Creating Singularity containers for HPC users

Martin Čuma
Center for High Performance Computing
University of Utah
m.cuma@utah.edu

Overview

- Why do we want to use containers?
- Containers basics
- Prepare your computer for containers
  - As a backup ssh to our lab
- Build and deploy a container
- Containers for complex software
- Containers for Windows programs

Hands on setup

1. Download the talk slides
   http://home.chpc.utah.edu/~mcuma/chpc/Containers_RMACC17.pdf
2. Get an user/password paper slip
3. Using terminal application (Mac terminal, PuTTY, GIT Shell)
   - `ssh userxx@linuxclass.chpc.utah.edu`
4. Make sure you can see singularity
   - `ls /usr/local/bin/singularity`
5. Make sure you can sudo singularity command
   - `sudo /usr/local/bin/singularity –version`

OR – if you have installed Singularity on your laptop, use it

Why to use containers?

Software dependencies

- Some programs require complex software environments
  - OS type and versions
  - Drivers
  - Compiler type and versions
  - Software dependencies
    - Python/R/MATLAB versions
    - glibc, stdlibc++ versions
    - Other libraries and executables
    - Python/R libraries
- We may not want to build everything from a scratch

Reproducible research

- Research outputs include software and data
- Software reproducibility
  - Software repositories (svn, git)
  - Good but often software has dependencies
- Data reproducibility
  - Data as publication supplementary info, centralized repositories (NCBI), …
  - Disconnected from the production environment
- Package data AND code AND compute environment in one file

Scalable research

- Develop a program / pipeline locally, run globally
- Scale to parallel resources
  - Run many times
  - Use local or national HPC resources
- Automate the process
  - Container/software building and deployment
  - Parallel pipeline

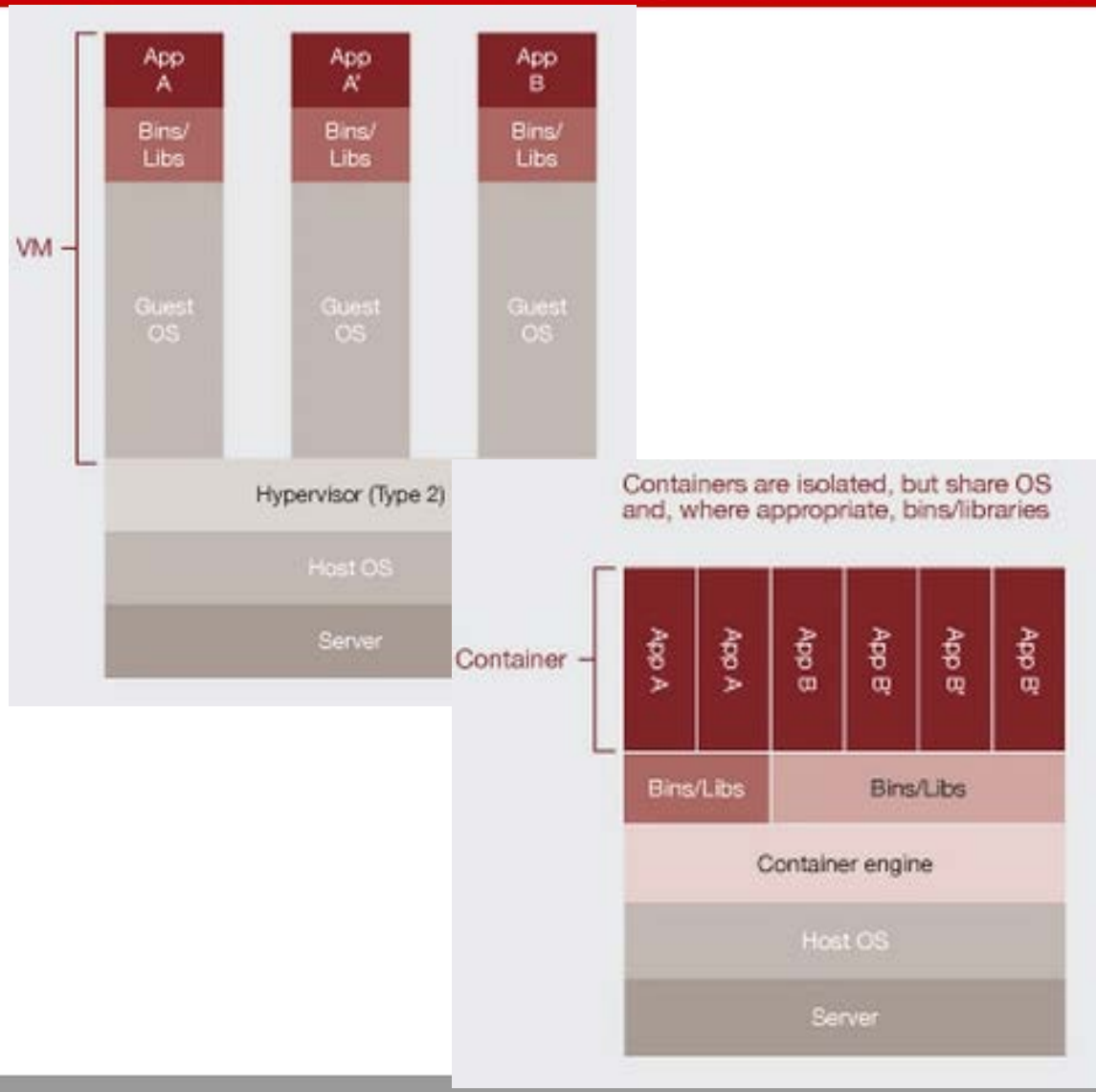
Additional bonus

- Old applications on old Linux versions

Container basics

Virtualization basics

- Hardware virtualization
  - Running multiple OSes on the same hardware
  - VMWare, VirtualBox
- OS level virtualization
  - run multiple isolated OS instances (guests) under a server OS (host)
  - Also called containers
  - Docker, Singularity



Containers

- Isolate computing environments
  - And allow for regenerating computing environments
- Guest OS running over host OS
  - Guest's OS can be different that host's
  - Low level operations (kernel, network, I/O) ran through the host
- From user standpoint guest OS behaves like standard OS

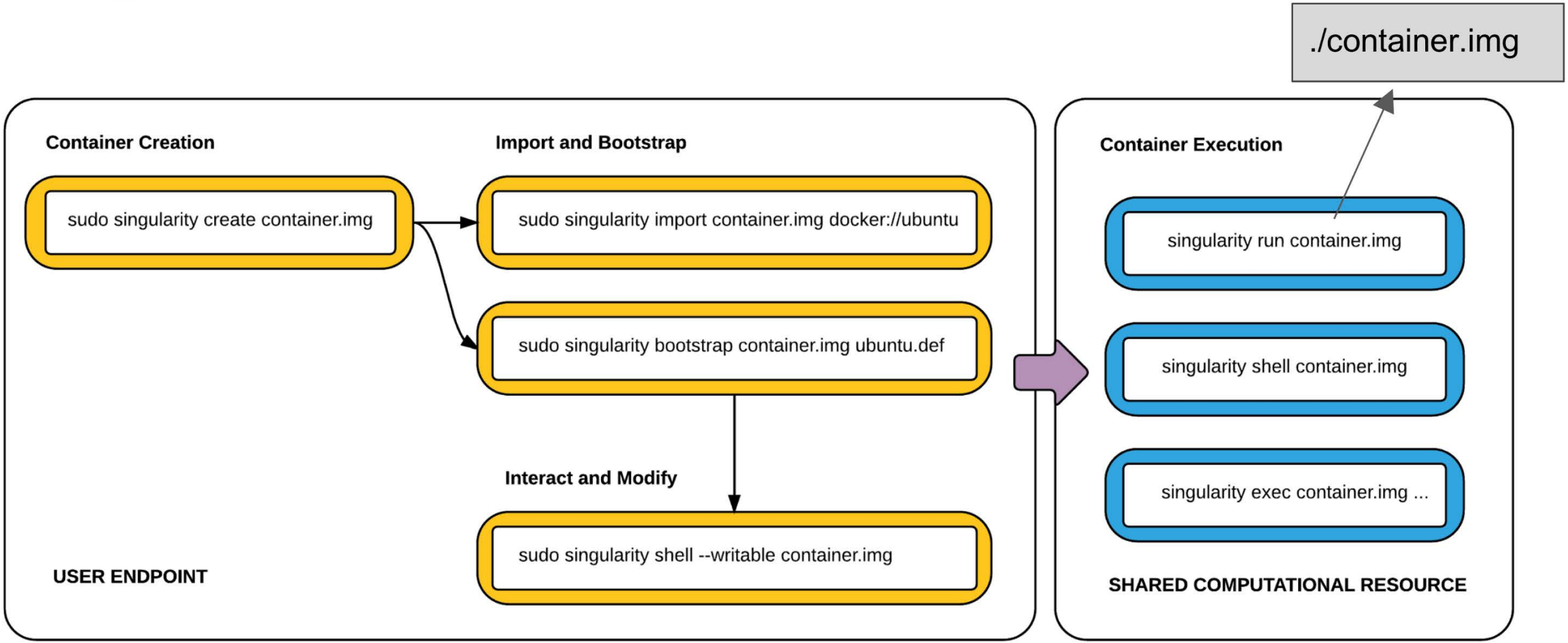
Container solutions

- Docker
  - Well established
  - Has docker hub for container sharing
  - Problematic with HPC
- Singularity
  - Designed for HPC
  - Support for SLURM, MPI, GPUs
  - Relatively new but rapidly expanding
- Shifter, Charliecloud
  - HPC oriented but less used presently

Singularity containers

- Integrate with traditional HPC
  - Same user inside and outside of the container
  - Same file systems (home, scratch), environment
  - Can integrate with existing software (CHPC sys branch)
- Portable and sharable
  - A container is a file
  - It can be built on one OS and run on another
- Only Linux support right now
  - But any Linux version – RHEL/CentOS, Ubuntu, Debian

Singularity workflow



Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific Containers for Mobility of Compute (under review)

Prepare your computer for Singularity containers

Linux

- We need to run Linux to build/run Singularity
  - If you already run Linux, make sure you have a root and then take a nap
  - On Windows and Mac, we need to install Linux first
- Install Linux in a VM
  - Windows – GIT Bash, Virtual Box and Vagrant
    - https://www.chpc.utah.edu/documentation/software/containers-localbuild.php#prep
  - Mac – Homebrew with Virtual Box and Vagrant
    - http://singularity.lbl.gov/install-mac

Install prerequisites

- Windows – GIT Bash, VirtualBox, Vagrant
  - GIT Bash provides a bash terminal on Windows
  - VirtualBox provides VM virtualization
  - Vagrant automates VM setup
  - Use Vagrant to install Ubuntu VM
- Mac – VirtualBox and Vagrant
  - Already have a terminal
  - Use Homebrew to install VirtualBox and Vagrant

https://www.chpc.utah.edu/documentation/software/containers-localbuild.php#prep

Install Singularity

- Linux laptop users – wake up
- In Ubuntu VM, or standalone Linux

```
$ git clone https://github.com/singularityware/singularity.git
$ cd singularity
$ ./autogen.sh
$ ./configure --prefix=/usr/local
$ make
$ sudo make install
```

Build and run containers

A few pre-requisites

- Building a container requires a root, or sudo
  - You can do that on your own machine
  - You can't do that at CHPC clusters
  - > build your containers locally
  - (we make an exception on our linuxclass machine)
- You can run a container as an user
  - You can run your own containers at CHPC
  - You can run CHPC provided containers at CHPC

Run someone else's container

- Singularity allows to run images from Docker hub (and Singularity hub)

```
$ singularity shell docker://ubuntu:latest
$ whoami
$ env | grep SINGULARITY
$ exit
```

- Other ways to run

```
$ singularity exec image program
$ singularity run image
```

Container build process

- All needs to be run as root or with sudo
- Create a bare container

```
$ sudo /usr/local/bin/singularity create --size 2048
```

- Install (bootstrap) the container

```
$ sudo /usr/local/bin/singularity bootstrap ubuntu16.img ubuntu16.def
```

- If additional installation is needed after the bootstrap
  - Shell into the container and do the install manually

```
$ sudo /usr/local/bin/singularity shell –w -s /bin/bash ubuntu16.img
```

  - Execute a script in the container

```
$ sudo /usr/local/bin/singularity exec –w ubuntu16.img myscript.sh
```

Container definition file

- Defines how the container is bootstrapped
  - Header – defines the core OS to bootstrap
  - Sections – scriptlets that perform additional tasks
- Header
  - Docker based (faster installation)

```
BootStrap: docker
From: ubuntu:16.10
```

  - Linux distro based

```
BootStrap: debootstrap
OSVersion: xenial
MirrorURL: http://us.archive.ubuntu.com/ubuntu/
```

Definition file sections

- `%setup` Runs on the host
  - Install host based drivers (e.g. GPU)
- `%post` Runs in the container
  - Install additional packages, configure, etc
- `%runscript` Defines what happens when container is run
  - Execution commands
- `%test` Runs tests after the bootstrap
  - Basic testing

Let's get a definition file

- Download CHPC containers GIT repo

```
$ git clone https://github.com/CHPC-UofU/Singularity-ubuntu-python
```

Go to the ubuntu_python directory and view what's in there

```
$ cd https://github.com/CHPC-UofU/Singularity-ubuntu-python
$ ls
$ cat build_container.sh
… replace singularity with /usr/local/bin/singularity
$ more Singularity
```

Build the container

- Simply type the build script

```
$ ./build_container.sh
```

- CHPC specific caveats
  - File server mount points

```
$ mkdir /uufs /scratch
```

Run the container

- Locally

```
$ singularity shell ubuntu_python.img
$ /usr/bin/python -c "import numpy as np;np.__config__.show()"
```

- At our linuxclass machine:

```
$ scp ubuntu_python.img userxx@linuxclass.chpc.utah.edu:~/
$ ssh userxx@linuxclass.chpc.utah.edu
$ singularity shell ubuntu_python.img
Singularity $ /usr/bin/python -c "import numpy as
np;np.__config__.show()"
```

Some useful tips

- Binding mount points

```
$ export SINGULARITY_BINDPATH="/scratch,/uufs/chpc.utah.edu"
$ singularity shell -B /scratch,/uufs/chpc.utah.edu
ubuntu_python.img
```

- Specifying shell

```
$ export SINGULARITY_SHELL=/bin/bash
$ singularity shell –s /bin/bash ubuntu_python.img
```

- More specialized topics – ask us
  - Using environment modules from the host
  - Using GPUs, MPI over InfiniBand

Using Lmod from containers

- Many Linux programs are binary compatible between distros
  - Most installed binaries are (Intel, PGI tools, DDT, …)
- No need to install these in the container – use our NFS mounted software stack through Lmod
  - Need to have separate Lmod installation for Ubuntu due to some files having different location
- In the container
  - Install Lmod dependencies
  - Modify /etc/bash.bashrc to source our Lmod

https://github.com/CHPC-UofU/Singularity-ubuntu-python/blob/master/Singularity

Using GPUs

- Need to bring in the Nvidia driver stack
  - Pre Singularity 2.3 – explicitly install – make sure to have the same driver version on the host and in the container
  - Singularity 2.3+ - -nv flag

https://github.com/CHPC-UofU/Singularity-tensorflow/blob/master/Singularity

MPI over InfiniBand

- Need to bring in the InfiniBand driver stack
  - For Ubuntu based on https://community.mellanox.com/docs/DOC-2431

https://github.com/CHPC-UofU/Singularity-tensorflow/blob/master/Singularity

Containers for complex software

When to use containers

- Complex software dependencies
  - Especially Python and R packages
    - bioBakery – intricate dependencies of Python and R which did not build on CentOS
    - SEQLinkage – instructions to build on Ubuntu using its packages
- Quick deployment
  - Some Linux distros provide program packages while others don't
    - paraview-python on Ubuntu via apt-get
- Deploying your own code or pipeline

Container build strategy

- Bootstrap the basic container
- Shell into the container
  - Install additional needed programs
    - If they have dependencies, install the dependencies – google for the OS provided packages first and install with apt-get/yum if possible
  - Put the commands in the `%post` scriptlet
- Build the container again
  - Now with the additional commands in the `%post`
  - If something fails, fix it, build container again
- Iterate until all needed programs are installed

Example - bioBakery

- Instructions say to
  - Install VirtualBox, Vagrant, and bioBakery from an archive
    - Great for a desktop, but, not for an HPC cluster
  - Further below they mention Google Cloud
- So we download the bioBakery archive, unpack it and look inside
  - Great, there is `google_cloud/build_biobakery.sh` script
  - In that file, Ubuntu 16.04 is mentioned

Building bioBakery container

- Build base Ubuntu 16.04 container
- sudo shell into the container
  - Start executing the lines of the build_biobakery.sh script, one after another
  - Some dependencies pop up, install them
  - Another caveat – Linuxbrew requires to be installed as non-root
  - Do some web searching and figure how to add a new user and run Linuxbrew as this user
  - In the end, add the correct paths to the container environment

```
$ echo "export PATH=/usr/local/bin:$PATH” >> /environment
```

Building bioBakery container

- Once everything installs in the container
  - Run the bioBakery tests
  - Add %test section that run the bioBakery tests
  - Build the container again, now it will run the tests (will take a few hours)
- Create a module file or an alias to start the container
- See it all at

https://github.com/CHPC-UofU/Singularity-bioBakery

Resources

- http://singularity.lbl.gov
- https://singularity-hub.org
- https://www.chpc.utah.edu/documentation/software/containers.php
- https://github.com/mcuma/chpc_singularity

Windows in a container?

Windows and HPC

- What, Windows?
  - There are programs that researchers use that only run on Windows
  - E.g. data processing that comes with an instrument
- Our current approach
  - Tell them to run on our only Windows server
    - Gets oversubscribed quickly
  - Build a specific VM
    - Resource intensive for us, not high performing
- What if we could run Windows programs on our Linux clusters

Wine

- Windows compatibility layer on Linux
  - https://www.winehq.org/
  - Not an emulator – translates Windows system calls to Linux, provides alternative Windows system libraries,…
  - Actively developed, under CodeWeavers company umbrella
  - Windows ABI completely in user space
  - Most Linux distros come with some version of Wine
  - Generally better to use recent Linux distros for more recent Wine version (https://www.winehq.org/download)

Winetricks

- While Wine provides the basic Windows support, Winetrics is a set of scripts that install additional Windows libraries
  - Like library dependencies in Linux
  - `winetricks list` – to list available libraries
  - Most commonly used libraries are DirectX, .NET, VB or C runtimes

Wine and Singularity

- Poached out of http://dolmades.org/
- Basic Singularity container
  - Recent Ubuntu or Fedora
  - Some winetricks work better on Fedora than Ubuntu, and vice versa
  - Include Wine repo from winehq to get the latest Wine version
  - Some experimentation is needed but if the Windows program is not complicated, success chances are there

%post section

- Install Wine and Winetricks

```
dpkg --add-architecture i386
apt update
apt -y install wget less vim software-properties-common
python3-software-properties apt-transport-https winbind
wget https://dl.winehq.org/wine-builds/Release.key
apt-key add Release.key
apt-add-repository https://dl.winehq.org/wine-builds/ubuntu/
apt update
apt install -y winehq-stable winetricks
```

User space

- User application
  - Done in `%runscript` section
  - First container launch creates WINEPREFIX (Windows file space), then installs the needed applications, and tars the whole WINEPREFIX for future use
  - Subsequent container launch untars WINEPREFIX and launches program

%runscript section

```
TEMPDIR="$(mktemp -d)"
APPDIR="$HOME/WINE/Topofusion"
PROFILEDIR="$HOME/WINE/PROFILES/${USER}@${HOSTNAME}"
…
export WINEPREFIX="$TEMPDIR/wineprefix"
export WINEARCH="win32"

if [ -f "$APPDIR/wineprefix.tgz" ]; then
    echo "Found existing wineprefix - restoring it..."
    mkdir -p "$WINEPREFIX“; cd "$WINEPREFIX“; tar xzf "$APPDIR/wineprefix.tgz"
else
  wineboot --init
  echo "Installing TopoFusion and its dependencies ..."
  winetricks dlls directx9 vb6run
  wget http://topofusion.com/TopoFusion-Demo-Pro-5.43.exe
fi
wine ./TopoFusion-Demo-Pro-5.43.exe
```

Examples

- IDL 6.4 runtime + PeakSelector
  - IDL runtime under Linux crashes due to IDL bug
  - Windows runtime works fine, older IDL (ca. 2010)
  - https://github.com/CHPC-UofU/Singularity-ubuntu-wine-peakselector
- Topofusion
  - My favorite GPS mapping program, e.g. http://home.chpc.utah.edu/~mcuma/summer16/madison/wed/
  - Needs DirectX and VB runtime
  - https://github.com/CHPC-UofU/Singularity-ubuntu-wine-topofusion

Caveats (failed examples)

- Very new application (Win10 like)
  - Installer was not functional under Wine
- Complex scientific application
  - .NET – did not install on Ubuntu, worked on Fedora
  - Microsoft SQL did not install – show stopper
- Wine application compatibility
  - https://appdb.winehq.org/
  - Notice a lot of games

Outlook

- Success rate 1 out of 3 is not that great
  - Still worth trying, the chances are there
  - Singularity makes it easier to experiment
- It would be nice to have a HPC support for Windows so that
  - We would not need to have specialized Win machines
  - We would not have to build special purpose VMs
- May still need to look into the direction of reconfigurable HPC clusters like Bridges or Jetstream

Open discussion

Open discussion

- What do you do with complicated installs?
- What is your experience with containers?
- How are you supporting other Oses?