Making a docker image to run the Waterfox browser

Neville Jackson

29 Sep 2022

Contents

1 Introduction

This project is about learning how to compose a Dockerfile which runs an X11 application. The Waterfox browser was chosen as an example X11 application, because Waterfox is not available as a package in most Linux distributions, so the result might actually be a useful means of obtaining Waterfox without having to download a tarfile and do a local install.

Getting a running Docker container to talk to the X11 server in the host system is relatively easy. Waterfox is a large and complex X application, and most of the challenge of this project is in setting up in the Docker image a working environmejnt for Waterfox.

2 What is involved in a normal Waterfox install?

There is a document [19] on that. Briefly the steps are

1. Download a tarfile from the Waterfox website [18] or from the Github site [20].
2. Unpack the tarfile in /usr/local/src

   ```
   bzip2 -dc waterfox-G4.1.1.1.en-US.linux-x86_64.tar.bz2 | tar xvf -
   ```

3. Make a link in /usr/local/bin to point to the Waterfox binary

   ```
   ln /usr/local/src/waterfox/waterfox /usr/local/bin/waterfox
   ```

4. Define the desktop icon by adding a file waterfox.desktop to the directory /usr/share applications. The file waterfox.desktop does not come with the tarfile, but is available here [19]

To make a docker image, we need to implement steps 1 to 3 in a Dockerfile. Step 4 can be omitted, as there will be no icon if running Waterfox in a container. The container will have to be started from the command line.

3 How to run a docker container requiring access to the host Xserver

We assume the host is running X11 (not Wayland). Set up the following primitive Dockerfile.

```
FROM debian
WORKDIR /home/demo
RUN apt-get -y update
RUN apt-get -y upgrade
RUN apt-get -y x11-apps
RUN groupadd -g 1000 demo
```

```
RUN useradd -d /home/demo -s /bin/bash -m demo -u 1000 -g 1000
USER demo
ENV HOME /home/demo
CMD /usr/bin/xcalc
```

Build this with

```
docker build -t nevj/test:v2 .
Sending build context to Docker daemon  2.048kB
Step 1/10 : FROM debian
 ---> 07d9246c53a6
Step 2/10 : WORKDIR /home/demo
 ---> Running in 2e0e40e0fd4c
..........
Successfully built 517c1937106a
Successfully tagged nevj/test:v2
```

Then run it with

```
[nevj@trinity dtest]$ xhost +
[nevj@trinity dtest]$ docker run -v /tmp/.X11-unix:/tmp/.X11-unix \
          -e DISPLAY=$DISPLAY -h $HOSTNAME \
          -v $HOME/.Xauthority:/home/nevj/.Xauthority nevj/test:v2
```

and the xcalc app appears in a separate X window, as shown in Figure 1

The *xhost +* statement is needed to give the container permission to communicate with the host X11 server. The purpose of the options included in the *docker run* statement are as follows

-v /tmp/.X11-unix:/tmp/.X11-unix defines a *mount* for a volume. The string */tmp/.X11-unix:/tmp/.X11-unix* is the mount point on the host. The part before *':'* is the volume name on the host. The part after *':'* is where the file or directory is mounted oinside the container. A volume is just a file or directory in the host filesystem.

-e DISPLAY=$DISPLAY sets the environment variable DISPLAY in the container

-h $HOSTNAME sets the container hostname

-v $HOME/.Xauthority:/home/nevj/.Xauthority sets the .Xauthority file of the container to the .Xauthority file of the host user who initiated the *run* statement again using a volume mount.

nevj/test:v2 is the name of the image.

Volumes are the mechanism used by Docker containers to share data between the host and the container (and between containers). They are needed in this case because the container is an X11 client, while the host acts as the X11 server, so information has to pass between container and host. A docker container can only write on the part of the host filesystem defined as a docker volume. There is also *tmpfs mount* which is a temporary ram disk used for non-permanent data within a container.

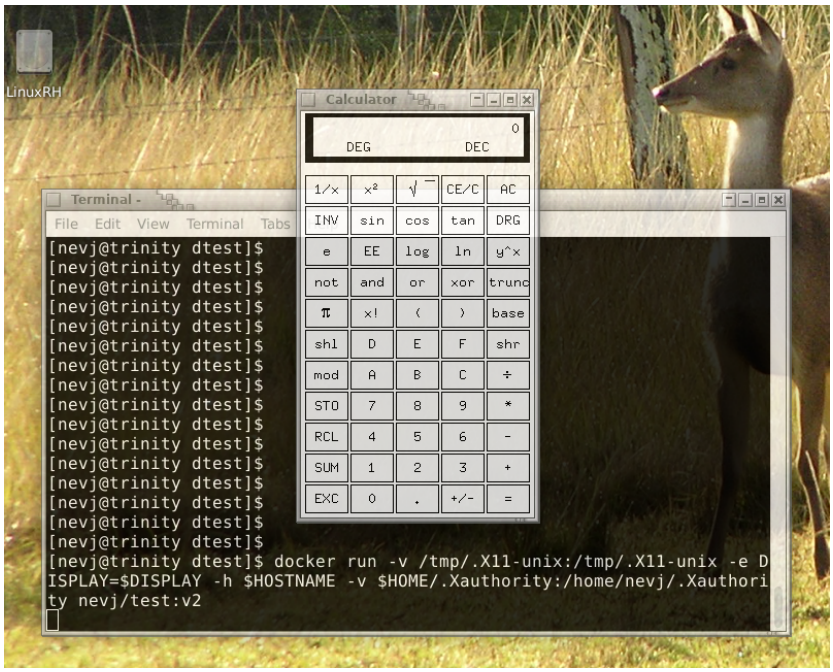

Figure 1: Xcalc app started from a Docker container

4 Attempt at a Waterfox Dockerfile, using a Debian parent image

There is no new principle involved in running Waterfox rather than the simple xcalc app, in a docker container. The Waterfox browser is a large program with many dependencies, so the Dockerfile will have to build up a complete working environment specifically for Waterfox.

I chose a Debian parent image for a first attempt, because Debian is familiar and is most likely to compatable with Waterfox. The Debian parent image is large. We shall try later to repeat the job with a smaller parent image.

4.1 Finding the dependencies of a binary program file

There is a really good article [17] on identifying dependencies of a runtime binary when making a dockerfile.

If one looks at the unpacked Waterfox distribution tarfile, it contains the files shown in Figure 2

The *.so files shown in green are dynamically loaded libraries. Presumably these are all dependencies of waterfox, but they may not be the only dependencies. There is a file TwemojiMozilla.ttf in the fonts subdirectory, and there are various default configuration files.

Figure 2: Files contained in the waterfox distribution tarfile

An initial check for dependencies can be obtained with *ldd*

```
nevj@mary:/usr/local/src/waterfox$ ldd waterfox
linux-vdso.so.1 (0x00007ffc3eb95000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f2a6d46c000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f2a6d466000)
libstdc++.so.6 => /lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f2a6d299000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f2a6d155000)
libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f2a6d13b000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f2a6cf76000)
/lib64/ld-linux-x86-64.so.2 (0x00007f2a6d569000)
```

This is best done in a Debian system in which waterfox has been installed There are 8 libraries called directly by waterfox, and none of them are present in the distribution tarfile. These are libraries required for waterfox to start up. Note that linux-vdso.so.1 is a virtual library, ie it is built into the kernel and does not exist as a package. The other 7 should each be supplied by a package. We shall leave the problem of finding the package names which supply each library later.

There may also be libraries which waterfoox loads dynamically while executing. To find these we do the following, again in a Debian system

```
nevj@mary:~$ LD_DEBUG=libs waterfox >&logfile

nevj@mary:~$ cat logfile
     2572:     find library=libpthread.so.0 [0]; searching
      2572:       search cache=/etc/ld.so.cache
      2572:        trying file=/lib/x86_64-linux-gnu/libpthread.so.0
      2572:
      2572:     find library=libdl.so.2 [0]; searching
      2572:       search cache=/etc/ld.so.cache
      2572:        trying file=/lib/x86_64-linux-gnu/libdl.so.2
      2572:
      2572:     find library=libstdc++.so.6 [0]; searching
      2572:       search cache=/etc/ld.so.cache
      2572:        trying file=/lib/x86_64-linux-gnu/libstdc++.so.6
```

```
 2572:
 2572:     find library=libm.so.6 [0]; searching
 2572:      search cache=/etc/ld.so.cache
 2572:       trying file=/lib/x86_64-linux-gnu/libm.so.6
 2572:
 2572:     find library=libgcc_s.so.1 [0]; searching
 2572:      search cache=/etc/ld.so.cache
 2572:       trying file=/lib/x86_64-linux-gnu/libgcc_s.so.1
2572:
 2572:     find library=libc.so.6 [0]; searching
 2572:      search cache=/etc/ld.so.cache
 2572:       trying file=/lib/x86_64-linux-gnu/libc.so.6
 2572:
 2572:
 2572:     calling init: /lib/x86_64-linux-gnu/libpthread.so.0
 2572:
 2572:
 2572:     calling init: /lib/x86_64-linux-gnu/libc.so.6
 2572:
 2572:
 2572:     calling init: /lib/x86_64-linux-gnu/libgcc_s.so.1
 2572:
 2572:
 2572:     calling init: /lib/x86_64-linux-gnu/libm.so.6
 2572:
 2572:
 2572:     calling init: /lib/x86_64-linux-gnu/libstdc++.so.6
 2572:
 2572:
 2572:     calling init: /lib/x86_64-linux-gnu/libdl.so.2
 2572:
 2572:
 2572:     initialize program: waterfox
 2572:
 2572:
 2572:     transferring control: waterfox
 2572:
 2572:
 2572:     calling init: /usr/local/src/waterfox/libnspr4.so
 2572:
 2572:
 2572:     calling init: /usr/local/src/waterfox/libplc4.so
 2572:
 2572:
 2572:     calling init: /usr/local/src/waterfox/libplds4.so
 2572:
```

```
     2572:     find library=librt.so.1 [0]; searching
     2572:      search cache=/etc/ld.so.cache
     2572:       trying file=/lib/x86_64-linux-gnu/librt.so.1
     2572:
     2572:
     2572:     calling init: /lib/x86_64-linux-gnu/librt.so.1
     2572:
     2572:
     2572:     calling init: /usr/local/src/waterfox/libmozsandbox.so
     2572:
     2572:
     2572:     calling init: /usr/local/src/waterfox/liblgpllibs.so
     2572:
     2572:
     2572:     calling init: /usr/local/src/waterfox/libnssutil3.so
     2572:
     2572:
     2572:     calling init: /usr/local/src/waterfox/libnss3.so
     2572:
 ..................
    2796:     calling init: /lib/x86_64-linux-gnu/libX11-xcb.so.1
     2796:
     2796:
     2796:     calling init: /usr/local/src/waterfox/libxul.so
     2796:
     2796:     /usr/local/src/waterfox/waterfox: error: symbol lookup error:
undefined symbol: nspr_use_zone_allocator (fatal)
     2796:
     2796:     calling init: /usr/local/src/waterfox/libsoftokn3.so
     2796:
     2796:
     2796:     calling init: /usr/local/src/waterfox/libfreeblpriv3.so
     2796:
JavaScript error: resource:///gre/modules/PartitioningExceptionListService.jsm
, line 69: NS_ERROR_MALFORMED_URI: Component returned failure code: 0x804b000
a (NS_ERROR_MALFORMED_URI) [nsIPartitioningExceptionListObserver.onExceptionL
istUpdate]
console.error: PushService:
  clearOriginData: Error clearing origin data:
  TypeError
JavaScript error: resource:///modules/Interactions.jsm, line 209: NS_ERROR_FA
ILURE: Component returned failure code: 0x80004005 (NS_ERROR_FAILURE) [nsIUse
rIdleService.removeIdleObserver]

###!!! [Parent][RunMessage] Error: Channel closing: too late to send/recv, me
ssages will be lost
```

Lots of messages, had to truncate it, was 5744 lines. The first 3 lines say it is searching for a system library called *libpthread*. The lines beginning *calling init* indicate the library has been found and give its absolute path. Some libraries may not be found, and they will have no line beginning *calling init*. We can get a list of libraries found with

```
nevj@mary:~$  cat out | grep init > outinit
nevj@mary:~$ wc outinit
  715  2878 45875 outinit
nevj@mary:~$  cat outinit
   2572: calling init: /lib/x86_64-linux-gnu/libpthread.so.0
      2572: calling init: /lib/x86_64-linux-gnu/libc.so.6
      2572: calling init: /lib/x86_64-linux-gnu/libgcc_s.so.1
      2572: calling init: /lib/x86_64-linux-gnu/libm.so.6
      2572: calling init: /lib/x86_64-linux-gnu/libstdc++.so.6
      2572: calling init: /lib/x86_64-linux-gnu/libdl.so.2
      2572: initialize program: waterfox
      2572: calling init: /usr/local/src/waterfox/libnspr4.so
      2572: calling init: /usr/local/src/waterfox/libplc4.so
      2572: calling init: /usr/local/src/waterfox/libplds4.so
      2572: calling init: /lib/x86_64-linux-gnu/librt.so.1
      2572: calling init: /usr/local/src/waterfox/libmozsandbox.so
      2572: calling init: /usr/local/src/waterfox/liblgpllibs.so
      2572: calling init: /usr/local/src/waterfox/libnssutil3.so
      2572: calling init: /usr/local/src/waterfox/libnss3.so
      2572: calling init: /usr/local/src/waterfox/libsmime3.so
      2572: calling init: /usr/local/src/waterfox/libmozsqlite3.so
      2572: calling init: /usr/local/src/waterfox/libssl3.so
      2572: calling init: /lib/x86_64-linux-gnu/libgpg-error.so.0
  ............
```

So there are still 715 of them. The first 6 are the startup dependencies, and the rest are runtime dependencies

There is also the question of configuration files. Most of these are hopefully included in the Waterfox tarfile. We can check

```
strace --trace=open,openat /usr/local/bin/waterfox >&strace.out

Lots of libraries which we can ignore
....
Conf files
openat(AT_FDCWD, "/etc/host.conf", O_RDONLY|O_CLOEXEC) = 7
openat(AT_FDCWD, "/etc/resolv.conf", O_RDONLY|O_CLOEXEC) = 7
openat(AT_FDCWD, "/etc/nsswitch.conf", O_RDONLY|O_CLOEXEC) = 7
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 7
......
openat(AT_FDCWD, "/home/nevj/.Xauthority", O_RDONLY) = 5
```

```
openat(AT_FDCWD, "/usr/share/X11/locale/locale.alias", O_RDONLY) = 5
openat(AT_FDCWD, "/usr/share/X11/locale/locale.alias", O_RDONLY) = 5
openat(AT_FDCWD, "/usr/share/X11/locale/locale.dir", O_RDONLY) = 5
openat(AT_FDCWD, "/usr/share/X11/locale/en_US.UTF-8/XLC_LOCALE", O_RDONLY) =
5
openat(AT_FDCWD, "/home/nevj/.Xdefaults-mary", O_RDONLY) = -1 ENOENT (No such
 file or directory)
....
openat(AT_FDCWD, "/home/nevj/.waterfox/profiles.ini", O_RDONLY) = 8
openat(AT_FDCWD, "/home/nevj/.Xauthority", O_RDONLY) = 9
....
openat(AT_FDCWD, "/usr/local/src/waterfox/omni.ja", O_RDONLY) = 14
openat(AT_FDCWD, "/usr/local/src/waterfox/browser/omni.ja", O_RDONLY) = 15
openat(AT_FDCWD, "/sys/devices/system/cpu", O_RDONLY|O_NONBLOCK|O_CLOEXEC|O_D
IRECTORY) = 16
openat(AT_FDCWD, "/usr/local/src/waterfox/defaults/pref", O_RDONLY|O_NONBLOCK
|O_CLOEXEC|O_DIRECTORY) = 16
openat(AT_FDCWD, "/usr/local/src/waterfox/defaults/pref/channel-prefs.js", O_
RDONLY) = 17
......
openat(AT_FDCWD, "/usr/share/fontconfig/conf.avail/11-lcdfilter-default.conf"
, O_RDONLY|O_CLOEXEC) = 38
openat(AT_FDCWD, "/etc/fonts/conf.avail/20-unhint-small-dejavu-lgc-sans-mono.
conf", O_RDONLY|O_CLOEXEC) = 38
.....
openat(AT_FDCWD, "/etc/os-release", O_RDONLY) = 37
openat(AT_FDCWD, "/sys/devices/system/cpu/present", O_RDONLY) = 37
openat(AT_FDCWD, "/usr/share/locale/locale.alias", O_RDONLY) = 38
openat(AT_FDCWD, "/usr/share/icons/Tango/index.theme", O_RDONLY) = 38
openat(AT_FDCWD, "/usr/share/icons/Tango/icon-theme.cache", O_RDONLY) = 38
......
openat(AT_FDCWD, "/usr/share/icons/index.theme", O_RDONLY) = -1 ENOENT (No su
ch file or directory)
openat(AT_FDCWD, "/usr/share/pixmaps/cursors/e-resize", O_RDONLY) = -1 ENOENT
 (No such file or directory)
....
penat(AT_FDCWD, "/home/nevj/.icons/Adwaita/cursors/e-resize", O_RDONLY) = -1
 ENOENT (No such file or directory)
openat(AT_FDCWD, "/home/nevj/.icons/Adwaita/index.theme", O_RDONLY) = -1 ENOE
NT (No such file or directory)
.....
openat(AT_FDCWD, "/home/nevj/.config/xfce-mimeapps.list", O_RDONLY) = -1 ENOE
NT (No such file or directory)
openat(AT_FDCWD, "/home/nevj/.config/mimeapps.list", O_RDONLY) = 80
....
openat(AT_FDCWD, "/usr/local/share/applications/xfce-mimeapps.list", O_RDONLY
```

```
) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/share/applications/mimeapps.list", O_RDONLY) = -
1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/share/applications/defaults.list", O_RDONLY) = -
1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/share/applications/mimeinfo.cache", O_RDONLY) =
-1 ENOENT (No such file or directory)
.....
openat(AT_FDCWD, "/usr/share/fonts/truetype/dejavu/DejaVuSans-Bold.ttf", O_RD
ONLY) = 88
JavaScript error: resource:///actors/AboutNewTabParent.jsm, line 23: InvalidS
tateError: An attempt was made to use an object that is not, or is no longer,
 usable
.....
JavaScript error: resource://gre/modules/PartitioningExceptionListService.jsm
, line 69: NS_ERROR_MALFORMED_URI: Component returned failure code: 0x804b000
a (NS_ERROR_MALFORMED_URI) [nsIPartitioningExceptionListObserver.onExceptionL
istUpdate]
....
penat(AT_FDCWD, "/home/nevj/.icons/Adwaita/cursors/hand2", O_RDONLY) = -1 EN
OENT (No such file or directory)
openat(AT_FDCWD, "/home/nevj/.icons/Adwaita/index.theme", O_RDONLY) = -1 ENOE
NT (No such file or directory)
.....
```

Some of the files have the message *(No such file or directory)*. They are probably not serious issues... the above was done in a Debian Waterfox install which works satisfactorily. I have only shown a selection. The full output above was 1402 lines.

4.2 Which Debian packages provide these files?

We have a long list of files which are dependencies. We need to convert this into a list of *.deb* packages which supply the necessary files. That should reduce the information to a much shorter list of packages. The tool for finding which package supplies a file is *apt-file*.

apt-file is not part of a standard Debian system. We need to install it

```
root@trinity:/home/nevj# apt-get install apt-file
......
The following additional packages will be installed:
  libapt-pkg-perl libexporter-tiny-perl liblist-moreutils-perl
  liblist-moreutils-xs-perl libregexp-assemble-perl
The following NEW packages will be installed:
  apt-file libapt-pkg-perl libexporter-tiny-perl liblist-moreutils-perl
  liblist-moreutils-xs-perl libregexp-assemble-perl
0 upgraded, 6 newly installed, 0 to remove and 6 not upgraded.
```

```
Need to get 322 kB of archives.
After this operation, 901 kB of additional disk space will be used.
The following additional packages will be installed:
  libapt-pkg-perl libexporter-tiny-perl liblist-moreutils-perl
  liblist-moreutils-xs-perl libregexp-assemble-perl
The following NEW packages will be installed:
  apt-file libapt-pkg-perl libexporter-tiny-perl liblist-moreutils-perl
  liblist-moreutils-xs-perl libregexp-assemble-perl
0 upgraded, 6 newly installed, 0 to remove and 6 not upgraded.
Need to get 322 kB of archives.
After this operation, 901 kB of additional disk space will be used.
The following additional packages will be installed:
  libapt-pkg-perl libexporter-tiny-perl liblist-moreutils-perl
  liblist-moreutils-xs-perl libregexp-assemble-perl
The following NEW packages will be installed:
  apt-file libapt-pkg-perl libexporter-tiny-perl liblist-moreutils-perl
  liblist-moreutils-xs-perl libregexp-assemble-perl
0 upgraded, 6 newly installed, 0 to remove and 6 not upgraded.
Need to get 322 kB of archives.
After this operation, 901 kB of additional disk space will be used.
The following additional packages will be installed:
  libapt-pkg-perl libexporter-tiny-perl liblist-moreutils-perl
  liblist-moreutils-xs-perl libregexp-assemble-perl
The following NEW packages will be installed:
  apt-file libapt-pkg-perl libexporter-tiny-perl liblist-moreutils-perl
  liblist-moreutils-xs-perl libregexp-assemble-perl
0 upgraded, 6 newly installed, 0 to remove and 6 not upgraded.
Need to get 322 kB of archives.
After this operation, 901 kB of additional disk space will be used.
The following additional packages will be installed:
  libapt-pkg-perl libexporter-tiny-perl liblist-moreutils-perl
  liblist-moreutils-xs-perl libregexp-assemble-perl
The following NEW packages will be installed:
  apt-file libapt-pkg-perl libexporter-tiny-perl liblist-moreutils-perl
  liblist-moreutils-xs-perl libregexp-assemble-perl
0 upgraded, 6 newly installed, 0 to remove and 6 not upgraded.
Need to get 322 kB of archives.
After this operation, 901 kB of additional disk space will be used.
......
```

Then we need to generate apt-file's database

```
root@trinity:/home/nevj# apt-file update
```

Then we can search

```
nevj@trinity:~$ apt-file search libpthread.so.0
```

```
libc6: /lib/x86_64-linux-gnu/libpthread.so.0
......
Lots of other libraries .... it is better to use dthe full path
nevj@trinity:~$ apt-file search /lib/x86_64-linux-gnu/libpthread.so.0
libc6: /lib/x86_64-linux-gnu/libpthread.so.0
 So we just get the relevant one
```

There is an alternative to apt-file

```
nevj@trinity:~$ dpkg -s libpthread.so.0
dpkg-query: package 'libpthread.so.0' is not installed and no information is available
Use dpkg --info (= dpkg-deb --info) to examine archive files.
```

So *dpkg -s* only works for installed library files, *apt-file works for all available packages*.

Now we have the tools , it is a mere matter of feeding several hundred library files to *apt-file search* .

```
#  a bit of hand editing
$  cp outinit outinit.edit
$  vi outinit.edit

# make a simple awk script to extract the 4th field
$ vi field4.awk
{print $4}

# run the script
$ awk -f field4.awk < outinit.edit > libfiles.txt

# and the output is
$ more libfiles.txt
/lib/x86_64-linux-gnu/libpthread.so.0
/lib/x86_64-linux-gnu/libc.so.6
/lib/x86_64-linux-gnu/libgcc_s.so.1
/lib/x86_64-linux-gnu/libm.so.6
/lib/x86_64-linux-gnu/libstdc++.so.6
/lib/x86_64-linux-gnu/libdl.so.2
/usr/local/src/waterfox/libnspr4.so
/usr/local/src/waterfox/libplc4.so
....... too many to list

  $ wc libfiles.txt
  711   711 26829 libfiles.txt
So 711 libraries in total
```

So now we have out list of libraries, we need a shell script to feed it to apt-file.

```
#! /bin/csh
```

```
foreach lib (`cat libfiles.txt`)
 apt-file --fixed-string -l search $lib >>& package.txt
end
```

Now if we run that script we get

```
chmod 755 packages.csh
./packages.csh
wc package.txt
 125  125 1124 package.txt
```

So we have reduced from 711 libraries to 125 packages. However there are duplicates so

```
sort <package.txt | uniq >uniqpackage.txt
wc uniqpackage.txt
 21  21 249 uniqpackage.txt
cat uniqpackage.txt
dconf-gsettings-backend
gvfs
gvfs-libs
libc6
libcom-err2
libdbus-1-3
libelogind0
libexpat1
libgcc-s1
libgdk-pixbuf-2.0-0
libgl1-mesa-dri
libgpg-error0
libkeyutils1
liblzma5
libnss-mdns
libpcre3
libpulse0
libselinux1
libtirpc3
python3-minimal
zlib1g
```

There are 21 unique packages reguired. Now we can write the Dockerfile with some confidence that it will supply an suitable environment for Waterfox.

4.3 First Debian parent Dockerfile

For a first attempt, we will use a copy of the waterfox tarfile which I have downloaded and placed in the Dockerfile directory. Later we will try to setup the Dockerfile to download it from the Waterfox site [20].

We will work within the container in a directory */wfox*. There will be a user called *wfox* and the container will execute the waterfox binary.

Our first Dockerfile is as follows

```
# get a parent image
FROM debian:stable-20220801

#set working dir inside container
WORKDIR /wfox

# get waterfox
COPY . .

# install waterfox.desktop file
RUN mkdir /usr/share/applications && \
  cp waterfox.desktop /usr/share/applications

#
RUN cd /etc/apt && \
# apt debian setup
  cp sources.list sources.list.orig && \
  sed 's/main/main contrib non-free/' sources.list.orig >sources.list && \
  cd /wfox && \
  apt-get update && \
  apt-get upgrade -y && \
# apt install libraries
  apt-get install -y  dconf-gsettings-backend gvfs gvfs-libs libc6 \
  libcom-err2 libdbus-1-3 libelogind0 libexpat1 libgcc-s1 \
  libgdk-pixbuf-2.0-0 libgl1-mesa-dri libgpg-error0 libkeyutils1 \
  liblzma5 libnss-mdns libpcre3 libpulse0 libselinux1 libtirpc3 \
  python3-minimal zlib1g

#    unpack waterfox distro tarfile
  cd /wfox && \
  bzip2 -dc waterfox-G4.1.4.en-US.linux-x86_64.tar.bz2 | tar xvf -
# setup environment
RUN groupadd -g 1000 wfox
RUN useradd -d /home/wfox -s /bin/bash -m wfox -u 1000 -g 1000
USER wfox
ENV HOME /home/wfox
CMD /wfox/waterfox/waterfox
```

Then build an image using the above Dockerfile and run it

```
# work directory in host system
cd ~/Waterfox.docker
ls
```

```
Dockerfile  junk      waterfox-G4.1.4.en-US.linux-x86_64.tar.bz2  wget
buildrun    libsused  waterfox.desktop                            x11docker

# docker build
docker build -t wfoxdeb:v1 .
.....
Successfully built 7d1e1798583a
Successfully tagged wfoxdeb:v1

#docker run
docker run -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY \
   -h $HOSTNAME -v $HOME/.Xauthority:/home/nevj/.Xauthority \
   xfoxdeb:v1

XPCOMGlueLoad error for file /wfox/waterfox/libmozgtk.so:
libgtk-3.so.0: cannot open shared object file: No such file or directory
Couldn't load XPCOM.
```

What the message means is that the library *libmozgtk.so* which comes with the waterfox tarfile requires the library *libgtk-3.so.0*, but it was not present in the container system.

So our wonderful, complicated procedure for reducing 711 libraries to 21 packages has missed something. What is missing is that we traced all the libraries called by the *waterfox* binary, but we neglected to trace all the libraries called by the 18 special waterfox libraries included in the waterfox tarfile.

We need to do the library tracing and package finding all over again for each of the 18 special libraries.

4.4 Dependencies of the 18 special *waterfox* libraries.

We setup a list of the 18 special libraries in a file *list18libs.txt*. We need to process each library in the list with *ldd* so we setup a script. We can not execute a library, so we cant use *strace* or *LD_DEBUG* , but we can use *ldd*.

```
cd /usr/local/src/waterfox
ls | grep so
libfreeblpriv3.so
liblgpllibs.so
libmozavcodec.so
libmozavutil.so
libmozgtk.so
libmozsandbox.so
libmozsqlite3.so
libmozwayland.so
libnspr4.so
libnss3.so
libnssckbi.so
libnssutil3.so
```

```
libplc4.so
libplds4.so
libsmime3.so
libsoftokn3.so
libssl3.so
libxul.so
```

Store this on a file

```
ls | grep so > list18libs.txt
# make a script to process each of these with ldd
#! /bin/csh
foreach lib (`cat list18libs.txt`)
  ldd lib >>& list18libsdeps.txt
end
```

How many libraries found?

```
wc list18libsdeps.txt
  291  1107 20639 list18libsdeps.txt
```

What packages contain these libraries? t have a look at list18libsdeps.txt

```
head list18libsdeps.txt
linux-vdso.so.1 (0x00007fff591c5000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f346a0d9000)
libnspr4.so => /lib/x86_64-linux-gnu/libnspr4.so (0x00007f346a098000)
libplc4.so => /lib/x86_64-linux-gnu/libplc4.so (0x00007f346a091000)
libplds4.so => /lib/x86_64-linux-gnu/libplds4.so (0x00007f346a08c000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f346a086000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f3469ec1000)
/lib64/ld-linux-x86-64.so.2 (0x00007f346a1eb000)
linux-vdso.so.1 (0x00007ffec61a5000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007ffaa2b5d000)
......
```

We need the third field from the file list18libsdeps.txt, so use our awk script again, after removing the first line

```
cat field3.awk
{print $3}

awk -f field3.awk <list18libsdeps.txt2 >list18libsdeps.txt3
head list18libsdeps.txt3
/lib/x86_64-linux-gnu/libpthread.so.0
/lib/x86_64-linux-gnu/libnspr4.so
/lib/x86_64-linux-gnu/libplc4.so
/lib/x86_64-linux-gnu/libplds4.so
/lib/x86_64-linux-gnu/libdl.so.2
/lib/x86_64-linux-gnu/libc.so.6
```

```
/lib/x86_64-linux-gnu/libpthread.so.0
/lib/x86_64-linux-gnu/libdl.so.2
.......
```

We have extracted the full pathname of each library, there are a few blank line, so we edit those out (may not actually matter, then

```
wc list18libsdeps.txt3
 245  245 8897 list18libsdeps.txt3
```

There are 245 library paths left. We can process this with *apt-file* to see what packages supply these libraries

```
cat packages18.csh
#! /bin/csh
foreach lib (‘cat list18libsdeps.txt3‘)
 apt-file --fixed-string -l search $lib >>& package18.txt
end

./packages18.

wc package18.txt
 85  85 596 package18.txt

head package18.txt
libc6
libc6
libc6
libc6
libc6
libc6
libgcc-s1
libc6
libc6
libc6
......
```

We have reduced te 245 library paths to 85 packages, but we can see from the partial listing there are lots of duplicates, so

```
sort package18.txt | uniq > uniqpackage18.txt
wc uniqpackage18.txt
 10  10 101 uniqpackage18.txt
cat uniqpackage18.txt
libc6
libdbus-1-3
```

```
libelogind0
libexpat1
libgcc-s1
libgpg-error0
liblzma5
libpcre3
libselinux1
zlib1g
```

There are 10 unique package names, and all 10 are already present in our previous list of 21 required packages. So this does not solve the dependency problem ?

An investigation of each of the above steps reveals that the *apt-file* statement in the script packages18.csh does not require the option *–fixed-string*. Running this step again leads to

```
cat packages18.csh
#! /bin/csh
foreach lib (‘cat list18libsdeps.txt3‘)
 apt-file  -l search $lib >>& package18.txt
end

./packages18.chh

wc package18.txt
 338  338 3128 package18.txt
```

So now we have 338 packages instead of 245. Removing duplicates again gives

```
sort package18.txt | uniq >uniqpackage18.txt

wc uniqpackage18.txt
 72  72 853 uniqpackage18.txt

head uniqpackage18.txt
libatk1.0-0
libatk-bridge2.0-0
libatspi2.0-0
libblkid1
libbrotli1
libbsd0
libc6
libcairo2
libcairo-gobject2
libdatrie1
......
```

We now have 72 package some of which are not included in the 21 package dependencies of the waterfox binary. We need to remove from these 72 those which are in both lists

```
sort uniqpackage.txt > sorteduniqpackage.txt
sort uniqpackage18.txt > sorteduniqpackage18.txt
comm -3 sorteduniqpackage.txt sorteduniqpackage18.txt >commpackage.txt

wc commpackage.txt
 71  71 921 commpackage.txt
```

So we still have 71 packages to add to the Dockerfile.

4.5 Second Debian parent Dockerfile attempt

We now add an apt-get install for the 71 packages required to satisfy Waterfox's 18 special libraries. So version 2 of our Dockerfile looks as follows

```
FROM debian:stable-20220801
#set working dir inside container
WORKDIR /wfox
# get waterfox
COPY . .
# install waterfox.desktop file
RUN mkdir /usr/share/applications && \
  cp waterfox.desktop /usr/share/applications
#
RUN cd /etc/apt && \
# apt debian setup
  cp sources.list sources.list.orig && \
  sed 's/main/main contrib non-free/' sources.list.orig >sources.list && \
  cd /wfox && \
  apt-get update && \
  apt-get upgrade -y && \
# apt install packages for waterfox
  apt-get install -y  dconf-gsettings-backend gvfs gvfs-libs libc6 \
   libcom-err2 libdbus-1-3 libelogind0 libexpat1 libgcc-s1 \
   libgdk-pixbuf-2.0-0 libgl1-mesa-dri libgpg-error0 libkeyutils1 \
   liblzma5 libnss-mdns libpcre3 libpulse0 libselinux1 libtirpc3 \
   python3-minimal zlib1g && \
# apt install packages for waterfox libraries
  apt-get install -y libatk1.0-0 libatk-bridge2.0-0 libatspi2.0-0 \
   libblkid1 libbrotli1 libbsd0 libc6 libcairo2 libcairo-gobject2 \
   libdatrie1 libdbus-1-3 libdbus-glib-1-2 libelogind0 libepoxy0 \
   libexpat1 libffi7 libfontconfig1 libfreetype6 libfribidi0 libgcc-s1 \
   libgcrypt20 libgdk-pixbuf-2.0-0 libglib2.0-0 libglib2.0-dev \
```

```
   libgpg-error0 libgraphite2-3 libgtk-3-0 libharfbuzz0b libice6 \
   liblz4-1 liblzma5 libmd0 libmount1 libnspr4 libnss3 libpango-1.0-0 \
   libpangocairo-1.0-0 libpangoft2-1.0-0 libpcre2-8-0 libpcre3 \
   libpixman-1-0 libpng16-16 libselinux1 libsm6 libstdc++6 libsystemd0 \
   libthai0 libuuid1 libwayland-client0 libwayland-cursor0 libwayland-egl1 \
   libx11-6 libx11-xcb1 libxau6 libxcb1 libxcb-render0 libxcb-shm0 \
   libxcomposite1 libxcursor1 libxdamage1 libxdmcp6 libxext6 libxext6-dbg \
   libxfixes3 libxi6 libxinerama1 libxkbcommon0 libxrandr2 libxrender1 \
   libxt6 libzstd1 zlib1g && \
# needed to unpack waterfox tarfile
 apt-get install -y bzip2 && \
#    unpack waterfox distro tarfile
  cd /wfox && \
  bzip2 -dc waterfox-G4.1.4.en-US.linux-x86_64.tar.bz2 | tar xvf -
# setup environment
RUN groupadd -g 1000 wfox
RUN useradd -d /home/wfox -s /bin/bash -m wfox -u 1000 -g 1000
USER wfox
ENV HOME /home/wfox
# exec waterfox
CMD /wfox/waterfox/waterfox
```

We build this Dockerfile with

```
docker build -t wfoxdeb:vt2 .
```

and it fails to build because the packages libelogind0 and libsystemd0 have conflicts and apt refuses to install both. We choose to delete libelogind0. Then the build works , and we are able to run its docker image with

```
xhost +
docker run -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY -h $HOSTNAME -v $HOME/.Xauth
```

This time we are successful. The container executes the Waterfox binary and we ger an X11 window with a blank welcome page shown in Figure 3, a correct release page shown in Figure 4, all the search functions and page displays seem to work as shown in Figure 5

There are messages in the container terminal as follows

```
[nevj@trinity Waterfox.docker]$ docker run -v /tmp/.X11-unix:/tmp/.X11-unix \
  -e DISPLAY=$DISPLAY -h $HOSTNAME -v $HOME/.Xauthority:/home/nevj/.Xauthority
  wfoxdeb:vt2
Authorization required, but no authorization protocol specified
Unable to init server: Could not connect: Connection refused
Error: cannot open display: :0.0
[nevj@trinity Waterfox.docker]$ xhost +
access control disabled, clients can connect from any host
[nevj@trinity Waterfox.docker]$
```

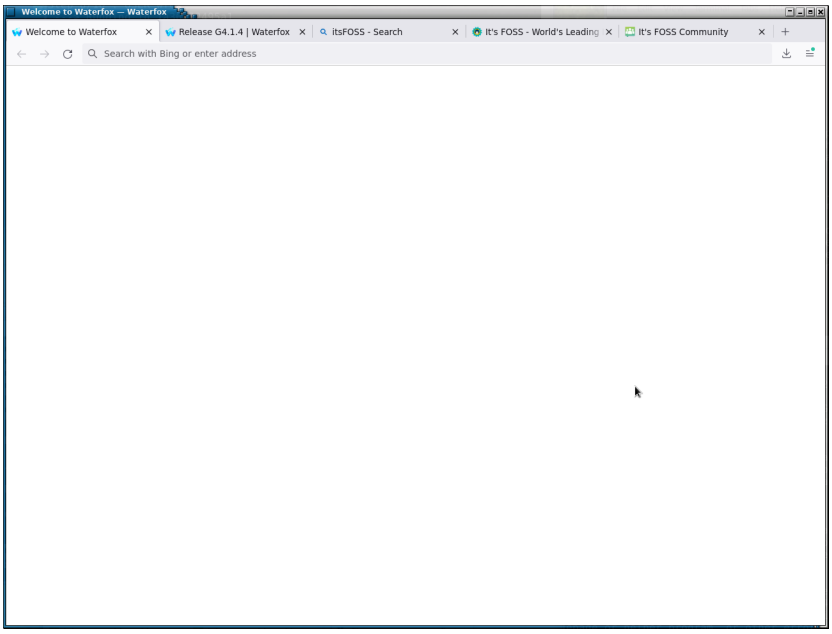

Figure 3: Waterfox Welcome screen from Dockerfile attempt No 2

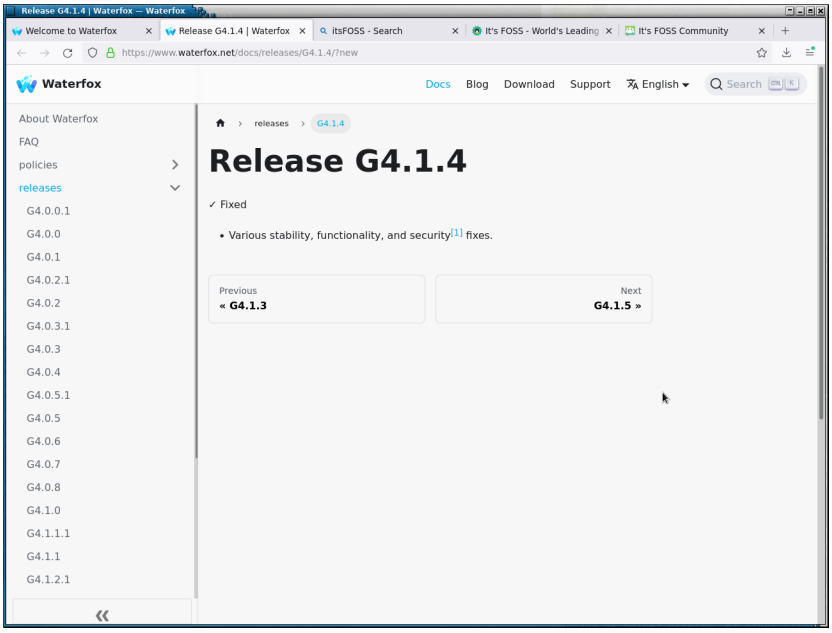

Figure 4: Waterfox Release screen from Dockerfile attempt No 2

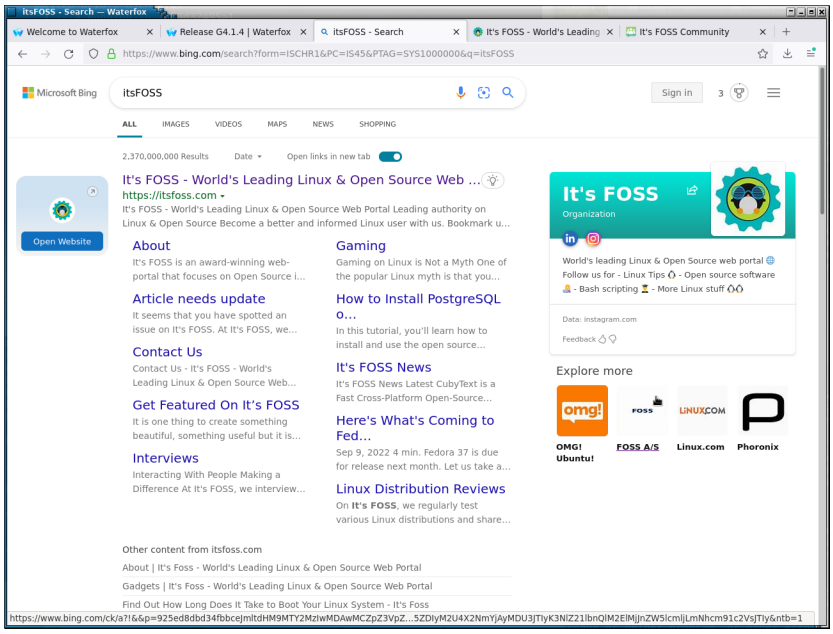

Figure 5: Waterfox Google Search screen from Dockerfile attempt No 2

```
[nevj@trinity Waterfox.docker]$ docker run -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISP
Crash Annotation GraphicsCriticalError: |[0][GFX1-]: glxtest: libpci missing (t=0.361222) [G
Crash Annotation GraphicsCriticalError: |[0][GFX1-]: glxtest: libpci missing (t=0.361222) ||
Crash Annotation GraphicsCriticalError: |[0][GFX1-]: glxtest: libpci missing (t=0.361222) ||
Crash Annotation GraphicsCriticalError: |[0][GFX1-]: glxtest: libpci missing (t=0.361222) ||
Crash Annotation GraphicsCriticalError: |[0][GFX1-]: glxtest: libpci missing (t=0.361222) ||
console.warn: SearchSettings: "get: No settings file exists, new profile?" (new NotFoundErro
JavaScript error: resource://gre/modules/XPCOMUtils.jsm, line 161: NS_ERROR_XPC_GS_RETURNED_
JavaScript error: , line 0: InternalError: Promise rejection value is a non-unwrappable cros
JavaScript error: resource://gre/modules/PromiseWorker.jsm, line 106: Error: Could not get c
JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile di
JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile di
JavaScript error: resource://gre/modules/PartitioningExceptionListService.jsm, line 69: NS_E
console.warn: LoginRecipes: "getRecipes: falling back to a synchronous message for:" "https:
JavaScript error: https://p.ad.gt/api/v1/p/450, line 1: TypeError: t.vendor is undefined
JavaScript warning: https://pagead2.googlesyndication.com/bg/5BBnFljR3G8Y2LtXULQJm9Fu_ODS9Xr
JavaScript warning: https://pagead2.googlesyndication.com/bg/5BBnFljR3G8Y2LtXULQJm9Fu_ODS9Xr
* WebglAllowWindowsNativeGl:false restricts context creation on this system. ()
* Exhausted GL driver options. (FEATURE_FAILURE_WEBGL_EXHAUSTED_DRIVERS)
JavaScript error: , line 0: TypeError: NetworkError when attempting to fetch resource.
console.error: PushService:
  clearOriginData: Error clearing origin data:
```

```
  TypeError
JavaScript error: resource:///modules/Interactions.jsm, line 209: NS_ERROR_FAILURE: Componer
JavaScript error: , line 0: TypeError: NetworkError when attempting to fetch resource.
[Parent 7, IPC I/O Parent] WARNING: FileDescriptorSet destroyed with unconsumed descriptors:

###!!! [Child][RunMessage] Error: Channel closing: too late to send/recv, messages will be l


###!!! [Child][RunMessage] Error: Channel closing: too late to send/recv, messages will be l
```

It would seem we have not found all dependencies yet. It says that *libpci* and *libEGL* are missing, and there are some Javascript resources required. Instead of just adding those 2 extra libraries/packages, we need to search out why our dependencies searches missed them.

4.6 Finding missing dependencies after attempt No. 2

A rerun of all the searches fails to reveal any hint of *libpci* and em libEGL being required. We conclude that some process other than waterfox which runs when the container runs, must be requiring them, perhaps the video drivers?

We can try and see what extra processes are running when the waterfox image is run in a docker container. All tha tcould be found using *pa ax* was

```
a4170 ?        Sl     0:00 /usr/bin/containerd-shim-runc-v2 -namespace moby -id 33ef059b4eda
 4408 ?        Sl     0:00 /wfox/waterfox/waterfox -contentproc -childID 1 -isForBrowser -pr
.......   5 more like the above
4488 ?        S      0:00 dbus-launch --autolaunch=796ea3dd4303ebc6c386a3176
4494 ?        Ss     0:00 /usr/bin/dbus-daemon --syslog-only --fork --print-
```

So running a container only adds the */usr/bin/containerd-shim-runc-v2* process, and *dbus-launch* and *dbus-daemon* apart from several processes associates with waterfox.

We can see what they require

```
ldd /usr/bin/containerd-shim-runc-v2
linux-vdso.so.1 (0x00007fffc3118000)
libdl.so.2 => /usr/lib/libdl.so.2 (0x00007fb56d758000)
libpthread.so.0 => /usr/lib/libpthread.so.0 (0x00007fb56d737000)
libc.so.6 => /usr/lib/libc.so.6 (0x00007fb56d571000)
/lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linux-x86-64.so.2 (0x00007fb56d773000)

ldd /usr/bin/dbus-daemon
linux-vdso.so.1 (0x00007ffefdb46000)
libdbus-1.so.3 => /usr/lib/libdbus-1.so.3 (0x00007f8b5d62e000)
libexpat.so.1 => /usr/lib/libexpat.so.1 (0x00007f8b5d5fd000)
libpthread.so.0 => /usr/lib/libpthread.so.0 (0x00007f8b5d5dc000)
libc.so.6 => /usr/lib/libc.so.6 (0x00007f8b5d416000)
```

```
/lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linux-x86-64.so.2 (0x00007f8b5d6d3000)

ldd /bin/dbus-launch
linux-vdso.so.1 (0x00007ffd8226e000)
libdbus-1.so.3 => /usr/lib/libdbus-1.so.3 (0x00007fc4e7b6a000)
libX11.so.6 => /usr/lib/libX11.so.6 (0x00007fc4e7a26000)
libc.so.6 => /usr/lib/libc.so.6 (0x00007fc4e7860000)
libpthread.so.0 => /usr/lib/libpthread.so.0 (0x00007fc4e783f000)
libxcb.so.1 => /usr/lib/libxcb.so.1 (0x00007fc4e7814000)
libdl.so.2 => /usr/lib/libdl.so.2 (0x00007fc4e780e000)
/lib64/ld-linux-x86-64.so.2 => /usr/lib64/ld-linux-x86-64.so.2 (0x00007fc4e7bdf000)
libXau.so.6 => /usr/lib/libXau.so.6 (0x00007fc4e7807000)
libXdmcp.so.6 => /usr/lib/libXdmcp.so.6 (0x00007fc4e77ff000)
```

but these would be requirements in the host system, not in the waterfox container. The command *docker logs ...* gives the same information as appears in the command line window. There seems to be no way to trace the source of these remaining dependencies, which I assume are inside the container.

I tried using *strace* on the *docker run* statement

```
strace docker run -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY -h $HOSTNAME -v $HOME
execve("/bin/docker", ["docker", "run", "-v", "/tmp/.X11-unix:/tmp/.X11-unix", "-e", "DISPLA
brk(NULL)                               = 0x308b000
.....
```

There is a lot of output, but the critical messages are the same as appears on the terminal screen.

The only remaining option is to add things by trial and error combined with a touch of intuition.

4.7 Third Debian parent dockerfile attempt

Add the packages needed to provide Libpci and libEGL and the Dockerfile becomes

```
M debian:stable-20220801
#set working dir inside container
WORKDIR /wfox
# get waterfox
COPY . .
# install waterfox.desktop file
RUN mkdir /usr/share/applications && \
  cp waterfox.desktop /usr/share/applications
#
RUN cd /etc/apt && \
# apt debian setup
  cp sources.list sources.list.orig && \
```

```
  sed 's/main/main contrib non-free/' sources.list.orig >sources.list && \
  cd /wfox && \
  apt-get update && \
  apt-get upgrade -y && \
# apt install packages for waterfox
  apt-get install -y  dconf-gsettings-backend gvfs gvfs-libs libc6 \
   libcom-err2 libdbus-1-3 libelogind0 libexpat1 libgcc-s1 \
   libgdk-pixbuf-2.0-0 libgl1-mesa-dri libgpg-error0 libkeyutils1 \
   liblzma5 libnss-mdns libpcre3 libpulse0 libselinux1 libtirpc3 \
   python3-minimal zlib1g && \
# apt install packages for waterfox libraries
  apt-get install -y libatk1.0-0 libatk-bridge2.0-0 libatspi2.0-0 \
   libblkid1 libbrotli1 libbsd0 libc6 libcairo2 libcairo-gobject2 \
   libdatrie1 libdbus-1-3 libdbus-glib-1-2 libepoxy0 libexpat1 libffi7 \
   libfontconfig1 libfreetype6 libfribidi0 libgcc-s1 libgcrypt20 \
   libgdk-pixbuf-2.0-0 libglib2.0-0 libglib2.0-dev libgpg-error0 \
   libgraphite2-3 libgtk-3-0 libharfbuzz0b libice6 liblz4-1 liblzma5 \
   libmd0 libmount1 libnspr4 libnss3 libpango-1.0-0 libpangocairo-1.0-0 \
   libpangoft2-1.0-0 libpcre2-8-0 libpcre3 libpixman-1-0 libpng16-16 \
   libselinux1 libsm6 libstdc++6 libsystemd0 libthai0 libuuid1 \
   libwayland-client0 libwayland-cursor0 libwayland-egl1 libx11-6 \
   libx11-xcb1 libxau6 libxcb1 libxcb-render0 libxcb-shm0 libxcomposite1 \
   libxcursor1 libxdamage1 libxdmcp6 libxext6 libxext6-dbg libxfixes3 \
   libxi6 libxinerama1 libxkbcommon0 libxrandr2 libxrender1 libxt6 \
   libzstd1 zlib1g && \
# needed to unpack waterfox tarfile
 apt-get install -y bzip2 && \
# Mystery dependencies
  apt-get install -y libpci3 libpciaccess0 && \
  apt-get install -y libgl1 libegl1 libgl-dev libegl-dev  && \
#    unpack waterfox distro tarfile
  cd /wfox && \
  bzip2 -dc waterfox-G4.1.4.en-US.linux-x86_64.tar.bz2 | tar xvf -
# setup environment
RUN groupadd -g 1000 wfox
RUN useradd -d /home/wfox -s /bin/bash -m wfox -u 1000 -g 1000
USER wfox
ENV HOME /home/wfox
# exec waterfox
CMD /wfox/waterfox/waterfox
```

Build this Dockerfile to make version 3 of the image, and run it and we obtain

```
docker run -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY \
   -h $HOSTNAME -v $HOME/.Xauthority:/home/nevj/.Xauthority \
   wfoxdeb:vt3
```

```
console.warn: SearchSettings: "get: No settings file exists, new profile?" (new NotFoundErro
JavaScript error: resource://gre/modules/XPCOMUtils.jsm, line 161: NS_ERROR_XPC_GS_RETURNED_
JavaScript error: , line 0: InternalError: Promise rejection value is a non-unwrappable cros
JavaScript error: resource://gre/modules/PromiseWorker.jsm, line 106: Error: Could not get c
JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile di
JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile di
JavaScript error: resource://gre/modules/PartitioningExceptionListService.jsm, line 69: NS_E
```

Same JavaScript messages as before but there are no missing libraries now. As far as I can tell Waterfox runs exactly as before. The Welcome Screen is still blank, the rest is OK.

4.8 Fourth attempt at Dockerfile

We need to try and satisfy the missing Java resources. I know nothing about JacaScript, so I can only proceed by guessing the required packages. An *apt-cache search javascript* yields a very long list of packages. It is impossible to know what is needed. I resort to searching the internet and find that the minumal requirement for JavaScript in Debian is two packages *nodejs* which provides the JavaScript runtime environment and *npm* which is a package registry providing access to shared JavaScript software.

So we add these 2 packages to the Dockerfile.

```
FROM debian:stable-20220801
#set working dir inside container
WORKDIR /wfox
# get waterfox
COPY . .
# install waterfox.desktop file
RUN mkdir /usr/share/applications && \
  cp waterfox.desktop /usr/share/applications
#
RUN cd /etc/apt && \
# apt debian setup
  cp sources.list sources.list.orig && \
  sed 's/main/main contrib non-free/' sources.list.orig >sources.list && \
  cd /wfox && \
  apt-get update && \
  apt-get upgrade -y && \
# apt install packages for waterfox
  apt-get install -y  dconf-gsettings-backend gvfs gvfs-libs libc6 \
   libcom-err2 libdbus-1-3 libelogind0 libexpat1 libgcc-s1 \
   libgdk-pixbuf-2.0-0 libgl1-mesa-dri libgpg-error0 libkeyutils1 \
   liblzma5 libnss-mdns libpcre3 libpulse0 libselinux1 libtirpc3 \
   python3-minimal zlib1g && \
# apt install packages for waterfox libraries
  apt-get install -y libatk1.0-0 libatk-bridge2.0-0 libatspi2.0-0 \
   libblkid1 libbrotli1 libbsd0 libc6 libcairo2 libcairo-gobject2 \
```

```
   libdatrie1 libdbus-1-3 libdbus-glib-1-2 libepoxy0 libexpat1 libffi7 \
   libfontconfig1 libfreetype6 libfribidi0 libgcc-s1 libgcrypt20 \
   libgdk-pixbuf-2.0-0 libglib2.0-0 libglib2.0-dev libgpg-error0 \
   libgraphite2-3 libgtk-3-0 libharfbuzz0b libice6 liblz4-1 liblzma5 \
   libmd0 libmount1 libnspr4 libnss3 libpango-1.0-0 libpangocairo-1.0-0 \
   libpangoft2-1.0-0 libpcre2-8-0 libpcre3 libpixman-1-0 libpng16-16 \
   libselinux1 libsm6 libstdc++6 libsystemd0 libthai0 libuuid1 \
   libwayland-client0 libwayland-cursor0 libwayland-egl1 libx11-6 \
   libx11-xcb1 libxau6 libxcb1 libxcb-render0 libxcb-shm0 libxcomposite1 \
   libxcursor1 libxdamage1 libxdmcp6 libxext6 libxext6-dbg libxfixes3 \
   libxi6 libxinerama1 libxkbcommon0 libxrandr2 libxrender1 libxt6 \
   libzstd1 zlib1g && \
# needed to unpack waterfox tarfile
 apt-get install -y bzip2 && \
# Mystery dependencies
  apt-get install -y libpci3 libpciaccess0 && \
  apt-get install -y libgl1 libegl1 libgl-dev libegl-dev  && \
# Javascript packages
  apt-get install -y nodejs npm  && \
#    unpack waterfox distro tarfile
  cd /wfox && \
  bzip2 -dc waterfox-G4.1.4.en-US.linux-x86_64.tar.bz2 | tar xvf -
# setup environment
RUN groupadd -g 1000 wfox
RUN useradd -d /home/wfox -s /bin/bash -m wfox -u 1000 -g 1000
USER wfox
ENV HOME /home/wfox
# exec waterfox
CMD /wfox/waterfox/waterfox
```

then build the image and run the container

```
docker build -t wfoxdeb:vt4 .
.....
docker run -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY \
   -h $HOSTNAME -v $HOME/.Xauthority:/home/nevj/.Xauthority \
   wfoxdeb:vt4
Console.warn: SearchSettings: "get: No settings file exists, new profile?" (new NotFoundErro
JavaScript error: resource://gre/modules/XPCOMUtils.jsm, line 161: NS_ERROR_XPC_GS_RETURNED_
JavaScript error: , line 0: InternalError: Promise rejection value is a non-unwrappable cros
JavaScript error: resource://gre/modules/PromiseWorker.jsm, line 106: Error: Could not get c
JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile di
JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile di
JavaScript error: resource://gre/modules/PartitioningExceptionListService.jsm, line 69: NS_E
```

Then when I do a search there are more messages

```
console.warn: LoginRecipes: "getRecipes: falling back to a synchronous message for:" "https:
```

```
JavaScript error: https://p.ad.gt/api/v1/p/450, line 1: TypeError: t.vendor is undefined
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to load driver: nouveau
```

Waterfox still has a blank Welcome page, but it seems to work properly. The most concerning is the *libGL error*. It only happened after I did a search. I added *libgl* in the third Dockerfile attempt because I was unsure whether it was needed as well as *libEGL*. I did not notice this message, probably because I did not do a search. The messages from second attempt at a Dockerfile only said *libEGL* and *libpci* were missing. So I think I should try removing *libGL*

4.8.1 Removing libGL

```
FROM debian:stable-20220801
#set working dir inside container
WORKDIR /wfox
# get waterfox
COPY . .
# install waterfox.desktop file
RUN mkdir /usr/share/applications && \
  cp waterfox.desktop /usr/share/applications
#
RUN cd /etc/apt && \
# apt debian setup
  cp sources.list sources.list.orig && \
  sed 's/main/main contrib non-free/' sources.list.orig >sources.list && \
  cd /wfox && \
  apt-get update && \
  apt-get upgrade -y && \
# apt install packages for waterfox
  apt-get install -y  dconf-gsettings-backend gvfs gvfs-libs libc6 \
   libcom-err2 libdbus-1-3 libelogind0 libexpat1 libgcc-s1 \
   libgdk-pixbuf-2.0-0 libgl1-mesa-dri libgpg-error0 libkeyutils1 \
   liblzma5 libnss-mdns libpcre3 libpulse0 libselinux1 libtirpc3 \
   python3-minimal zlib1g && \
# apt install packages for waterfox libraries
  apt-get install -y libatk1.0-0 libatk-bridge2.0-0 libatspi2.0-0 \
   libblkid1 libbrotli1 libbsd0 libc6 libcairo2 libcairo-gobject2 \
   libdatrie1 libdbus-1-3 libdbus-glib-1-2 libepoxy0 libexpat1 \
   libffi7 libfontconfig1 libfreetype6 libfribidi0 libgcc-s1 \
   libgcrypt20 libgdk-pixbuf-2.0-0 libglib2.0-0 libglib2.0-dev \
   libgpg-error0 libgraphite2-3 libgtk-3-0 libharfbuzz0b libice6 \
   liblz4-1 liblzma5 libmd0 libmount1 libnspr4 libnss3 libpango-1.0-0 \
   libpangocairo-1.0-0 libpangoft2-1.0-0 libpcre2-8-0 libpcre3 \
   libpixman-1-0 libpng16-16 libselinux1 libsm6 libstdc++6 libsystemd0 \
```

```
   libthai0 libuuid1 libwayland-client0 libwayland-cursor0 \
   libwayland-egl1 libx11-6 libx11-xcb1 libxau6 libxcb1 libxcb-render0 \
   libxcb-shm0 libxcomposite1 libxcursor1 libxdamage1 libxdmcp6 \
   libxext6 libxext6-dbg libxfixes3 libxi6 libxinerama1 libxkbcommon0 \
   libxrandr2 libxrender1 libxt6 libzstd1 zlib1g && \
# needed to unpack waterfox tarfile
 apt-get install -y bzip2 && \
# Mystery dependencies
  apt-get install -y libpci3 libpciaccess0 && \
  apt-get install -y libegl1 libegl-dev  && \
# Javascript packages
  apt-get install -y nodejs npm  && \
#    unpack waterfox distro tarfile
  cd /wfox && \
  bzip2 -dc waterfox-G4.1.4.en-US.linux-x86_64.tar.bz2 | tar xvf -
# setup environment
RUN groupadd -g 1000 wfox
RUN useradd -d /home/wfox -s /bin/bash -m wfox -u 1000 -g 1000
USER wfox
ENV HOME /home/wfox
# exec waterfox
CMD /wfox/waterfox/waterfox

docker build -t wfoxdeb:vt4 .
....
docker run -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY \
   -h $HOSTNAME -v $HOME/.Xauthority:/home/nevj/.Xauthority \
   wfoxdeb:vt4
Console.warn: SearchSettings: "get: No settings file exists, new profile?" (new NotFoundErro
JavaScript error: resource://gre/modules/XPCOMUtils.jsm, line 161: NS_ERROR_XPC_GS_RETURNED_
JavaScript error: , line 0: InternalError: Promise rejection value is a non-unwrappable cros
JavaScript error: resource://gre/modules/PromiseWorker.jsm, line 106: Error: Could not get c
JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile di
JavaScript error: resource://gre/modules/XULStore.jsm, line 66: Error: Can't find profile di
JavaScript error: resource://gre/modules/PartitioningExceptionListService.jsm, line 69: NS_E
console.warn: LoginRecipes: "getRecipes: falling back to a synchronous message for:" "https:
JavaScript error: https://p.ad.gt/api/v1/p/450, line 1: TypeError: t.vendor is undefined
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to load driver: nouveau
```

I did a search, and now when I go to a website the libGL message is as before. So not adding *libGL* did not change anything... it is apparently unnecessary, but is not the source of the error message.

4.8.2 Error messages about .jsm files

We need to have a look at the messages about *.jsm* files

```
XPCOMUtils.jsm
PromiseWorker.jsm
XULStore.jsm
PartitioningExceptionListService.jsm
```

A web search reveals that *.jsm* files store Firefox Javascript modules. So we need to see if there are any missing packages related to Mozilla Firefox. There are no missing packages. Adding 'apt-get install firefox-esr' to the Dockerfile does not change the number of Javascript errer messages. If I force the execution of firefox in the container

```
[nevj@trinity Waterfox.docker]$ docker run -v /tmp/.X11-unix:/tmp/.X11-unix \
  -e DISPLAY=$DISPLAY -h $HOSTNAME \
  -v $HOME/.Xauthority:/home/nevj/.Xauthority wfoxdeb:ffox firefox
.....
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to load driver: nouveau
```

The Javascript errors disappear when running firefox in this container. So firefox is using the same JavasScript modules without error. That proves the JavaScript errors are a Waterfox problem , and are not solvable by adding any packages. I would like to acknowledge assistance from Laszlo Kovacs and Akito Kitsune in diagnosing the Javascript error messages with *.jsm* modules.

The libGL error however remains.

4.8.3 LibGL errors

We have these messages

```
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to load driver: nouveau
\end{verbatiom}
which occur after one does a search and goes to a website. The seem to indicate that some de
\begin{verbatim}
-t /dev/dri/card0:/dev/dri/card0
```

What it should do is make the device */dev/dri/card0* in the host available in the container under the same name. It failed but it changes the messages to

```
libGL error: failed to open /dev/dri/card0: Operation not permitted
libGL error: failed to load driver: nouveau
```

So failed again , but the message is different. I think the name of the device inside the container should probably be different, but I dont know what it should be.

I discovered another way of making a device available to a container

```
docker run ...... --device=/dev/dri/card0 ......
```

On adding this to the docker run statement I get

```
docker run -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY -h $HOSTNAME -v $HOME/.Xauth
.......
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to load driver: nouveau
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to open /dev/dri/card0: Permission denied
libGL error: failed to load driver: nouveau
console.error: PushService:
  clearOriginData: Error clearing origin data:
```

THere are now twice as many error messages.

We shall have to leave the libGL messages for now. It does not seem to disable operation of waterfox in the container

4.9 Fifth attempt at Dockerfile

to the *docker run statement*. What needs to be done now is to make the Dockerfile site independent by downloading the Waterfox tarfile from the Official Waterfox site, rather than copying a file which has been pre-downloaded into my working directory.

One way to download a file from the web is with *wget*

```
wget https://github.com/WaterfoxCo/Waterfox/releases/download/\
G4.1.5/waterfox-G4.1.5.en-US.linux-x86_64.tar.bz2
```

```
-2022-09-25 21:05:20--  https://github.com/WaterfoxCo/Waterfox/releases/download/G4.1.5/wate
Resolving github.com (github.com)... 20.248.137.48
Connecting to github.com (github.com)|20.248.137.48|:443... connected.
.......
waterfox-G4.1.5.en- 100%[===================>]  79.05M  4.20MB/s    in 24s


2022-09-25 21:05:46 (3.31 MB/s) - ‘waterfox-G4.1.5.en-US.linux-x86_64.tar.bz2’ saved [828929
```

So it works, the download is 79Mb and on my system took 24sec. It names a specific release of Waterfox. Because Waterfox keeps all old releases on its Github site, doing wget for a specific release will always work. Also we want the Dockerfile to use the release for which the environment (particularly dependencies) was built. Using a later release may introduce further dependencies. So this retrieval with wget would suit our Dockerfile.

Another option is *curl*.

```
curl  https://github.com/WaterfoxCo/Waterfox/releases/download\
/G4.1.5/waterfox-G4.1.5.en-US.linux-x86_64.tar.bz2 > wfox.tar.bz2
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
```

This attempt does not work. It makes an empty file? So we may as well use what works. Our fiftg Dockerfile attempt becomes

```
FROM debian:stable-20220801
#set working dir inside container
WORKDIR /wfox
# copy commented out
# COPY . .
# install waterfox.desktop file
# RUN mkdir /usr/share/applications && \
# cp waterfox.desktop /usr/share/applications
#
RUN cd /etc/apt && \
# apt debian setup
  cp sources.list sources.list.orig && \
  sed 's/main/main contrib non-free/' sources.list.orig >sources.list && \
  cd /wfox && \
  apt-get update && \
  apt-get upgrade -y && \
  apt-get install -y apt-utils && \
# apt install packages for waterfox
  apt-get install -y  dconf-gsettings-backend gvfs gvfs-libs libc6 \
   libcom-err2 libdbus-1-3 libelogind0 libexpat1 libgcc-s1 \
   libgdk-pixbuf-2.0-0 libgl1-mesa-dri libgpg-error0 libkeyutils1 \
   liblzma5 libnss-mdns libpcre3 libpulse0 libselinux1 libtirpc3 \
   python3-minimal zlib1g && \
# apt install packages for waterfox libraries
  apt-get install -y libatk1.0-0 libatk-bridge2.0-0 libatspi2.0-0 libblkid1 libbrotli1 libbs
# needed to unpack waterfox tarfile
 apt-get install -y bzip2 && \
# Mystery dependencies
  apt-get install -y libpci3 libpciaccess0 && \
  apt-get install -y libegl1 libegl-dev  && \
# apt-get install -y libgl1 libglfw3  && \
# video
# apt-get install -y xserver-xorg-video-nouveau && \
# Javascript packages
  apt-get install -y nodejs npm  && \
  apt-get install -y  gir1.2-javascriptcoregtk-4.0 javascriptcoregtk-4.0\
   javascript-common libnode72 && \
# firefox
# apt-get install -y firefox-esr && \
#  get waterfox distro tarfile and unpack
```

```
  apt-get install -y wget && \
  cd /wfox
RUN wget https://github.com/WaterfoxCo/Waterfox/releases/download/\
G4.1.5/waterfox-G4.1.5.en-US.linux-x86_64.tar.bz2 && \
  bzip2 -dc waterfox-G4.1.5.en-US.linux-x86_64.tar.bz2 | tar xvf -
# setup environment
RUN groupadd -g 1000 wfox
RUN useradd -d /home/wfox -s /bin/bash -m wfox -u 1000 -g 1000
USER wfox
ENV HOME /home/wfox
# exec waterfox
CMD /wfox/waterfox/waterfox
```

Note we have changed to version G4.1.5 of waterfox. It is only a minor update. Will check if dependencies are affected.

After building the above fifth Dockerfile attempt, and running it, Waterfox still works and the messages while running are exactly the same as for the fourth attempt. There are no additional missing libraries , so the version change to G4.1.5 is OK. The libGL error messages are still present.

We can conclude that the changeover from a local file to a download was successful.

4.10 Sixth attempt at a Dockefile - using slim Debian parent image

a The full Debian parent image is 124Mb. We can save space by using the Debian-slim image which is 80Mb. We can also make the image smaller by removing the downloaded Waterfox tarfile, after we have unpacked it. So our sixth Dockerfile attempt is as follows

```
FROM debian:stable-20220801-slim
#set working dir inside container
WORKDIR /wfox
#
RUN cd /etc/apt && \
# apt debian setup
  cp sources.list sources.list.orig && \
  sed 's/main/main contrib non-free/' sources.list.orig >sources.list && \
  cd /wfox && \
  apt-get update && \
  apt-get upgrade -y && \
  apt-get install -y apt-utils && \
# apt install packages for waterfox
  apt-get install -y  dconf-gsettings-backend gvfs gvfs-libs libc6 \
   libcom-err2 libdbus-1-3 libelogind0 libexpat1 libgcc-s1 \
   libgdk-pixbuf-2.0-0 libgl1-mesa-dri libgpg-error0 libkeyutils1 \
   liblzma5 libnss-mdns libpcre3 libpulse0 libselinux1 libtirpc3 \
   python3-minimal zlib1g && \
# apt install packages for waterfox libraries
```

```
  apt-get install -y libatk1.0-0 libatk-bridge2.0-0 libatspi2.0-0 libblkid1 libbrotli1 libbs
# needed to unpack waterfox tarfile
 apt-get install -y bzip2 && \
# Mystery dependencies
  apt-get install -y libpci3 libpciaccess0 && \
  apt-get install -y libegl1 libegl-dev  && \
# apt-get install -y libgl1 libglfw3  && \
# Javascript packages
  apt-get install -y nodejs npm  && \
  apt-get install -y  gir1.2-javascriptcoregtk-4.0 javascriptcoregtk-4.0\
   javascript-common libnode72 && \
# firefox
# apt-get install -y firefox-esr && \
#  get waterfox distro tarfile and unpack
  apt-get install -y wget && \
  cd /wfox
RUN wget https://github.com/WaterfoxCo/Waterfox/releases/download/\
G4.1.5/waterfox-G4.1.5.en-US.linux-x86_64.tar.bz2 && \
  bzip2 -dc waterfox-G4.1.5.en-US.linux-x86_64.tar.bz2 | tar xvf - && \
  rm waterfox-G4.1.5.en-US.linux-x86_64.tar.bz2
# setup environment
RUN groupadd -g 1000 wfox
RUN useradd -d /home/wfox -s /bin/bash -m wfox -u 1000 -g 1000
USER wfox
ENV HOME /home/wfox
# exec waterfox
CMD /wfox/waterfox/waterfox
```

So build and run this sixth Dockerfile attempt. The build works, the run gives exactly the same messages as with the fourth and fifth Dockerfiles, and Waterfox still works, the only issue being the blank welcome page.

So the full Debian parent image was unnecessary. The fifth image was 1.43Gb, the sixth is 1.16Gb.

4.11 Seventh Dockerfile attempt - setting locale

We have defined a user and a group and a home directory. We need to set the time zone and character set. There is an article [13] on setting locale in a Ubuntu container. It recommends the following Dockerfile statements

```
# Set the locale
RUN locale-gen en_US.UTF-8
ENV LANG en_US.UTF-8
ENV LANGUAGE en_US:en
ENV LC_ALL en_US.UTF-8
```

Another reference [14] recommends installing *locales* and *locales-all* in the container

```
RUN apt-get install -y locales locales-all
ENV LC_ALL en_US.UTF-8
ENV LANG en_US.UTF-8
ENV LANGUAGE en_US.UTF-8
```

That does not set the timezone. That is done with

```
ENV TZ Australia/Sydney
```

The above will result in the environment values being hardcoded ito the Dockerfile. So if a user wishes to change one or more values, they have to edit the Dockerfile.

It is possible to set the environment values dynamically when the container is run. To do this the ENV statements inside Dockerfile should be

```
RUN apt-get install -y locales locales-all
ENV LC_ALL=$[LC_ALL]
ENV LANG=$[lang]
ENV LANGUAGE=$[language]
ENV TZ=$[TZ]
```

Then the *docker run* statement can pass these values using the *-e* option as follows

```
docker run -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY \
   -e LC_ALL=en_AU.UTF-8 LANG=en_AU.UTF-8 LANGUAGE=en_AU.UTF-8 \
   TZ=Australia/Sydney -h $HOSTNAME \
   -v $HOME/.Xauthority:/home/nevj/.Xauthority imagename
```

We shall use the simpler method, so our seventh Dockerfile attempt becomes

```
FROM debian:stable-20220801-slim
#set working dir inside container
WORKDIR /wfox
#
RUN cd /etc/apt && \
# apt debian setup
  cp sources.list sources.list.orig && \
  sed 's/main/main contrib non-free/' sources.list.orig >sources.list && \
  cd /wfox && \
  apt-get update && \
  apt-get upgrade -y && \
  apt-get install -y apt-utils && \
# apt install packages for waterfox
  apt-get install -y  dconf-gsettings-backend gvfs gvfs-libs libc6 \
   libcom-err2 libdbus-1-3 libelogind0 libexpat1 libgcc-s1 \
   libgdk-pixbuf-2.0-0 libgl1-mesa-dri libgpg-error0 libkeyutils1 \
   liblzma5 libnss-mdns libpcre3 libpulse0 libselinux1 libtirpc3 \
   python3-minimal zlib1g && \
# apt install packages for waterfox libraries
  apt-get install -y libatk1.0-0 libatk-bridge2.0-0 libatspi2.0-0 libblkid1 libbrotli1 libbs
```

```
# needed to unpack waterfox tarfile
 apt-get install -y bzip2 && \
# Mystery dependencies
  apt-get install -y libpci3 libpciaccess0 && \
  apt-get install -y libegl1 libegl-dev  && \
# apt-get install -y libgl1 libglfw3  && \
# Javascript packages
  apt-get install -y nodejs npm  && \
  apt-get install -y  gir1.2-javascriptcoregtk-4.0 javascriptcoregtk-4.0\
   javascript-common libnode72 && \
# firefox
# apt-get install -y firefox-esr && \
# locale
  apt-get install -y locales locales-all && \
#  get waterfox distro tarfile and unpack
  apt-get install -y wget && \
  cd /wfox
RUN wget https://github.com/WaterfoxCo/Waterfox/releases/download/\
G4.1.5/waterfox-G4.1.5.en-US.linux-x86_64.tar.bz2 && \
  bzip2 -dc waterfox-G4.1.5.en-US.linux-x86_64.tar.bz2 | tar xvf - && \
  rm waterfox-G4.1.5.en-US.linux-x86_64.tar.bz2
# setup environment
RUN groupadd -g 1000 wfox
RUN useradd -d /home/wfox -s /bin/bash -m wfox -u 1000 -g 1000
RUN rm -fr /tmp/* /var/tmp/* /var/cache/*/*
USER wfox
ENV HOME /home/wfox
ENV LC_ALL en_AU.UTF-8
ENV LANG en_AU.UTF-8
ENV LANGUAGE en_AU.UTF-8
ENV TZ=Australia/Sydney
# exec waterfox
CMD /wfox/waterfox/waterfox
```

A build and run of this Dockerfile works and gives all the same messages afds the sixth attempt. THe image size is now 1.4Gb, compared to 1.16Gb for sixth attempt, the increase being the added locale packages.

5 Attempt at a Waterfox Dockerfile using a Void Linux parent image

This work has been removed to a separate document. Void offers the possibility of a much smaller parent image than Debian. However all the work of tracing the dependencies has to be done again, because packages in Void's *xbps* package system have different names to Debian's packages.

Void may offer better support for Waterfox, because its packages are more up to date. In particular the JavaScript errors may be resolved.

6 Testing

The whole idea of using docker is to be able to build and run images in any Linux system. My Dockerfile should work in any Linux with a reasonably recent kernel. The Dockef ile was developed in a Void host system, so I know it works in Void.

I tested the Dockerfile in Debian, Devuan, and Solus. There were no issues.

In Debian and Devuan the *libGL* error messages were slightly different

```
libGL error: MESA-LOADER: failed to retrieve device information
MESA-LOADER: failed to retrieve device information
MESA-LOADER: failed to retrieve device information
```

I think it means the same. Things in Debian have different names.

In Solus I chose to install docker from the GUI Software Centre. The install required numerous dependencies be installed first, including firefox and thunderbird, which were already installed? I cant imagine why docker would have firefox or thunderbird as dependencies? The libGL error messages in Solus were the same as in Void Linux.

So my Dockerfile passes the test of operating in various Linux versions. They were all $x86_64 architecture. I am not sure about different architectures$?

7 Discussion

I have attempted to lay out a systematic way of determining what packages should be in the docker image environment, using Waterfox as an example. For a large app like Waterfox, it is important to trace dependencies systematically.

It turns out that dependencies can be reliably traced, but it is a large job for a big package like Waterfox.

The issue of running in a container a program which has GUI output with X11 is quite simple. One just has to provide the image at run time with access to a couple of jost system X11 files. This is done by defining *volumes* in the *run* statement.

There are some unresolved issues, particularly those labelled *libGL error*. They do not seem to stop Waterfox working.

If one were going to tackle a project like this, it would be almost essential to have an install of Waterfox in a Debian system (assuming one were using a Debian parent image). One needs to have a working Waterfox to trace dependencies, and it needs to be running in the same Linux as used for the parent image.

One of the reasons for undertaking this project was to see if it woulld be feasable to use docker to share software setups among users. It is possible, but too complicated for any sort of automatic use.

References

[1] Docker tutorial. URL https://www.guru99.com/docker-tutorial.html

[2] Docker get started URL https://docs.docker.com/get-started/

[3] Docker Desktop URL https://docs.docker.com/desktop/install/linux-install/

[4] Docker Hub URL https://hub.docker.com/

[5] Official Dockerfile documnet URL https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

[6] Dockerfile Guide URL https://medium.com/@BeNitinAgarwal/best-practices-for-working-with-dockerfil es-fb2d22b78186

[7] Docker Basics: How to use Dockerfiles URL https://thenewstack.io/docker-basics-how-to-use-dockerfiles/

[8] A Beginners Guide to Understanding and Building Docker Images URL https://jfrog.com/knowledge-base/a-beginners-guide-to-understanding-and-building-docker-images/

[9] Best practices for writing Dockerfiles URL https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

[10] A Beginners Guide to Understanding and Building Docker Images https://jfrog.com/knowledge-base/a-beginners-guide-to-understanding-and-building-docker-images/

[11] Creating a Docke Image for your Application URL https://www.stereolabs.com/docs/docker/creating-your-image/

[12] Docker Containerization Cookbook" - Hot Recipes for Docker Automation URL https://distrowatch.tradepub.com/free/w_java39/prgm.cgi?a=1

[13] Markell J.(2014) Docker and Locales URL http://jaredmarkell.com/docker-and-locales/

[14] URL https://stackoverflow.com/questions/28405902/how-to-set-the-locale-inside-a-debian-ubuntu-docker-container

[15] Void Linux Docker Images URL https://github.com/void-linux/void-docker

[16] LibreWolf source code website. URL https://gitlab.com/librewolf-community/browser/source

[17] Rehn, A. (2021) Identifying application runtime dependencies: A toolkit for identifying the runtime libraries and associated data that applications require in order to run correctly inside containers. URL https://unrealcontainers.com/blog/identifying-application-runtime-dependencies/

[18] Waterfox website. URL https://www.waterfox.net

[19] Install the Waterfox browser on a Linux system. URL https://github.com/nevillejackson/Unix/blob/main/waterfox/waterfox.pdf

[20] WaterfocCo Github site URL https://github.com/WaterfoxCo/Waterfox/releases