using Linux under Windows

introduction

Windows has had a native Linux capability for the last few years which has proven to be stable and quite capable.

"Windows Services for Linux" (WSL) can run Linux most distributions without problems in most of the ways we might want to use it as Linux users and provides a useful adjunct to those saddled with needing to use a Windows laptop for whatever reason.

why?

There are a number of reasons, but many of these simply come down to convenience.

For most uses it can't replace a full Linux desktop but for cases where you just want to use a Unix-based workflow for certain things such as text processing as this comes more naturally, then being able to work on the same files from within Windows and Linux without having to transfer them back and forth can be quite appealing.

It is also very useful as a development environment for devops work as it's local and can be created and destroyed relatively easily, mostly without need for Windows Administrator privileges after the initial setup and required Windows features are enabled.

version differences

WSL version 1 is more like a "reverse Wine" in the sense that it's an environment that provides a system call ABI interface to a Linux binary and translates these to the equivalent calls on Windows to provide similar functionality, but this approach tends to only be able to extend so far.

WSL version 2 is a para-virtualisation environment that operates at the kernel level to provide low-level functionality into the Windows environment and is apparently based on Microsoft's work with Xen in their Azure cloud infrastructure. This allows WSL2 VMs to host X11 clients and present these on the Windows desktop alongside other native applications. WSL2 is reported to support being able to read and write native Linux filesystems directly on devices (although I've not tried this)

installing WSL

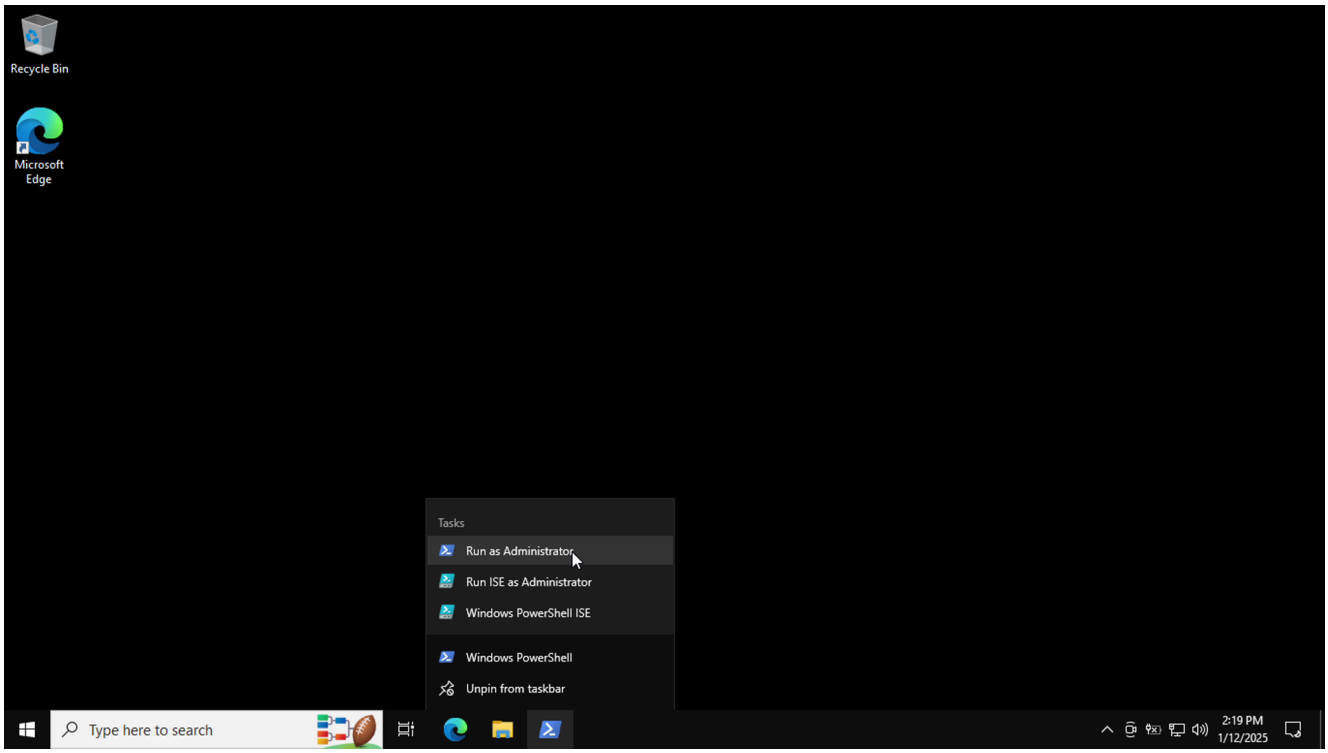
Microsoft do have a number of web pages on WSL and the install process however this page assumes some settings about your desired installation which may not be wanted

I'll be going through a manual process and describing things as we go, mostly following
the manual install process

installing WSL manually

enable virtual machine features

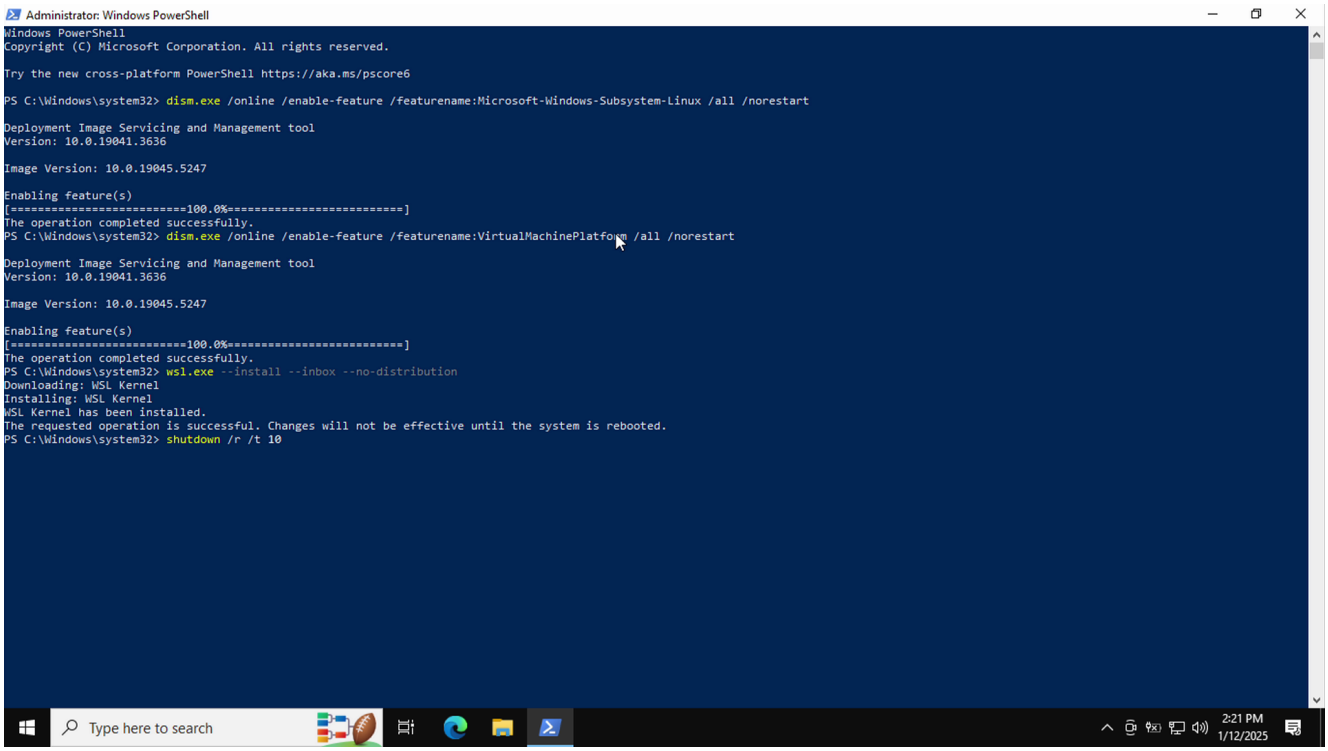
- open powershell as Administrator:



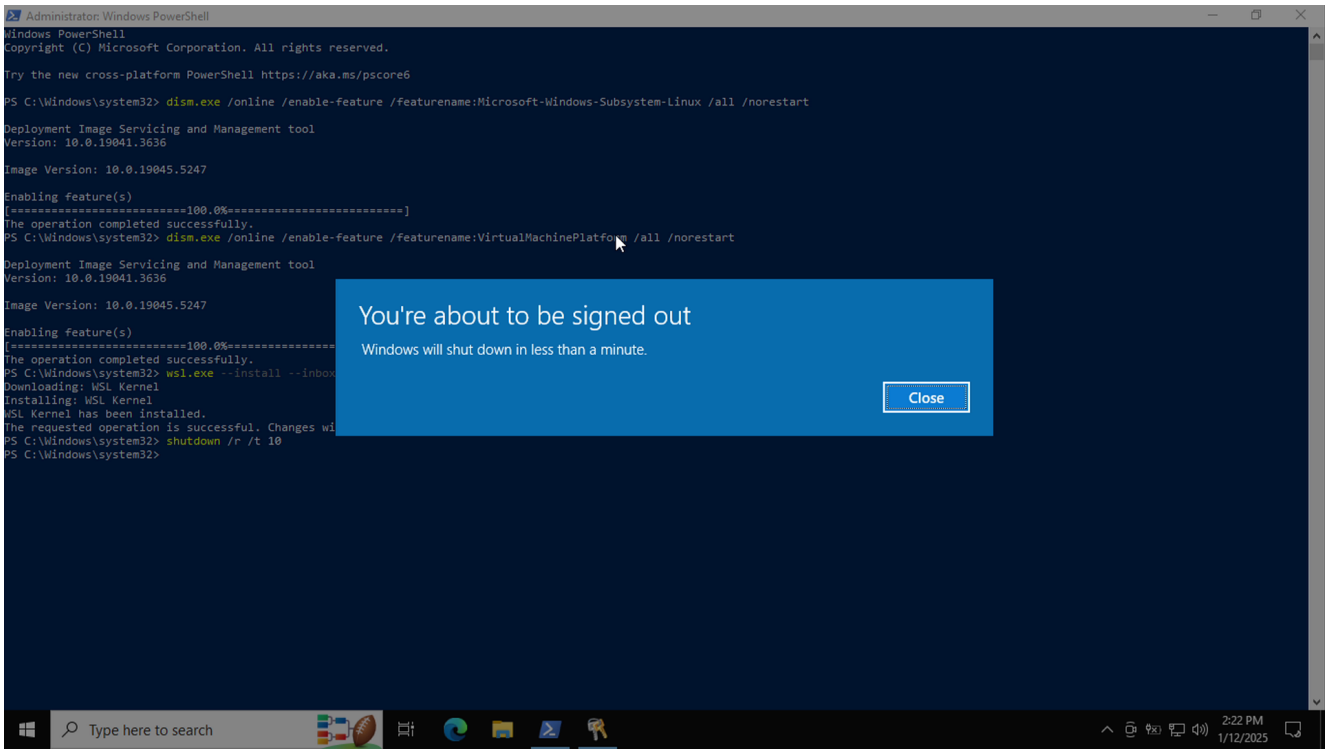
- enable virtual machine features and reboot the host

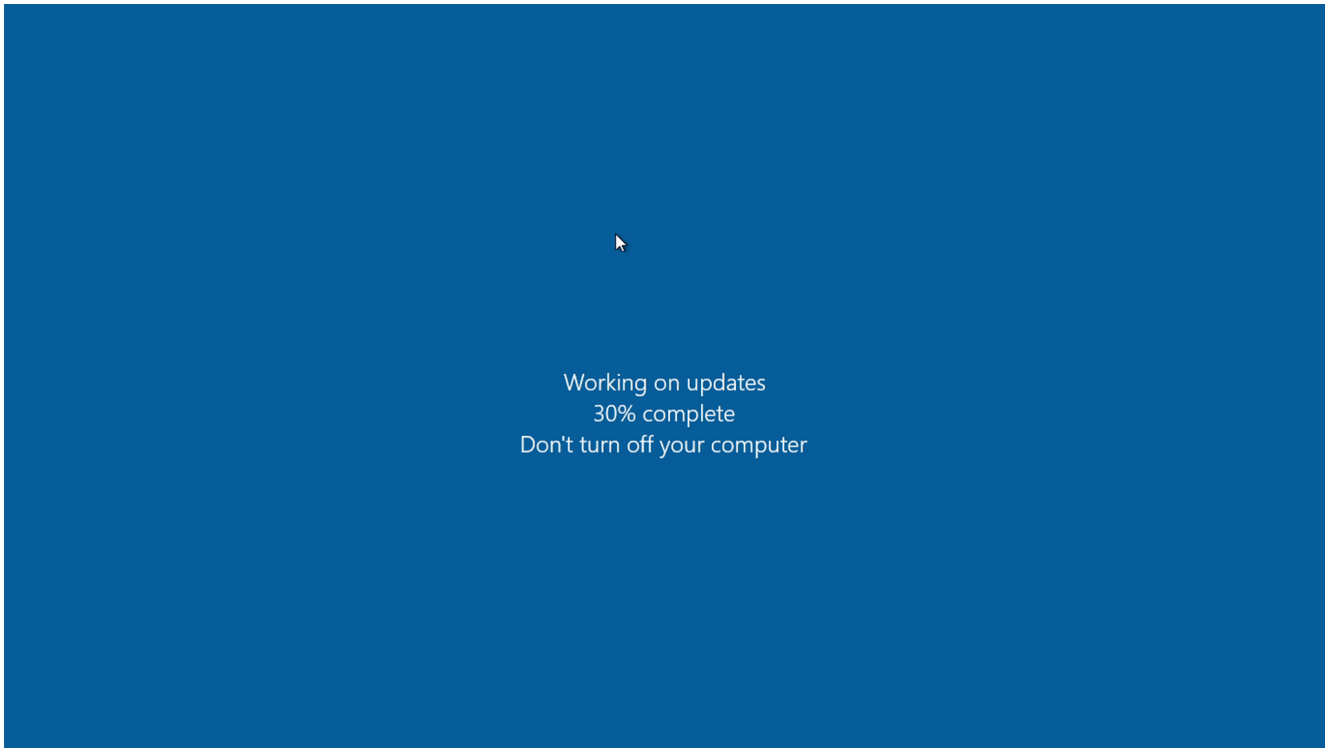
```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
wsl.exe --install --inbox --no-distribution
shutdown /r /t 10
```

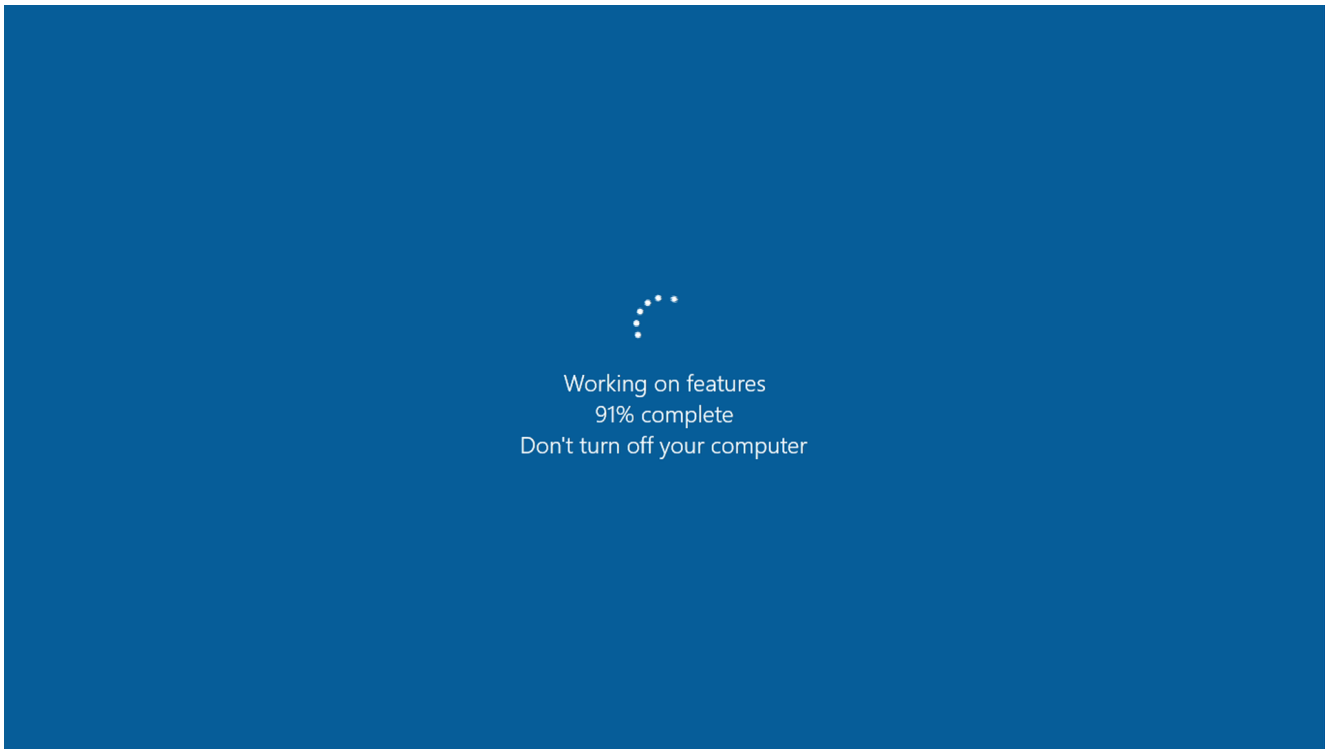
- for example:



- your Windows host will perform a reconfigure reboot

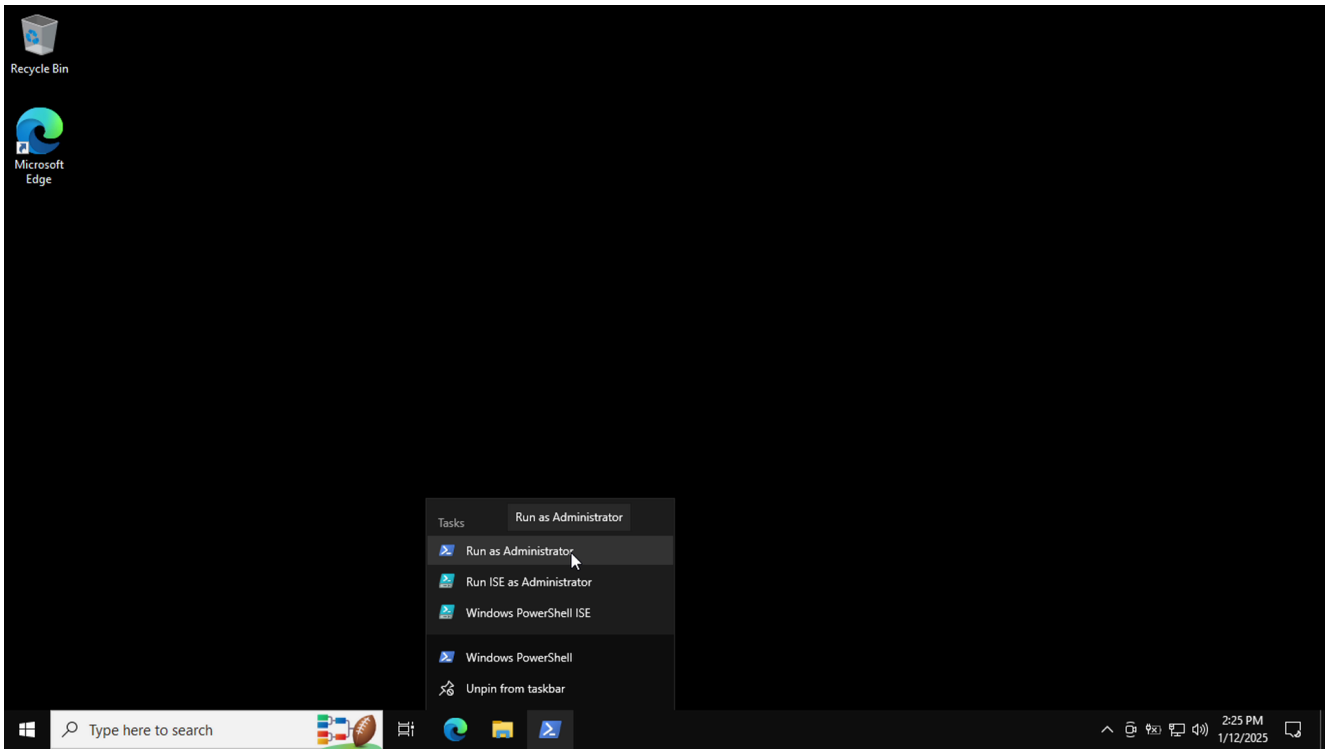

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Windows\system32> dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
Deployment Image Servicing and Management tool
Version: 10.0.19041.3636
Image Version: 10.0.19045.5247
Enabling feature(s)
[==========================100.0%==========================]
The operation completed successfully.
PS C:\Windows\system32> dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
Deployment Image Servicing and Management tool
Version: 10.0.19041.3636
Image Version: 10.0.19045.5247
Enabling feature(s)
The operation completed successfully.
PS C:\Windows\system32> wsl.exe --install --inbox
Downloading: WSL Kernel
Installing: WSL Kernel
WSL Kernel has been installed.
The requested operation is successful. Changes wi
PS C:\Windows\system32> shutdown /r /t 10
PS C:\Windows\system32>
You're about to be signed out
Windows will shut down in less than a minute.
Close
Type here to search
2:22 PM
1/12/2025


Working on updates
30% complete
Don't turn off your computer



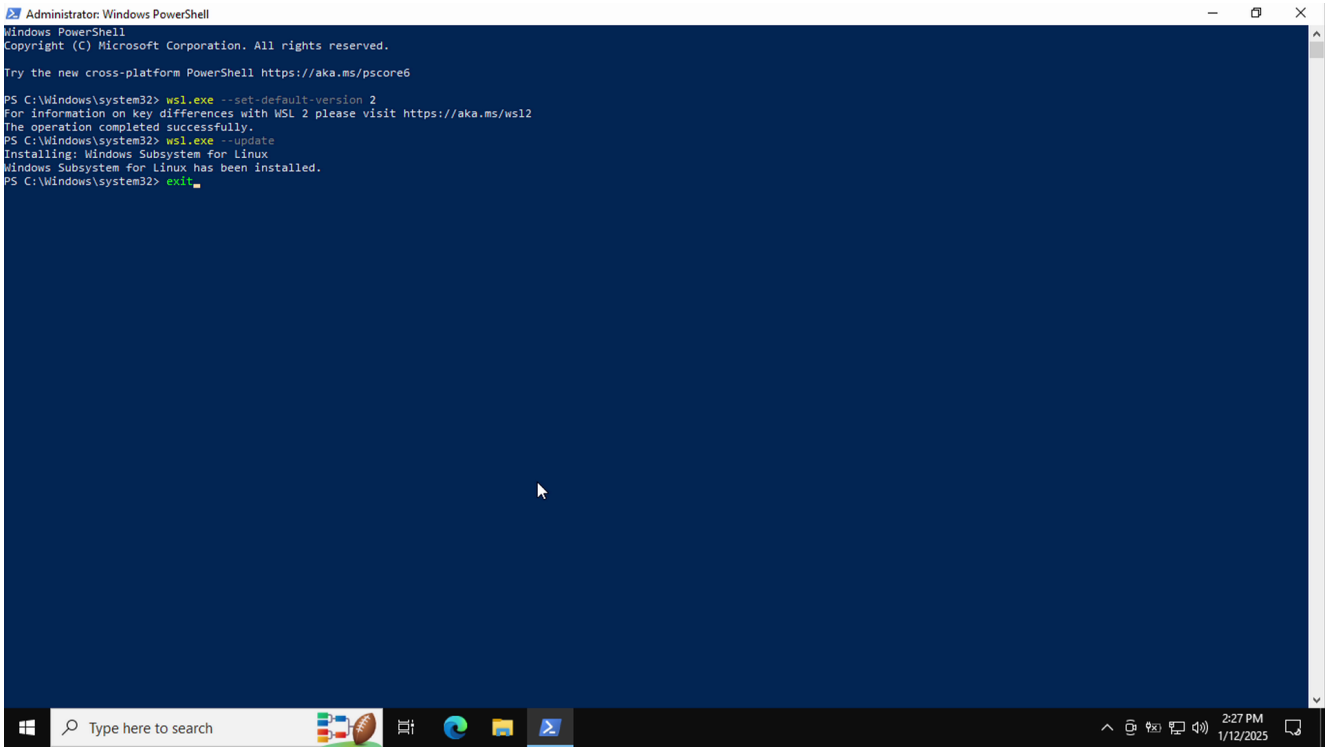
configure WSL

- start Powershell as Administrator



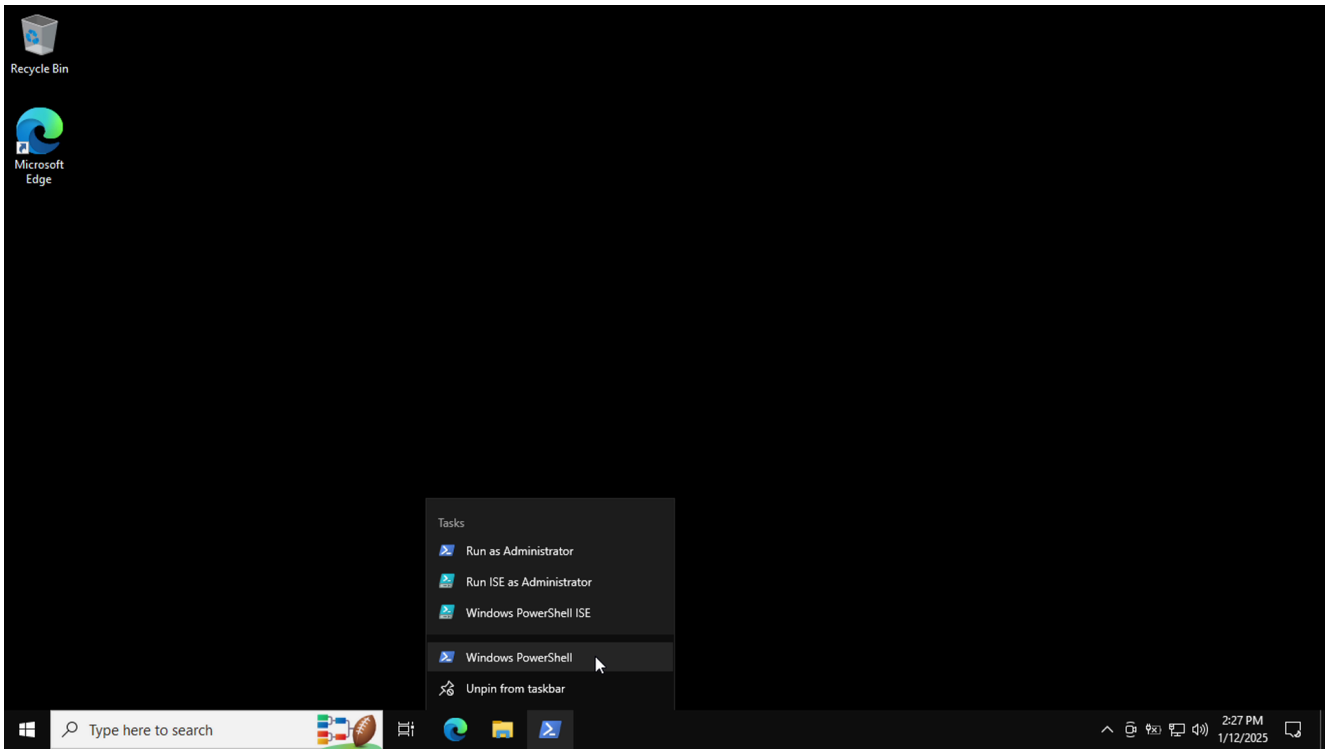
- set default version of WSL, update it and then exit the PowerShell session

```
wsl.exe --set-default-version 2
wsl.exe --update
exit
```



install image

- start Powershell as a non-privileged user



- list the OS images available online, install Ubuntu, then exit the PowerShell session:

```
wsl.exe --list --online
wsl.exe --install --no-launch --distribution ubuntu-22.04
exit
```

- for example:

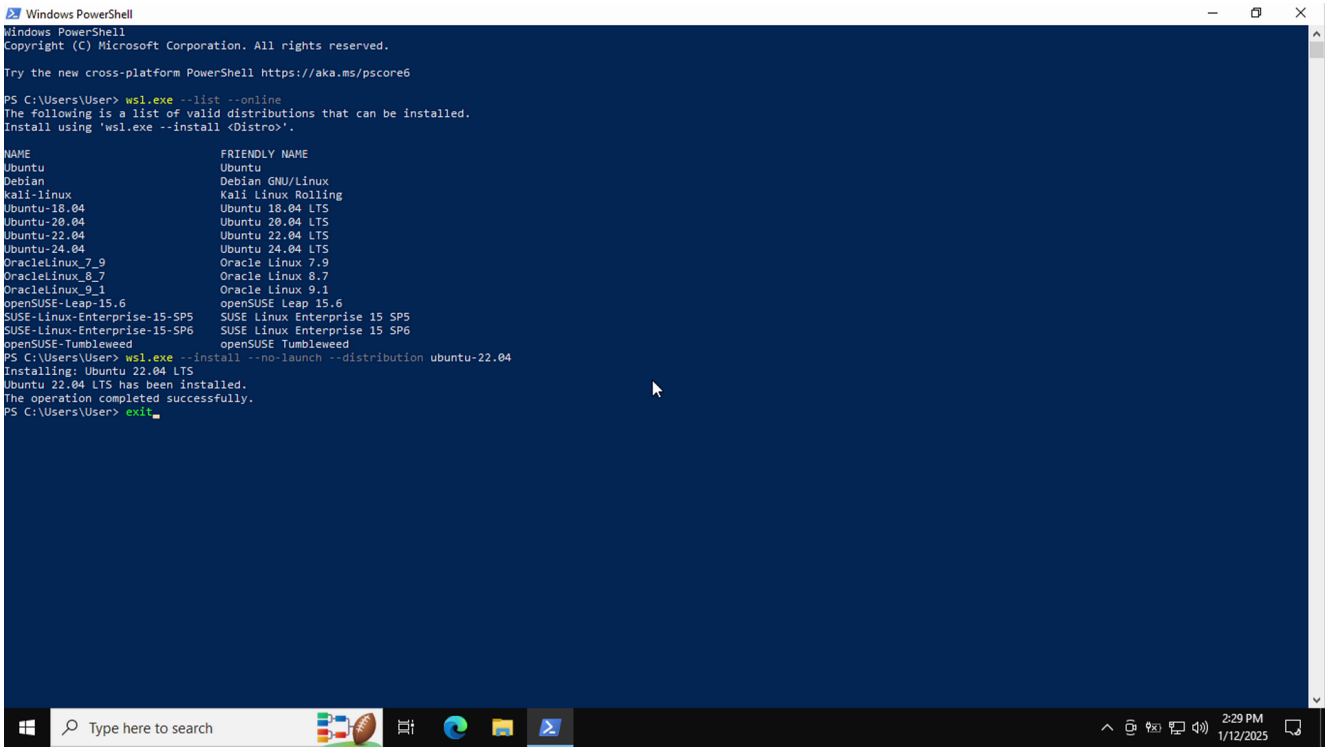
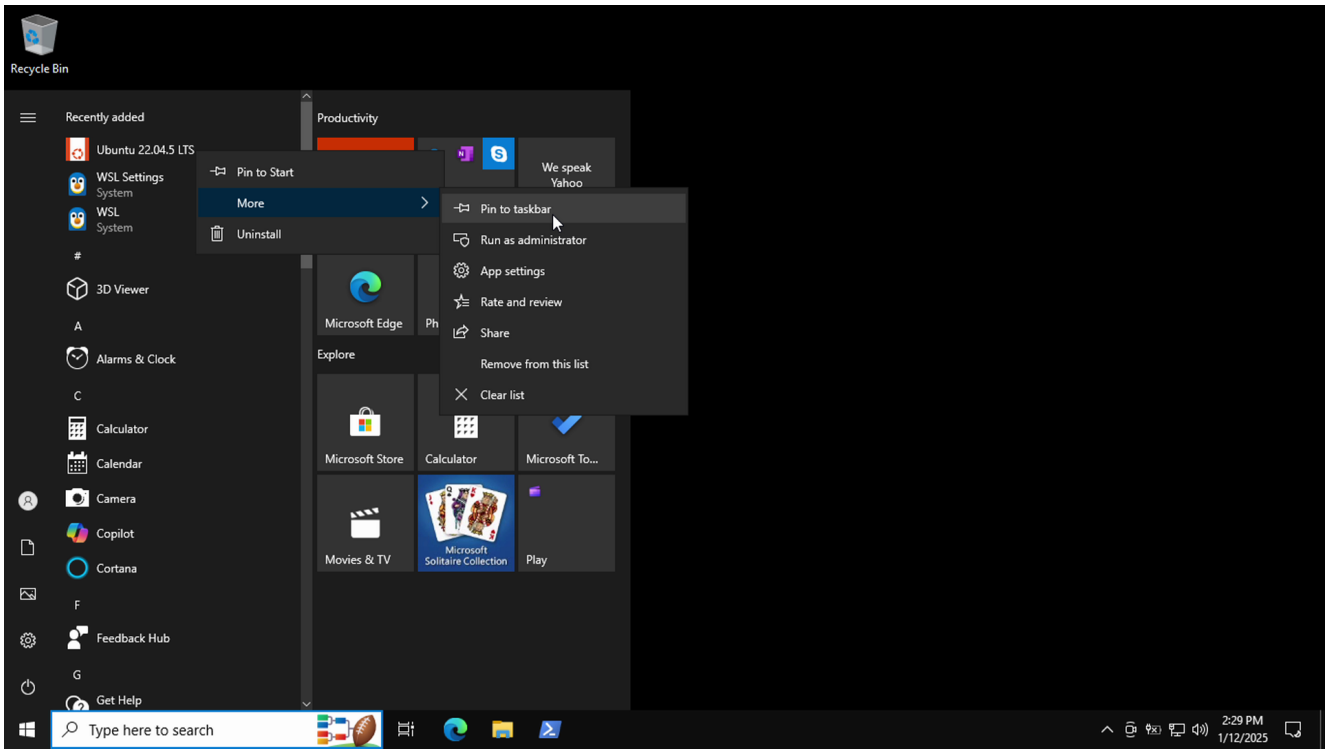
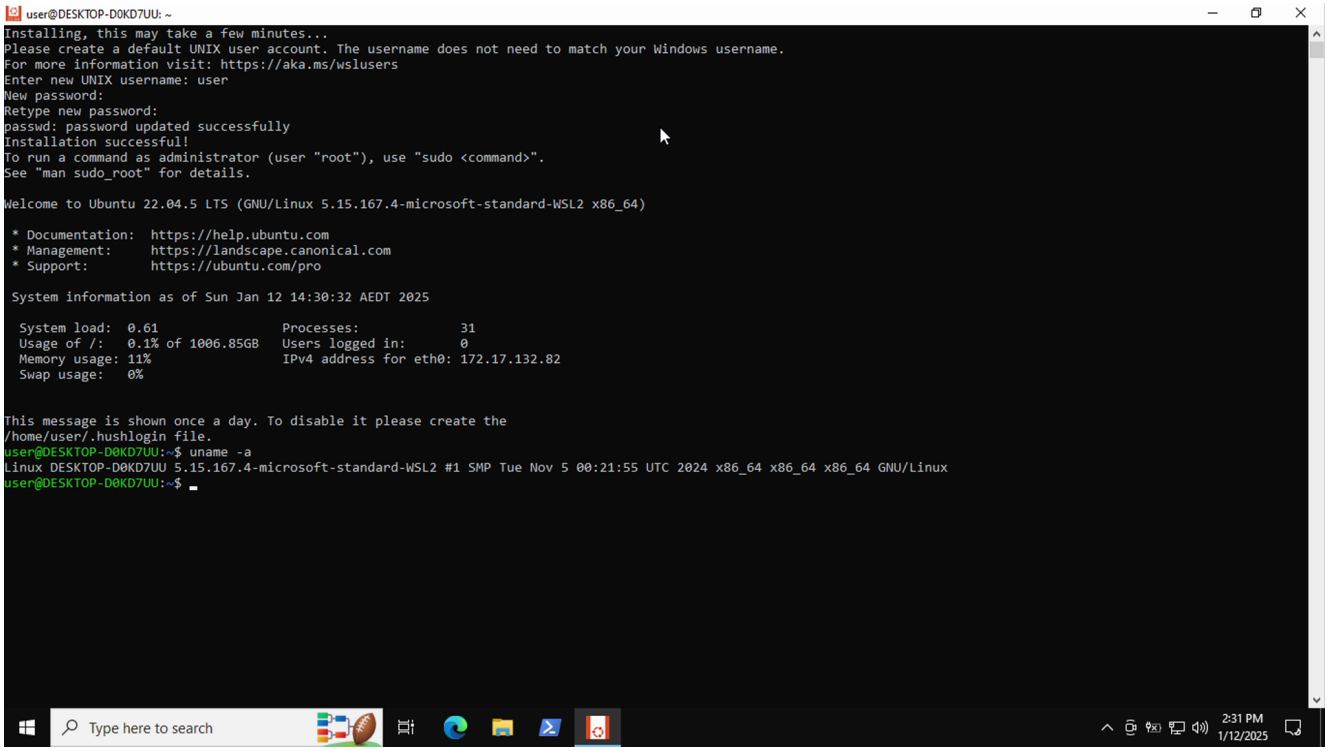


image configuration

- from the main menu, locate the new Ubuntu "app" (taskbar pinning is a convenience)



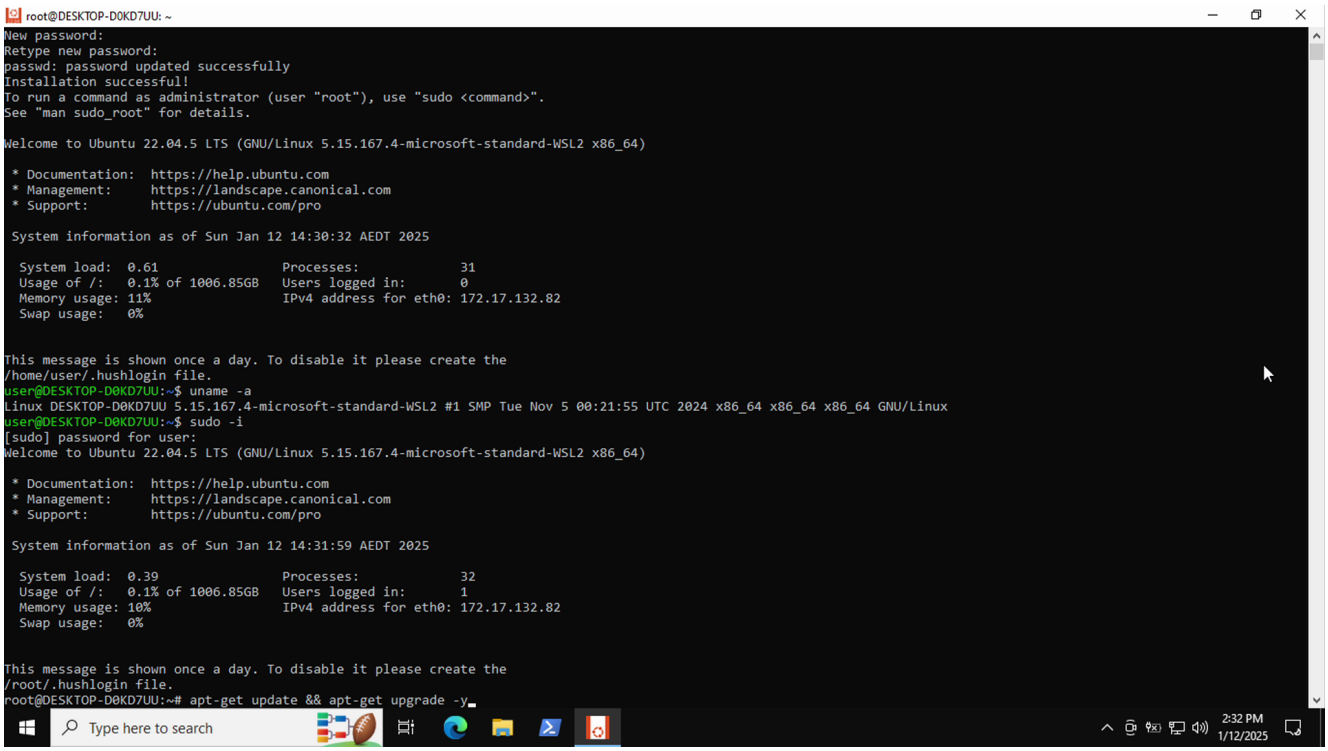
- start the app, which will then do the initial OS build:
  - use a proper non-privileged username (don't use `root`)
  - use a unique password
- for example:

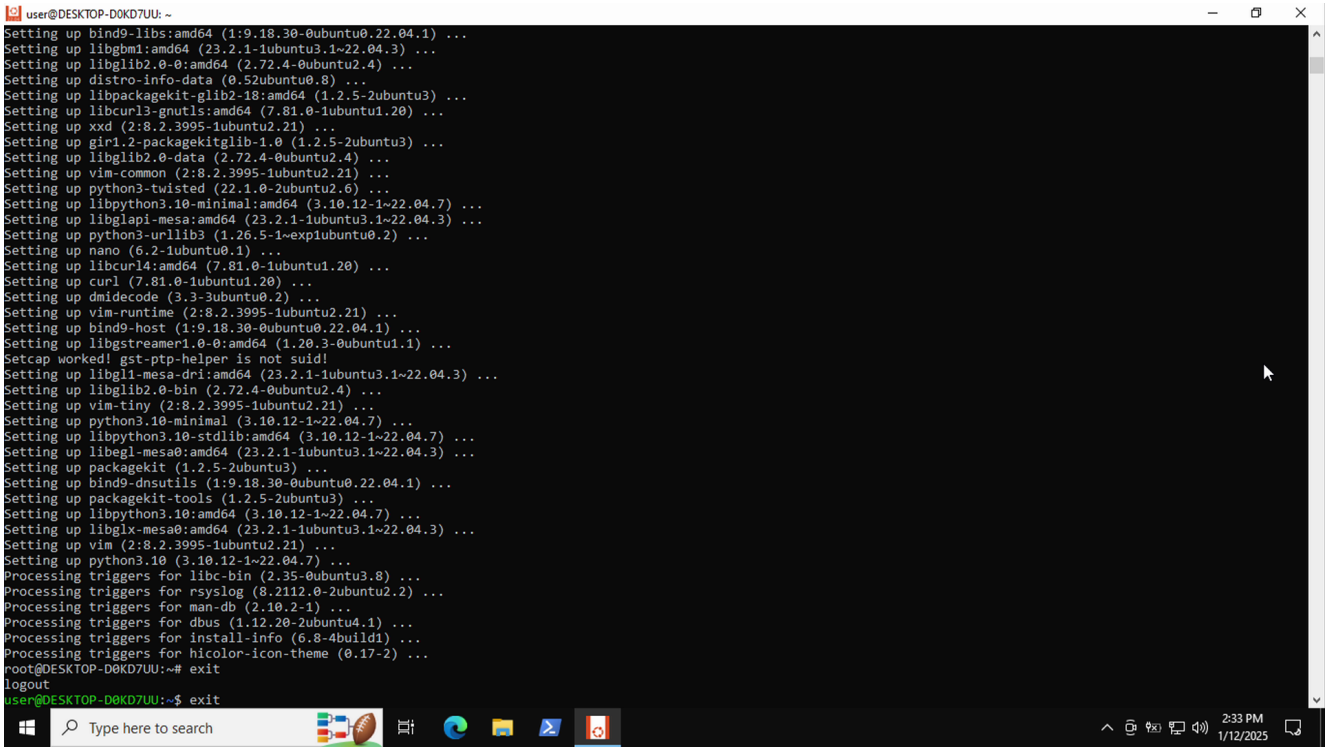


- do the usual maintenance:

```
sudo -i
apt-get update && apt-get upgrade -y
```

- for example:



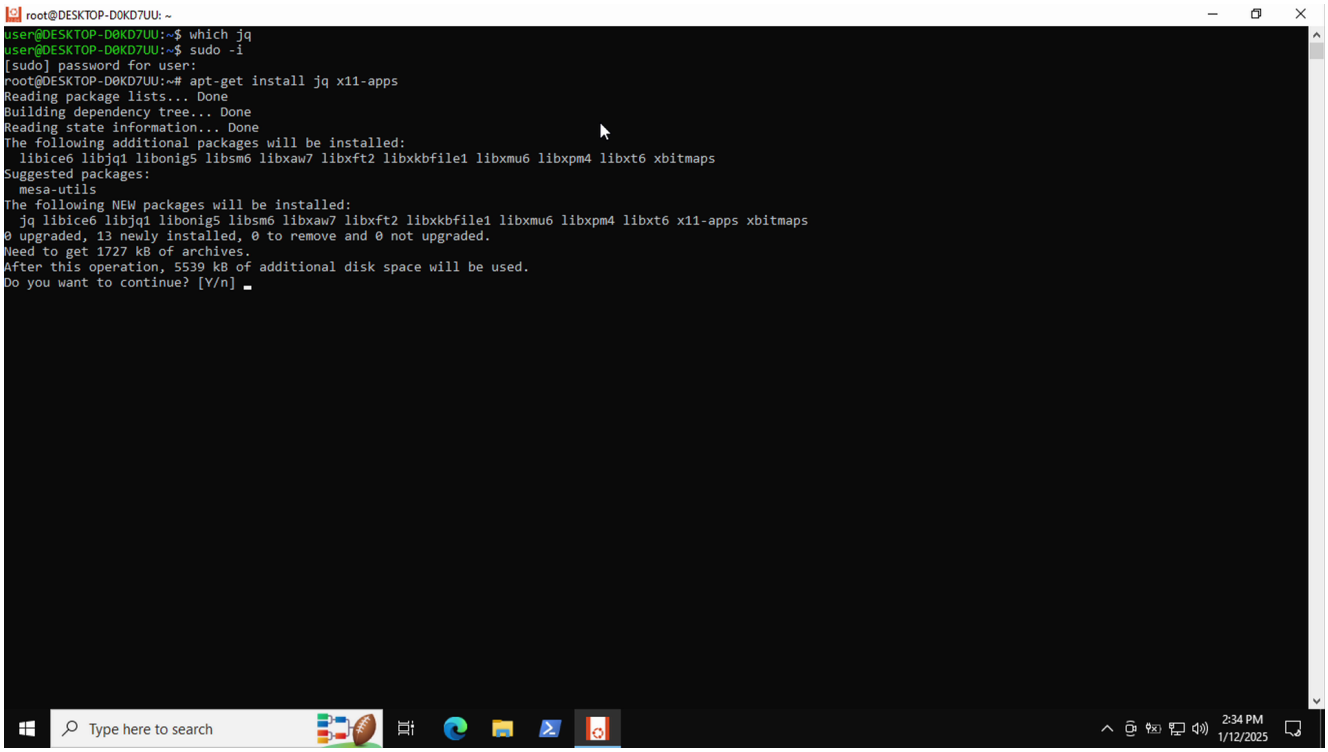


X11 support

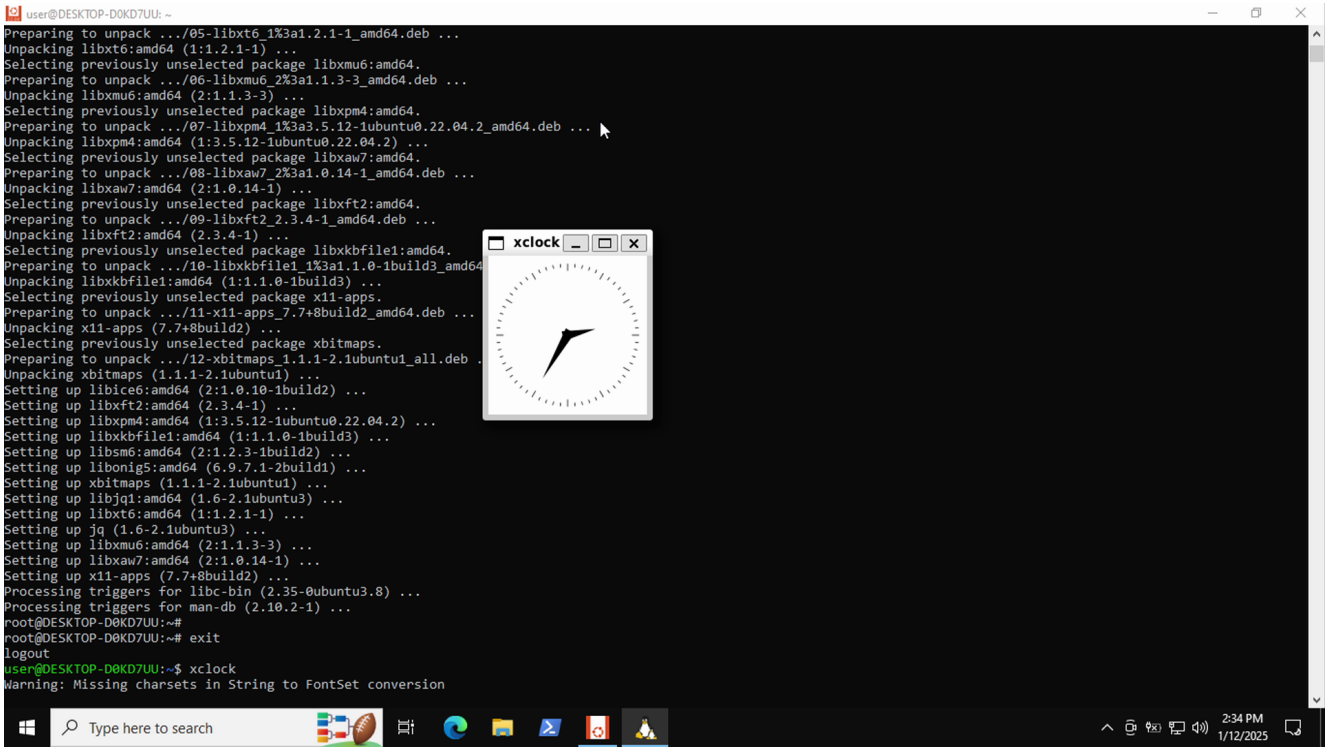
- install the x11-apps package and dependencies

```
sudo -i
apt-get install x11-apps
```

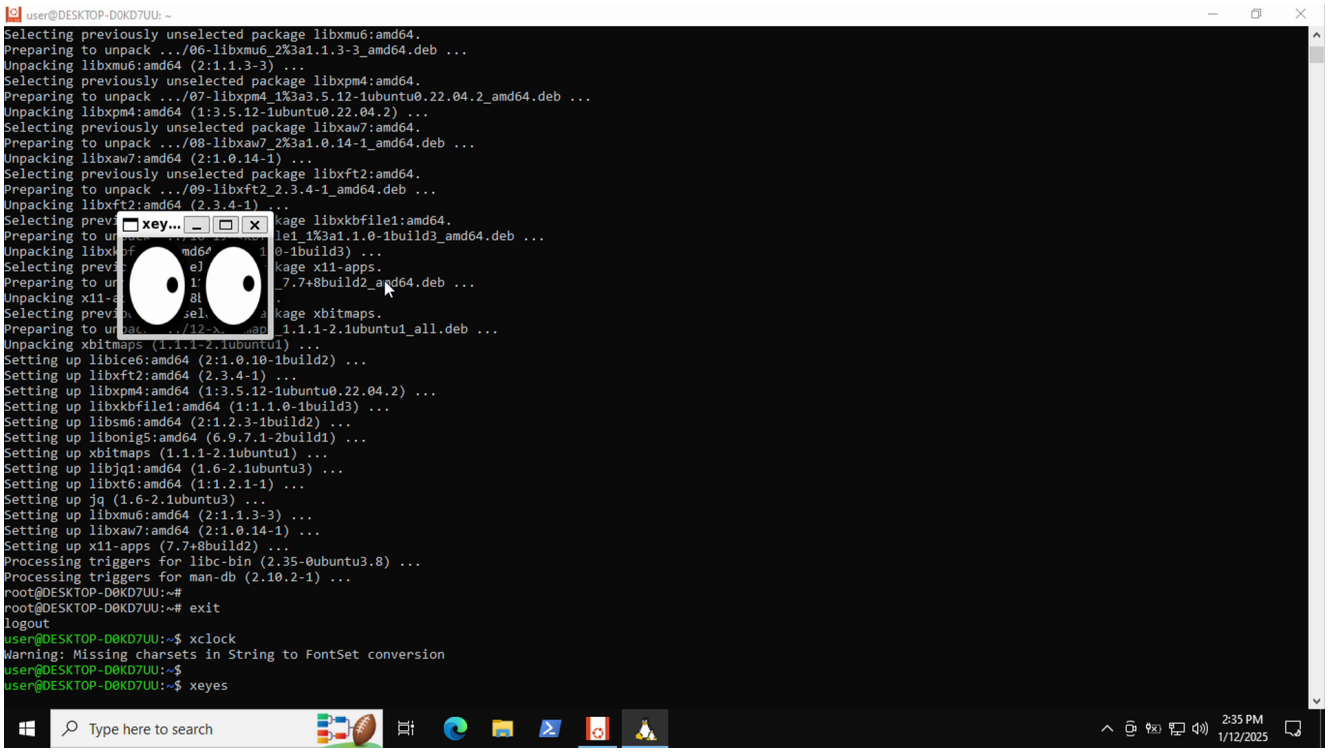
- for example:



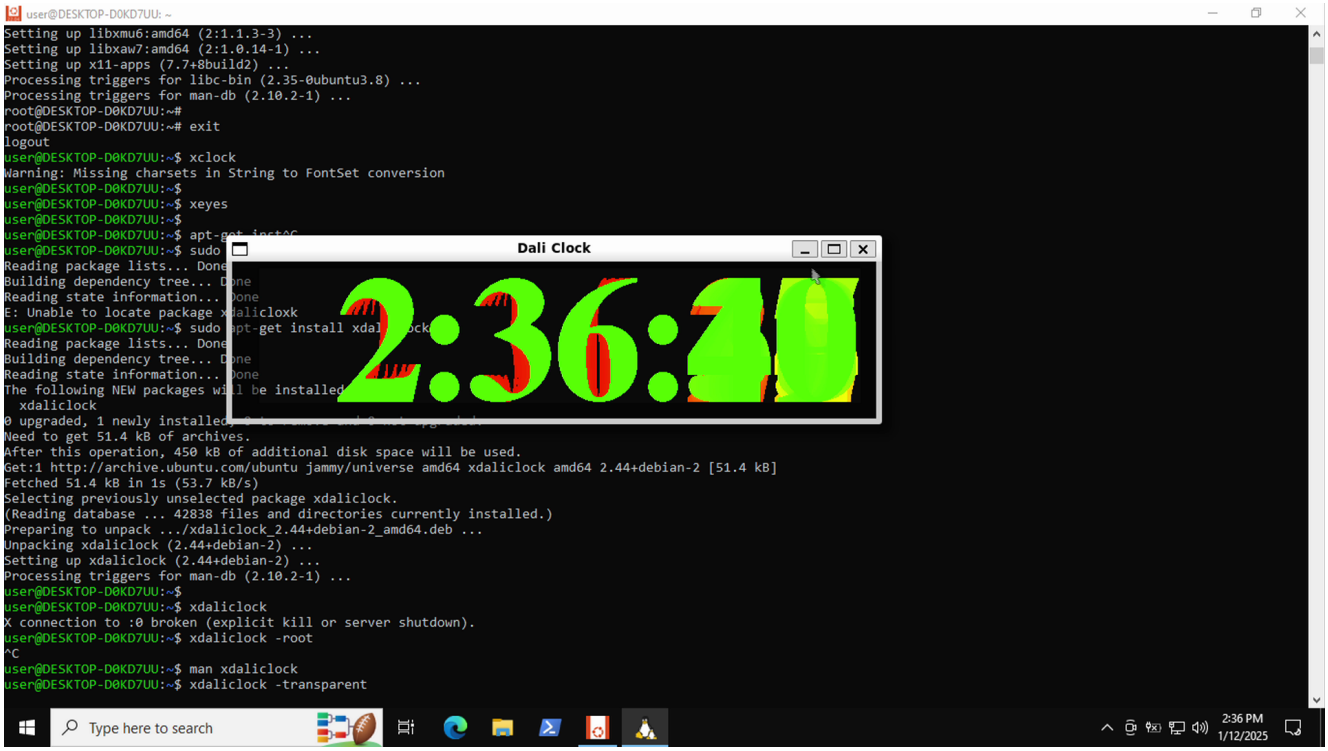
- confirm that `xclock` is functional



- `xeyes` sort of works also



- `xdaliclock` also sort of works also



caveats

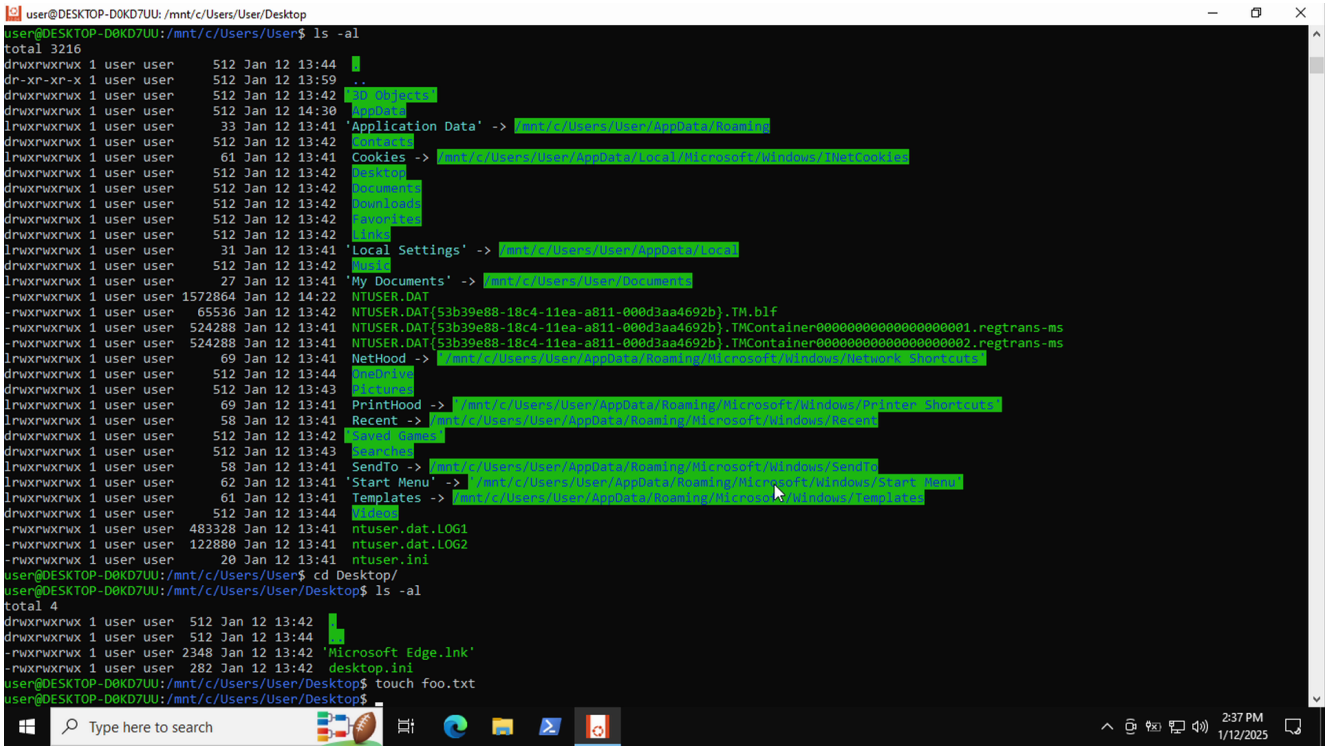
There is no apparent support for some things

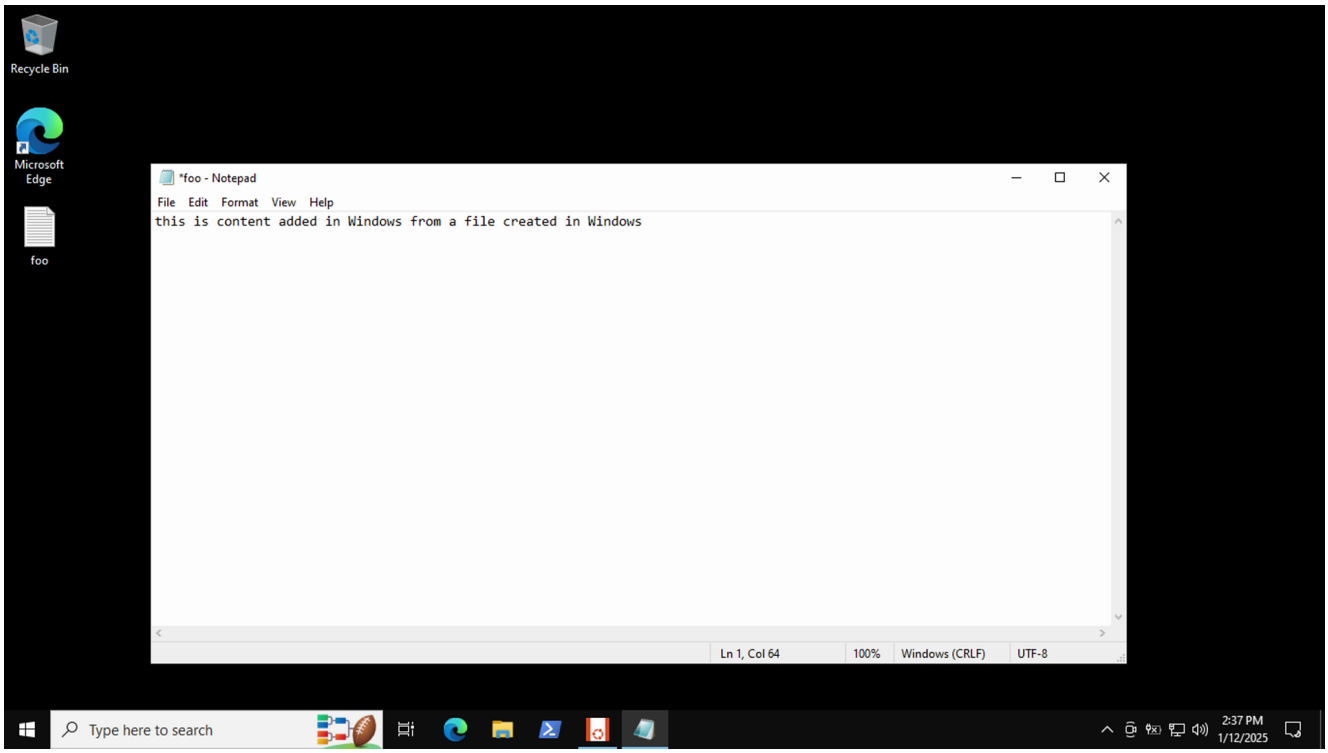
- root window, so no changing background parameters
- limited shape support, `xdaliclock` does work, but I'd not rely on it
- transparency support seems to be non-existent
- client snooping the mouse pointer outside of their own window (as per `xeyes`)

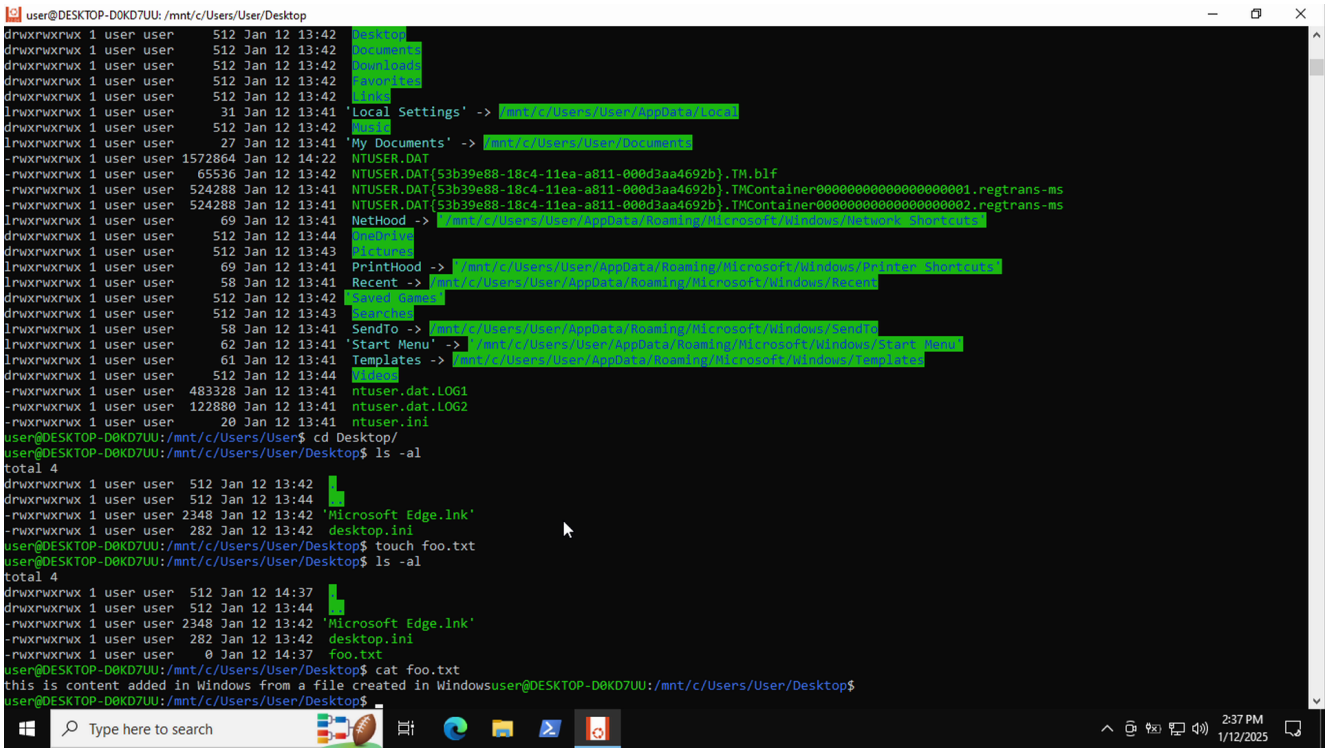
interoperability

file sharing

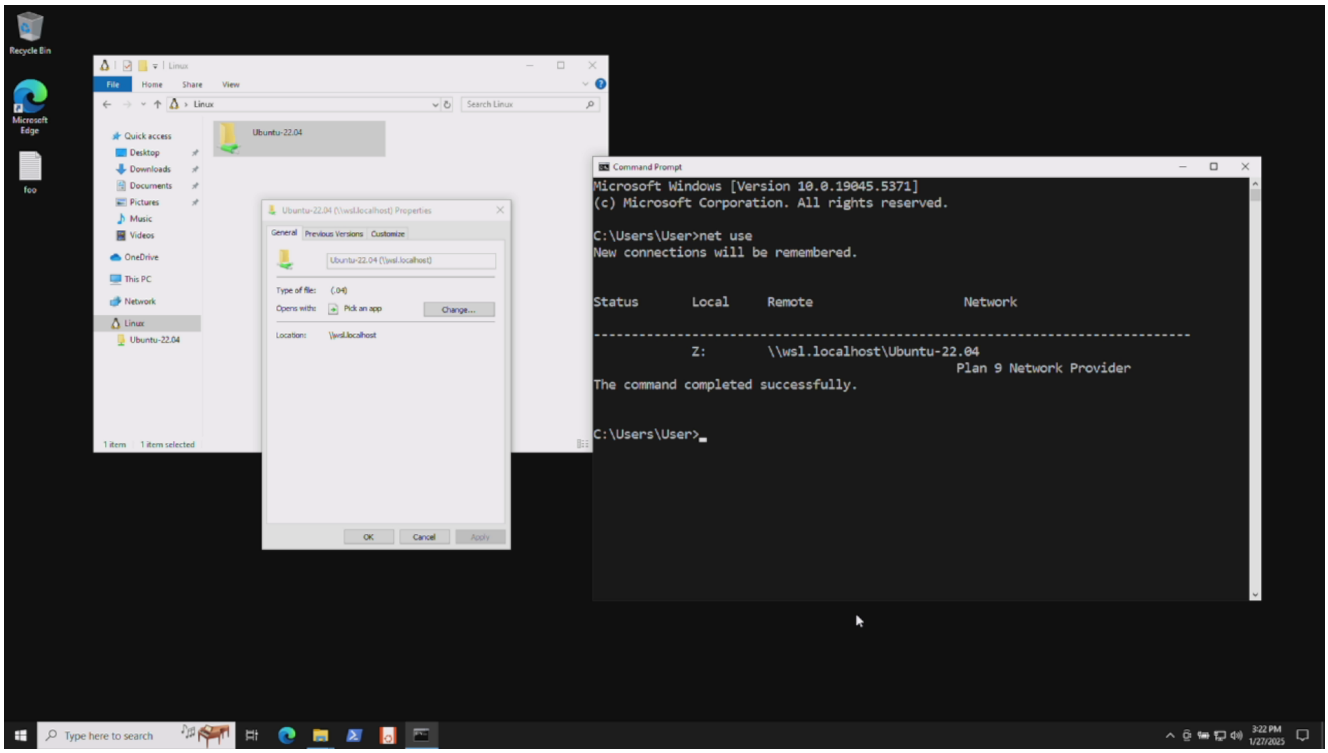
- Windows file system is available from within WSL as `/mnt/c`

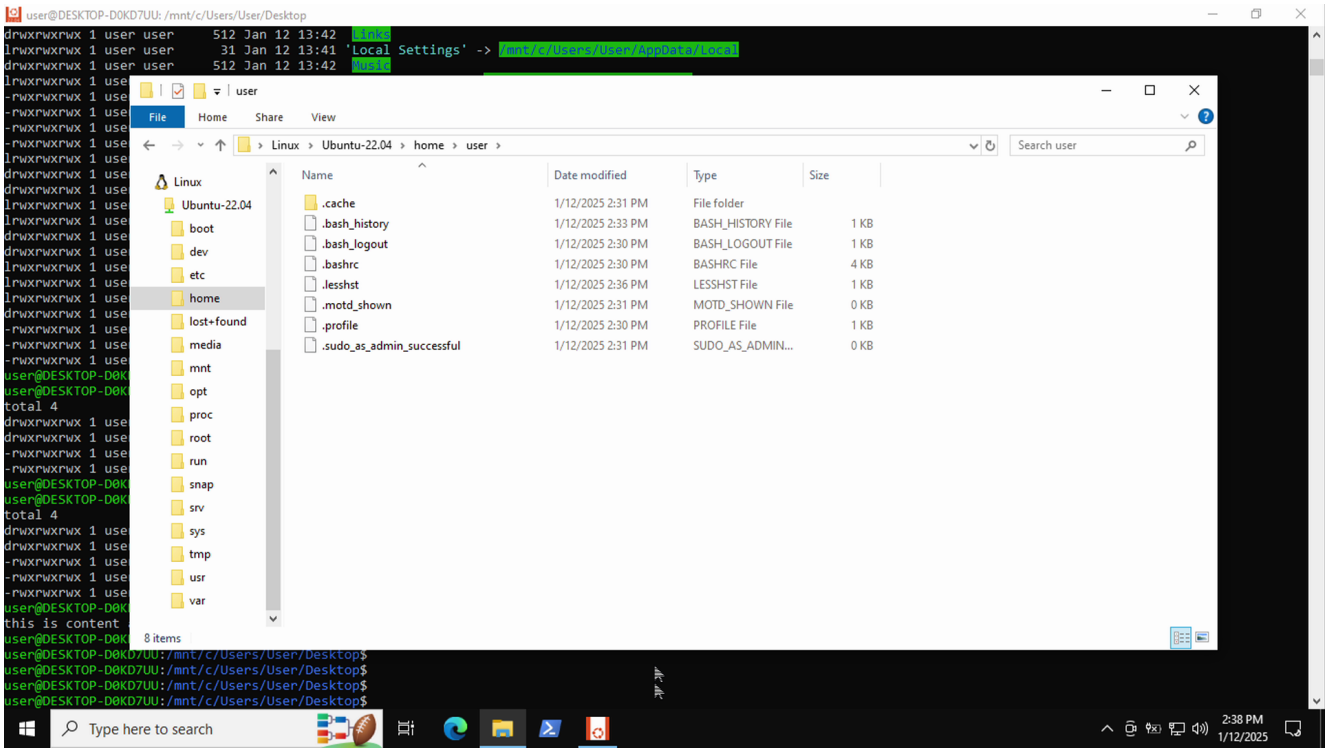


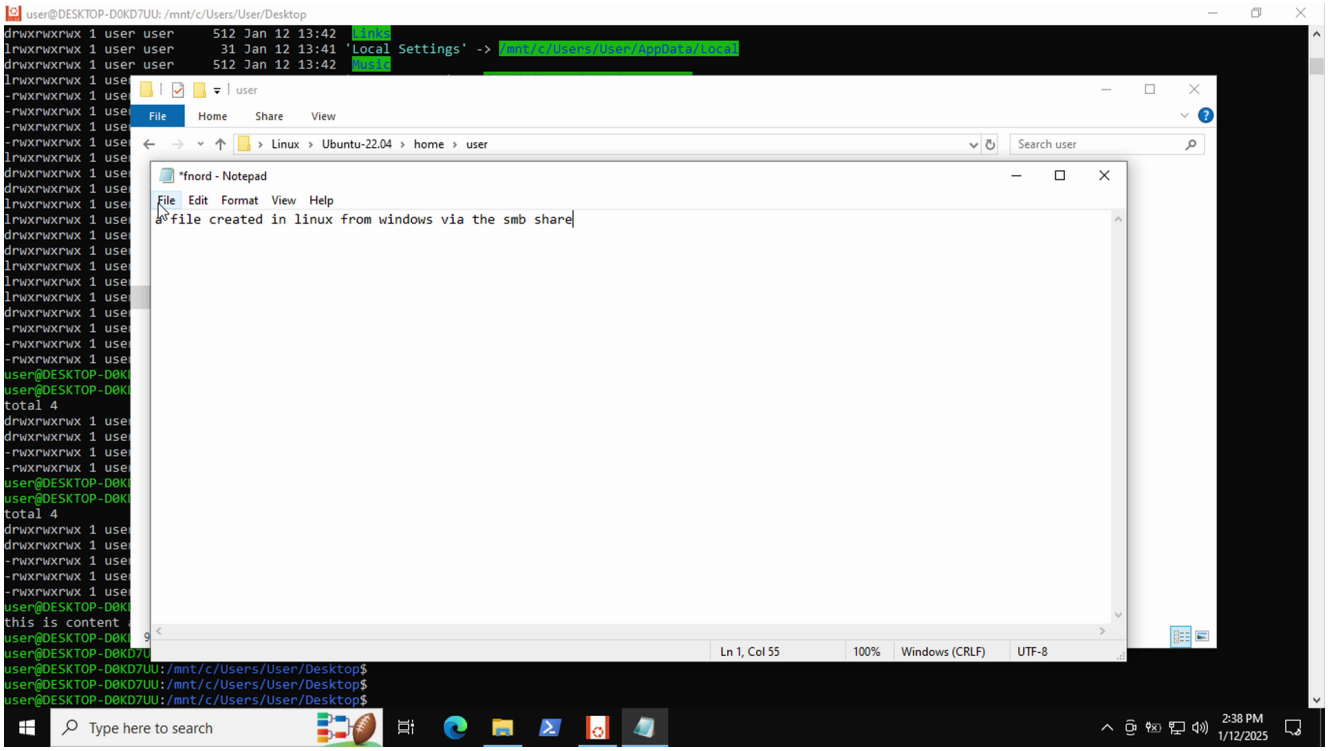


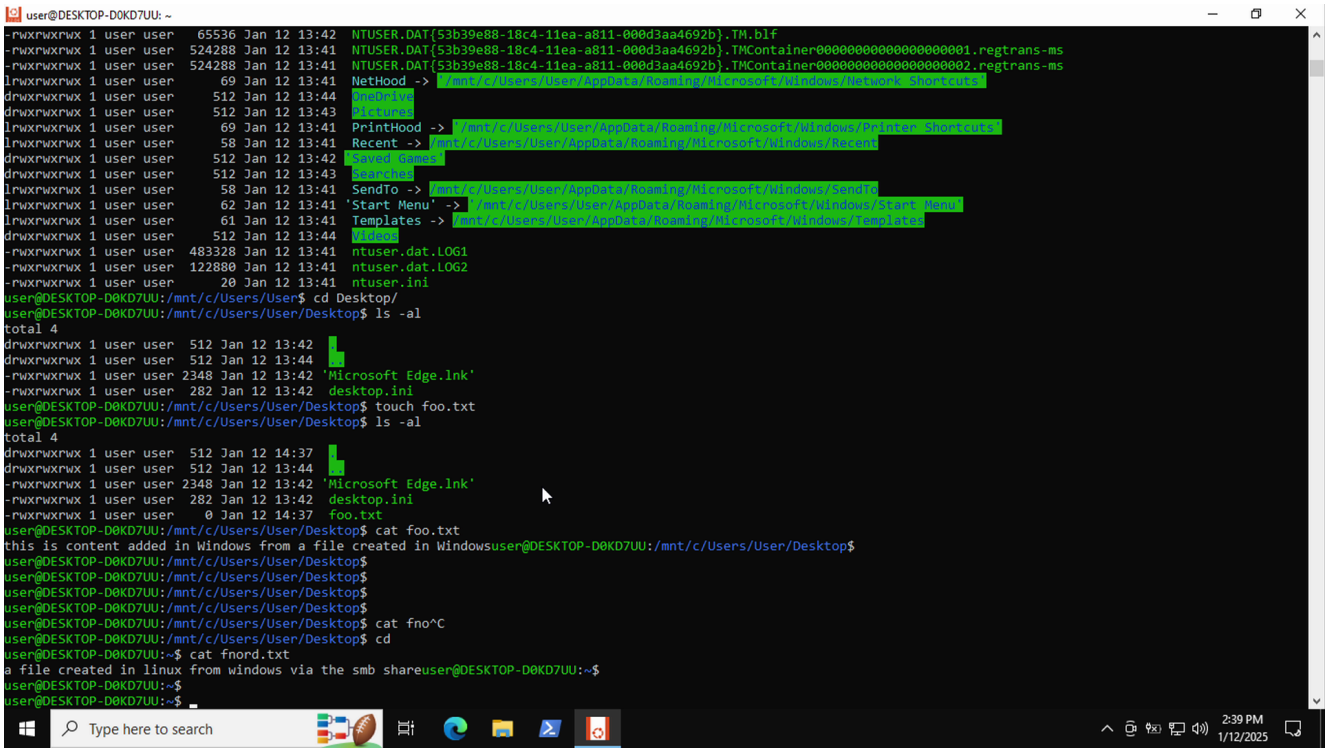


- WSL file system is available via a network share `\\wsl.localhost\<image-name>` as follows:


Recycle Bin
Microsoft Edge
foo
Linux
File Home Share View
Ubuntu-22.04
Ubuntu-22.04 (\\wsl.localhost) Properties
General Previous Versions Customize
Command Prompt
Microsoft Windows [Version 10.0.19045.5371]
(c) Microsoft Corporation. All rights reserved.
C:\Users\User>net use
New connections will be remembered.
Status Local Remote Network
Z: \\wsl.localhost\Ubuntu-22.04 Plan 9 Network Provider
The command completed successfully.
C:\Users\User>
Type here to search


user@DESKTOP-D0KD7UU: /mnt/c/Users/User/Desktop
drwxrwxrwx 1 user user 512 Jan 12 13:42 Links
lrwxrwxrwx 1 user user 31 Jan 12 13:41 'Local Settings' -> /mnt/c/Users/User/AppData/Local
drwxrwxrwx 1 user user 512 Jan 12 13:42 Music
total 4
this is content
user@DESKTOP-D0KD7UU:/mnt/c/Users/User/Desktop$
File Home Share View
Linux › Ubuntu-22.04 › home › user ›
Search user
Name Date modified Type Size
.cache 1/12/2025 2:31 PM File folder
.bash_history 1/12/2025 2:33 PM BASH_HISTORY File 1 KB
.bash_logout 1/12/2025 2:30 PM BASH_LOGOUT File 1 KB
.bashrc 1/12/2025 2:30 PM BASHRC File 4 KB
.lesshst 1/12/2025 2:36 PM LESSHST File 1 KB
.motd_shown 1/12/2025 2:31 PM MOTD_SHOWN File 0 KB
.profile 1/12/2025 2:30 PM PROFILE File 1 KB
.sudo_as_admin_successful 1/12/2025 2:31 PM SUDO_AS_ADMIN... 0 KB
Linux
Ubuntu-22.04
boot
dev
etc
home
lost+found
media
mnt
opt
proc
root
run
snap
srv
sys
tmp
usr
var
8 items
Type here to search
2:38 PM
1/12/2025





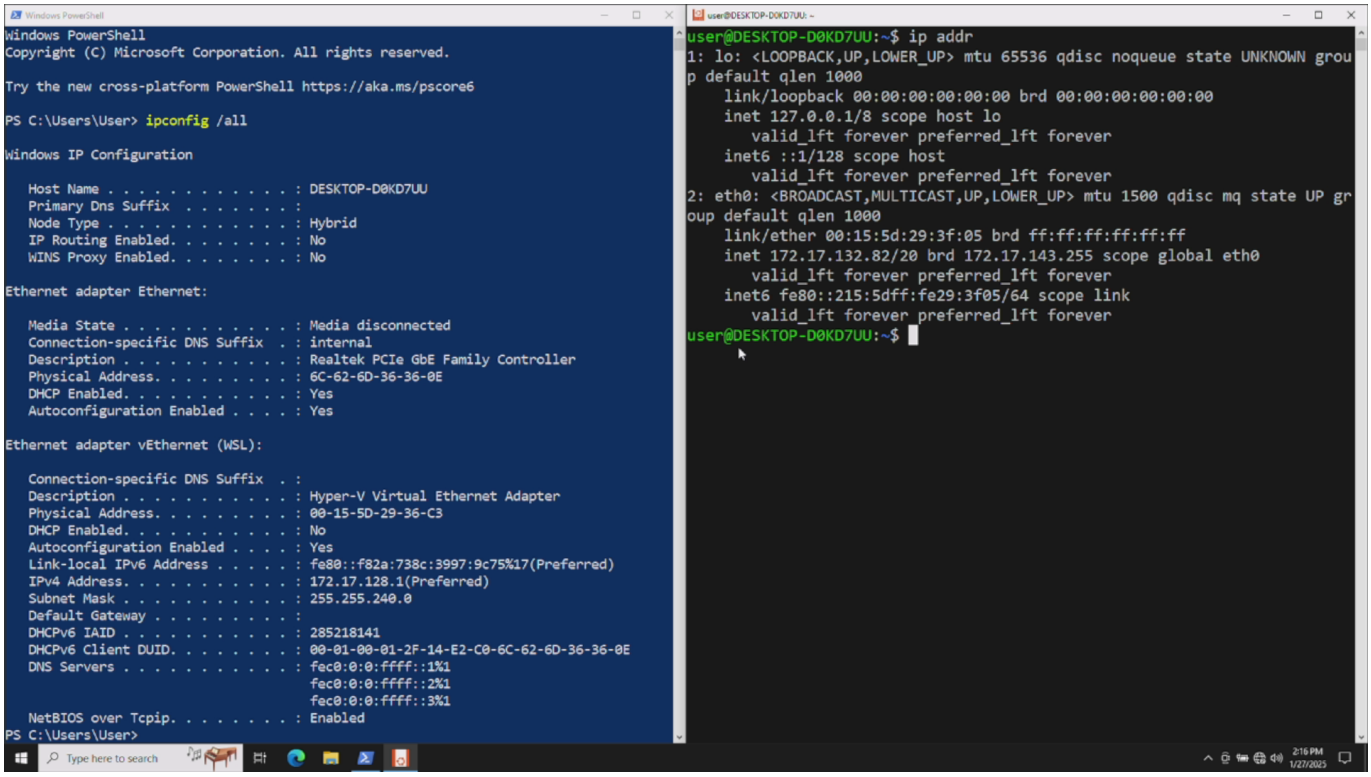
filesharing caveats

Modern Windows command-line tools output text as UTF-16 (for example, `netstat -r`) while Linux text tools tend to use UTF-8 - so you will need to use `iconv` on files generated under Windows or Linux tools such as `grep` will be unable to work with them:

```
$ iconv -f UTF-16le -t UTF-8
```

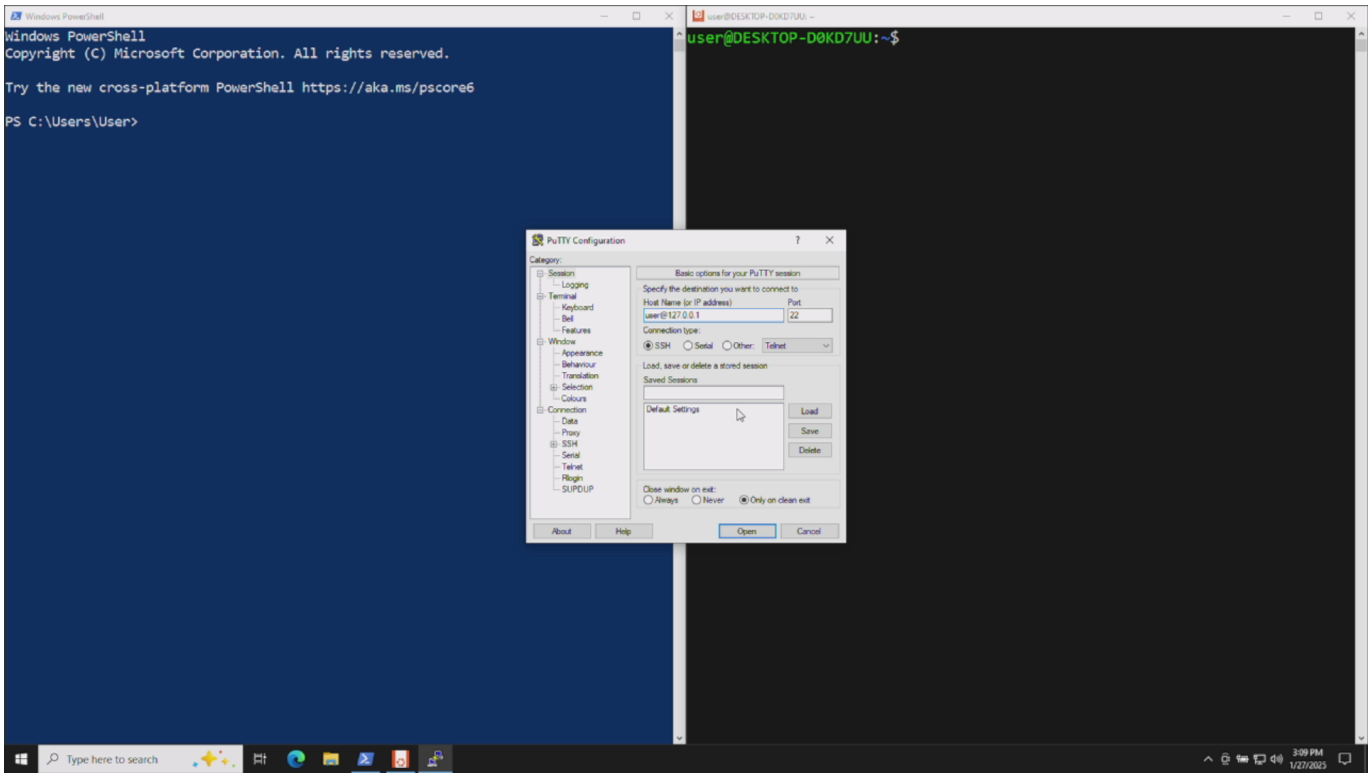
network

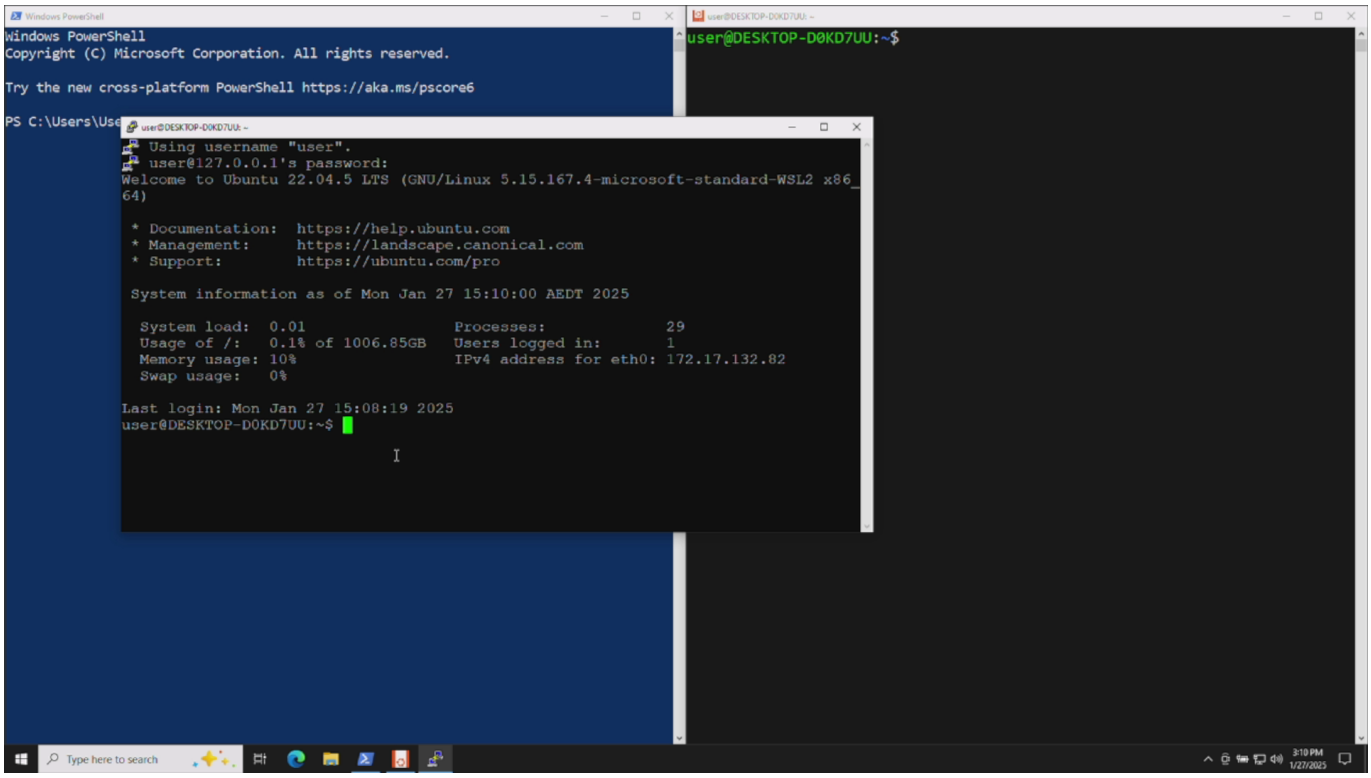
In the default configuration it seems that the WSL guest uses NAT to talk to the outside world as below:



access from Windows

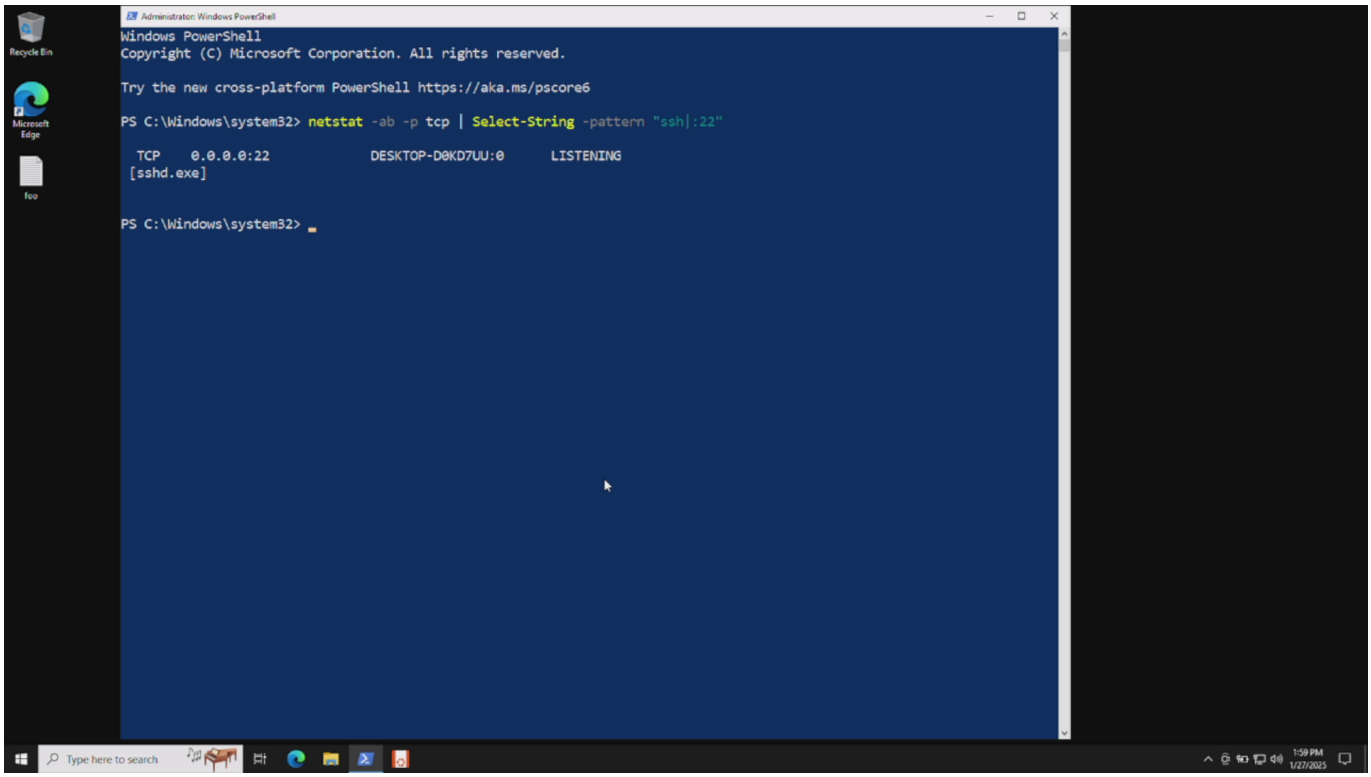
Oddly, WSL2 guest bindings to 127.0.0.1 seem to be honoured with routing to take precedence over those of the local Windows machine, because you can (for example) totally use `PuTTY` to talk to your Ubuntu guest directly:





external access

In the default configuration, getting at your guest from outside the Windows host can be a tad tricky due to the NAT, however this is doable when you install the Windows `openssh` server feature. This binds to `0.0.0.0:22` as below (this required admin privileges)



Once configured, you can get to your Ubuntu guest with the following

```
$ ssh -J <windows-user>@<windows-external-ip> <ubuntu-user>@<ubuntu-internal-ip>
```

For example, this is from my laptop:

```
mjch@yotta:~$ ssh -J user@172.31.4.140 user@172.17.132.82
user@172.31.4.140's password:
The authenticity of host '172.17.132.82 (<no hostip for proxy command>)' can't be established.
ED25519 key fingerprint is SHA256:HO320P4YgBHbBqFGpCPYlILQzVRXZ7pmp/VmovzPAm4.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.17.132.82' (ED25519) to the list of known hosts.
user@172.17.132.82's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Mon Jan 27 14:19:42 AEDT 2025

  System load:  0.0                Processes:             29
  Usage of /:   0.1% of 1006.85GB  Users logged in:       1
  Memory usage: 11%                IPv4 address for eth0: 172.17.132.82
  Swap usage:   0%

Last login: Mon Jan 27 14:15:04 2025
user@DESKTOP-D0KD7UU:~$ sudo shutdown -h -P now
[sudo] password for user:
Connection to 172.17.132.82 closed by remote host.
Connection to 172.17.132.82 closed.
mjch@yotta:~$
```

networking caveats

You will probably want to secure an OpenSSH daemon running on windows as this demo assumes that we're fine with allowing password-based logins which might not be to your taste

You might not even need to do this if the Ubuntu guest can be given an actual NIC, or at least a bridged virtual NIC, but I haven't explored that.

live demo

- bwahahahah

breadcrumbs

- https://learn.microsoft.com/en-us/windows/wsl/install
- https://learn.microsoft.com/en-us/windows/wsl/install-manual
- https://learn.microsoft.com/en-us/windows-server/administration/openssh/openssh_install_firstuse