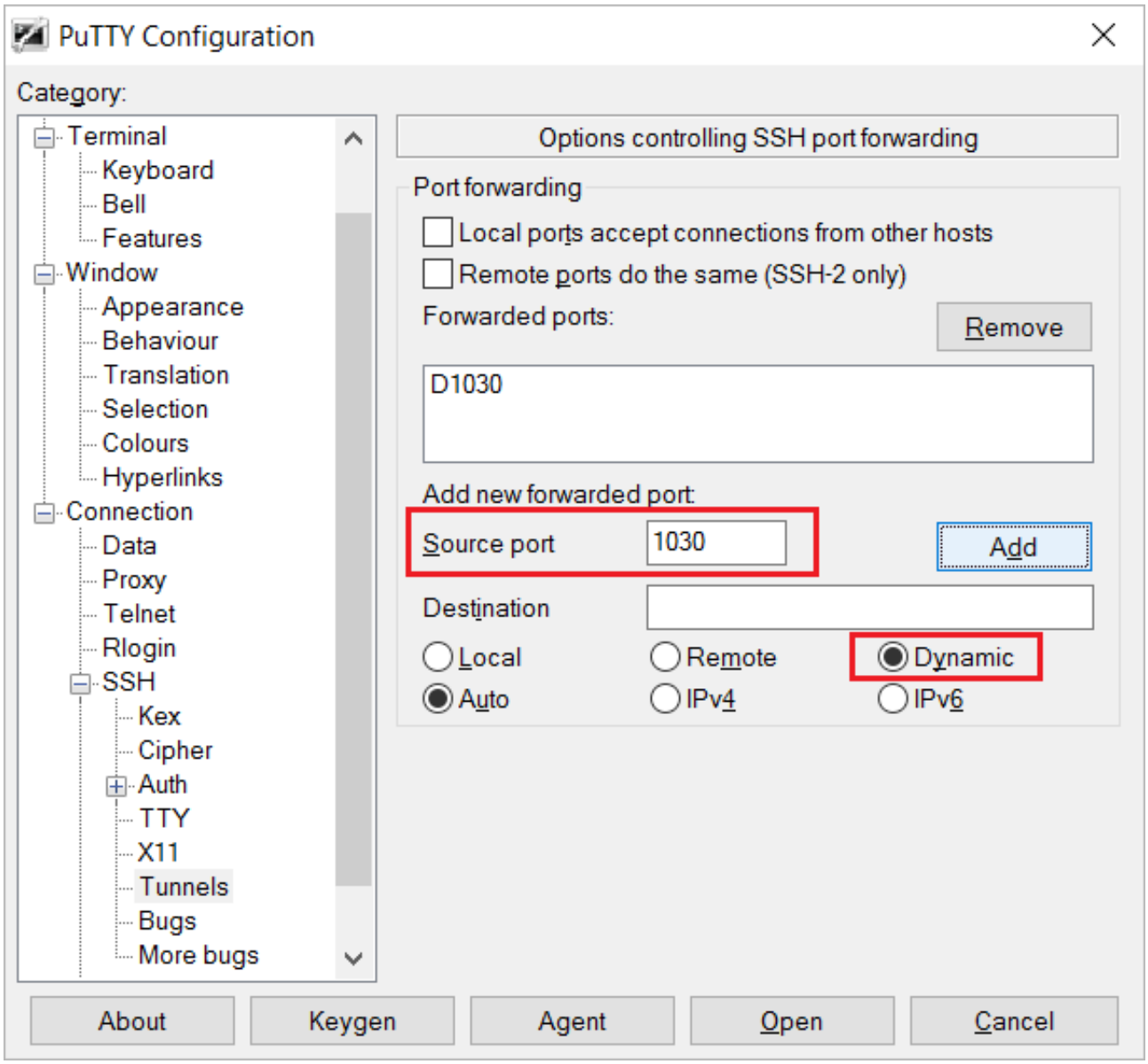
Second we need to add the corporate proxy, because Putty needs to tunnel through the HTTP port the same way your browser would if you visited an https:// website. Putty too will set up an SSL connection just like your browser would. The difference is we're not getting and sending a single https page, we're getting and sending **all** HTTP(S) traffic through this SSL connection.

Go to Connection -> Proxy -> Leave everything at default except Proxy type, port, Proxy hostname and your login details. Often this is the same as your windows login. Although uncommon, Proxy type can differ from HTTP, so change accordingly.



You can test if the proxy is configured correctly by simply starting an SSH session to your server by clicking *Open*. If it connects it means your company's proxy accepted your login and your ssh server accepted the connection on port 443. Most of the work is now done!

Third, the magic trick: configure putty to redirect all HTTP requests on *localhost* to the remote SSH server. That's right, HTTP requests from the browser will need to go to localhost instead of the outside world. Luckily we can configure any browser to do this automatically. Go to Connection -> SSH -> Tunnels -> and add a Dynamic port 1030 and leave the Destination empty and the rest on default:



So what did we just do? We told Putty to do a Dynamic port forward for everything requested using the SOCKS proxy protocol on port 1030 of the local client (so localhost:1030 or 127.0.0.1:1030). That's right, we just configure Putty to act as a SOCKS proxy! By just starting the SSH session, this proxy will be running automatically as part of the session.

Quoting the PuTTY's documentation:

> Set one of the 'Local' or 'Remote' radio buttons, depending on whether you want to forward a local port to a remote destination ('Local') or forward a remote port to a local destination ('Remote'). Alternatively, select 'Dynamic' if you want PuTTY to provide a local SOCKS 4/4A/5 proxy on a local port (note that this proxy only supports TCP connections; the SSH protocol does not support forwarding UDP).

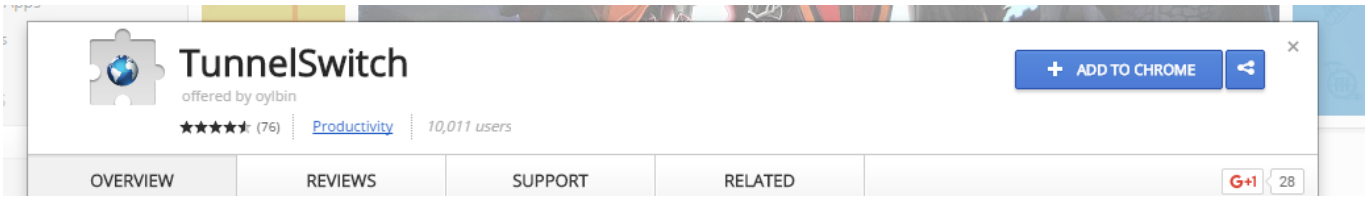
Configure your browser to use the proxy

Now that we have a SOCKS proxy running over port 443 using your normal proxy, we should start directing traffic through it from the browser.

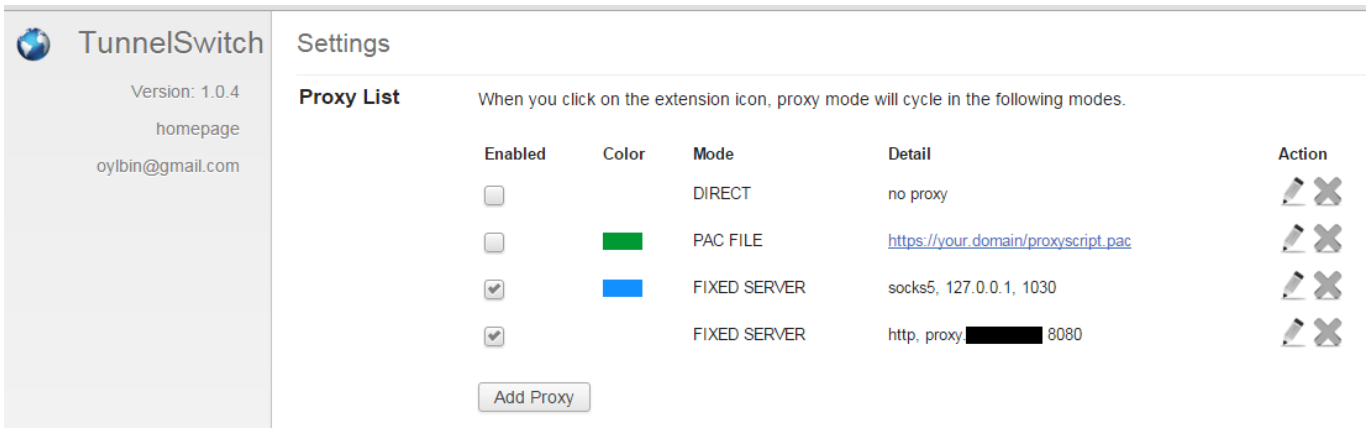
Configuring proxy settings in the various browsers in not a trivial matter. I've spent hours on end trying to configure Chrome and Firefox and various versions of them to use my custom proxy. The settings changed over the versions too and debugging these settings is terrible.

Install TunnelSwitch for Chrome

Luckily, there are some helpful tools out there today. The one we're going to use is the awesome TunnelSwitch for Chrome. In TunnelSwitch you can configure various proxy servers and modes. Then the TunnelSwitch tray icon allows you to cycle through these with ease. So go ahead and add this plugin to Chrome.



For our setup, we don't use *DIRECT*, because we are behind a restricting proxy already. We could use a Proxy auto-config (or PAC) script with TunnelSwitch that dynamically assigns the corporate proxy or our tunnel proxy based on a URL or REGEX. I had some issues getting this to work in Chrome. I settled for two *FIXED* proxy servers between which I can switch: the corporate proxy and my tunnel proxy. This first one is a regular *HTTP* proxy, while my tunnel is a *SOCKS5* proxy (which is easier for Chrome to play nice with).



Cycle through these modes by clicking on the TunnelSwitch tray icon:

 tunnel modes

To automatically start tunnel on windows startup

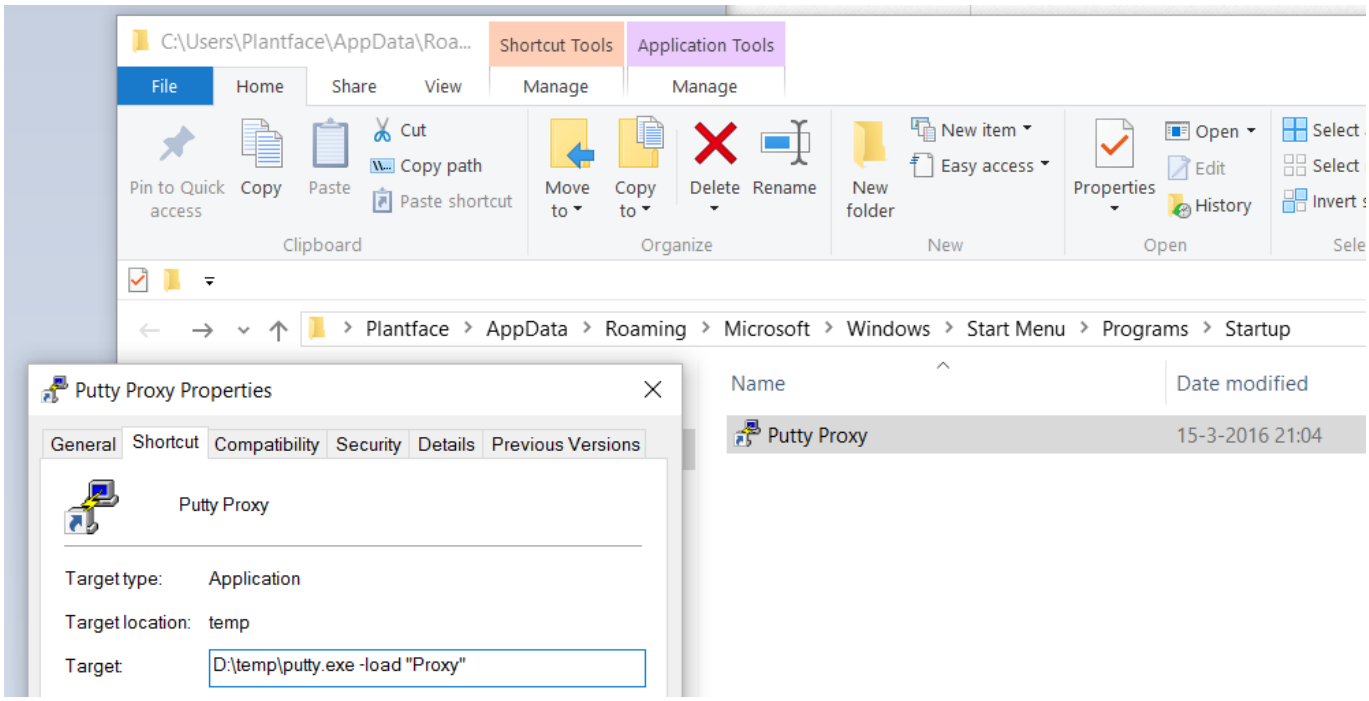
Start with Windows

Create a shortcut to your putty installation (be it PuTTY or PuTTYTray) and modify it's target by adding -load "proxy" at the end or however you named the putty session earlier.

Now run the following command in windows (via Start Menu -> Run or WIN-R key):

```
shell:startup
```

This will open the folder with items that are automatically run when Windows boots. Move the shortcut into this folder. Now every time windows boots, the tunnel will also start and wait for your login. Next we'll see how we can even automate this step.

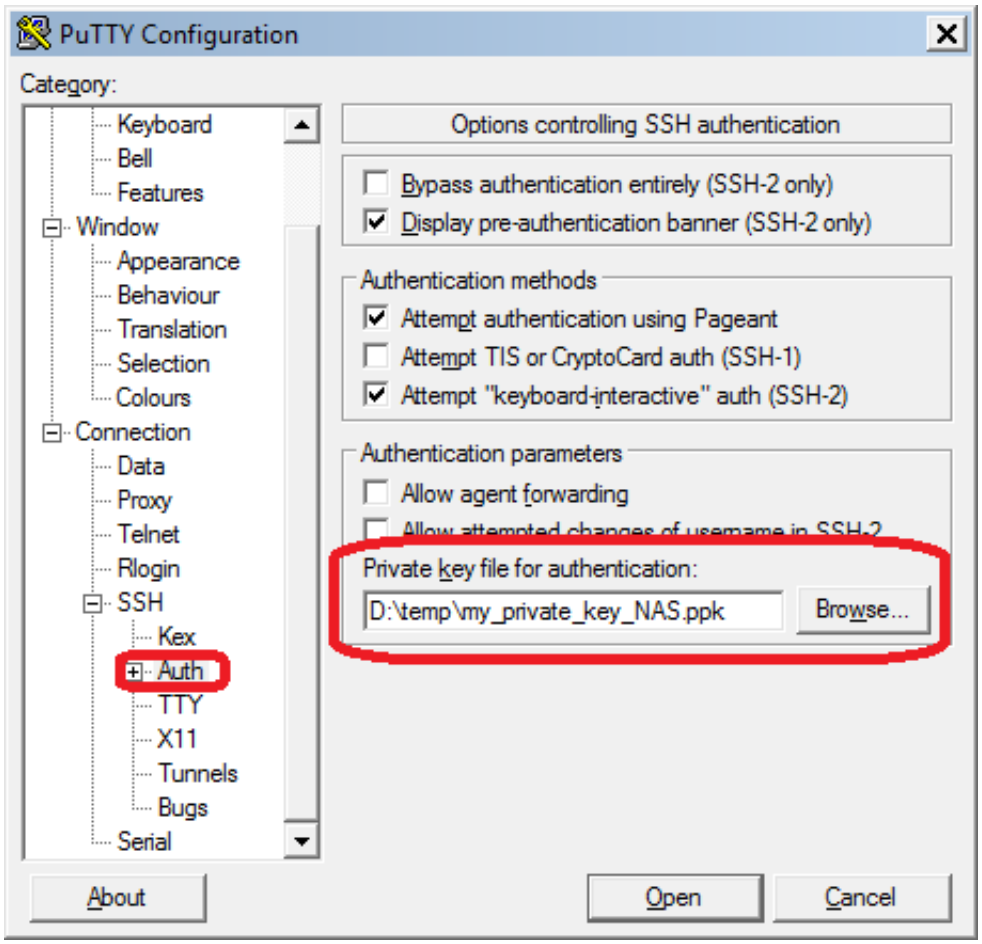


Use SSH key for complete automation

Logging in with your user / pass is manual labor and quite insecure. A more modern and usable way is to use public / private keys. Initial installation is a bit more complex now, but you'll be able to start a tunnel without logging in and have PuTTYTray minimize to tray directly rather than hovering on your desktop waiting for your login input.

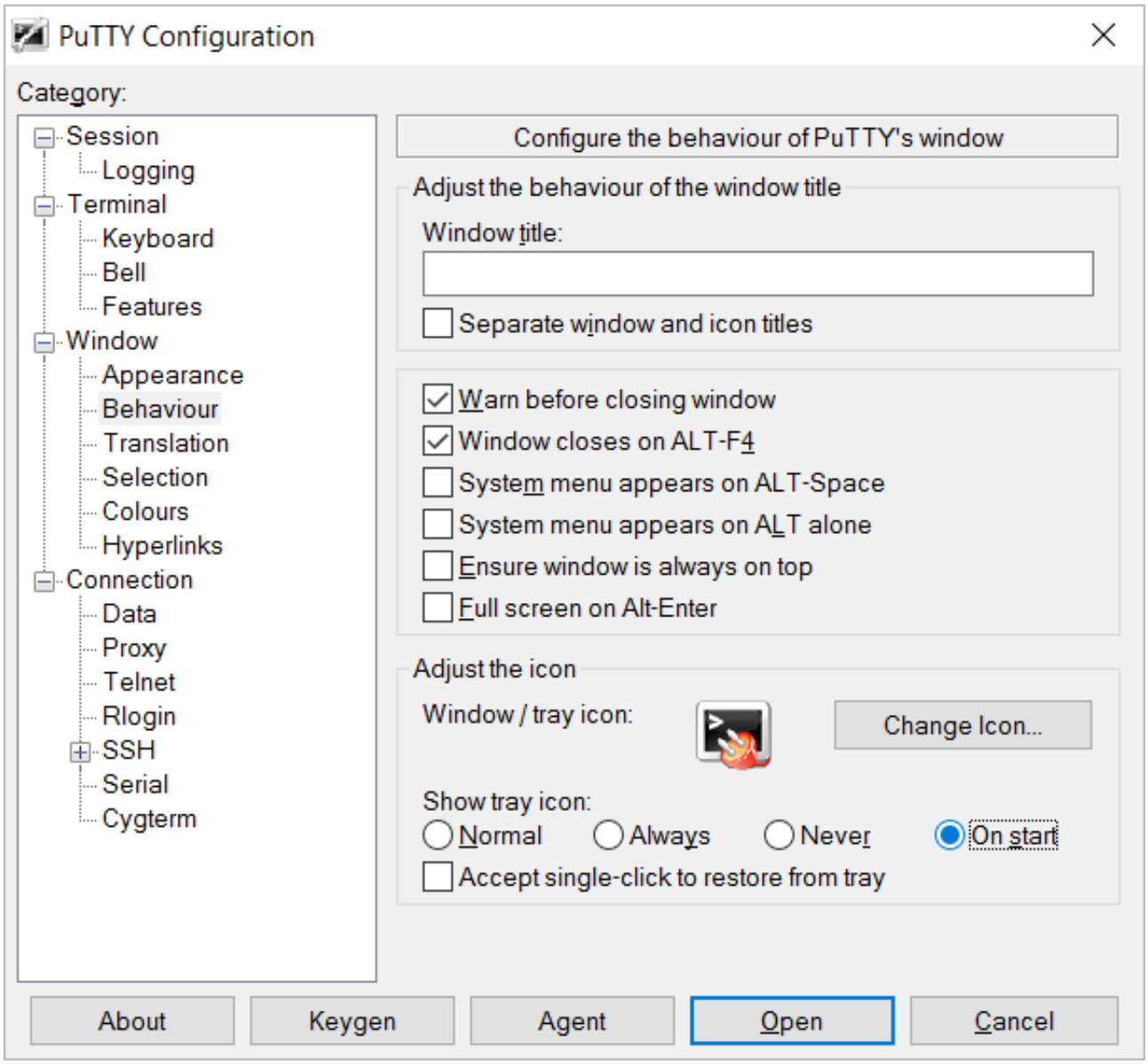
Generate a private / public key pair, using putty's own generator, PuTTYgen. Keep the public key on your SSH server and use the private key in putty to log in. I'm not detailing how to install SSH keys, but to use the private key in PuTTY, you can either use putty's own agent called Pageant to manage your keys outside of putty or use can configure a specific session to use a specific key. The first approach is handy if you want to reuse keys or if you have multiple keys you want to keep together, the second approach is useful for simple per-session configurations.

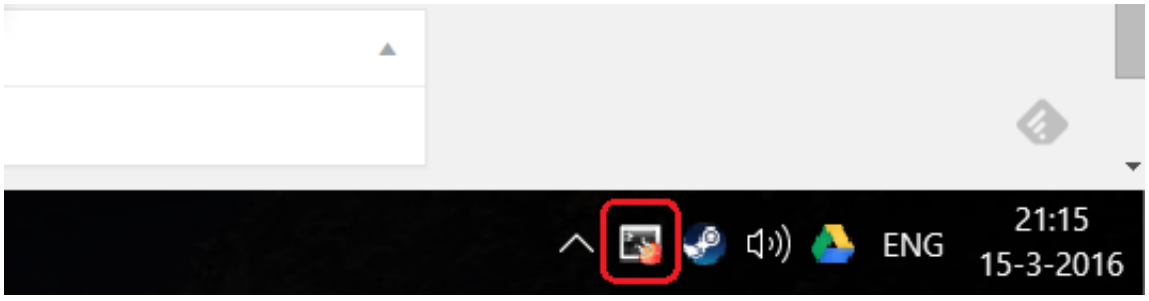
Here's a screenshot of where you use the private key in putty:



Start PuTTYTray minimized

Finally, as no manual interaction is necessary anymore, we need to configure PuTTYTray to start minimized. Load the previously session so all the configuration is restored, then go to Window -> Behaviour -> Adjust the icon -> set “Show tray icon” to “On start”.





PuTTRay SSH Proxy now starts minimized

What have we accomplished now?

We are starting a minimized tunnel when windows start, which does an automatic login using a private SSH key and sits snugly in the windows tray. Finally we have a nice switch in Chrome to switch between proxy modes whenever we need to.

Pretty awesome right?

PDF generated by Kalin's PDF Creation Station