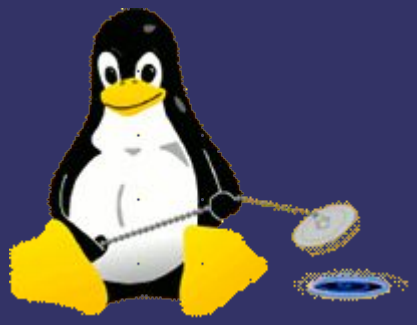


Remote Administration of Desktop Systems

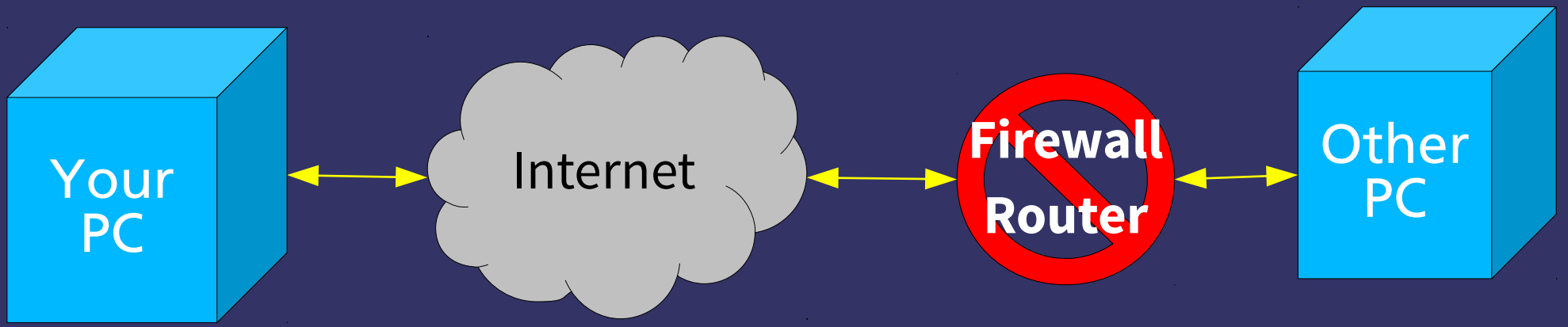
Adam John Trickett

www.iredale.net
adam.trickett@iredale.net
PGP Key: 0xAF0DB8C8

General problem

- You need to connect to a remote system
- You need to administer the system (upgrade, repair, extend etc.)
- You need to see the desktop as the user sees it
- The end user may not be technical

Topology



Technical problems

- Where is the other PC?
  - Most ISP only offer dynamic IP
- How do I get through the firewall?
  - Each make and model is different
- How do I reach the PC on the inside?
  - Most networks use dynamic & private IPs on the inside
- What needs to be installed on the target system?
  - Not all systems have everything installed by default

Where is the other PC

- The best solution is a static IP for the router/firewall
  - Standard with some ISPs
  - Optional cost extra with others
- If dynamic is the only option, then:
  - Some routers/firewalls will auto-update Dynamic DNS services
  - You can install a dynamic DNS client on the target PC
  - You can create a script to email you the external IP

Firewall - rules

- Most sane routers allow:
  - All ports outbound
  - All ports inbound that are part of an outbound pair
  - All ports inbound that are not part of a pair are denied
- You will need to tell it to allow at least one port inbound:
  - Some have virtual “DMZ”
  - Some have general rules

Router - Forwarding

- The remote system's firewall/router needs to forward incoming connections:
  - of type X, e.g. tcp
  - of port Y, e.g, 22
  - to IP address Z, e.g. 192.168.0.10
- External port number and internal port number are the same by default

Router – NAT/DHCP

- You need to ensure that the PC you want to reach has the same private IP so that the NAT rule points to the correct system every time:
  - DHCP reservation using MAC address
  - Static configuration in router and PC

Basic tools - SSH

- Secure Shell (“SSH”)
  - Replaces Telnet, rlogin, rsh, ftp etc
  - Standard on almost all Linux/Unix systems
  - Secure
  - Supports port forwarding
  - Creates a temporary on-demand instant “VPN-lite”

Extra tools

- Mobile Shell (“Mosh”)
  - Deals with lost connections better than SSH
  - Does not support port forwarding
- OpenVPN
  - Builds a permanent secure bridge between systems
  - Doesn't require user configuration to use
  - Requires administrative configuration to set-up
  - More complex than SSH

General installation

- OpenSSH server, though in all distros is not installed by default on all of them
- Mosh is widely available but not installed by default on most/all
- Sudo is widely available and installed by default on many but all
- Screen is widely available but not installed by default on most/all

Specific installation

- linuxvnc shares the physical console as VNC session, useful in emergencies or headless servers
- x11vnc shares the desktop X session as a VNC session and allows you to interact with the desktop at the same time as the user
- There are others but I'm not going to talk about them

Forwarding SSH ports

- The remote system's firewall/router needs to:
  - Forward TCP port on the external side to TCP port on the target PC
  - SSH normally uses tcp port 22
  - Mosh normally uses udp port 60001 (and up) plus SSH to start with only
- Many people change the external port to reduce the noise from script kiddies

Basic Administration

- Use SSH/Mosh to connect to the remote system
  - Default SSH configuration will work but you need to harden it
  - Run normal command line tools from login shell of your choice
  - Good for day to day administration and all standard tasks
  - No good if you need to see what the user sees or configure a desktop application

Harden SSH

- Open SSH is pretty good but it is not as secure as it can be out of the box on most Linux distributions:
  - Turn off password login – only allow SSH keys
  - Turn off root login – only allow real users
  - Specify the named users you want to allow
  - Turn off SSH protocol 1 – it may still be turned on in some distros

Configure SSH Client

- Edit your `~/.ssh/config` file:

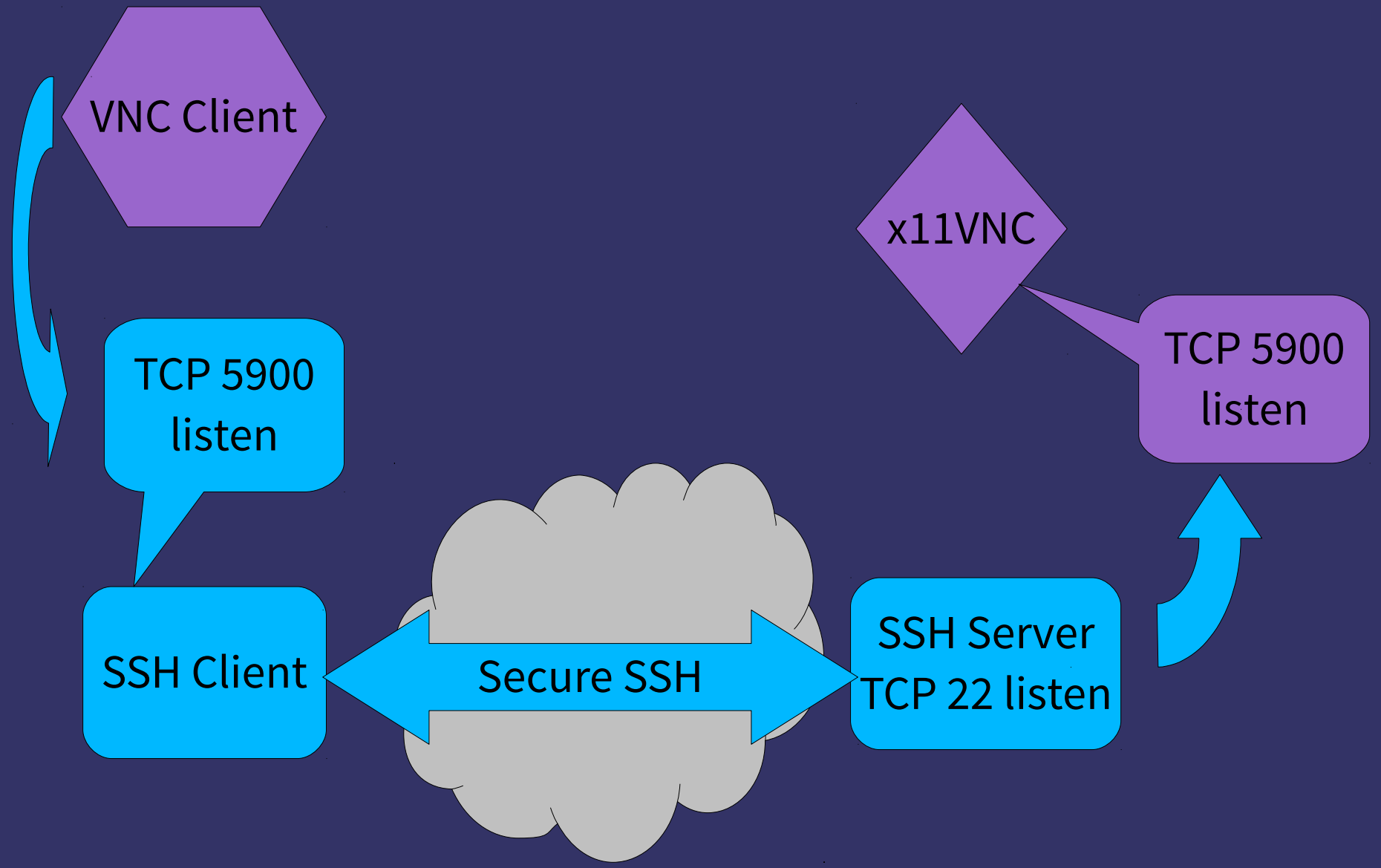
```
Host            <machinename>* <ip address>
HostName        <machinename.network.com>
user            <your username on machinename>
Port            <TCP port number>
ForwardX11      yes
Compression     yes
LocalForward    localhost:5900 localhost:5900
```

Procedure

- Add your SSH-Key to your SSH-Agent
- Start your SSH session to the other system
  - ssh machinename
- Your default shell starts at the other end
  - Start screen
  - Start any X programs
  - Start x11vnc or linuxvnc
- Start your VNC client on your desktop

What does SSH forwarding do?

- When you start x11vnc or linuxvnc they start to listen on the local host of the remote system on tcp port 5900 by default
- The SSH client on your PC also listens on TCP port 5900 locally, but forwards the packets to the remote system to its TCP port 5900
- That means an insecure protocol like VNC is now running over a secure and compressed SSH connection



x11vnc configuration

- To automate and get the best out of x11vnc without end user interaction – there are a lot of options!
- Something like:

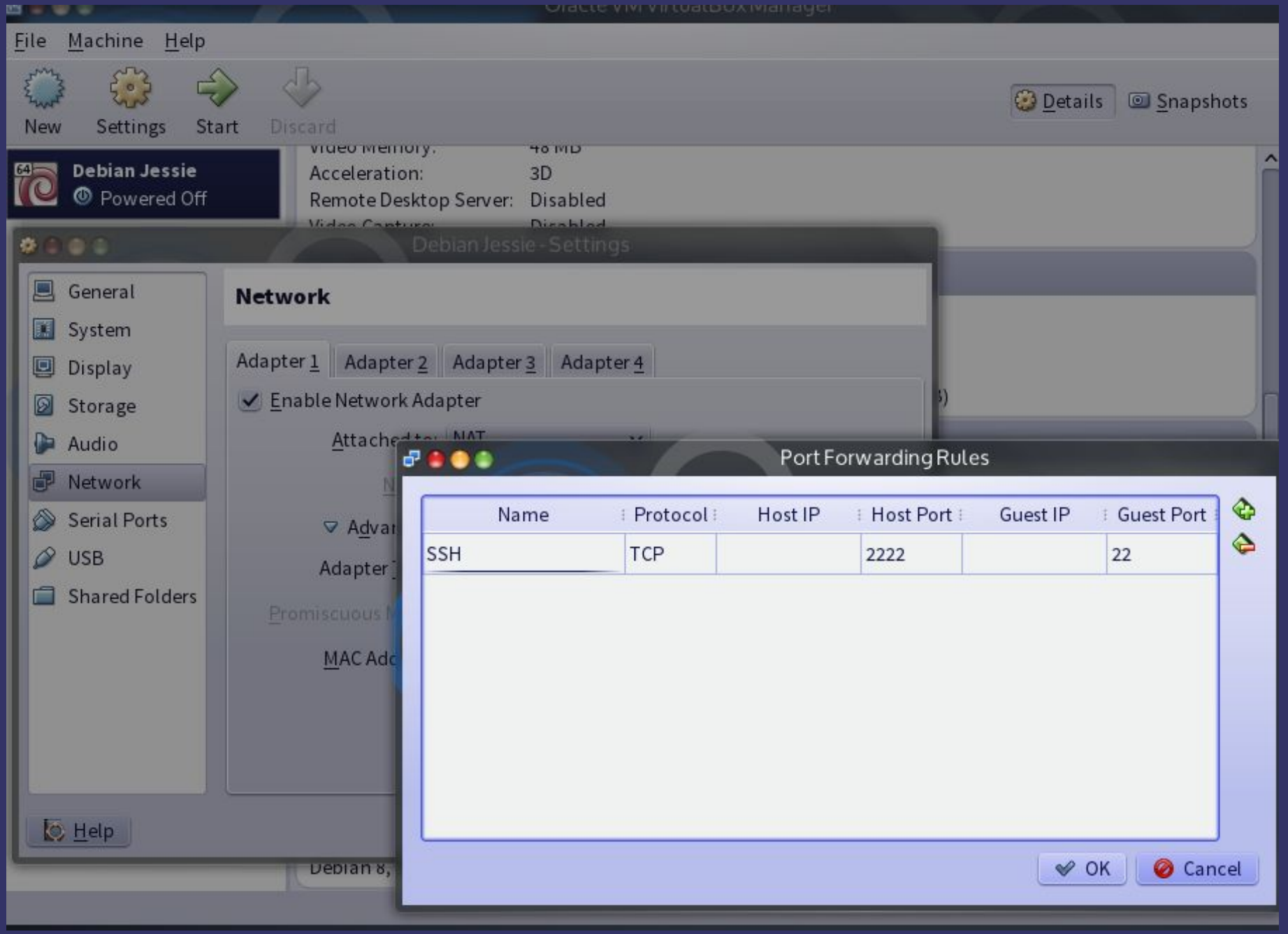
```
$ sudo x11vnc -nopw -localhost -ncache 10 -ncache_cr \
 -q -nodpms -auth <something>
```

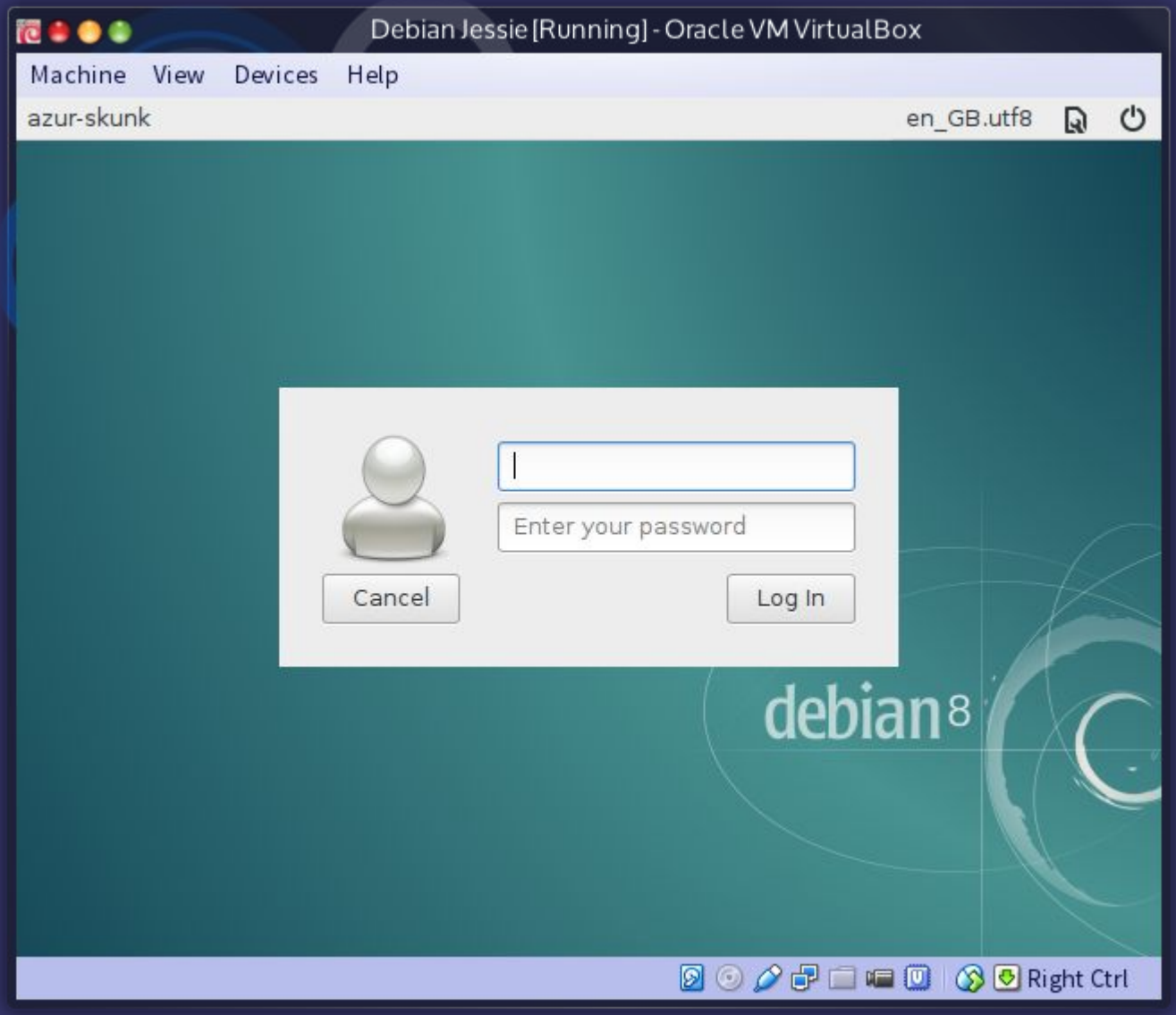
linuxvnc configuration

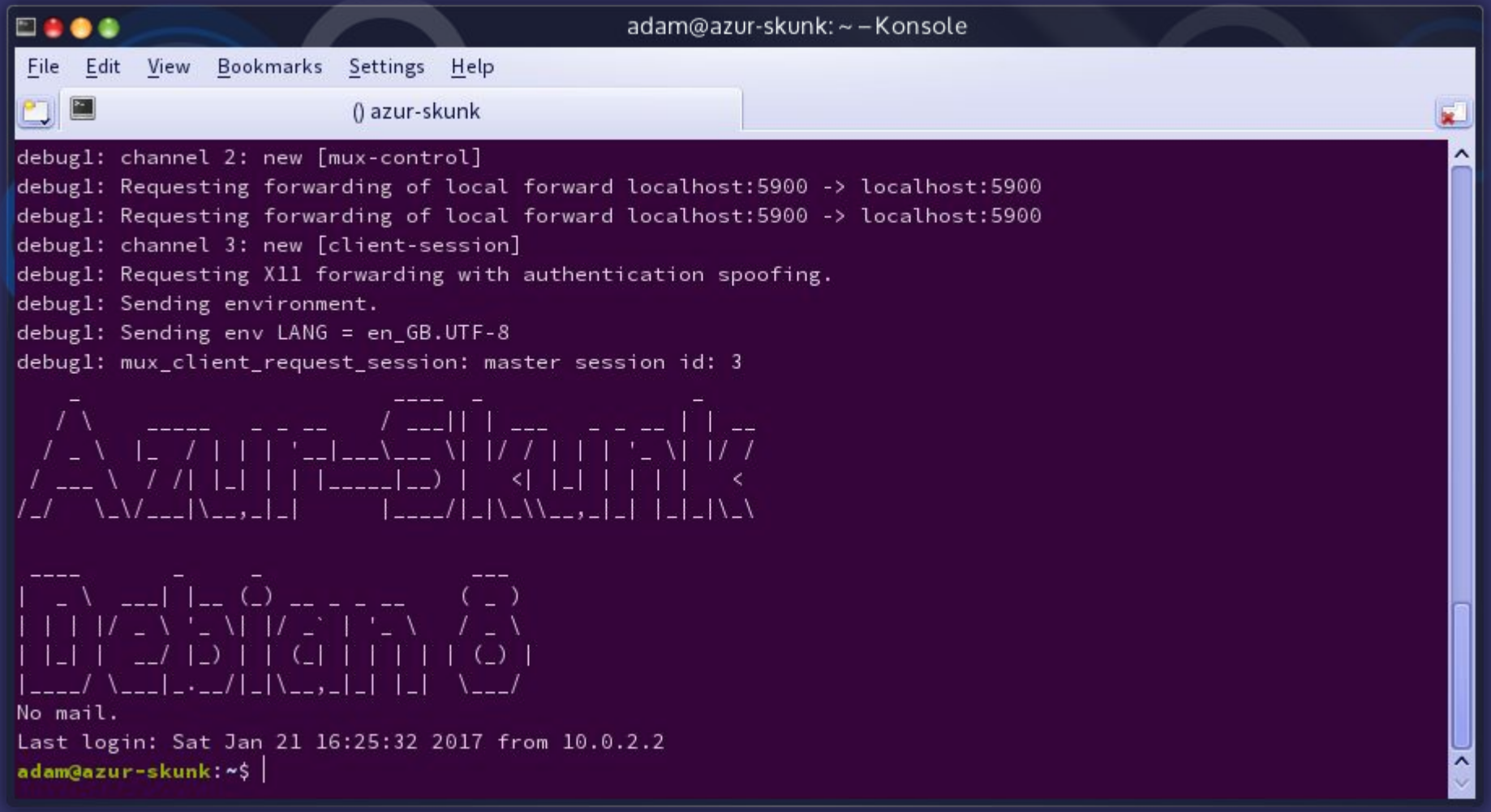
- Exports a physical terminal
- Useful if X has failed to start
- Allows you to see kernel messages etc
- Of only limited use, but nice to know

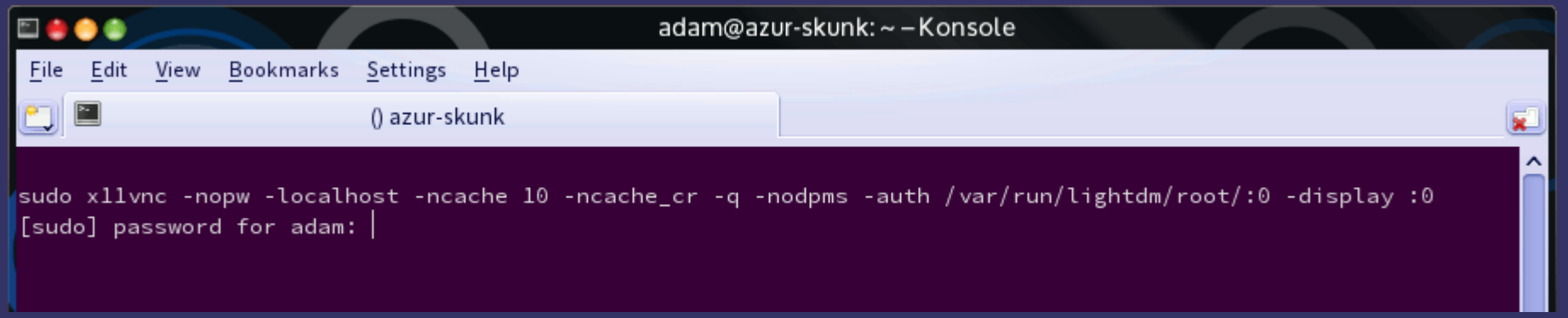
```
$ sudo linuxvnc 1 -alwaysshared
```

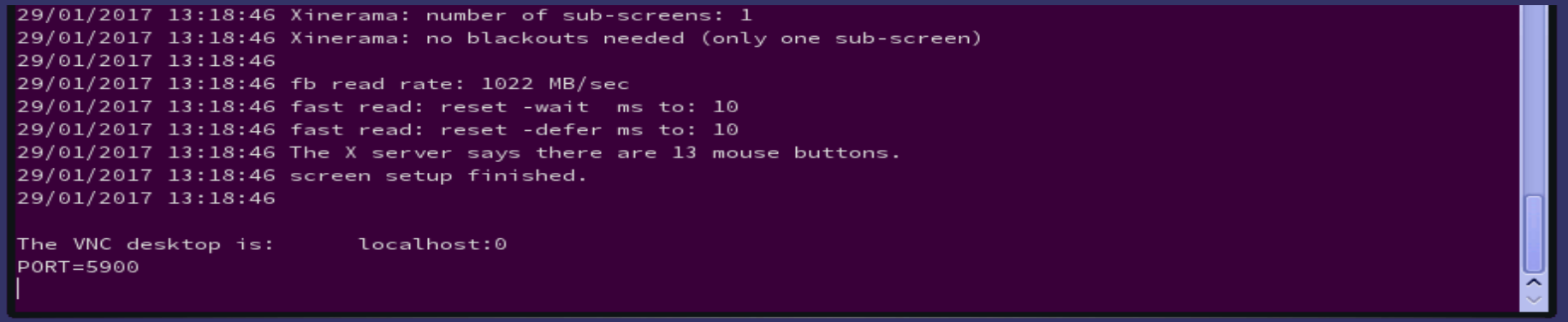
Demo

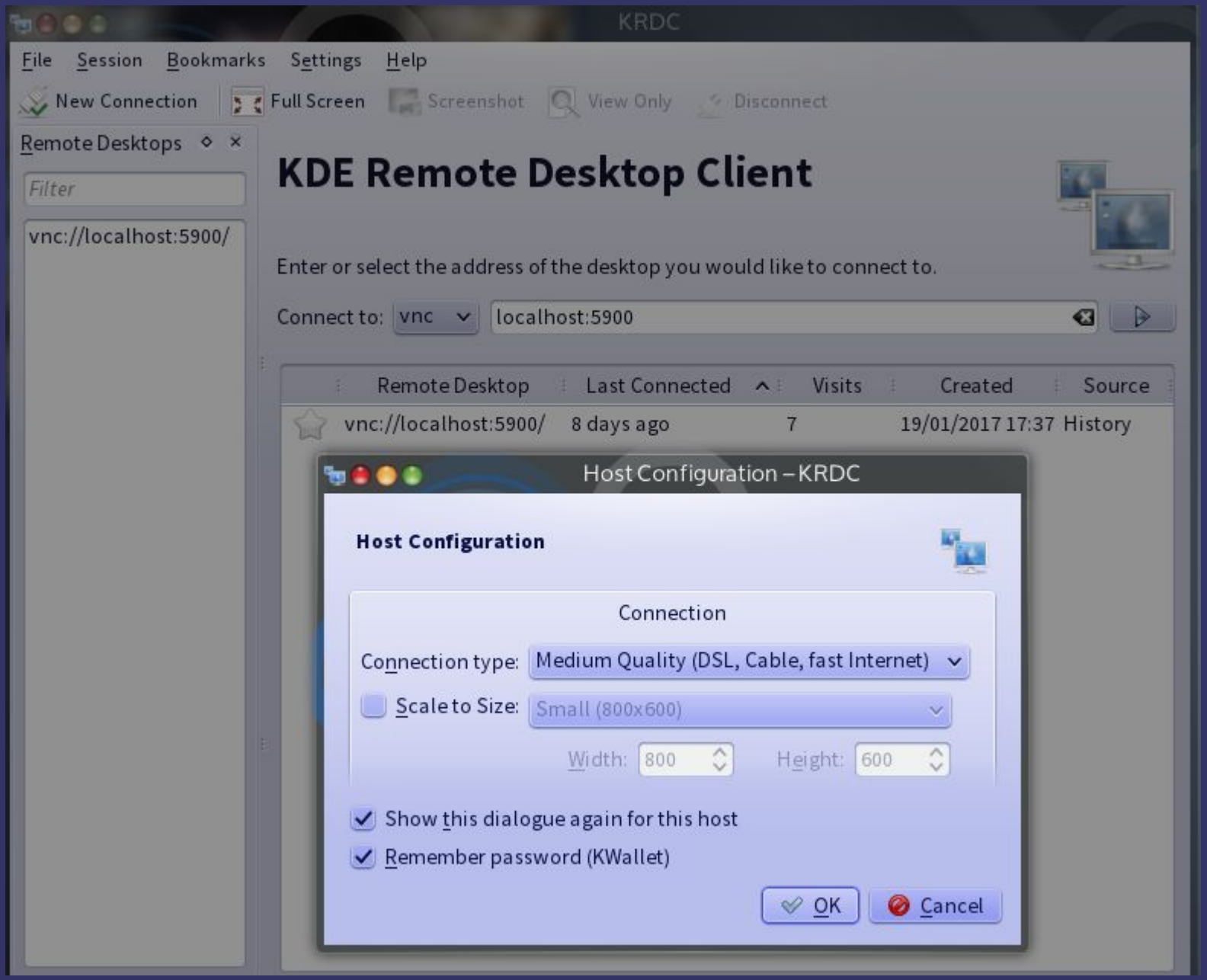


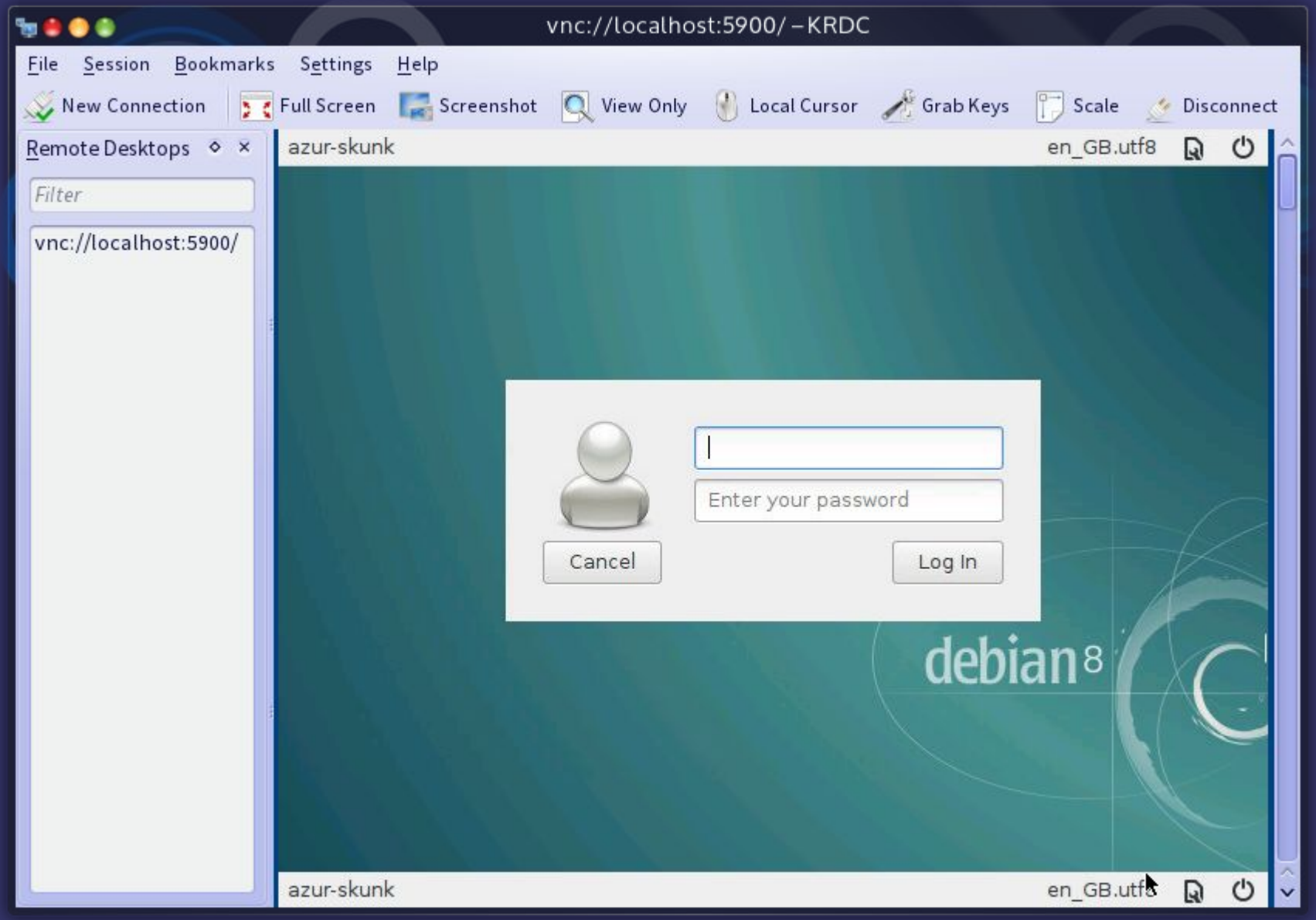


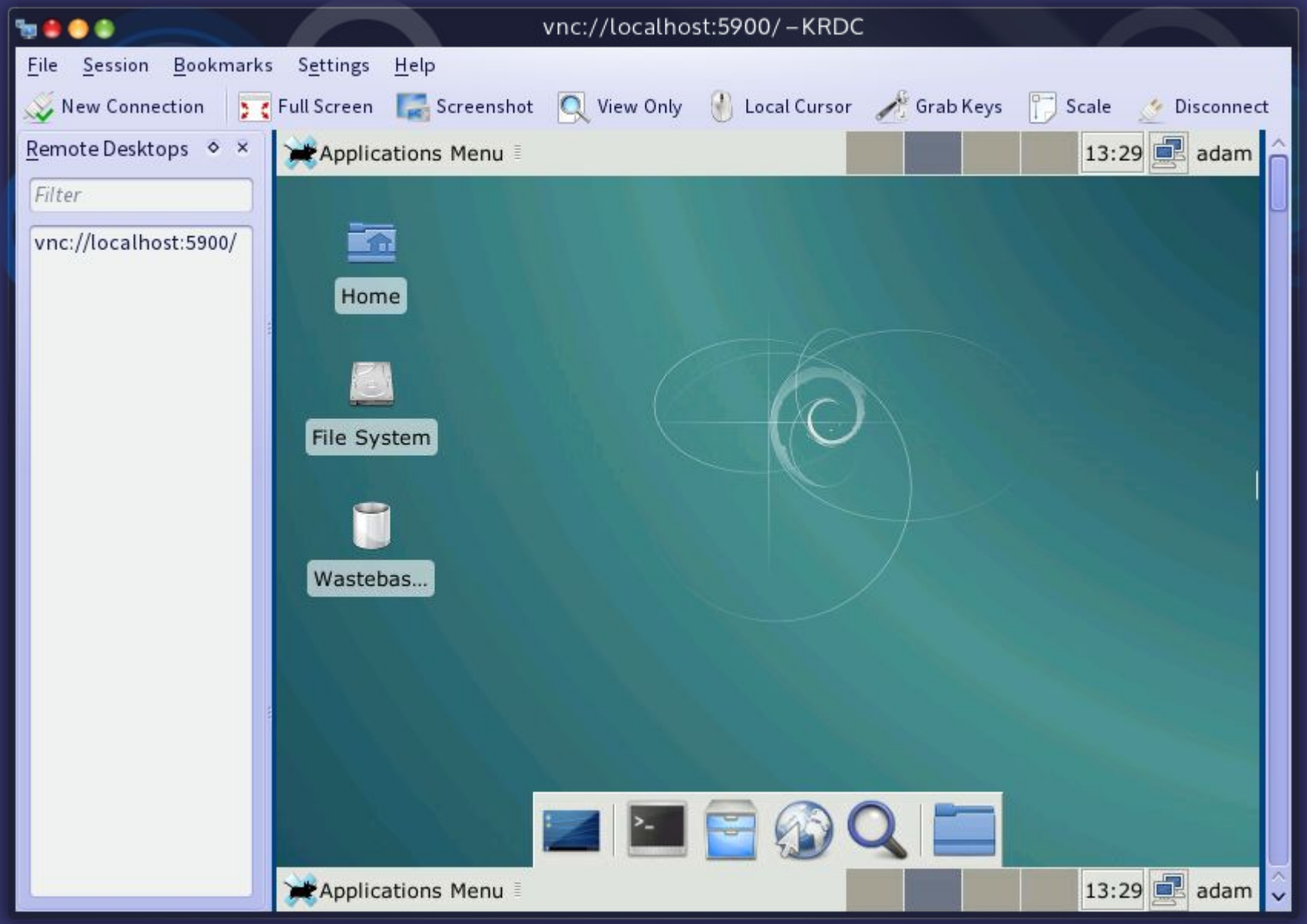


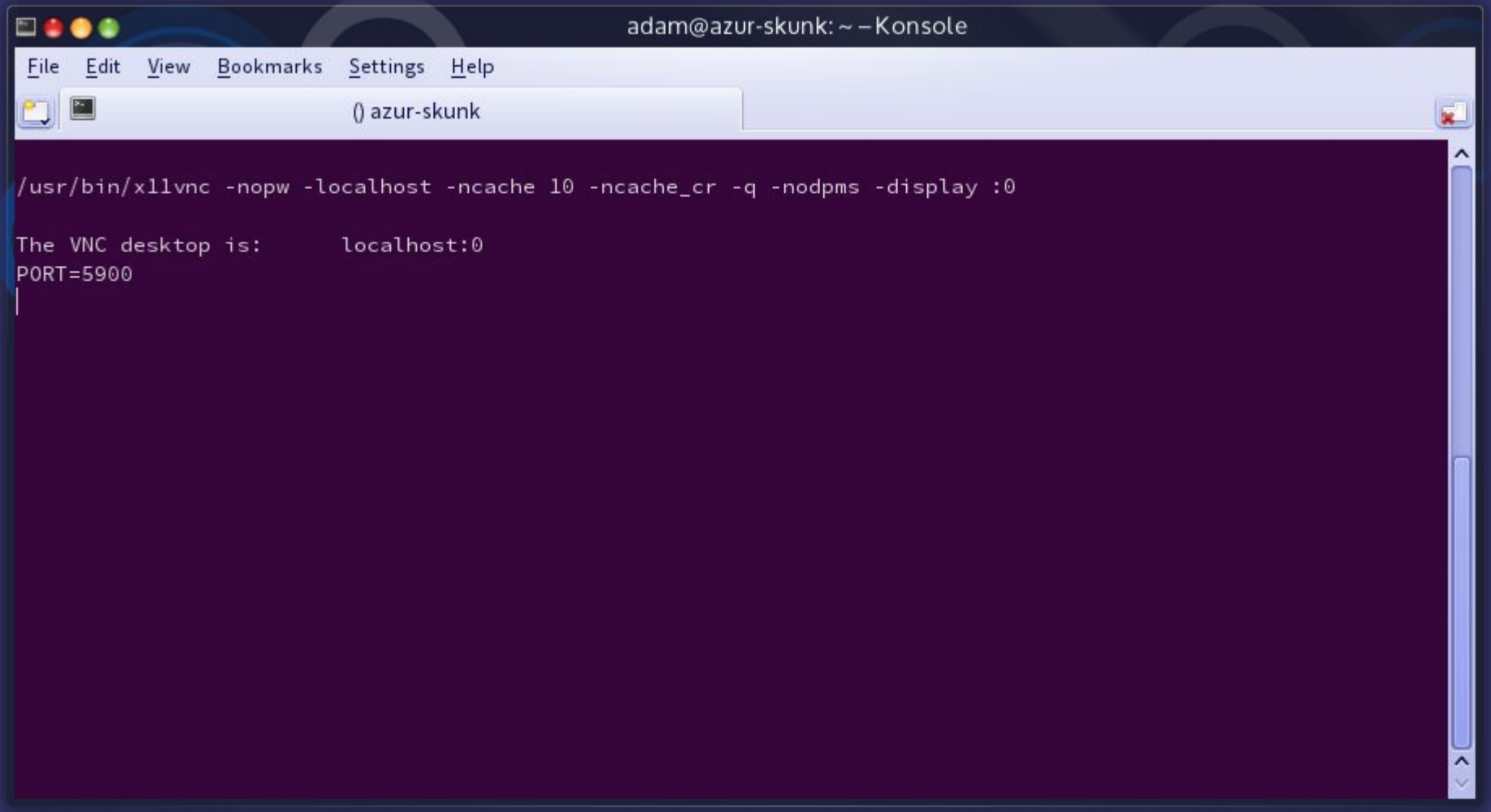


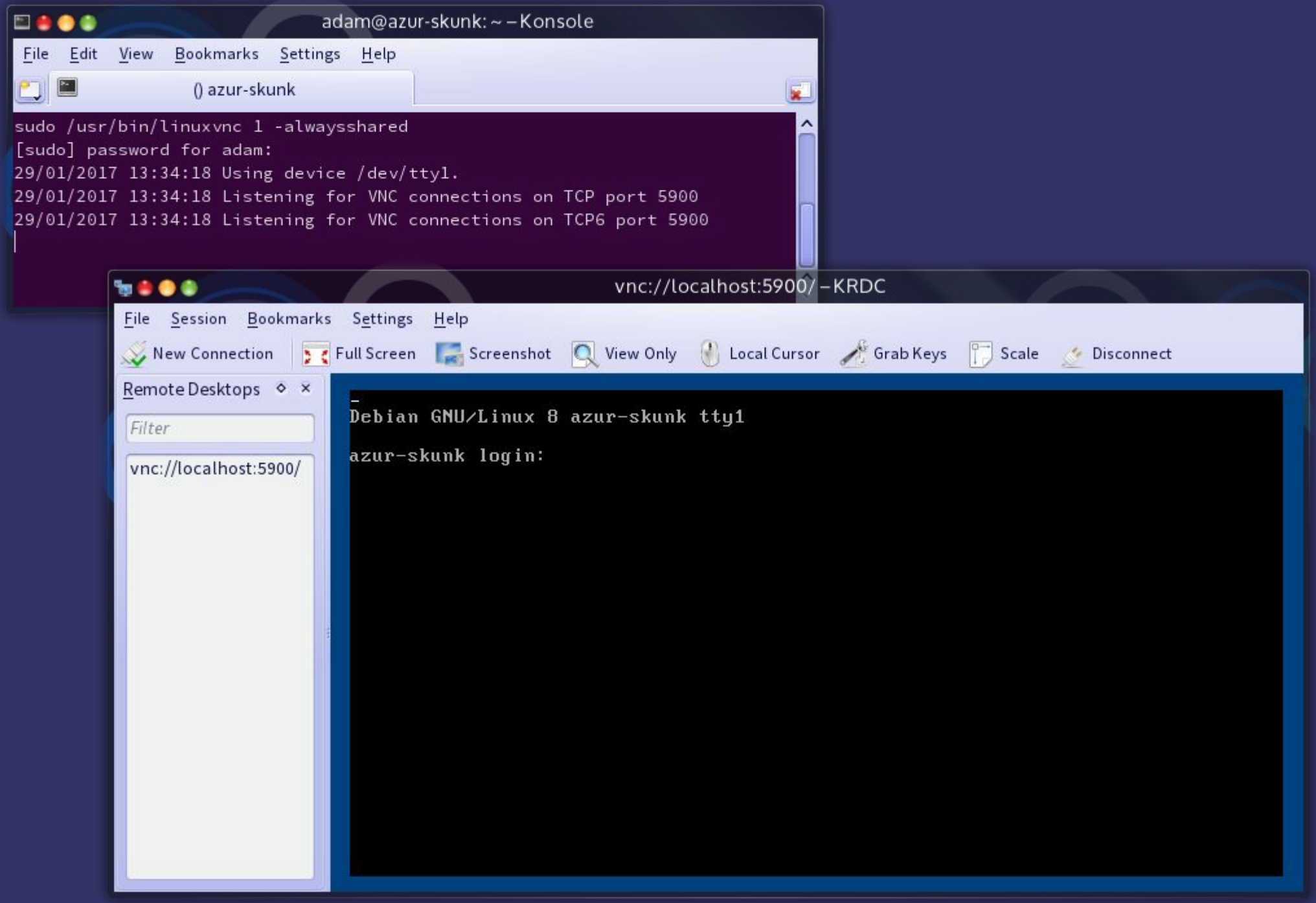






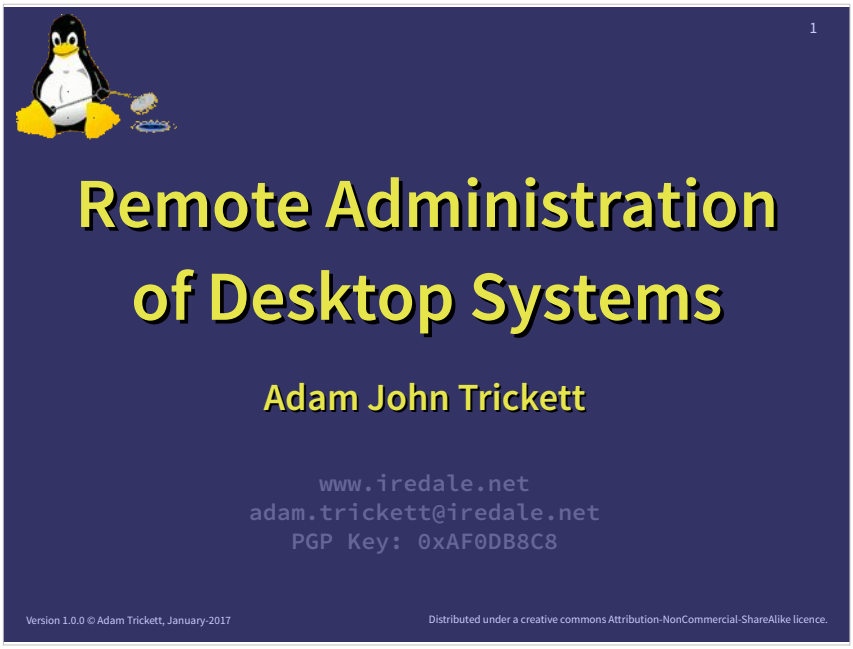






Thank You

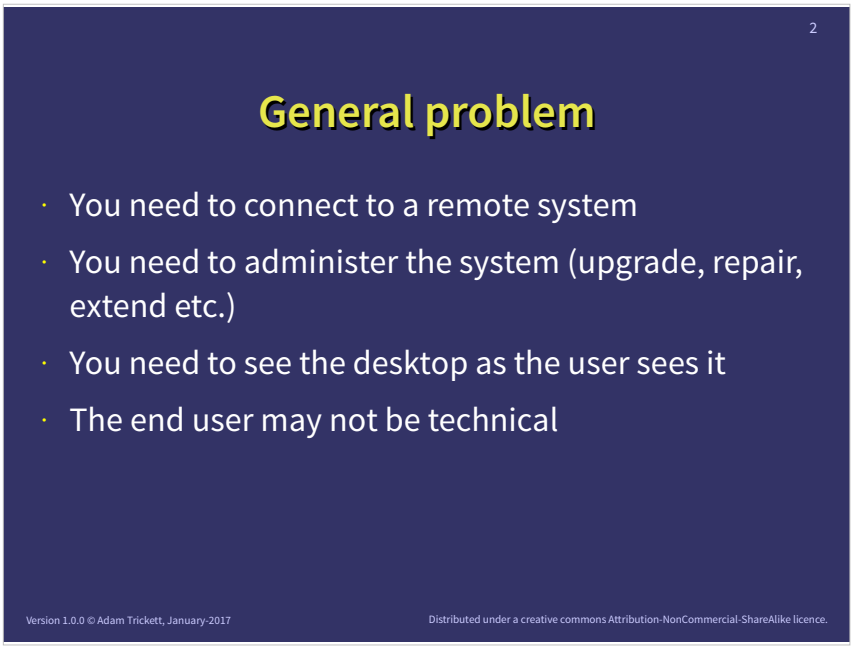
Any Questions?



Hello!

Hi it's me again.

This talk and all my other will be available on my web site, listed here. You can also email me any specific questions.

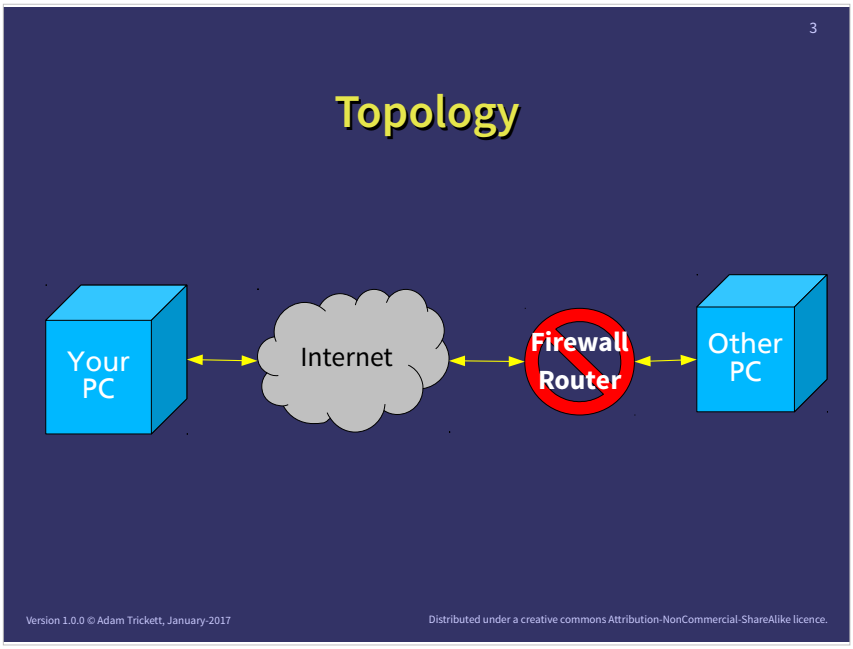


This problem arose many years ago when my mother used a computer – not being able to see what she was doing meant I couldn't help as I had no idea what she was doing or looking at.

When I gave my father a PC I went out of my way to get round this problem in a safe and reliable way that did not require any technical skill or input from my father.

The solution is generic and I've used it at work and to help friends out. In all cases it's the same issue:

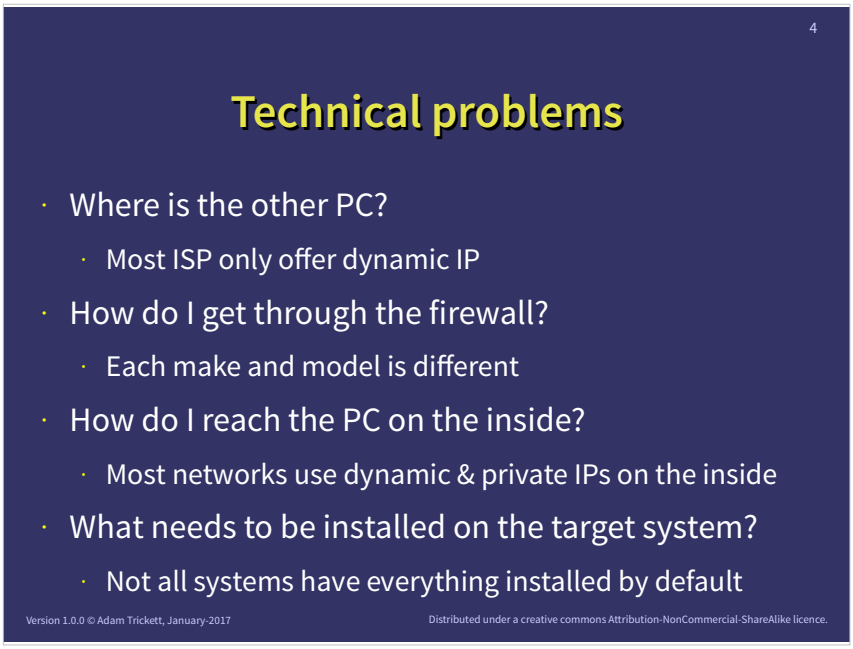
- **Connect in a safe way**
- **So stuff**
- **See what they see**
- **Fix things**



In most cases this is how we are connected. I've ignored my router/firewall for convenience and any firewall on the specific PC.

Basically I'm connected to the internet at all time via my ISP. That's pretty normal and nothing exciting.

The other computer should be connected by a router/firewall and may not be on all the time.

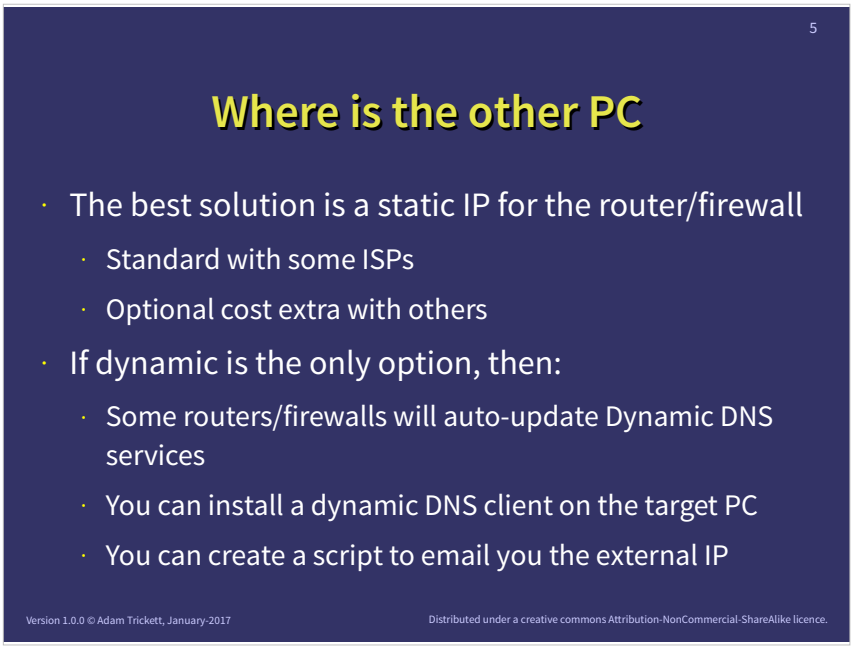


The first problem after getting the other computer to be turned on is where is it on the internet. In almost all cases it will have a private non-routable IP (10., 172.16 or 192.168.0) and a dynamic external (routabe) IP that maybe unknown to the internal machine.

The next problem is that the firewall shouldn't allow anything from the outside on the Internet into the private network which the PC lives on

By default the target system is probably on a dynamic IP on the private network, so even if I can get through the firewall, I don't know where to go on the inside to fine the right machine.

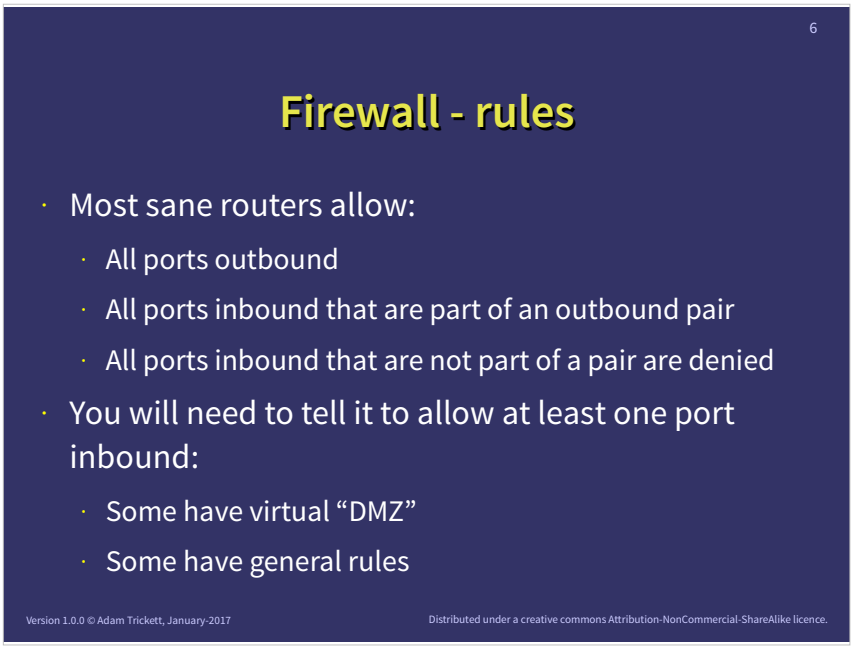
Finally unless the right software is installed and configured on the target system I won't be able to connect to it.



Most ISPs use dynamic IPs for most connections so that they can make efficient use of a limited number of IPv4 address available to them. Some offer a static IP which is more useful as it doesn't change from connection to connection – though you may have to ask for it and there may be an additional charge. The Phone and Broadband Co-operative charge 50p per month.

If you have only a dynamic IP, many routers can be configured to use a dynamic DNS service. So when they are allocated a new external IP address they connect to the dyn DNS system, and update their IP address. There are some free and some paid for options to do this.

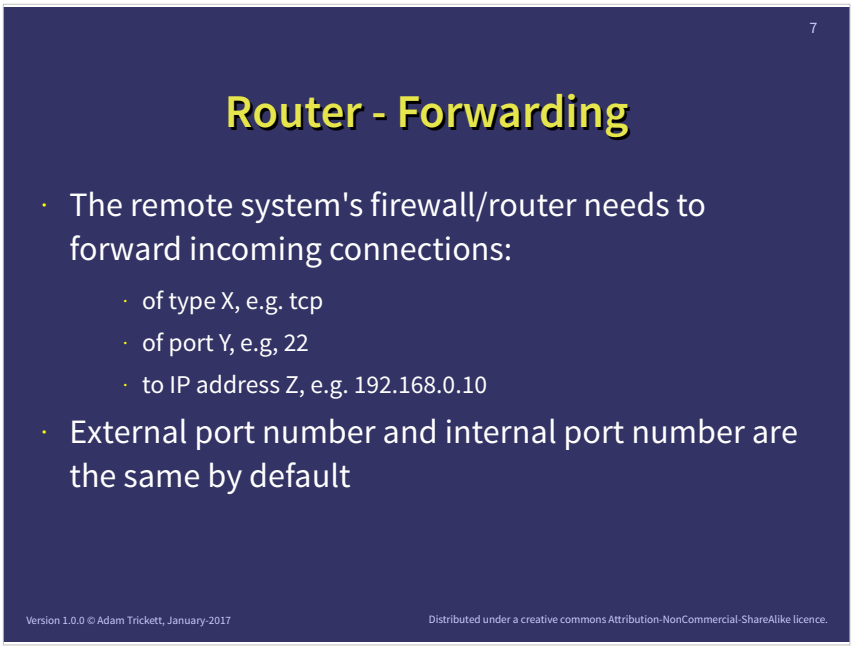
If your router doesn't have this option or not the one for the service you plan to use there are also scripts that can be run on the target PC and basically do a “What's my IP” query and them update the dyn DNS account from within the network.



Out of the box most consumer “modem”/firewall/router/WiFi boxes are set up to allow you out on the public Internet and the public Internet is NOT allowed back in.

Basically packets of data can go out of the network and only in if they belong to a a packet that just left. The packet matching is normally automatically turned on

To get into the target user's network you will need to open up a port on the outside and allow inbound traffic to come into the network. Depending on the router this may be a generic allow inbound port X or it may be directly connected to a routing rule to a specific IP address. It varies.



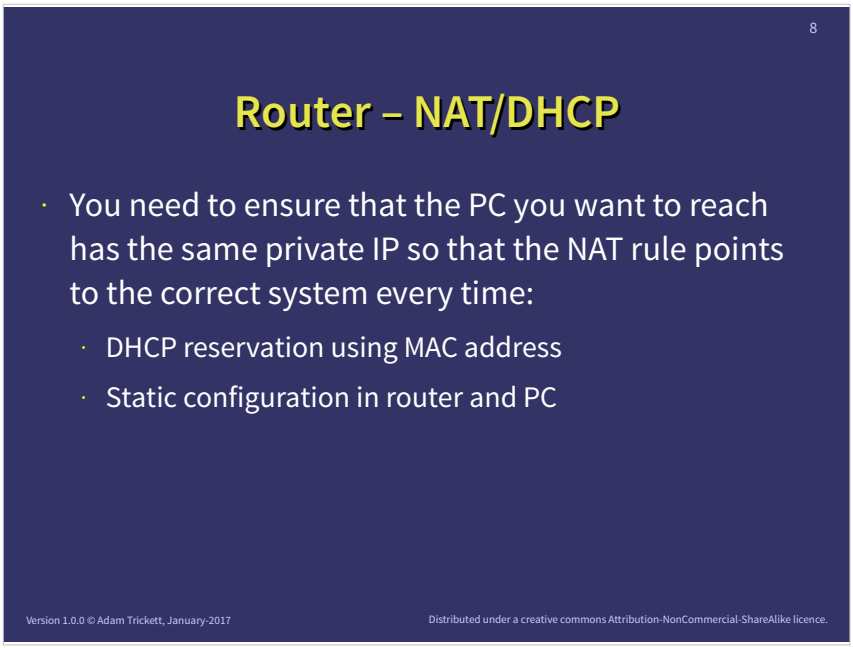
Most consumer routers running on IPv4 – which is the vast majority, use Network Address Translation, so the IP addresses on the inside are private and not the same as the external ones.

NAT connects internal address to the outside world and works out when a packet comes back which machine on the inside requested it so it can send it back there...

So once you have opened up a specific port, you need to send it some where on the inside – different routers use different rules, some have a virtual DMZ, some allow each individual port to be assigned to one IP, some have time rules and so on.

What you need to do is ensure that the port you want, e.g. SSH is open on the outside for the right type, e.g. tcp and is forwarded to the right IP address and port.

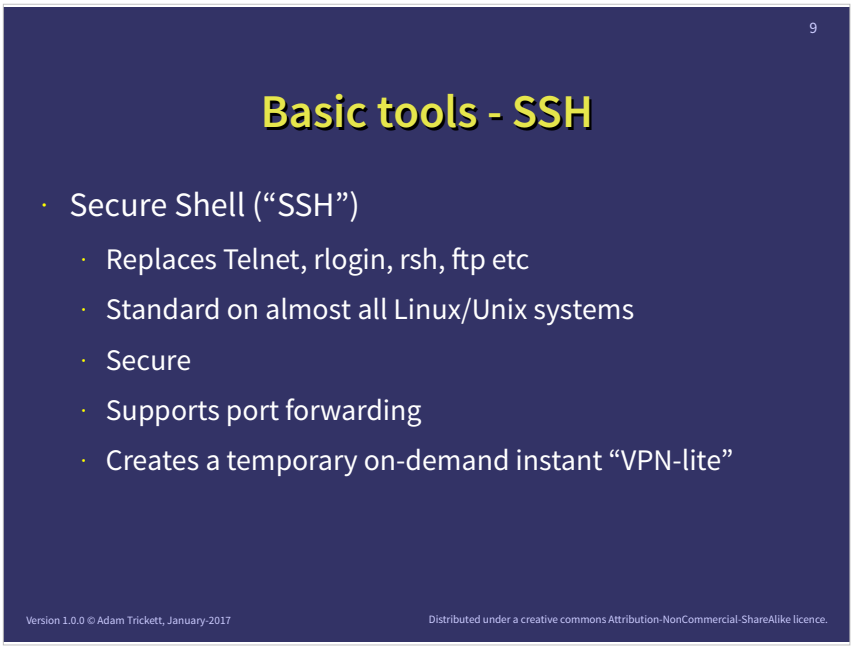
See https://portforward.com/ and other similar sites



By default most routers will provide a DHCP service which give a random private IP address to any machine on the private network that asks for one.

NAT ensure it can talk to the network and get it's response back without getting some other machines response back.

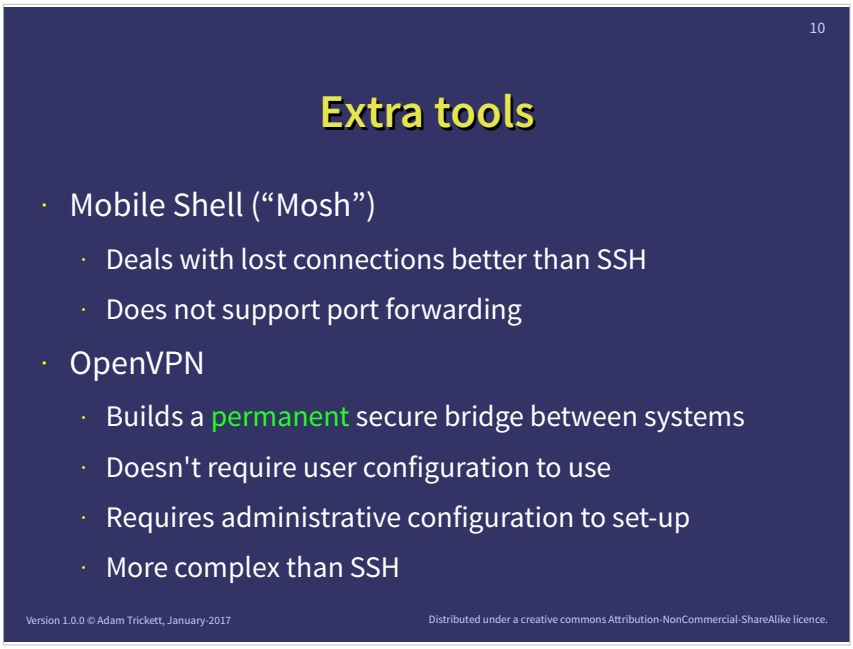
Because DHCP allocates at random you need to ensure that you know what the IP address will be so you can put something into the forwarding rule.



The modern default for connecting to another machine is SSH. It replaces almost all the older insecure tools and is the default on almost all Linux/BSD/Unix machines today. Windows doesn't have SSH but both client and server parts can be installed.

SSH does three things:

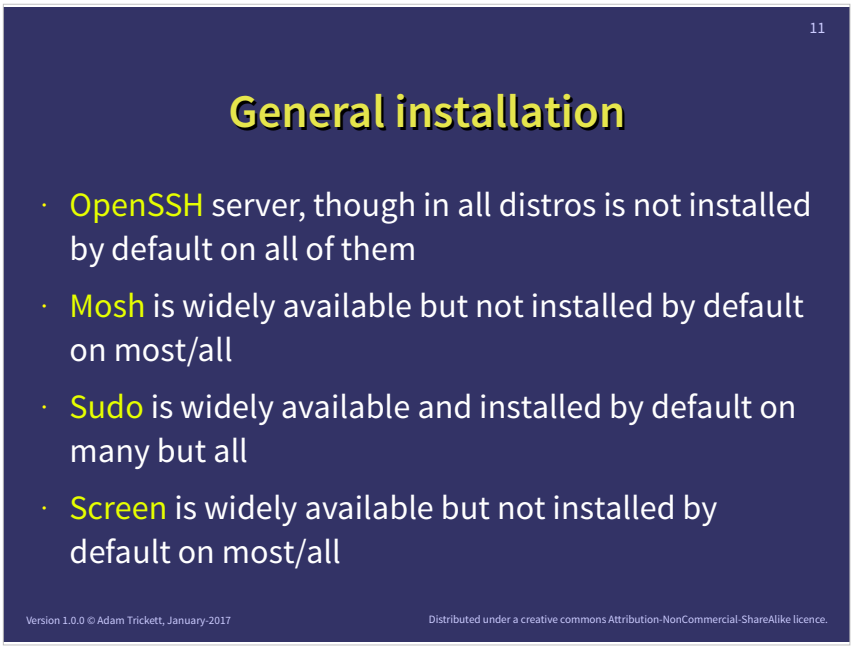
- **It connects to another system**
- **It connects in a safe and efficient way**
- **It connects and allows you to route other traffic through the connection at the same time**



Mosh is also useful to have a know. It doesn't do the port forwarding trick that SSH does but is is good at dealing with rubbish networks, e.g. public WiFi which is unreliable and keeps dropping and connecting.

An alternative to SSH with port forwarding and Mosh is a complete VPN, such as OpenVPN which is widely available. It is more complex to set up and configure than SSH, but once in place it is a permanent solution that can bridge your networks. Being permanent may or may not be a good thing!

I'm not really going to talk much about OpenVPN, because I know SSH better and it's more useful for what I want. However if you have OpenVPN configured then the VNC solution will work perfectly over it too.

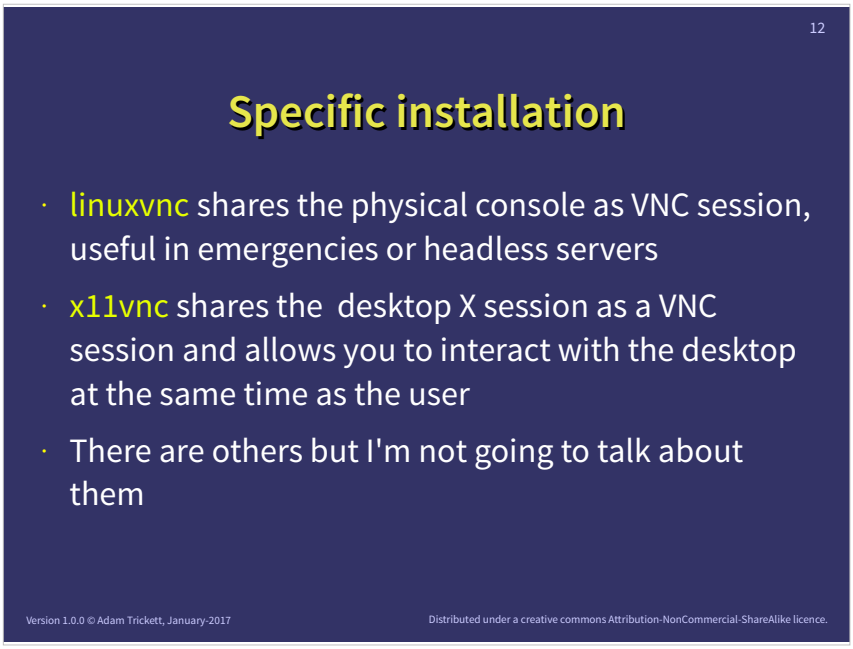


There are several SSH clients and servers available but almost all Linux and BSD distributions come with OpenSSH from the OpenBSD people. It is not automatically installed and the server and client are now in separate packages in Debian and derived distros. Look for openssh-server and openssh-client.

Mosh isn't installed by default but is useful to have, you will need SSH as well. Search for mosh.

It's never a good idea to login as root and not a good idea with SSH or Mosh so I strongly recommend you login as a user and then use sudo. Sudo is default on some distros and not on others, search for sudo.

Even if you are using Mosh, it's still worth using screen, so you can maintain several things at once with only on connection. It's not installed by standard on most systems, but it is pretty common.



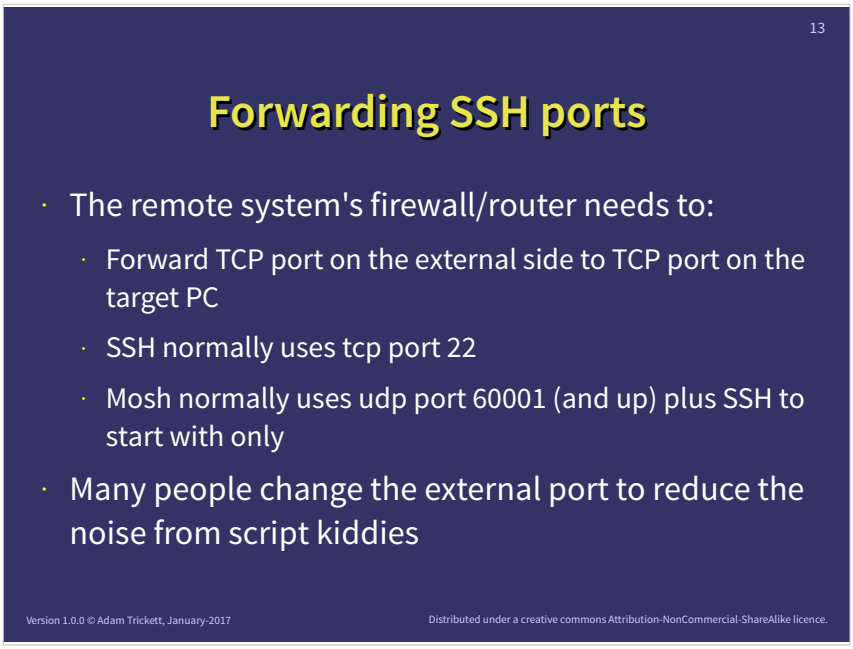
I recommend you install these two packages

Linuxvnc is a tool that turns an actual terminal screen into a VNC session. Sometimes it is all you have so it's useful to have as well as the more common X11 version.

x11vnc turns the X11 desktop into a VNC session. It can be used to start a new logon screen or more usefully "capture" the currently running one. It allows both the local and remote user to see the same thing and both an interact with the desktop at the same time.

On it's own the RFB/VNC protocol is quite efficient but it has no security built in – though some packages do provide this, but I atill I recommend only using it via SSH.

There are alternatives such as NX from NoMachine and RDP and so on but I'm not going to talks about them



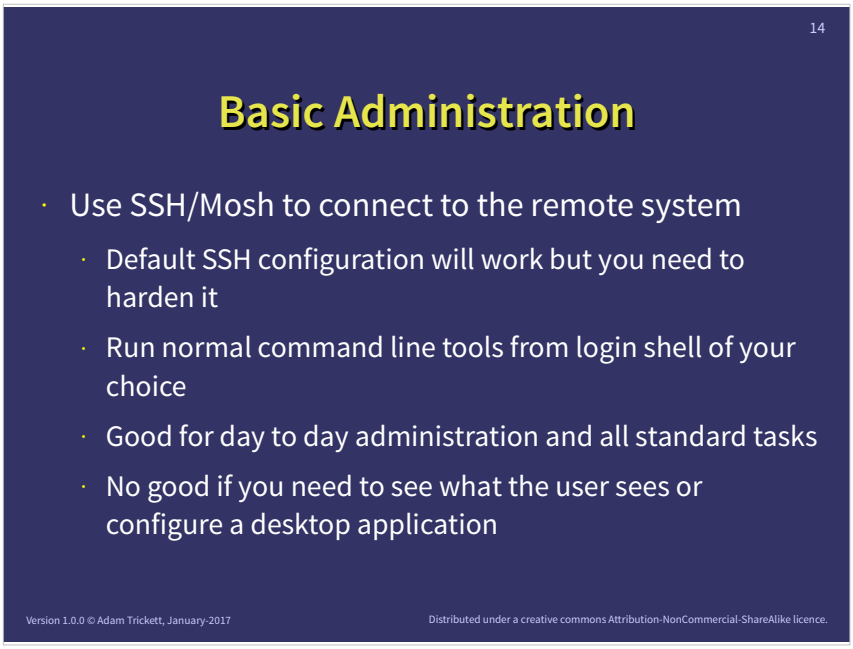
Assuming you are using SSH then you need to set your firewall/router to forward the ports.

SSH uses TCP port 22 by default, you can change the port number if you want – it will reduce the number of script kiddies doing port scans but it won't block a determined attacker, even an automated robot.

Assuming you have your target PC set up with a known IP on the inside it should be a simple case of creating a rule in your router to forward TCP port "whatever" to internal IP target on TCP port 22.

If you want to support Mosh as well you will need a single UDP port open as well, Mosh uses 60001 by default.

If you are only using SSH that's just one open inbound port – but be aware it's one that lots of people are looking for!

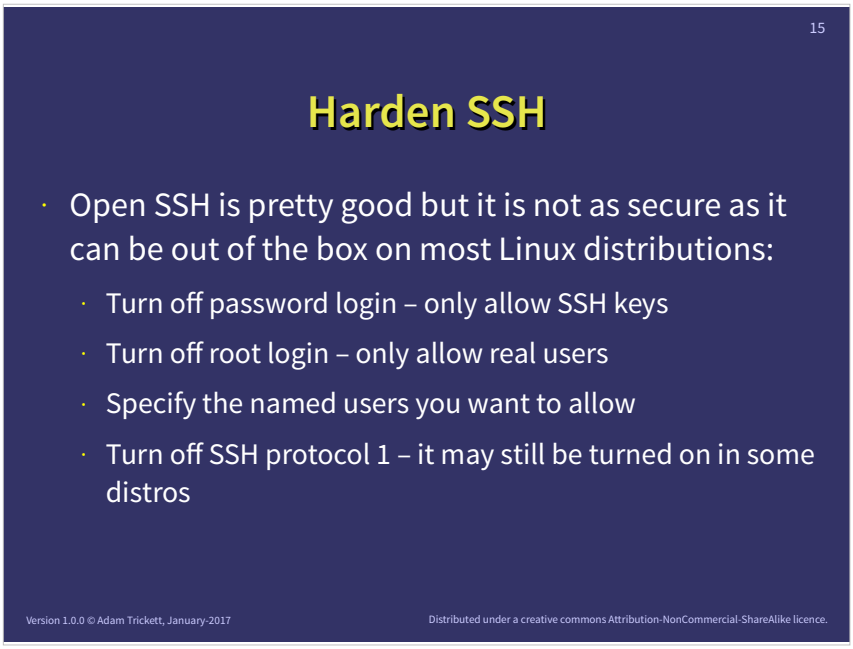


You can now use SSH or Mosh to connect to the remote system.

Probably best to “harden” SSH without using the public internet first if you can, but even if you can't you should do it as soon as you gain access.

You should now be able to do all the normal package installation tasks, user maintenance, system tweaks and optimisations. If you have SSH and X11Forwarding turned on you can also run GUI applications on your screen if you need to.

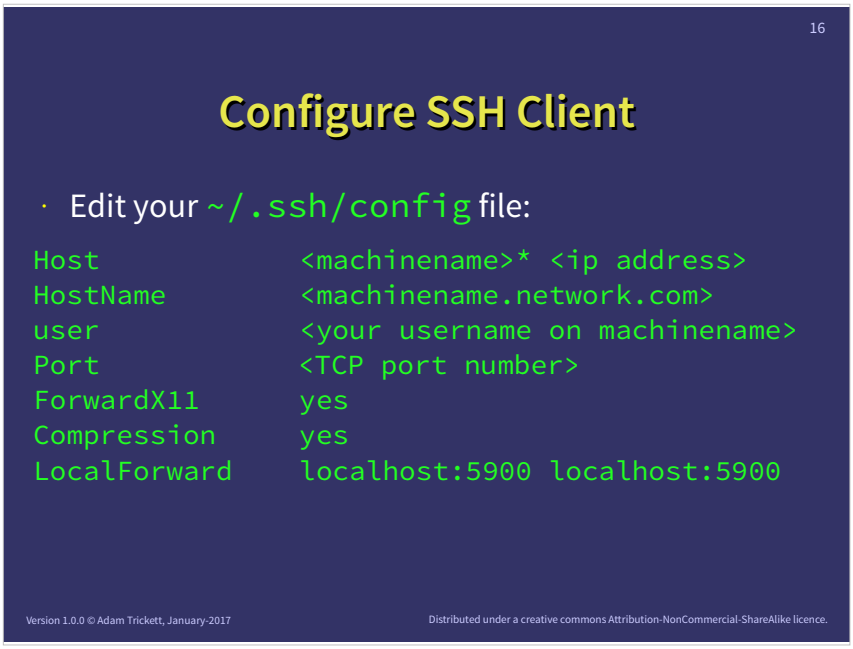
Basic SSH access won't allow you to see the screen of the loged in user user though.



I STRONGLY recommend anyone running a system connected to the Internet that SSH out of the box on a Linux system is not secure enough to keep automated robots and script kiddies out, if you have weak passwords or an old version of SSH.

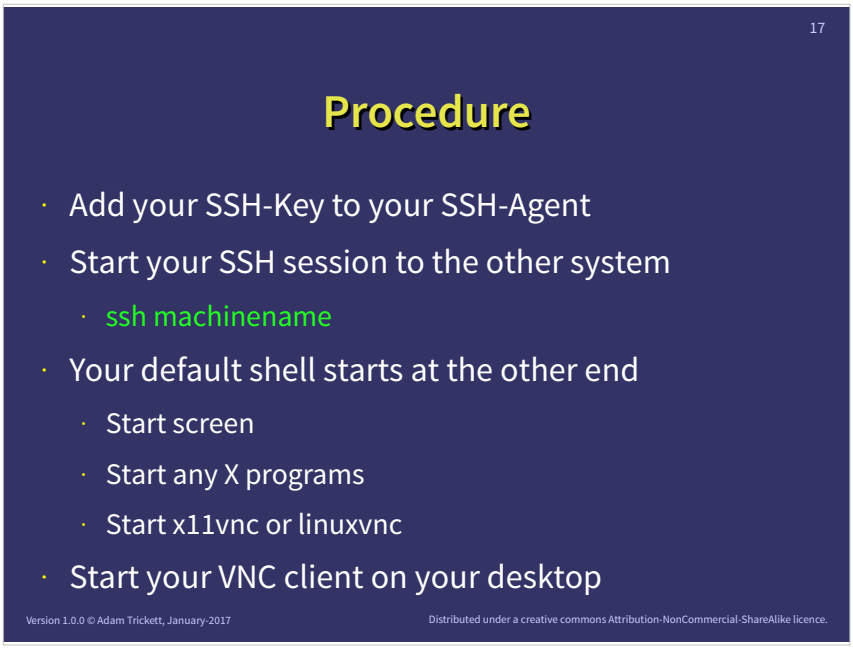
Your first task is always to harden the box, ideally before you expose it to the network:

- **Passwords are easy to guess so turn them off by default – only allow SSH-Keys which are MUCH more SECURE**
- **Don't allow root to remote login under any circumstance – there are special exceptions but they are very rare**
- **Only allow a limited list of named user account to login via SSH**
- **If SSH version 1 is allowed, make sure you disable it, only version 2 is safe.**



All of these settings can be done via the command line but putting your settings into a configuration file makes things a lot easier for you.

- The Host and HostName lines allow OpenSSH to confirm which system gets which set of rules
- If your username isn't the same on your PC as the remote one then you can specify it with the user command
- If you are using a port other than the default of 22 you can specify it here
- ForwardX11 isn't essential but useful
- Compressing is less important these days but can still help if you have a slow connection
- LocalForward is the bit that will make VNC work – this is not on by default and you will have to add something like it to make VNC work



I logon to my PC with SSH keys using pam-SSH but if you don't you need to add your SSH key to your SSH-Agent.

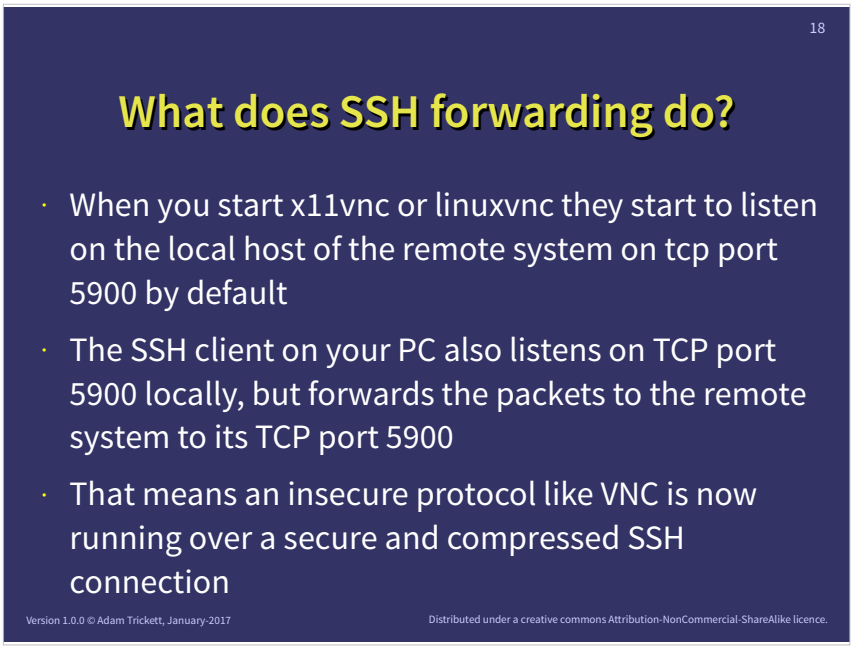
Start a console and execute your SSH session. Some desktop systems allow you to create a terminal profile that does it all in one go (Konsole in KDE for example)

A few seconds later you should be presented with your default shell prompt on the remote PC.

I recommend starting screen now, so you can do more than one thing at once, including running programs that don't let go of the console.

You can start X11 programs now, e.g. xterm

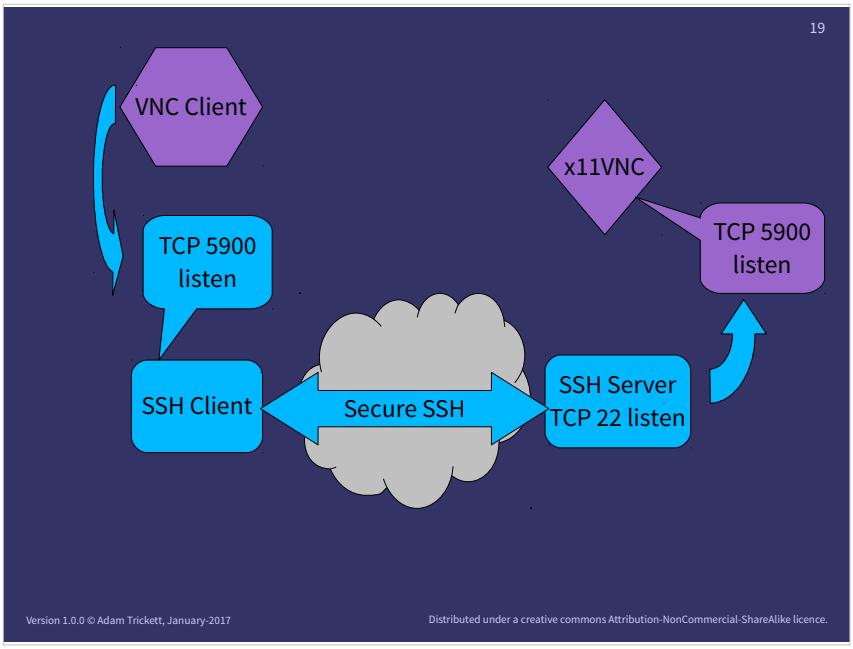
Finally start your x11vnc session on the server and connect to it from your desktop system, e.g. krdc



When you connect with a SSH forwarding you have created a secure tunnel between the SSH client running on your PC and the SSH client connected to the SSH server on the target system.

That means your SSH client is now listening to local port on your PC and then sending that data to the corresponding port on the target machine.

So when your VNC client speaks to localhost:5900, SSH forwards that to remote machine and connects it to localhost:5900 which is where a VNC server is listening.



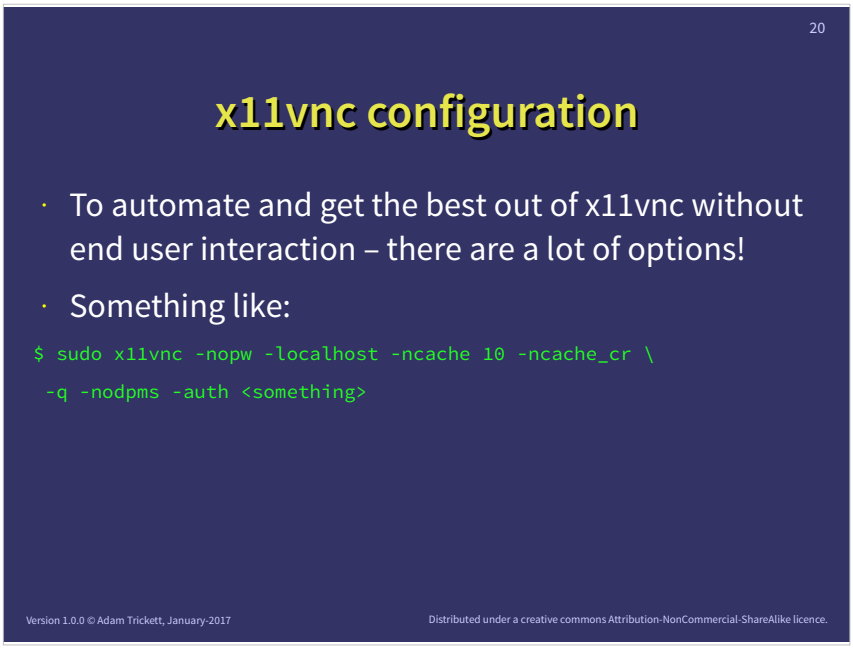
You point your VNC client at the local system on port tcp_5900

The SSH client is already listening on that port, picks up the request, and sends it down the tunnel to the other end.

At the other end it tries to forward the data packet on to a local port tcp_5900, where the x11vnc program is running.

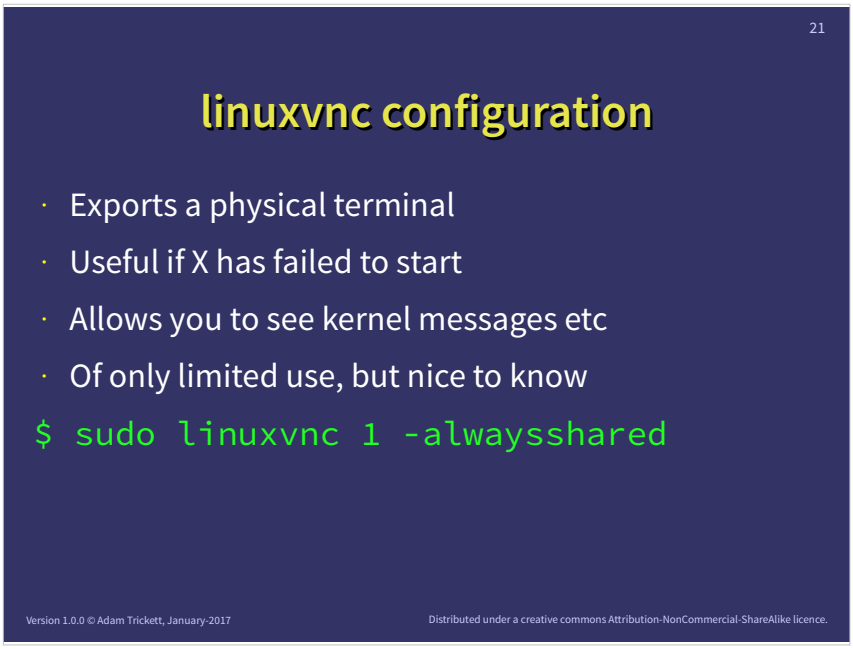
Your VNC client thinks it's talking locally on port tcp_5900 and to your SSH client, but practice it's actually talking to a VNC program somewhere else.

This may sometime confuse VNC as it expects the data to flow very quickly – so you need to make sure you tell the connection that your are on Internet speeds not LAN speeds.



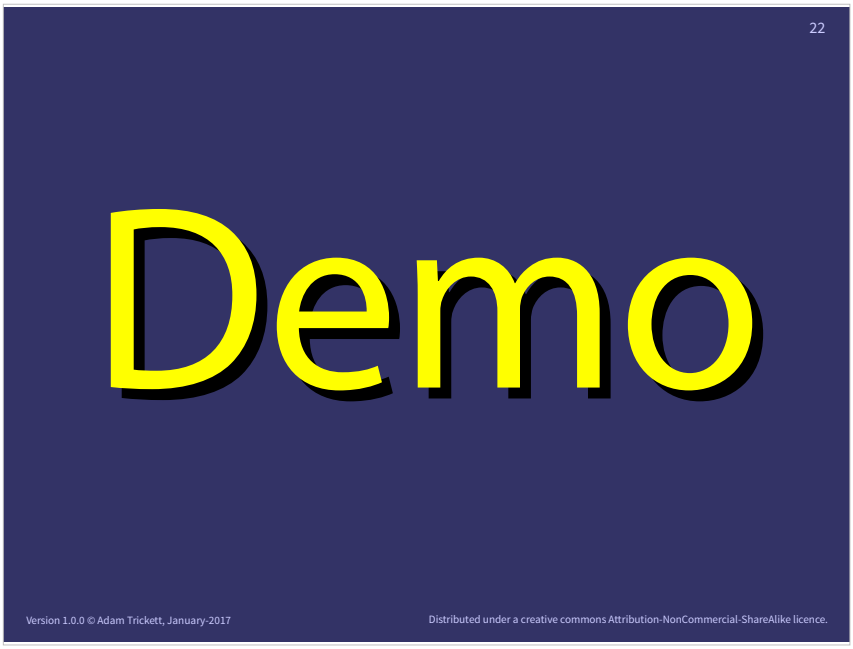
x11vnc can be run as the logged in user, in which case it automatically finds their X session.

You can also run it as root, and let it figure things out, the incantation shown here will work if you are using lightdm as commonly used by Debian, before the user has logged in.



Linuxvnc is cute and useful under only some circumstances.

It takes a physical console and turns it into a VNC session that can be shared. In most cases it's not much use, as you already have a pseudo console session via SSH or Mosh, BUT if X has failed to start and there are messages on the screen it's useful to be able to see them.

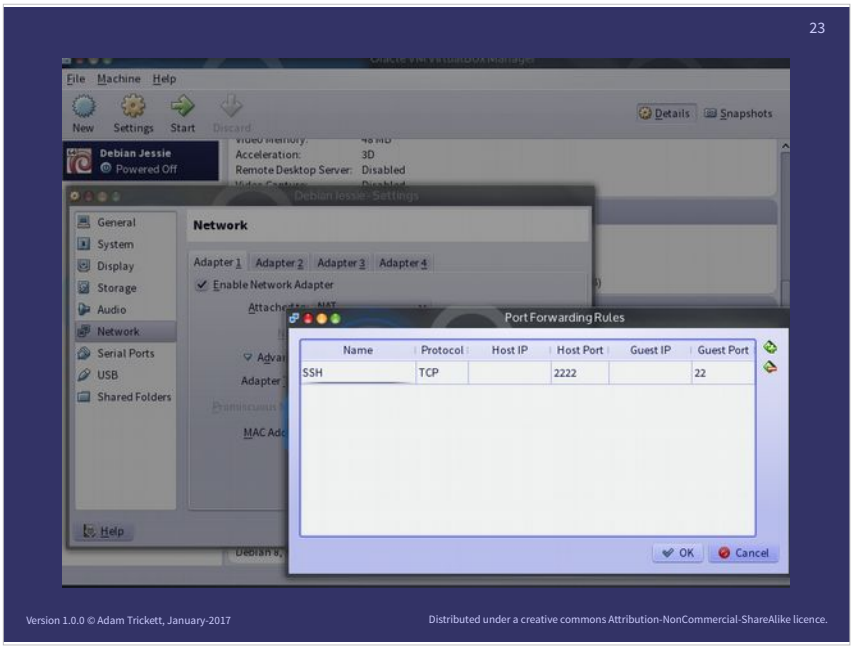


For the demo I didn't bring two laptops – that seemed like a lot of weight to cart about! So I will SSH onto a Virtual machine, that we can treat as a remote systems.

I will use my local laptop as the source system, it has the SSH client installed, and I login using my SSH keys (via pam-SSH) so they are installed in my SSH-Agent automatically and before I even started this talk.

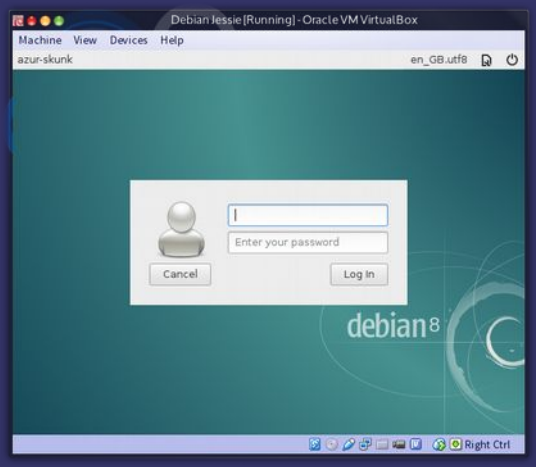
The virtual machine is running in Oracle VirtualBox, KVM/Qemu would work just as well on this laptop, but it's easier to show the router/firewall settings as part of the demo.

The target system is a stock Debian session, nothing fancy installed in it, other than the bits I wanted to add to make this demo work.



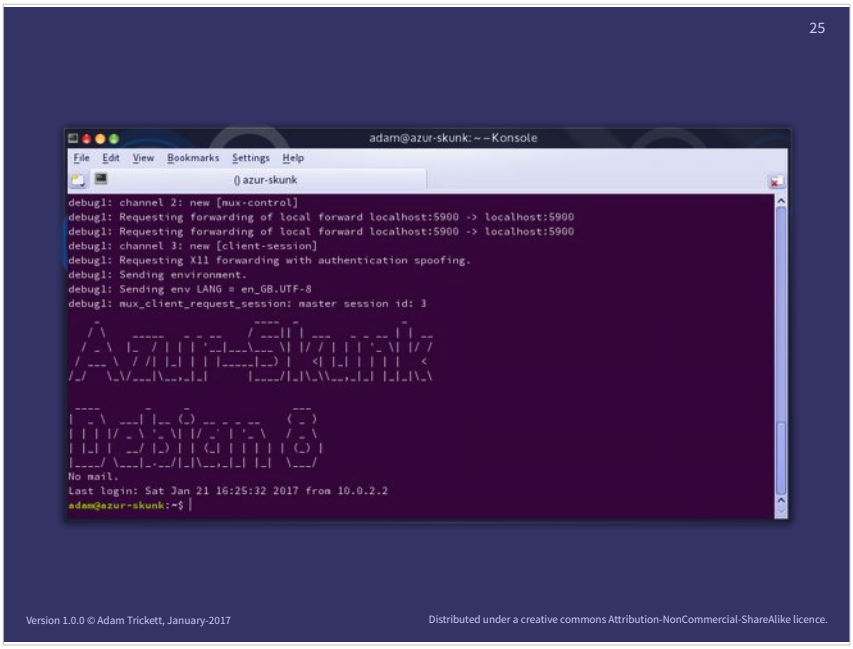
I'm using VirtualBox to simulate a system behind a firewall.

Here you can see I'm forwarding TCP port 2222 to port 22.



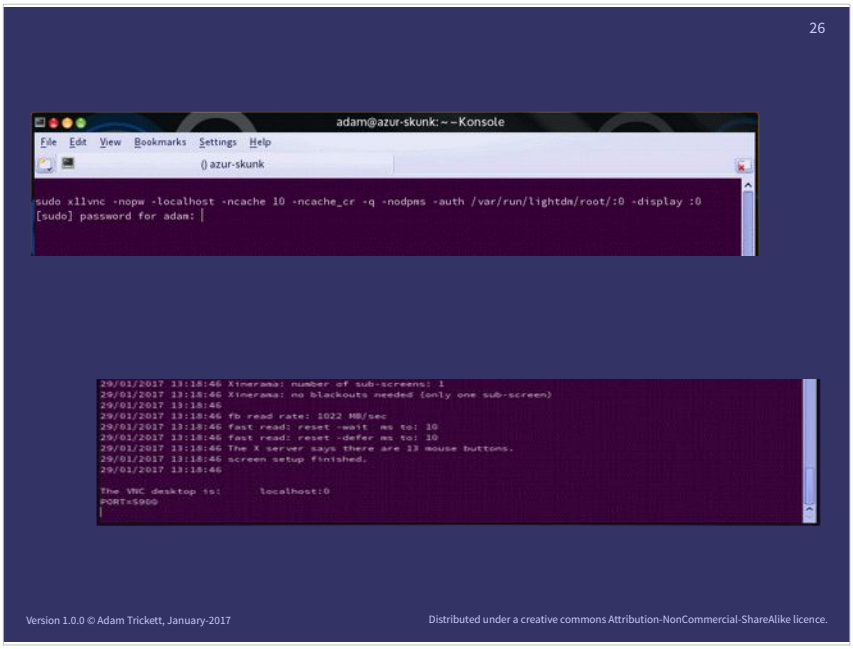
Here is the box booted up in VirtualBox.

No one has logged in yet.



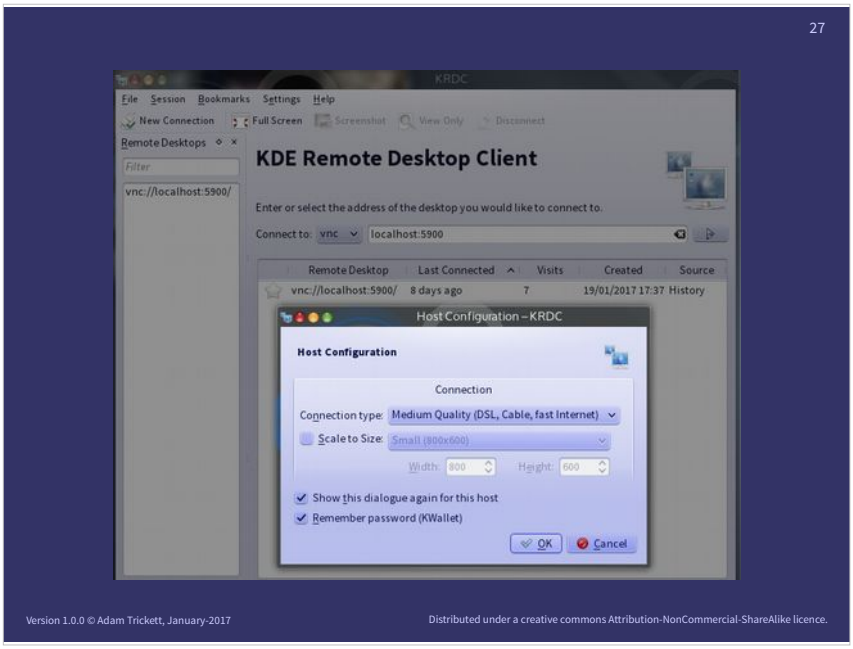
I've started a console session on my local machine and SSHed into the remote/virtual box.

Note I've used ssh -v “verbose” so you can see the port forwarding rules in the command stack.



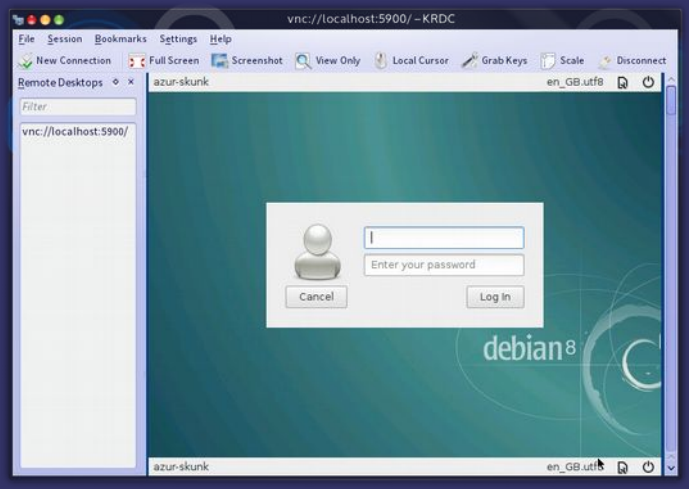
Starting the Xlogin session capture into X11vnc from the SSH session.

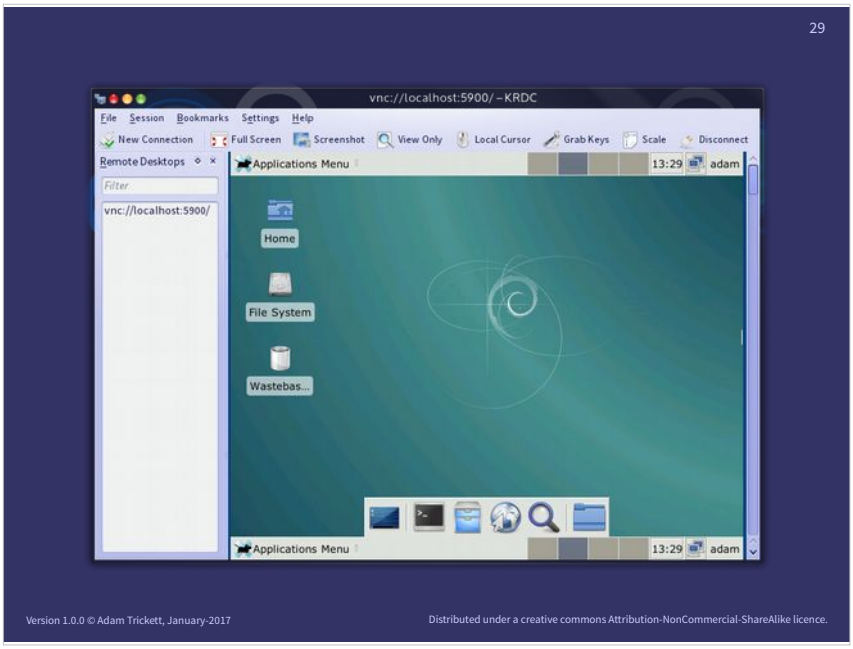
Top show command line, bottom shows result.



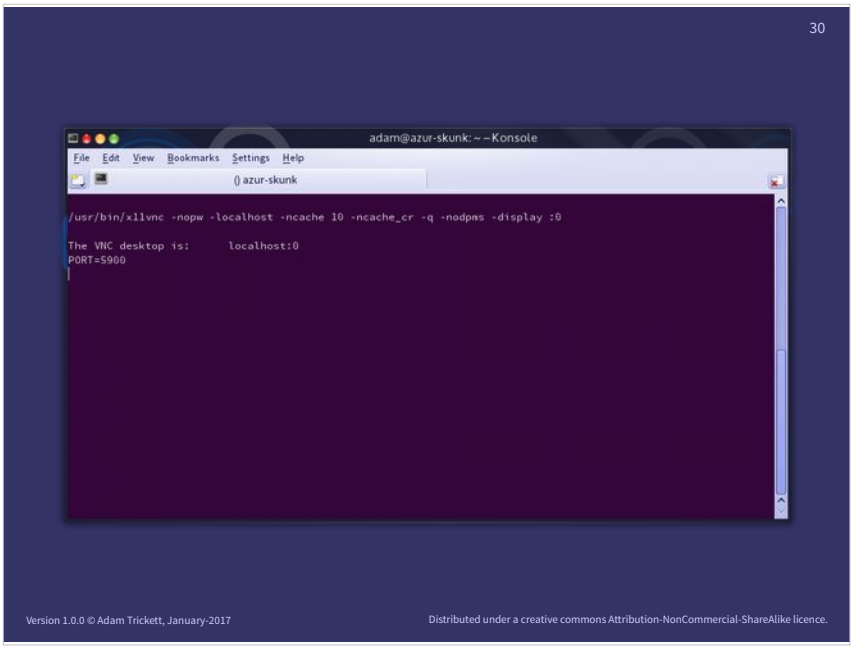
Here I'm using KRDC as a VNC client to connect to localhost port 2222, which is my SSH tunnel connected to the virtual/remote machine.

Most VNC clients default to high speed if they think they are going local, that isn't a good idea over the public internet.



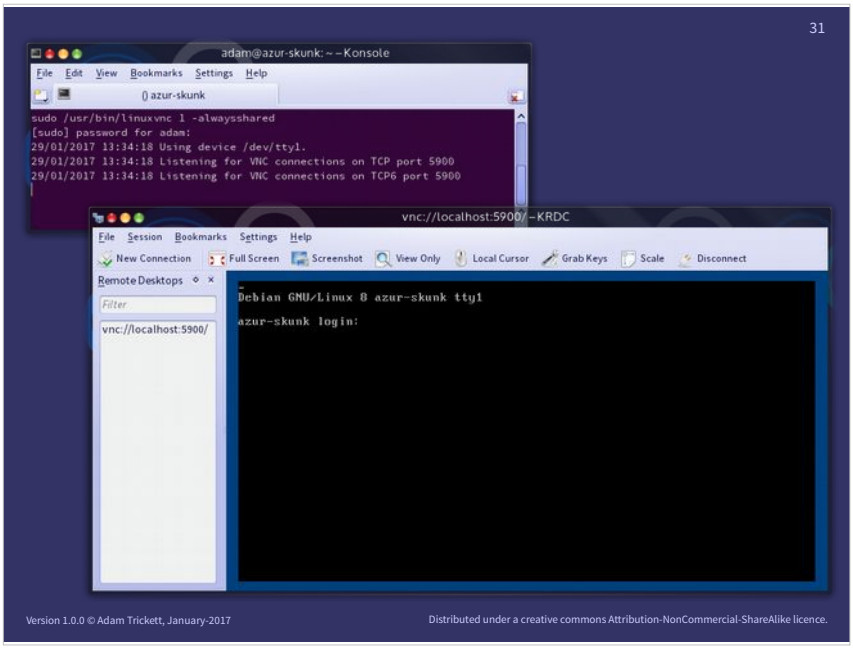


You can login as if you were looking at the machine.



Starting X11vnc from a running session.

Result as before.



For completeness here is linux vnc

Please feel free to ask anything...!