Table of Contents

RunningATLASOnWSL2

Introduction

This TWiki is intended to collect my notes on how to get ATLAS software to run on the "Windows Subsystem For Linux" (WSL) within Windows 10.

Most of this is based on the surprisingly useful documentation as well as random snippets found in the depths of the internets.

As of June 2020, the 'May 2020 Update' (v2004) update of windows 10 replaced the old Linux subsystem (WSL) by a revised version (WSL2). The new version, unlike the first iteration, actually consists of a custom linux kernel running in a very lightweight HyperV VM. This allows things to work that did not work before - like fuse and, with it, CVMFS...

Now what can we do if we get CVMFS and a linux kernel within windows... let the mayhem begin!

Please be aware that this is just my collection of notes, I can not take any responsibility for the correctness of what is below, always read the proper documentation made by people who actually know what they are talking about. I tried to provide as many refs as possible.

Prerequisites

You need

- surprisingly, a computer running windows 10. Ideally with decent amounts of RAM and cores.
- According to the FAQ, any edition of windows 10 should work.
- You should have the 'May 2020' Windows 10 update installed. If you go to the settings app, "System --> Info" it should say "Version 2004" or higher. If Windows update doesn't offer you the update yet, you can force it via the updator tool download
- You may need to enable the HyperV hypervisor
- A terminal client. I suggest the 'Windows Terminal', which is actually a huge improvement to the old built-in windows terminal clients. See the very useful documentation for a ton of examples on how to configure and customize it. You can also try alternatives, for example I had good experiences with Cmder.

Getting started

This page illustrates how to get the WSL installed. We need WSL2, so just follow the instructions - in particular do not forget

```
wsl --set-default-version 2
```

like Max did at first...

I suggest to **not** install a distribution from the Store, though I did get ATLAS to run also on the Ubuntu 18.04 offered in the store. The reason is that we can just as well get CentOS7, and avoid the hassle of running a singularity container within a virtual machine (centos inside ubuntu inside windows), in a horrible mixture of inception and various horror movies.

Installing a distribution - not from the store

What I ended up doing was to install a CentOS7 image directly, without the store.

You can get the image from this github project. Look for the latest production release of CentOS7.

Then, it is as simple as unzipping the archive and running the installer. Note that, afterwards, the installer also serves as a handy configuration program of your existing installation!

How to launch

After installing, running your new linux is very easy!

From the (Cmd.exe or powershell) prompt

In the windows shell or powershell, you can launch a new session by simply typing

```
wsl
```

In case you have previously installed other distros from the store, you can either specify what you would like to run, for example

```
wsl -d CentOS7
```

You can use

```
wsl -l -v
```

to list all installed distributions.

And you can set what is launched when calling wsl without args via

```
wsl -s <name of default distro you would like>
```

From the Windows Terminal client

In the Windows Terminal, you should automatically get a launcher for a WSL CentOS session. Check the "down arrow" in the tab bar. You can also set the terminal's default session type to be your new distro. To do this, open the settings - you will get a json file to edit, any changes will immediately affect the terminal when you save. Find your session in the "profiles" list and copy the 'guid' entry. Then, add a

```
    "defaultProfile": "{<guid of your linux session>}",
```

near the top, or replace the existing line of this appearance.

Further configuring your centos distribution

You will initially get a root session. We don't want that for day-to-day use. So let's create a user account via

```
adduser <your desired user name>
passwd <your desired user name>
[ enter desired password ]
```

We also want to be able to use sudo to run administrative tasks without logging into root, so

```
gpasswd -a <your user name> wheel
yum install sudo
```

will ensure we can do that.

Now, log out of the CentOS7 system and run from a windows command line session in the folder where you unzipped the installer:

```
CentOS7.exe config --default-user <your newly created user name>
```

This will make the container start the session under your fresh user account.

For interoperability between win10 and the WSL, I recommend to install the "wslu" package:

```
sudo yum-config-manager --add-repo https://download.opensuse.org/repositories/home:/wslutilities/
sudo yum install wslu
```

Also, add the following to your bashrc or equivalent:

```
export DISPLAY=$(grep nameserver /etc/resolv.conf  | awk '{print $2}'):0.0
export LIBGL_ALWAYS_INDIRECT=1
```

This ensures that the X11 clients on the WSL will be able to connect to an X server running within windows.

Setting up ATLAS

First, we will need CVMFS. This, we can get by following the instructions at the CVMFS homepage within our centos session.

So, an example

For CVMFS_HTTP_PROXY, you can set "DIRECT".

Mounting it via autofs will likely **not** work.

Therefore, we will do it the way recommended in the ATLAS docker TWiki, using a manual mount. Make a few directories,

```
sudo mkdir -p /cvmfs/atlas.cern.ch
sudo mkdir -p /cvmfs/atlas-condb.cern.ch
sudo mkdir -p /cvmfs/atlas-nightlies.cern.ch
sudo mkdir -p /cvmfs/sft.cern.ch
sudo mkdir -p /cvmfs/sft-nightlies.cern.ch
sudo mkdir -p /cvmfs/cernvm-prod.cern.ch
sudo mkdir -p /cvmfs/cvmfs-config.cern.ch
```

and define the following in your .bashrc (or equivalent):

```
function MC(){
    sudo cvmfs_config killall
    sudo mount -t cvmfs atlas.cern.ch /cvmfs/atlas.cern.ch
    sudo mount -t cvmfs atlas-condb.cern.ch /cvmfs/atlas-condb.cern.ch
    sudo mount -t cvmfs atlas-nightlies.cern.ch /cvmfs/atlas-nightlies.cern.ch
    sudo mount -t cvmfs cernvm-prod.cern.ch /cvmfs/cernvm-prod.cern.ch
    sudo mount -t cvmfs cvmfs-config.cern.ch  /cvmfs/cvmfs-config.cern.ch
    sudo mount -t cvmfs sft.cern.ch /cvmfs/sft.cern.ch
    sudo mount -t cvmfs sft-nightlies.cern.ch /cvmfs/sft-nightlies.cern.ch
}
```

This way, you can simply run 'MC' ("Mount Cvmfs") the first time you launch a session to get CFMVS mounted. If you open further sessions, the environment will be inherited, so no need to do it again unless the VM crashes or is manally shut down.

You can also automate this, by creating a shortcut in your autostart folder. In Win10, this is hidden from the user in a folder like C:\Users\\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup Creating a shortcut with the content

```
C:\Windows\System32\wsl.exe -u root --exec /home/goblirsc/.setup_cvmfs
```

where .setup_cvmfs is a bash script in your WSL home folder containing the equivalent of the MC function above.

This will automatically run the CVMFS setup commands as root (without bothering you for a PW) every time you start up Windows.

Next, you will likely be missing a ton of packages. I suggest at least HEP_OSlibs, which will give you a set of libraries that a lot of our stuff depends on. Please read the documentation, but in short:

```
yum install https://linuxsoft.cern.ch/wlcg/centos7/x86_64/wlcg-repo-1.0.0-1.el7.noarch.rpm
yum install HEP_OSlibs
```

if you have this and CVMFS in place, you should be good to go... try the usual

```
export ATLAS_LOCAL_ROOT_BASE=/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase
alias setupATLAS='source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh'
```

and hope for the best!!

Usage experience to date

As of the date of writing (June 2021), most ATLAS use cases seem to be working in the Centos7/WSL2 environment, including:

- lsetup and the other ATLAS command line tools
- version control with git
- Interactive ROOT file browsing and plotting with TBrowser
- Remote file access and transfers with xrootd / rucio
- Grid usage (pathena)
- compiling and running the full chain, in both release 21 and 22.
- Reproducible CPU profiling of the reco chain (performance numbers very close to native linux)
- Running docker within WSL (using the windows docker version as backend) to work with useful ATLAS images such as GeoModel or Persint
- Event displays and geometry visualisation with VP1, GMEX, Persint. (Note: VP1 needs some careful management of the rendering settings - see below)
- Typesetting with the TeXLive installations provided via CVMFS
- Synchronisation of files with EOS using Cernbox

Things I have not been lucky with so far:

- AFS
- EOS(X) - but setting this up is difficult also on native Centos7, so this might be independent of WSL2

Advice for these two would be welcome.

Not tried yet:

- GPU compute is advertised as 'coming soon' and already available for testing in the Windows Insider program.

Note on running VP1

While it seems direct X11 / GPU support for WSL is coming "soon", for now I needed a small workaround to get it to run. This is based on this StackOverflow.

Two things seem to be needed:

- export LIBGL_ALWAYS_INDIRECT=0 on the Centos7 side (in the session where you plan to run VP1)
- Launch your X server **without** selecting native openGL (or without -wGL if you use the CLI) on the Win10 side

Then, you should get a fairly smooth usage experience.

More (possibly) useful stuff

Where can I see my Windows drives / files in?

They should show up under `/mnt/` Please note that using the windows filesystem for development is currently very slow (git clone of Athena can take a huge amount of time). So it's best to work under WSL2 (and if you're using Visual Studio Code, this is well supported).

Where can I see my files / files in Windows?

Note that MS recommends to not edit files on your centos virtual file system with windows applications directly. You can still access everything by either typing

```
explorer.exe .
```

within your linux session (**yes**, you can run all windows programs from there!) or by opening the explorer on the windows end and navigating to

```
\\wsl$\CentOS7\<linux path>
```

within the address bar.

X11 on windows

To get graphical output (mainly ROOT's TBrowser) of X, you need an X server on the Windows side. See for example here for documentation. I personally had success with VcXsrv as well as XMing.

Protip: Add your X-server to autostart, it hardly takes any background resources. This can be done by adding a shortcut to your 'Autostart' folder (C:\Users\\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup), with the target

```
"C:\Program Files (x86)\Xming\Xming.exe" -multiwindow -ac -clipboard
```

or the same for VcXsrv. Even without autostart, the option -ac is necessary for the X-server to work!

One important thing not covered on the source above is that - at least in my case - I had to add a firewall rule to **allow** inbound TCP traffic to my VcXsrv.exe (or XMing.exe) program from the subnet the WSL virtual machine uses. See this stackoverflow for some detail or ask Max if you are not sure what to do By default, my firewall would otherwise block all inbound traffic from the VM to the x server, preventing it from working. This was since reproduced on another machine - if in doubt, check the firewall rules, and look for any that affect your X server.

This might in the future get easier, MS seems to be planning "native" X11 support: https://docs.microsoft.com/en-us/windows/wsl/tutorials/gui-apps

WSL in VS code

Another neat feature is that visual studio code has WSL integration. There is a "remote WSL" plugin. If you launch VS code from the **linux** session, it will automatically launch the editor on the **Windows** end and the vs code server on the linux side. As a result, you will be able to browse your linux file system as if you were running within linux, but using your already configured VS code installation from Win10.

My VM is misbehaving / using naughty words / taking over the world, how do I stop it?

If your VM misbehaves or you changed the settings (see next item) and you want to restart it, you can kill all running sessions with

```
wsl --shutdown
```

Opening a new session will then automatically launch a new VM.

My sessions are being killed while compiling / running memory-intensive work

If you find your sessions being killed in memory-heavy operation such as building atlas releases, you may be encountering RAM starvation. By default, the VM is greedy and eats up to 80% of the system RAM, which can lead to it being killed by Win if the computer runs out of RAM. See here for a ticket on this. What you can do about it is to create a `.wslconfig` file inside your Windows 10 user directory containing the following lines

```
[wsl2]
memory=10GB # Limits VM memory in WSL 10 GB. I found this effective on my 16GB box, adapt to suit
swap=16GB  # also allow for a bit of swap space on a second virtual disk
```

One underlying reason for this is that the Windows host by default does not attempt to clear the linux cache to free RAM. This is further explained here. As mentioned there, it's worth trying to manually clear the cache with

```
echo 1 > /proc/sys/vm/drop_caches
```

as root (sudo didn't work at least for me) when you find your vmmem process is growing too large.

My windows partition is small, how can I move my VM's virtual disk somewhere else?

See here - `wsl import` and `wsl export` are your friends! Note that this is also useful to back up your fully configured setup, or to copy it to other computers! It is indeed possibly to transfer your entire, working setup from one of your computers to another with minimal effort using this approach (or to create a backup). You'll just need to connect all the plumbing on the Windows end (X server, firewall, WSL2 installation, etc)

setupATLAS gives me weird warnings and / or my PATH gets corrupted!

Unfortunately, Microsoft LOVES whitespaces in directory names. More unfortunately, several of our ATLAS scripts do not properly escape arguments in export / test / etc statements. As a result, horrible things may happen.

You can prevent them by running the CentOS installer / config updator with

```
Centos7.exe config --append-path off
```

which will remove your windows PATH from the WSL one.

You may want to manually add back some useful stuff, like the windows installation of VS code or the windows system folders, to get explorer.exe for example. What you can do if your favourite program has whitespaces in its path is to just make a symlink to its folder without whitespaces and manually add this to PATH in your .bashrc! For example in my case:

```
# Expand path with folders from windows 10
export PATH="/home/goblirsc/auxFolders/VSC/bin:$PATH"   #points to VS code
export PATH="/home/goblirsc/auxFolders/WindowsApps:$PATH"   # points to the WindowsApps folder
export PATH="/mnt/c/WINDOWS/system32:$PATH"
export PATH="/mnt/c/WINDOWS:$PATH"
export PATH="/mnt/c/WINDOWS/System32/OpenSSH:$PATH"
export PATH="/mnt/c/Python27/:/mnt/c/Python27/Scripts:$PATH"
```

where I have two symlinks (auxFolders/VSC and auxFolders/WindowsApps) pointing to the folders where VS code and some windows applications are stored (taken from the old PATH with the whitespaces).

-- MaximilianGoblirschKolb - 2021-06-18

This topic: Sandbox > RunningATLASOnWSL2
Topic revision: r8 - 2021-08-19 - NoraEmiliaPettersson