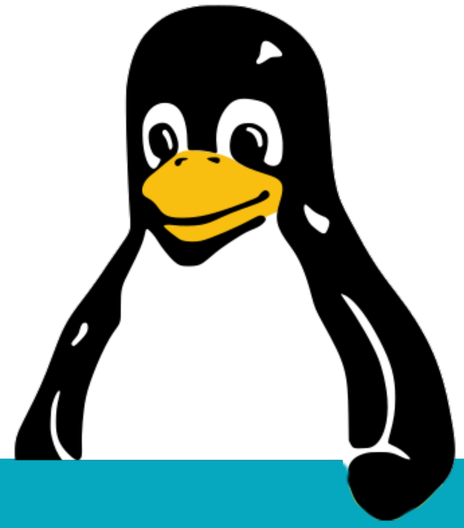
# Linux, day 2
# Advanced SSH operations



# SSH and its possibilities

# Secure SHell

- We just met "SSH", the Secure SHell.
- Currently Linux’ most popular way of remote login.
  - Offers encrypted communications.
  - Lots of cool options.
  - Lots of dangerous functionality.

# SSH authentication

- Logging in we can use:
  - Password
  - SSH key pair
  - Certificates
  - MFA (multi-factor auth.)

# You try!

- From your host OS, SSH to the VM.

```
$ ssh -p 2222 tess@localhost
```

- Or from the Linux VM, *to* the Linux VM.

```
$ ssh tess@ubuntu
```

# SSH key pairs

- Asymmetric encryption
  - Private key remains on your "source" host.
  - Public key distributed to all "target" hosts.
- Login uses your private key to encrypt a secret,
  - Which the target host verifies with your pubkey.

See: SSH keys for dummies

# You try!

- On your Fedora VM:

```
$ ssh-keygen -t rsa
```

- Copy the public key to your Ubuntu VM:

```
$ ssh-copy-id -i ~/.ssh/id_rsa.pub \
  tess@ubuntu
```

# You try!

- Now you can login with the private key!

```
$ ssh -i ~/.ssh/id_rsa \
  tess@ubuntu
```

# SSH is a "pipe"

- We use SSH to login and run a shell,
  - But we can do so much more!
- SSH offers an encrypted pipe for:
  - Shells and commands
  - File transfer
  - Port forwarding

# Transferring files

- MacOS and Linux have the tools available.
- Windows needs WinSCP.

See: Download WinSCP

# You try!

- Sending a file from the host OS, to the VM.

```
$ echo "Hallo" > ~/testfile
$ scp -P 2222 ~/testfile tess@localhost:/tmp

$ rm ~/testfile
```

# You try!

- You can transfer files with SFTP (again, from host OS).

```
$ sftp -P 2222 tess@localhost
sftp> cd /tmp
sftp> lcd ~
sftp> lpwd
sftp> get testfile
sftp> bye
```

# You try!

- You can run one-off commands.

```
$ ssh -p 2222 tess@localhost "ls -al /tmp"
```

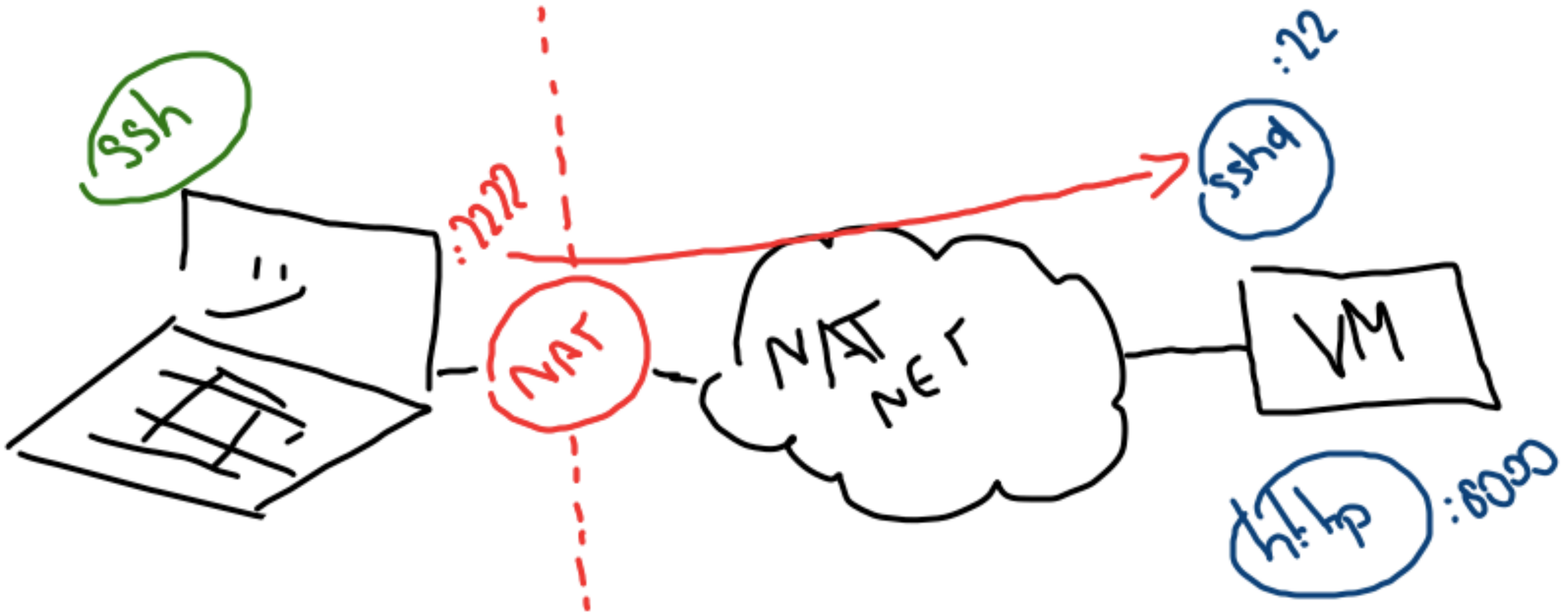
- Ansible is built on this!

# Scary stuff!

- SSH can be used to setup port forwards.
  - Both forward (outbound, from the source)
  - And reverse (inbound, from the target)
  - X11 as a special use-case
  - SOCKS5 proxy for fun-and-profit

# SSH port forwarding

- Let's access a web server behind NAT!



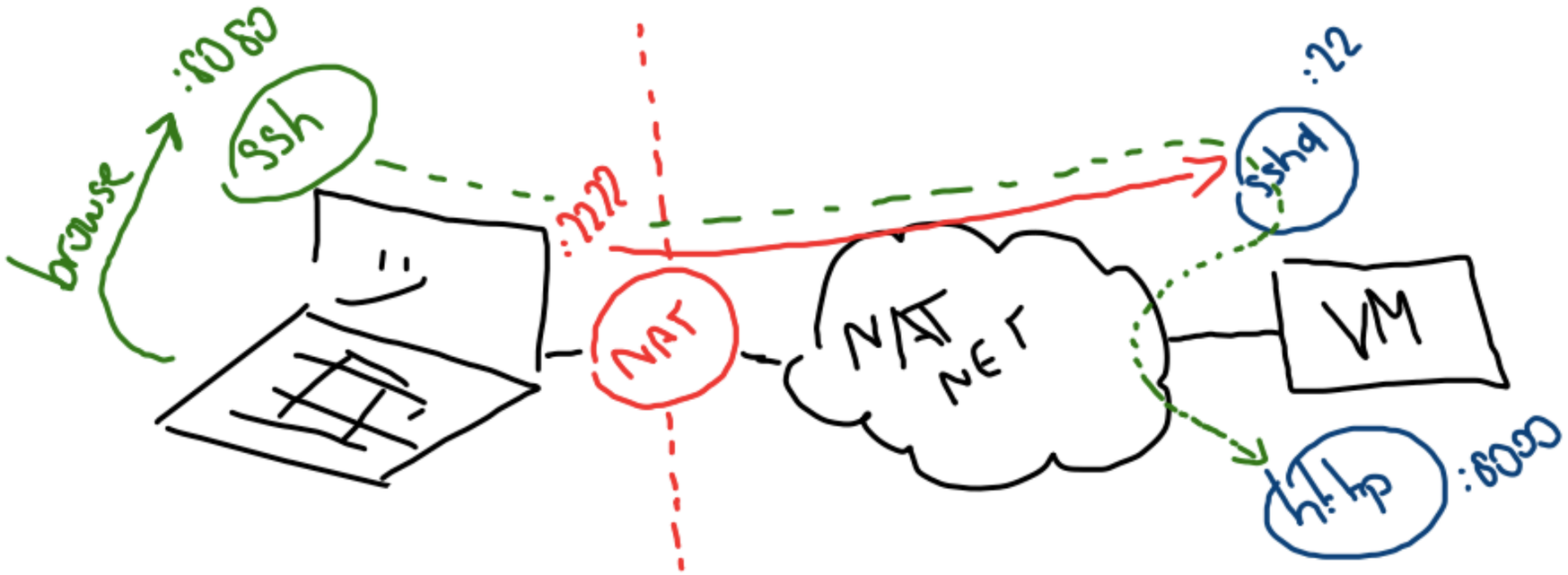
# SSH port forwarding

- On your VM run:

```
$ cd ~/Downloads
$ echo "Secrets!" > index.html
$ python3 -m http.server 8000
```

# SSH port forwarding

- Let’s lay some pipes!



# SSH port forwarding

- Defining a port forward with -L:
  - <local port>:<target IP>:<target port>

- This means:
  - SSH to the remote host, then build a forward.
  - Traffic will flow through the remote SSH box.

# SSH port forwarding

- On your host OS run (adjust guest IP):

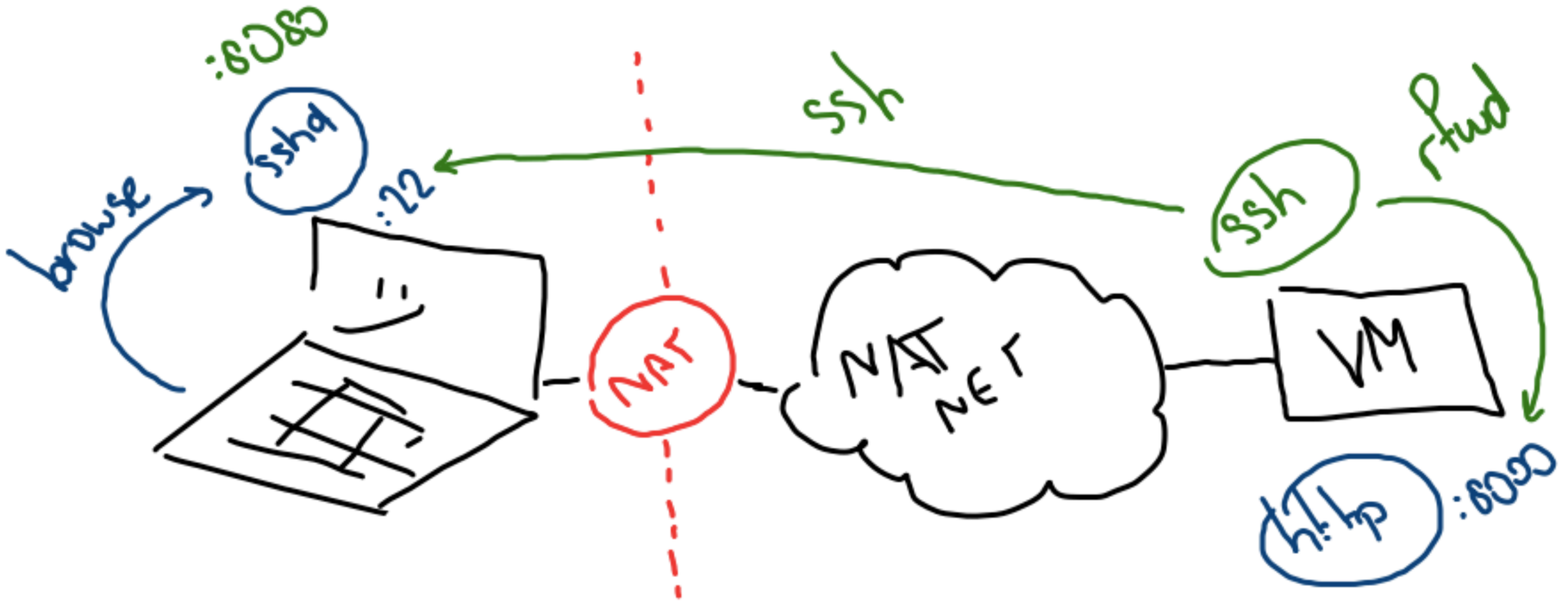
```
$ ssh -p 2222 -L 8080:10.0.2.15:8000 \
    tess@localhost
```

- On the host OS browse http://localhost:8080

```
$ curl http://localhost:8080
```

# SSH reverse port forward

- Even scarier…



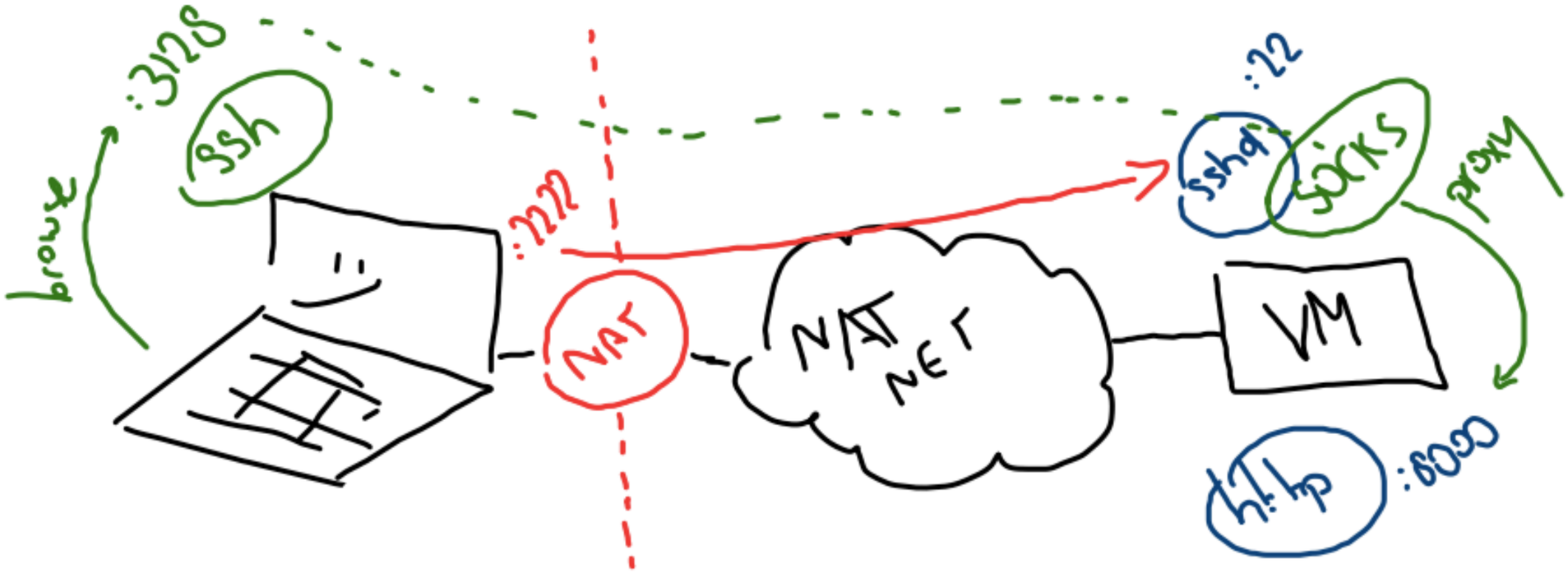
# SSH reverse port forwarding

- On your host OS install & run an sshd.
- Find your host OS IP address.
- On the Fedora VM run:

```
$ ssh -R 8080:10.0.2.15:8000 tess@laptop
```

- Then on the host OS, browse http://localhost:8080.

# SOCKS5 proxy



# SOCKS5 proxy

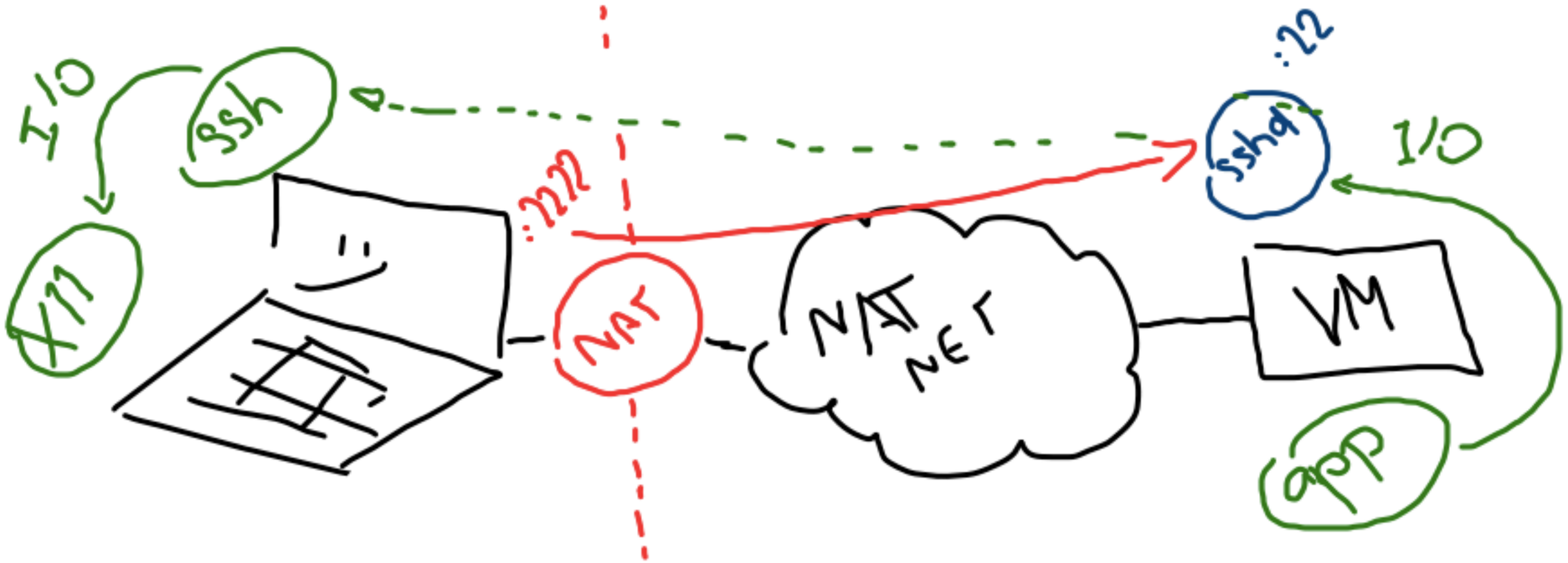
- On your host OS run:

```
$ ssh -p 2222 -D 3128 tess@localhost
```

- Browse to http://10.0.2.15:8000 using the proxy:

```
$ curl --preproxy http://localhost:3128 \
       http://10.0.2.15:8000
```

# X11 tunneling



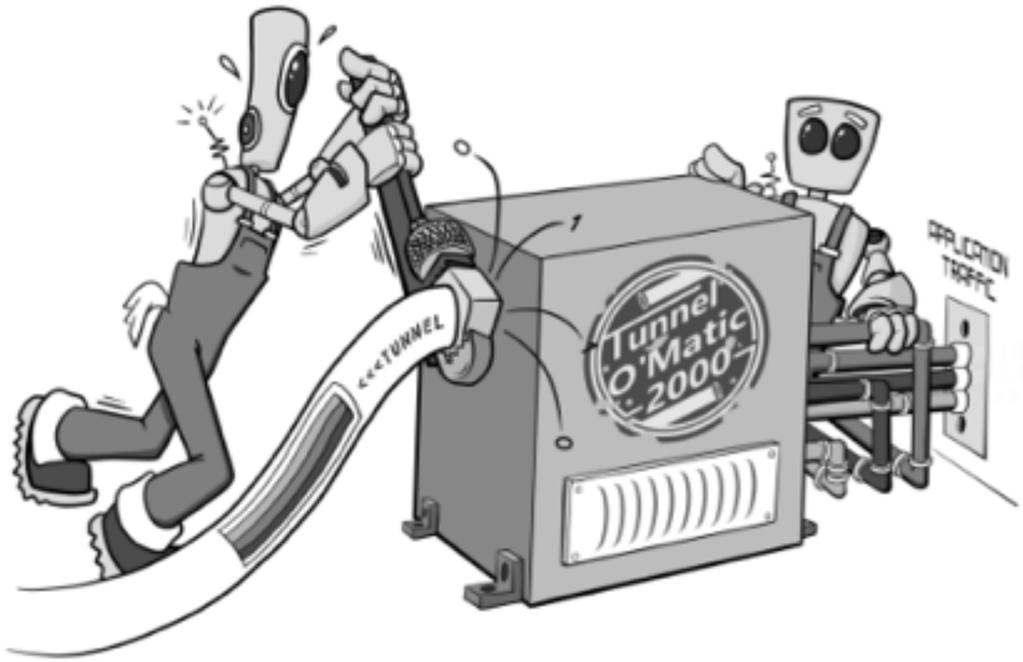
# X11 Tunneling

- This requires an X11 server on your host OS.
- On your host OS run:

```
$ ssh -p 2222 -Y tess@localhost "xclock"
```

# The SSH bible: CPH

- Brennon Thomas' awesome book.
- Free for students.
- Explains all cool SSH options.




- See: Cyber Plumber's Handbook

# SSH keys and ssh-agent

# Setup

- Ensure that you have two Linux VMs.
- And that you have an account on both.

# Assignment

- Double-check that SSHd runs on both servers.
- Generate a new key pair on one of the accounts.
  - Make it type ECDSA, with a password.
  - Setup its pub.key for authentication on the other VM.
  - Test your SSH key authentication.

# Assignment

- Start "*eval $(ssh-agent)*".
- Add / load the private key you generated into the running "*ssh-agent*", with the "ssh-add" command.
  - This should ask your password once.
- Try SSH-ing to the other VM again.
  - This should not ask your password.

# Assignment

- Reconfigure "*sshd_config*" on one of the VMs,
  - So it will only allow group "*sshusers*" to login.
- Give your own account the new group "*sshusers*"
- Restart the SSH daemon and test that you can login.
  - Also make sure that another user cannot.

# LAB: SSH as proxy

# Can you perform:

- An NMap portscan,
  - Of your Linux VM,
  - From your host OS?
  - For example to find your Python httpd on port 8000.

- Unfortunately Windows can’t play. So make teams!

# Hints

- You will need "proxychains" or "proxychains-ng".
- SOCKS is best suited for TCP connect scans, use "-sT".
- First limit to known-open ports (like 8000).
- A ping will fail, so use "-Pn".

# Solution

- Let SSH open a SOCKS5 proxy, with "-D 3128"
- Configure proxychains to use:
  - socks5 localhost 3128

```
$ proxychains nmap -Pn -p 8000 -sT 10.0.2.15
```

# Reference materials

# Resources

- VirtualBox networking modes
- Download Putty
- Download WinSCP
- SSH keys for dummies
- Cyber Plumber’s Handbook