Short Linux Guides

All things Linux that don't deserve more than one page each!

X11VNC SystemD Service

XUbuntu 18.04 (LightDM)

Have you always wanted to remote into your Linux both with VNC on the same XServer as the desktop is current running on? It's a common issue that most VNC software start new XSessions when all you want to see is what is currently displaying on the screen.

That's where X11VNC comes in.

X11VNC allows you to VNC into an existing XSession so that you can see what is currently being displayed on your Linux box's output. The only issue is that if you want the VNC server running all the time you'll have implement your own script to start it up on boot.

Looking around I found a few guides here and there for setting X11VNC up correctly for Ubuntu 18.04 as a SystemD service however most were outdated or had annoying issues such as the SystemD unit hanging.

Note: if you have not done so already you'll need a password for your VNC server: `x11vnc -storepasswd`

Using an amalgamation of knowledge from a lot of different places I came up with my own Unit script.

1. Use your favourite editor and create a new file: `sudo nano /etc/systemd/system/x11vnc-h.service`
   (I called my service `x11vnc-h` just to be different from the existing script)

```
[Unit]
Description=x11vnc VNC Server for X11
Requires=lightdm.service
```

```
After=lightdm.service
[Service]
Type=simpleExecStart=/usr/bin/x11vnc -auth /var/run/lightdm/root/:0 -display WAIT:0 -forever -
shared -rfbauth /home/user/.vnc/passwd -rfbport 5900
ExecStop=/usr/bin/killall x11vnc
Restart=on-failure
RestartSec=2
SuccessExitStatus=3
[Install]WantedBy=graphical.target
```

Now the magic here is `Type=simple`. This little type switch will stop the unit hanging when you go to start it using the normal forking mode. Another useful part of the unit are the restart switches which make sure that the VNC server is always up and restarts if it crashes.

Note: `-rfbauth` is the path to the VNC password file that you generated earlier.

~~**Note**: If you're an Arch user running LightDM you might not have anything inside the `/var/run/lightdm/` folder to which I have not found a fix for just yet.~~

Update 22/Dec/2018: On further testing the path may not exist on Manjaro / Arch based systems correctly but the same path still seems to work.

After you've saved your unit script reload SystemD with `systemctl daemon-reload` and fire it up with `systemctl start x11vnc-h`.

If all went well you should have an output like this: `systemctl status x11vnc-h`

```
● x11vnc-h.service - x11vnc VNC Server for X11   Loaded: loaded (/etc/systemd/system/x11vnc-
h.service; disabled; vendor preset: enabled)   Active: active (running) since Fri 2018-12-21
21:12:29 GMT; 3s ago  Process: 23528 ExecStop=/usr/bin/killall x11vnc (code=exited,
status=0/SUCCESS)
 Main PID: 23530 (x11vnc)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/x11vnc-h.service           └─23530 /usr/bin/x11vnc -auth
/var/run/lightdm/root/:0 -display WAIT:0 -forever -shared -rfbauth /home/user/.vnc/passwd -
rfbport 5900
```

```
Dec 21 21:12:29 xee x11vnc[23530]: 21/12/2018 21:12:29Dec 21 21:12:29 xee x11vnc[23530]:
21/12/2018 21:12:29 initialize_screen: fb_depth/fb_bpp/fb_Bpl 24/32/2560Dec 21 21:12:29 xee
x11vnc[23530]: 21/12/2018 21:12:29Dec 21 21:12:29 xee x11vnc[23530]: 21/12/2018 21:12:29
Listening for VNC connections on TCP port 5900Dec 21 21:12:29 xee x11vnc[23530]: 21/12/2018
21:12:29 Listening for VNC connections on TCP6 port 5900Dec 21 21:12:29 xee x11vnc[23530]:
21/12/2018 21:12:29 listen6: bind: Address already in useDec 21 21:12:29 xee x11vnc[23530]:
21/12/2018 21:12:29 Not listening on IPv6 interface.Dec 21 21:12:29 xee x11vnc[23530]:
21/12/2018 21:12:29
Dec 21 21:12:29 xee x11vnc[23530]: The VNC desktop is:      xee:0Dec 21 21:12:29 xee
x11vnc[23530]: PORT=5900
```

Manjaro i3 - GitKraken failing due to unset SSH_AUTH_SOCK

I came across on an annoying issue with the `SSH_AUTH_SOCK` variable not being set correctly on my Manjaro i3wm work machine. This is an issue as the variable needs to be set in order to use GitKraken properly. It turns out that the software wasn't even being correctly loaded in the first place on the community i3 version of Manjaro probably due to the minimalist intended design so I set out to fix that.

First of all we actually need `gnome-keyring` so using a terminal run `pacman -S gnome-keyring` to install it.

Next up we need to make some PAM changes so following this guide on the Arch Linux forums I changed the following file:

```
# /etc/pam.d/login
#%PAM-1.0
auth       required     pam_securetty.so
auth       requisite    pam_nologin.so
auth       include      system-local-loginauth       optional
pam_gnome_keyring.so              #<-- Add this lineaccount    include      system-local-login
session    include      system-local-login
session    optional     pam_gnome_keyring.so auto_start   #<-- Add this line
```

The two important lines highlighted in the file load the gnome keyring pam libraries.

Next we need to add a few lines to the `.xinitrc` file which can be found in your user's home.

```
# /home/user/.xinitrc
# get_session(){
#   ... bla ...
# }
...
# Start Gnome keyring
dbus-update-activation-environment --systemd DISPLAYeval $(/usr/bin/gnome-keyring-daemon --
start --components=pkcs11,secrets,ssh)
export SSH_AUTH_SOCK
...# exec ...
```

The lines between the dots are added just after the `get_session()` function but before the `exec` call to start i3wm. An extra line is also needed in the form of a dbus-update call for i3wm specific reasoning.

After that is sorted we need to add a few lines to the Bash `.profile` file in order to export the variable for Bash sessions.

```
# /home/user/.profile
# At the bottom of the file
# ...
# Start gnome ssh keyring daemon
if [ -n "$DESKTOP_SESSION" ];then    eval $(/usr/bin/gnome-keyring-daemon --start --
components=pkcs11,secrets,ssh)
    export SSH_AUTH_SOCKfi°
```

At the bottom of the file add the above 5 lines which makes sure that the keyring daemon is actually running and exports the variable we require for GitKraken (and various other programs).

You can now safely restart the system or logout and back in again to load everything up and we should be good to go. We can of course test this by echoing out the variable in the terminal like so:

```
```

```
[user@machine ~]$ echo $SSH_AUTH_SOCK/run/user/1000/keyring/ssh
```