Intro to Singularity Containers

Fuzzy Rogers
Research Computing Administrator
Materials Research Laboratory (MRL)
Center for Scientific Computing (CSC)
fuz@ucsb.edu
MRL 2066B

Sharon Solis
Research Computing Consultant
Enterprise Technology Services (ETS)
Center for Scientific Computing (CSC)
swsolis@ucsb.edu
Elings 3229

Paul Weakliem
Research Computing Administrator
CNSI Research Computing Support
Center for Scientific Computing (CSC)
weakliem@cnsi.ucsb.edu
Elings Hall 3231

Ack!

- Acknowledgements - http://csc.cnsi.ucsb.edu/pubs

Please acknowledge the CSC in publications and presentations if you are using our facilities computing resources (including staff involvement) in your research.

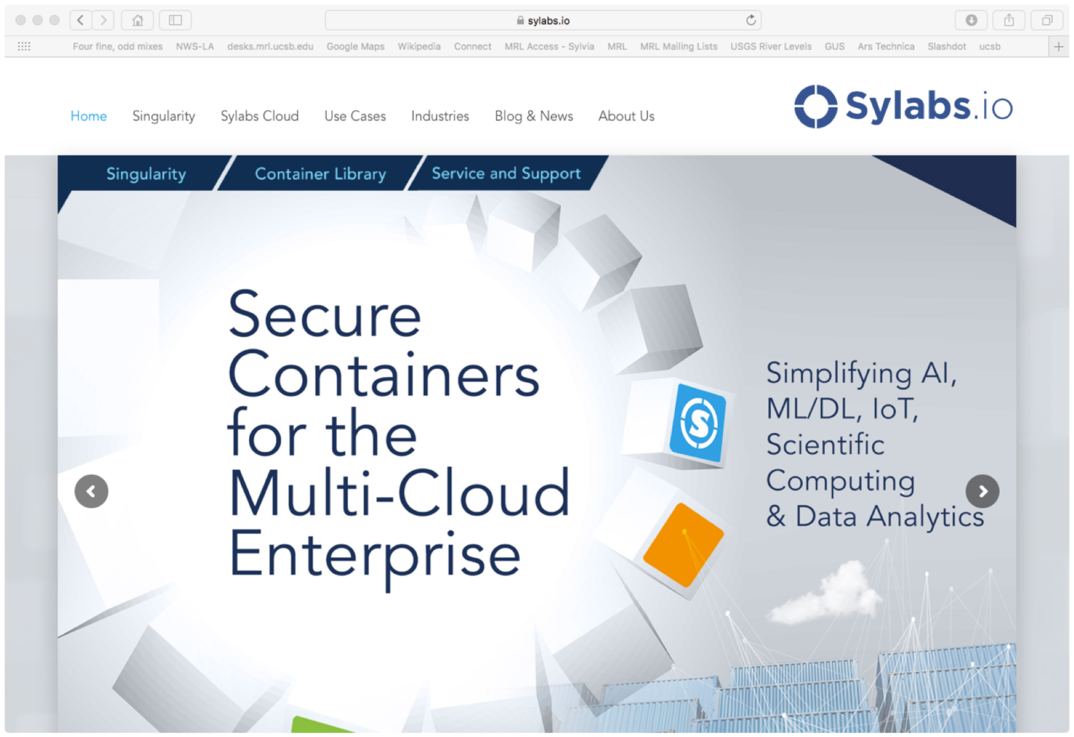
“We acknowledge support from the Center for Scientific Computing from the CNSI, MRL: an NSF MRSEC (DMR-1720256) and NSF CNS-1725797.”

What is this thing you call a container?

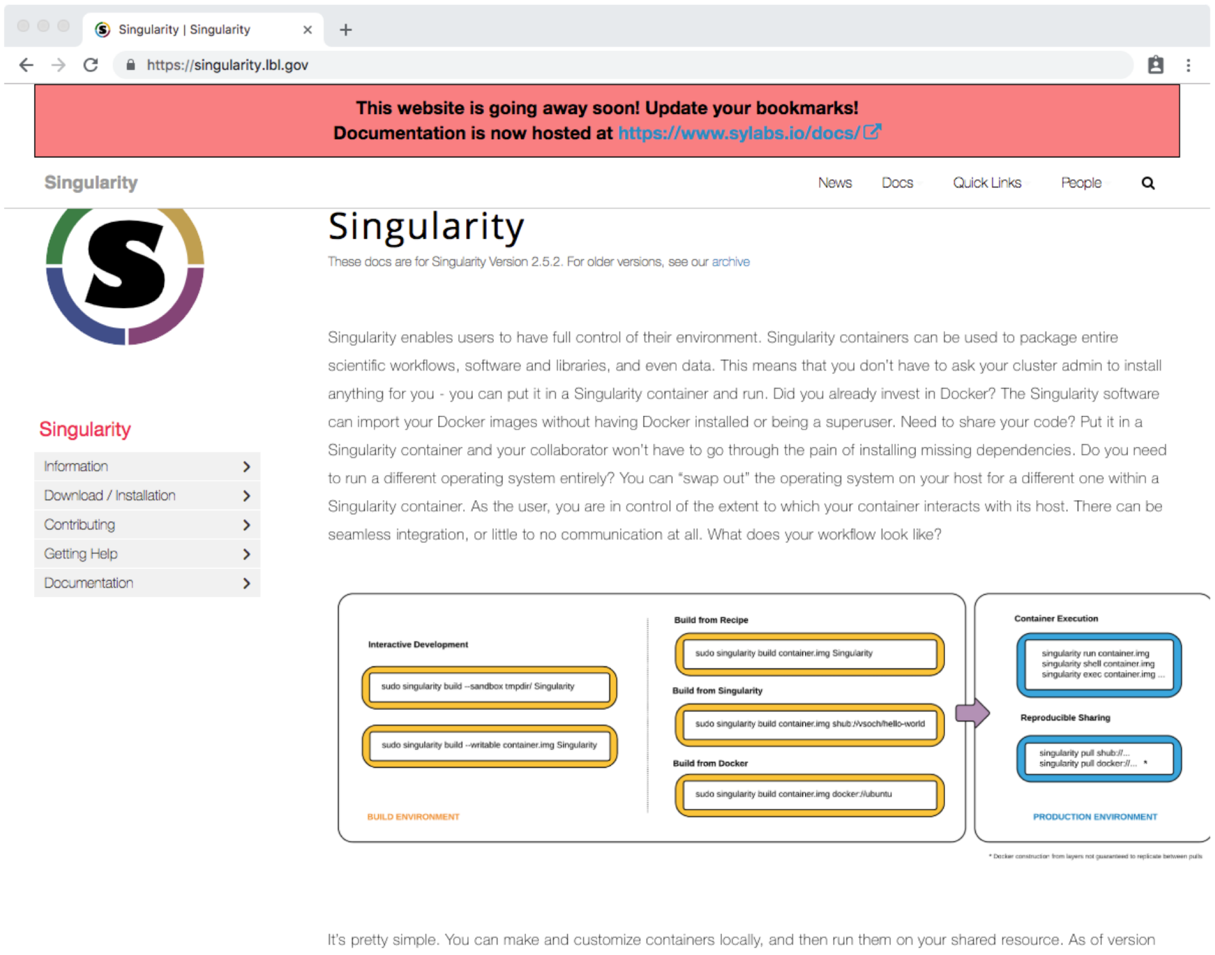
- Containers are linux software environments where the user can have control over everything but the kernel.
- Singularity containers can be used to package entire scientific workflows, software and libraries, and even data. This means that you don’t have to ask your cluster admin to install anything for you - you can create a software workflow in a Singularity container and run it on the clusters.
- That said, we admins have put together some containers and we can and will help you with more complex containers (i.e. Tensorflow/Keras).

Singularity has become it's own Company

(though the Software is still OpenSource)



Originally developed at LBL



This website is going away soon! Update your bookmarks!
Documentation is now hosted at https://www.sylabs.io/docs/

Singularity

These docs are for Singularity Version 2.5.2. For older versions, see our archive

Singularity enables users to have full control of their environment. Singularity containers can be used to package entire scientific workflows, software and libraries, and even data. This means that you don't have to ask your cluster admin to install anything for you - you can put it in a Singularity container and run. Did you already invest in Docker? The Singularity software can import your Docker images without having Docker installed or being a superuser. Need to share your code? Put it in a Singularity container and your collaborator won't have to go through the pain of installing missing dependencies. Do you need to run a different operating system entirely? You can "swap out" the operating system on your host for a different one within a Singularity container. As the user, you are in control of the extent to which your container interacts with its host. There can be seamless integration, or little to no communication at all. What does your workflow look like?

```
sudo singularity build --sandbox tmpdir/ Singularity
sudo singularity build --writable container.img Singularity
sudo singularity build container.img Singularity
sudo singularity build container.img shub://vsoch/hello-world
sudo singularity build container.img docker://ubuntu
singularity run container.img
singularity shell container.img
singularity exec container.img ...
singularity pull shub://...
singularity pull docker://... *
```

* Docker construction from layers not guaranteed to replicate between pulls

It's pretty simple. You can make and customize containers locally, and then run them on your shared resource. As of version

How to use a Singularity Container

(Knot-GPU2)

- export PATH=/sw/csc/singularity/bin:$PATH
- singularity shell /sw/csc/SingularityImg/ubuntu-tf17-knot-gpu2.img
- source ~/.bashrc
- Congratulations – you are now in a singularity container optimized for the GPUs on knot-gpu2.cnsi.ucsb.edu.
- Note that this is an interactive session (i.e. the “shell” in the 2nd line).
- Note that you must have anaconda installed with tensorflow. Say what?

Installing Anaconda Tensorflow

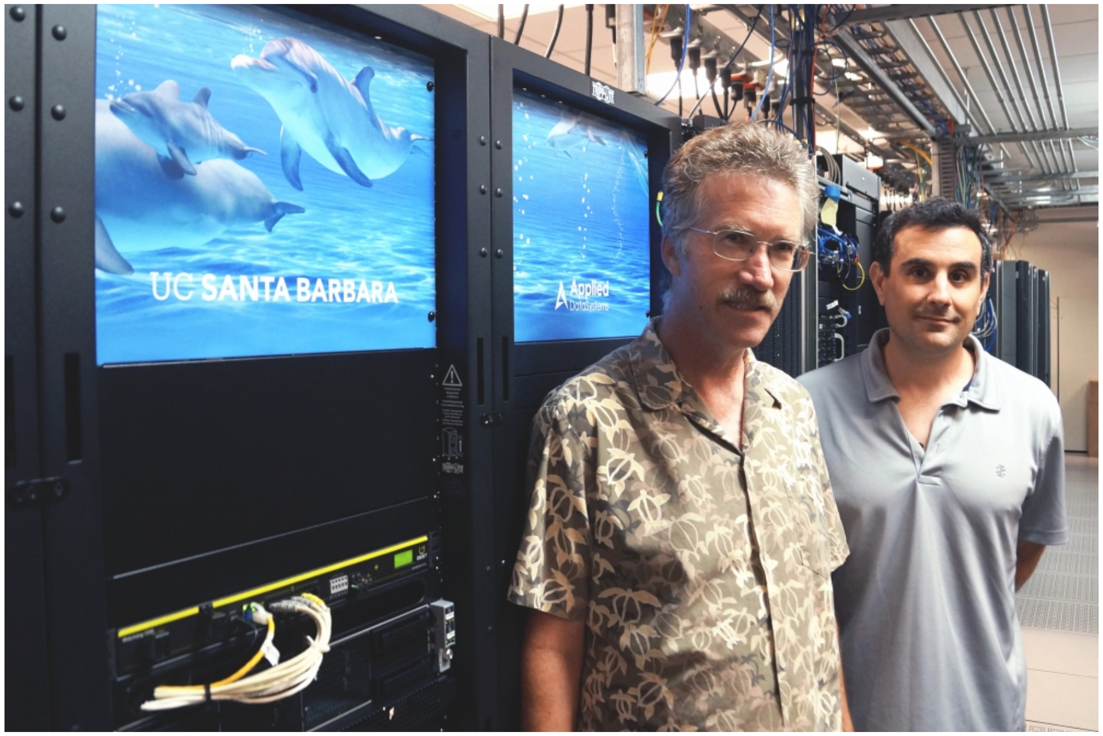
- Download anaconda (https://www.anaconda.com/download/#linux)
  - You'll want the 64bit x86 installer – I've normally used 2.7
  - wget https://repo.anaconda.com/archive/Anaconda2-5.3.0-Linux-x86_64.sh
  - sh Anaconda2-5.3.0-Linux-x86_64.sh (let anaconda modify your .bashrc)
  - source .bashrc (to make sure anaconda is the chosen python)
    - I've noticed that sometimes .bashrc isn't always sourced, so sometimes explicitly issuing an
      - export PATH=~/username/anaconda2/bin:$PATH

      fixes the issue.
  - Verify that by typing which python and it should say ~/anaconda2/bin/python
  - pip install tensorflow-gpu (if it gives some errors on certain packages, just pip install those like… pip install argparse and pip install PyHamcrest , etc.)
  - A few items to add to the top of your .bashrc file to locate NVIDIA stuff

```
export PATH=/usr/local/nvidia:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
```

TF example

Let’s classify an image and see what a TF model sees in this photo.



```
[fuz@sinode170 ~]$ source activate tensorflow

(tensorflow) [fuz@sinode170 ~]$ /sw/csc/singularity-2.4.5/bin/singularity shell /sw/csc/singularity-images/ubuntu-tf1.5-
GPU9.img

Singularity: Invoking an interactive shell within container...

Singularity ubuntu-tf1.5-GPU9.img:~> import tensorflow as tf

bash: import: command not found

Singularity ubuntu-tf1.5-GPU9.img:~> which python

/usr/bin/python

Singularity ubuntu-tf1.5-GPU9.img:~> source .bashrc

Singularity ubuntu-tf1.5-GPU9.img:~> which python

/home/fuz/anaconda2/bin/python

Singularity ubuntu-tf1.5-GPU9.img:~/tensorflow-inception/models-master/tutorials/image/imagenet> python
classify_image.py

>> Downloading inception-2015-12-05.tgz 100.0%

Successfully downloaded inception-2015-12-05.tgz 88931400 bytes.

CRITICAL:tensorflow:File does not exist /tmp/imagenet/fluoromax.jpeg

Singularity ubuntu-tf1.5-GPU9.img:~/tensorflow-inception/models-master/tutorials/image/imagenet> mv ~/pod-weakliem-
brown-uc-santa-barbara.jpg /tmp/imagenet/fluoromax.jpeg
```

```
Singularity ubuntu-tf1.5-GPU9.img:~/tensorflow-inception/models-master/tutorials/image/imagenet> python classify_image.py
2018-11-01 16:07:01.444163: I tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2018-11-01 16:07:02.015955: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1212] Found device 0 with properties:
name: Tesla P100-PCIE-16GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:02:00.0
totalMemory: 15.89GiB freeMemory: 14.57GiB
2018-11-01 16:07:02.949295: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1212] Found device 1 with properties:
name: Tesla P100-PCIE-16GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:03:00.0
totalMemory: 15.89GiB freeMemory: 14.57GiB
2018-11-01 16:07:03.256965: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1212] Found device 2 with properties:
name: Tesla P100-PCIE-16GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:82:00.0
totalMemory: 15.89GiB freeMemory: 14.57GiB
2018-11-01 16:07:03.565751: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1212] Found device 3 with properties:
name: Tesla P100-PCIE-16GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:83:00.0
totalMemory: 15.89GiB freeMemory: 14.57GiB
```

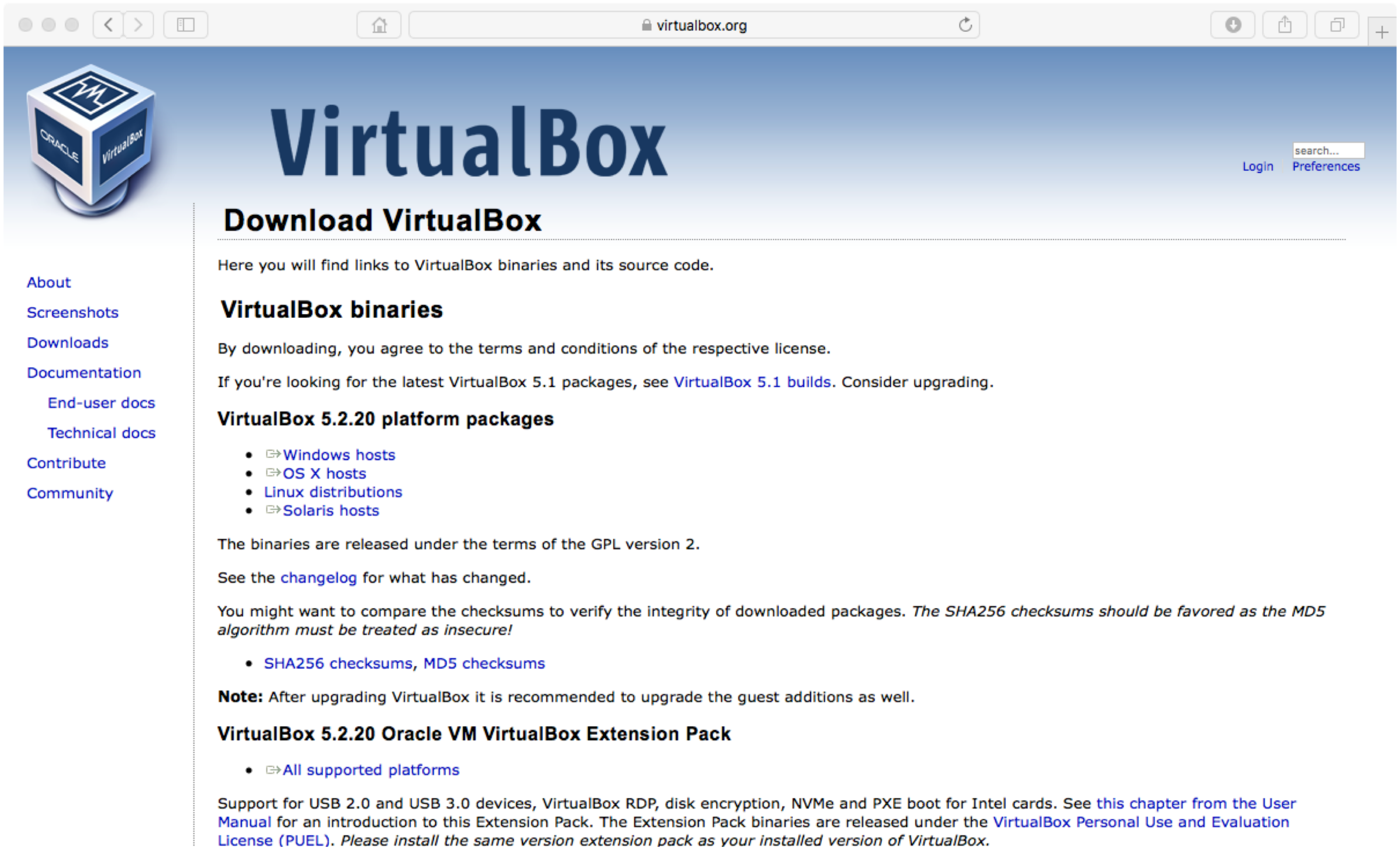
```
2018-11-01 16:07:03.568612: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1227] Device peer to peer matrix
2018-11-01 16:07:03.568718: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1233] DMA: 0 1 2 3
2018-11-01 16:07:03.568730: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1243] 0:   Y Y N N
2018-11-01 16:07:03.568738: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1243] 1:   Y Y N N
2018-11-01 16:07:03.568745: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1243] 2:   N N Y Y
2018-11-01 16:07:03.568752: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1243] 3:   N N Y Y
2018-11-01 16:07:03.568768: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1312] Adding visible gpu devices: 0, 1, 2, 3

2018-11-01 16:07:09.039015: I tensorflow/core/common_runtime/gpu/gpu_device.cc:993] Creating TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:0 with 14123 MB memory) -> physical GPU (device: 0, name: Tesla P100-PCIE-16GB, pci bus id:
0000:02:00.0, compute capability: 6.0)

2018-11-01 16:07:11.583130: I tensorflow/core/common_runtime/gpu/gpu_device.cc:993] Creating TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:1 with 14123 MB memory) -> physical GPU (device: 1, name: Tesla P100-PCIE-16GB, pci bus id:
0000:03:00.0, compute capability: 6.0)

2018-11-01 16:07:13.224968: I tensorflow/core/common_runtime/gpu/gpu_device.cc:993] Creating TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:2 with 14123 MB memory) -> physical GPU (device: 2, name: Tesla P100-PCIE-16GB, pci bus id:
0000:82:00.0, compute capability: 6.0)

2018-11-01 16:07:19.085977: I tensorflow/core/common_runtime/gpu/gpu_device.cc:993] Creating TensorFlow device
(/job:localhost/replica:0/task:0/device:GPU:3 with 14119 MB memory) -> physical GPU (device: 3, name: Tesla P100-PCIE-16GB, pci bus id:
0000:83:00.0, compute capability: 6.0)

2018-11-01 16:07:22.424384: W tensorflow/core/framework/op_def_util.cc:343] Op BatchNormWithGlobalNormalization is deprecated. It will cease
to work in GraphDef version 9. Use tf.nn.batch_normalization().
```

```
military uniform (score = 0.33332)
minibus (score = 0.05024)
crutch (score = 0.04971)
amphibian, amphibious vehicle (score = 0.02861)
jeep, landrover (score = 0.02484)
```

Obviously the training algorithm didn’t account for groups of people – garbage in, garbage out!

Making your own Singularity Containers

- The Workflow – Step 1 – Build a linux VM so you can be root
  - Download and install a Virtual Machine application (I chose VirtualBox)
  - For pod.cnsi.ucsb.edu, build a CentOS 7 virtual machine
    - Choose your HD size so that it can accommodate your OS *and* your singularity images that you will create (i.e. 20GBs or so)
    - I chose CentOS-7-x86_64-Everything-1804.iso as the base
      - My favorite mirror is http://mirrors.oit.uci.edu/centos/7/isos/x86_64/
      - Remember that you want to install the Development Tools (Compute Node has it on the left)
    - Then we'll download and build singularity as we'll be root on our own little linux machine.
    - And then you can build singularity images to fit your exact needs. Once you've tested your workflow, you can copy those images to pod.cnsi.ucsb.edu and create jobs for them to run.



Download VirtualBox

Here you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

If you're looking for the latest VirtualBox 5.1 packages, see VirtualBox 5.1 builds. Consider upgrading.

VirtualBox 5.2.20 platform packages

- Windows hosts
- OS X hosts
- Linux distributions
- Solaris hosts

The binaries are released under the terms of the GPL version 2.

See the changelog for what has changed.

You might want to compare the checksums to verify the integrity of downloaded packages. *The SHA256 checksums should be favored as the MD5 algorithm must be treated as insecure!*

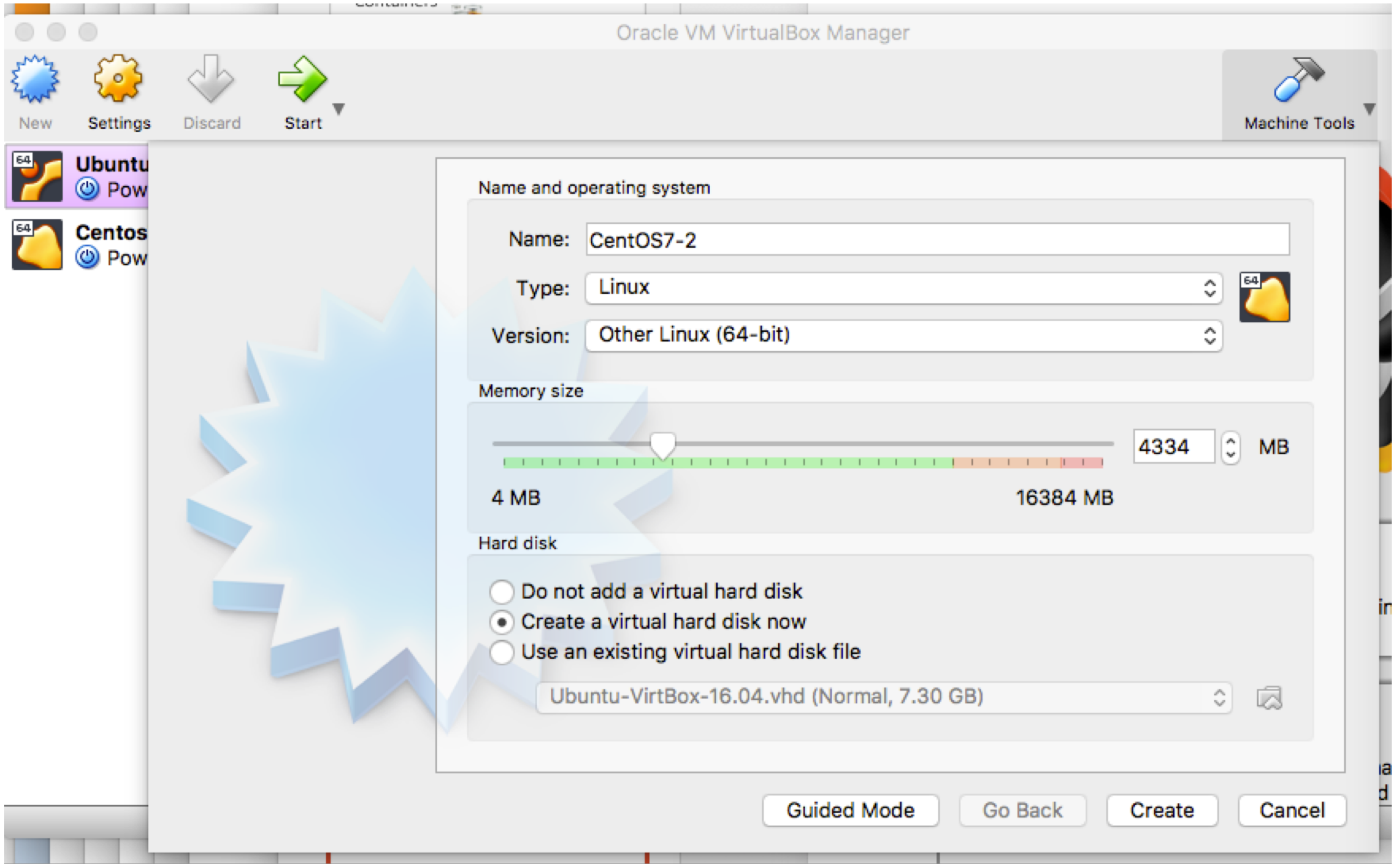
- SHA256 checksums, MD5 checksums

Note: After upgrading VirtualBox it is recommended to upgrade the guest additions as well.

VirtualBox 5.2.20 Oracle VM VirtualBox Extension Pack

- All supported platforms

Support for USB 2.0 and USB 3.0 devices, VirtualBox RDP, disk encryption, NVMe and PXE boot for Intel cards. See this chapter from the User Manual for an introduction to this Extension Pack. The Extension Pack binaries are released under the VirtualBox Personal Use and Evaluation License (PUEL). *Please install the same version extension pack as your installed version of VirtualBox.*



Name and operating system

Name: CentOS7-2

Type: Linux

Version: Other Linux (64-bit)

Memory size

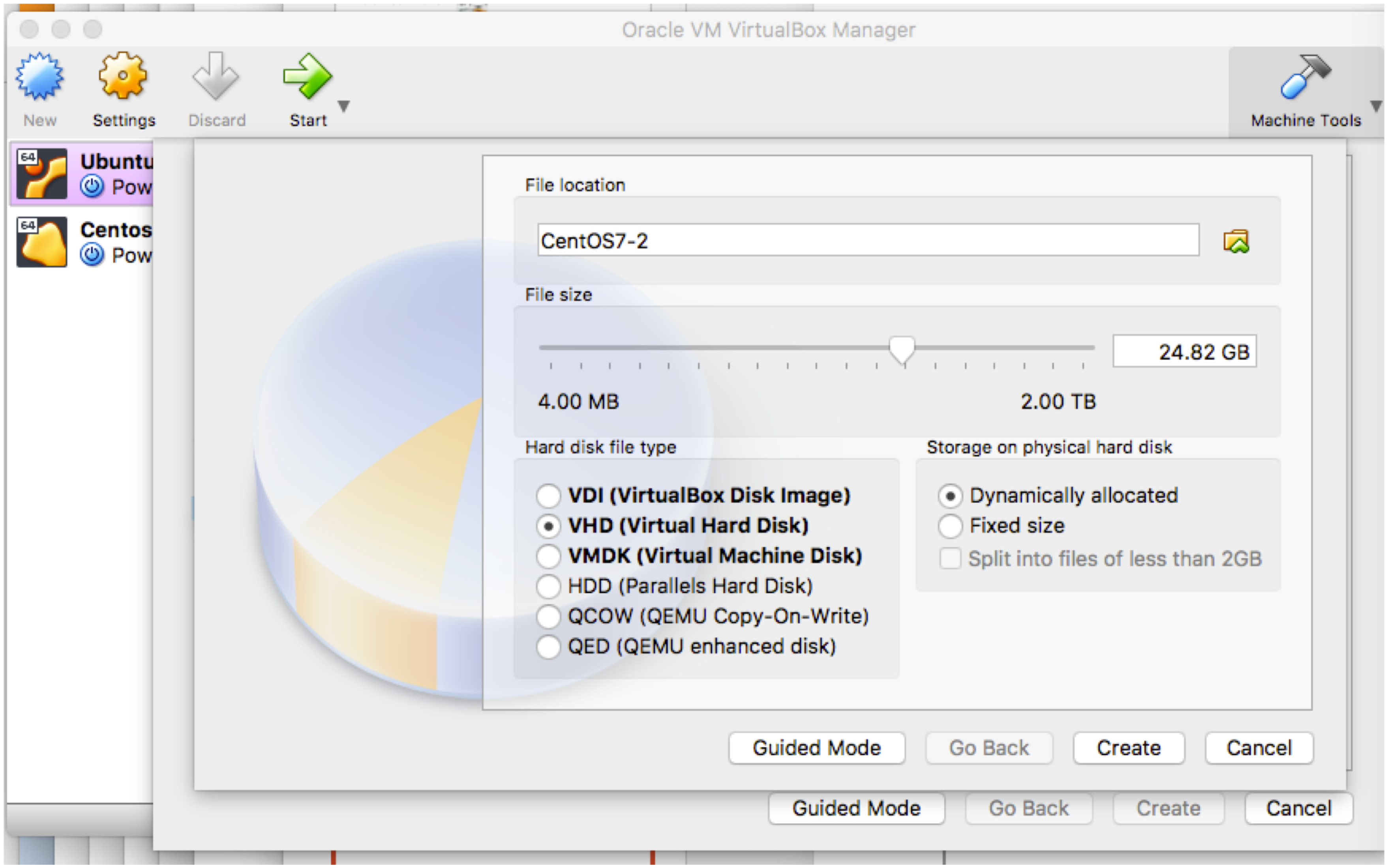
4334 MB

4 MB 16384 MB

Hard disk

- Do not add a virtual hard disk
- [✓ Create a virtual hard disk now]
- Use an existing virtual hard disk file

Ubuntu-VirtBox-16.04.vhd (Normal, 7.30 GB)


Oracle VM VirtualBox Manager

File location

CentOS7-2

File size

24.82 GB

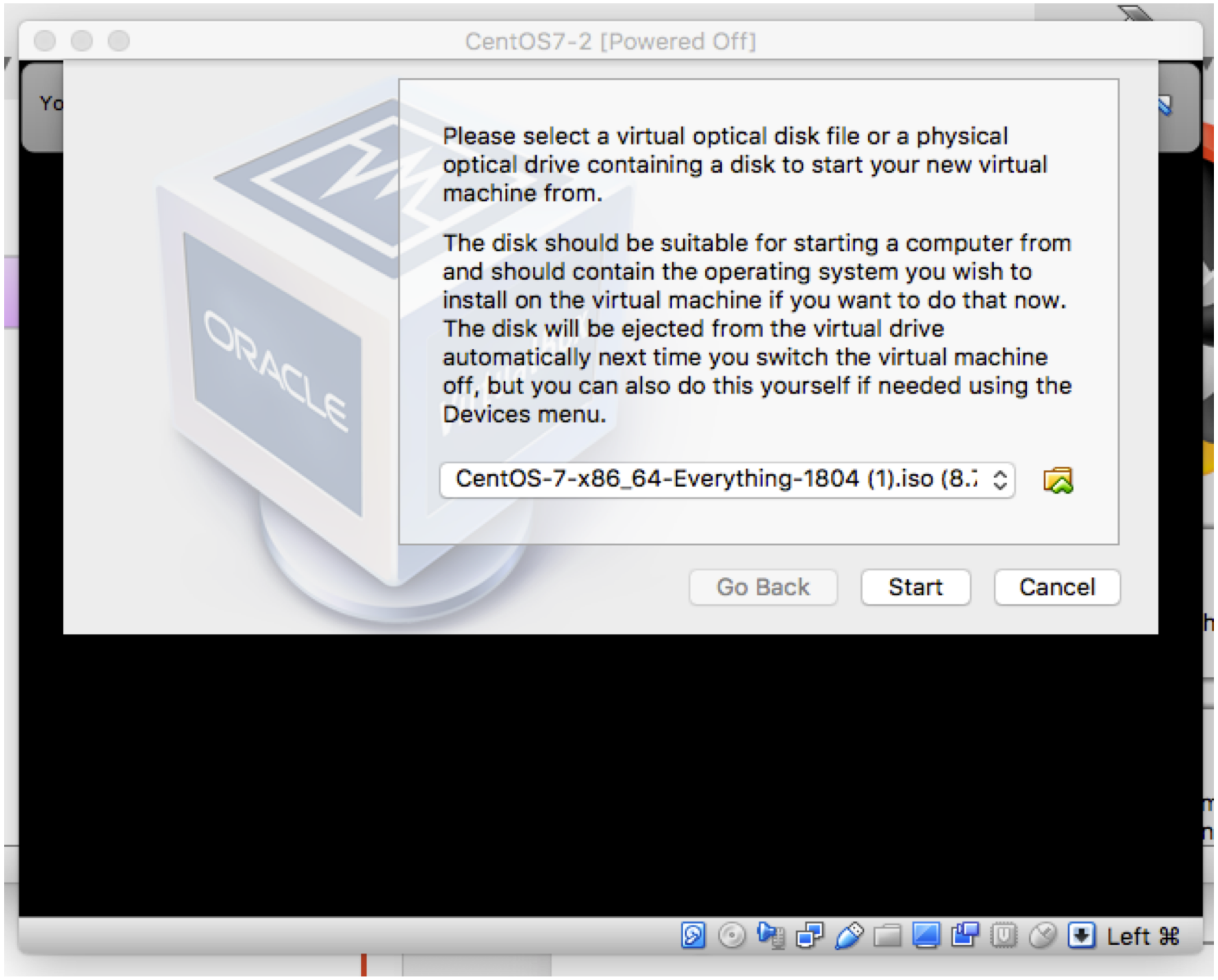
4.00 MB 2.00 TB

Hard disk file type

- VDI (VirtualBox Disk Image)
- [✓ VHD (Virtual Hard Disk)]
- VMDK (Virtual Machine Disk)
- HDD (Parallels Hard Disk)
- QCOW (QEMU Copy-On-Write)
- QED (QEMU enhanced disk)

Storage on physical hard disk

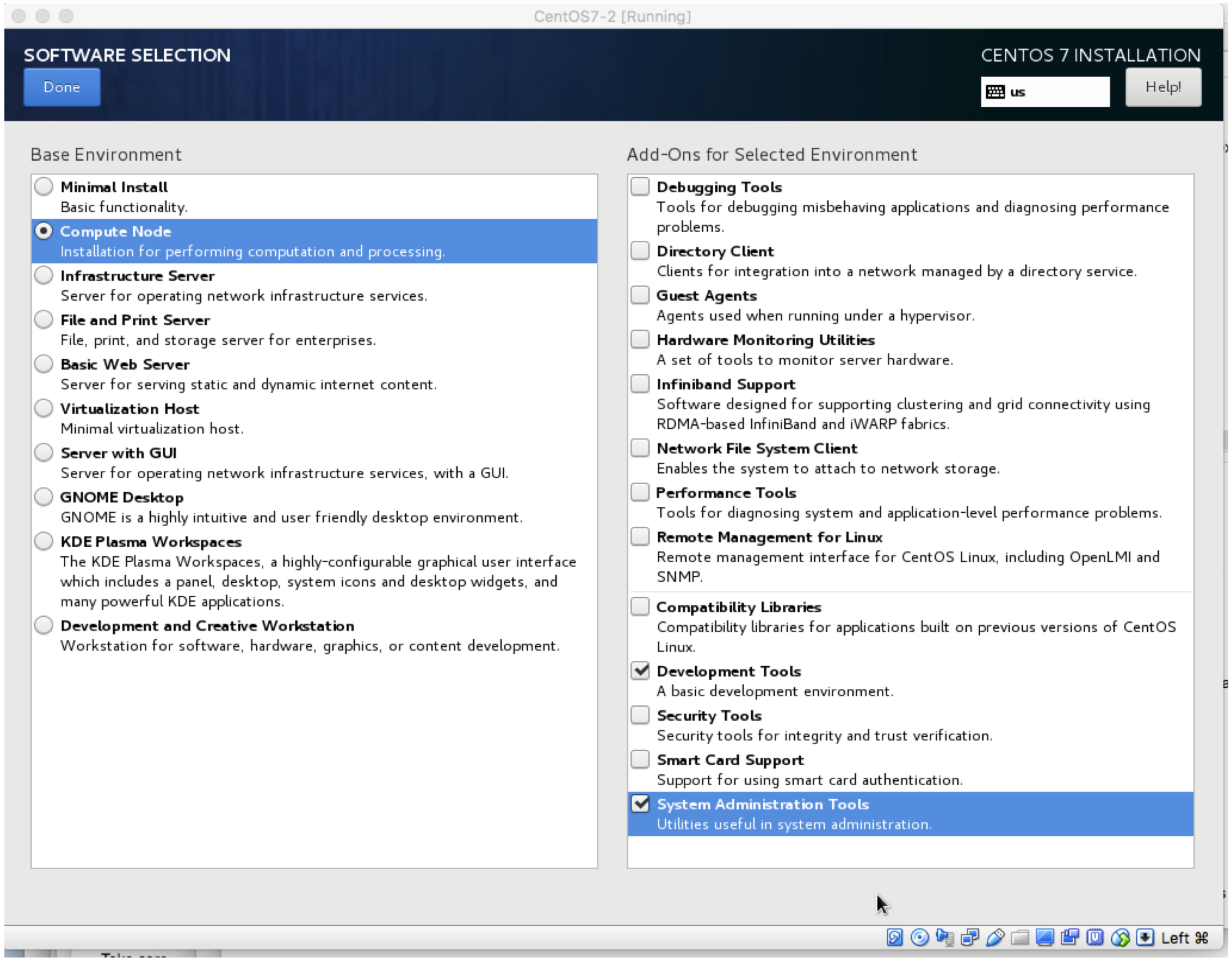
- [✓ Dynamically allocated]
- Fixed size
- Split into files of less than 2GB



Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.

The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.

CentOS-7-x86_64-Everything-1804 (1).iso (8.



CentOS7-2 [Running]

SOFTWARE SELECTION

Done

CENTOS 7 INSTALLATION

us

Help!

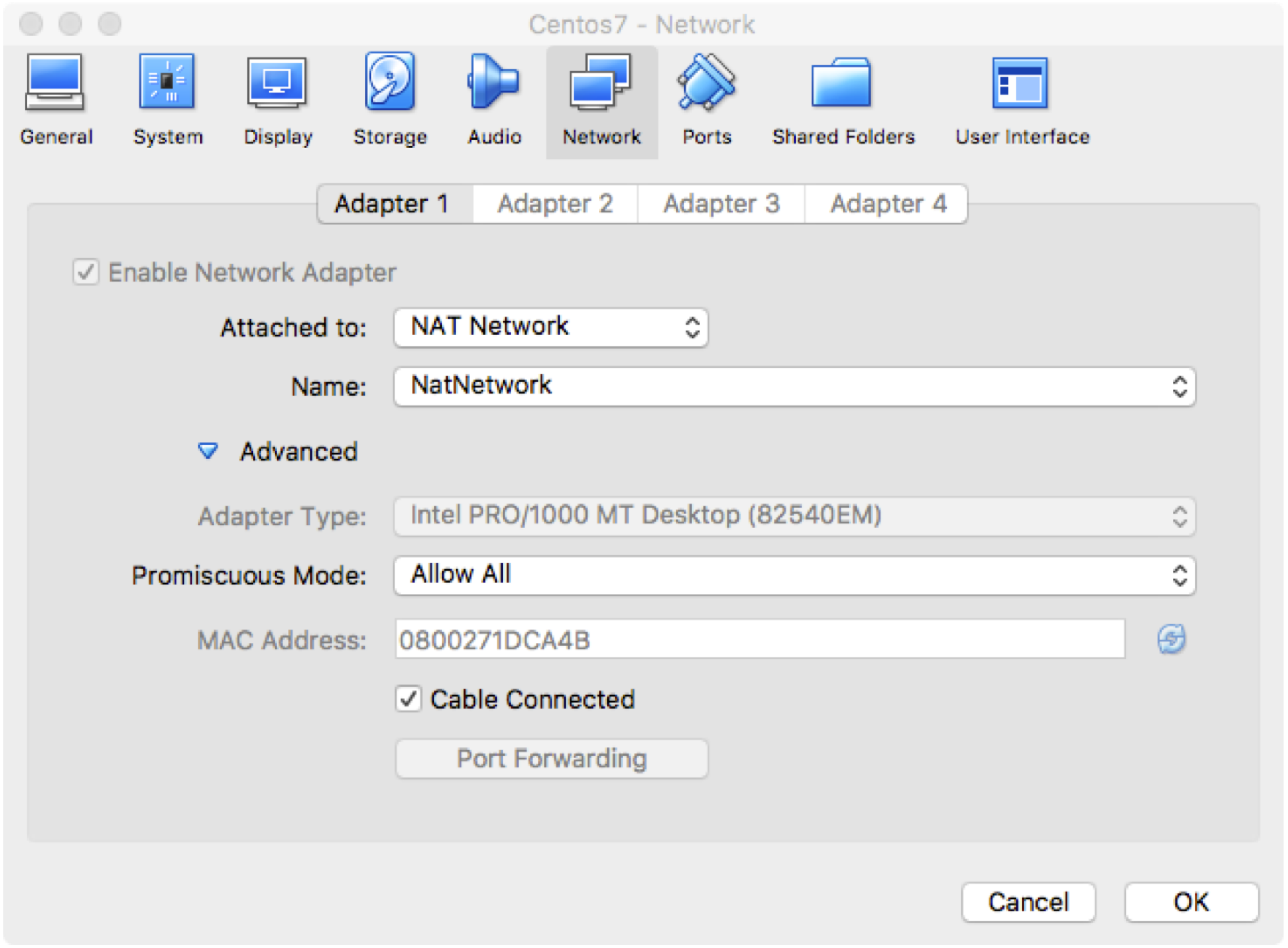
Base Environment

- ○ **Minimal Install**
  Basic functionality.
- ● **Compute Node**
  Installation for performing computation and processing.
- ○ **Infrastructure Server**
  Server for operating network infrastructure services.
- ○ **File and Print Server**
  File, print, and storage server for enterprises.
- ○ **Basic Web Server**
  Server for serving static and dynamic internet content.
- ○ **Virtualization Host**
  Minimal virtualization host.
- ○ **Server with GUI**
  Server for operating network infrastructure services, with a GUI.
- ○ **GNOME Desktop**
  GNOME is a highly intuitive and user friendly desktop environment.
- ○ **KDE Plasma Workspaces**
  The KDE Plasma Workspaces, a highly-configurable graphical user interface which includes a panel, desktop, system icons and desktop widgets, and many powerful KDE applications.
- ○ **Development and Creative Workstation**
  Workstation for software, hardware, graphics, or content development.

Add-Ons for Selected Environment

- ☐ **Debugging Tools**
  Tools for debugging misbehaving applications and diagnosing performance problems.
- ☐ **Directory Client**
  Clients for integration into a network managed by a directory service.
- ☐ **Guest Agents**
  Agents used when running under a hypervisor.
- ☐ **Hardware Monitoring Utilities**
  A set of tools to monitor server hardware.
- ☐ **Infiniband Support**
  Software designed for supporting clustering and grid connectivity using RDMA-based InfiniBand and iWARP fabrics.
- ☐ **Network File System Client**
  Enables the system to attach to network storage.
- ☐ **Performance Tools**
  Tools for diagnosing system and application-level performance problems.
- ☐ **Remote Management for Linux**
  Remote management interface for CentOS Linux, including OpenLMI and SNMP.
- ☐ **Compatibility Libraries**
  Compatibility libraries for applications built on previous versions of CentOS Linux.
- ✓ **Development Tools**
  A basic development environment.
- ☐ **Security Tools**
  Security tools for integrity and trust verification.
- ☐ **Smart Card Support**
  Support for using smart card authentication.
- ✓ **System Administration Tools**
  Utilities useful in system administration.

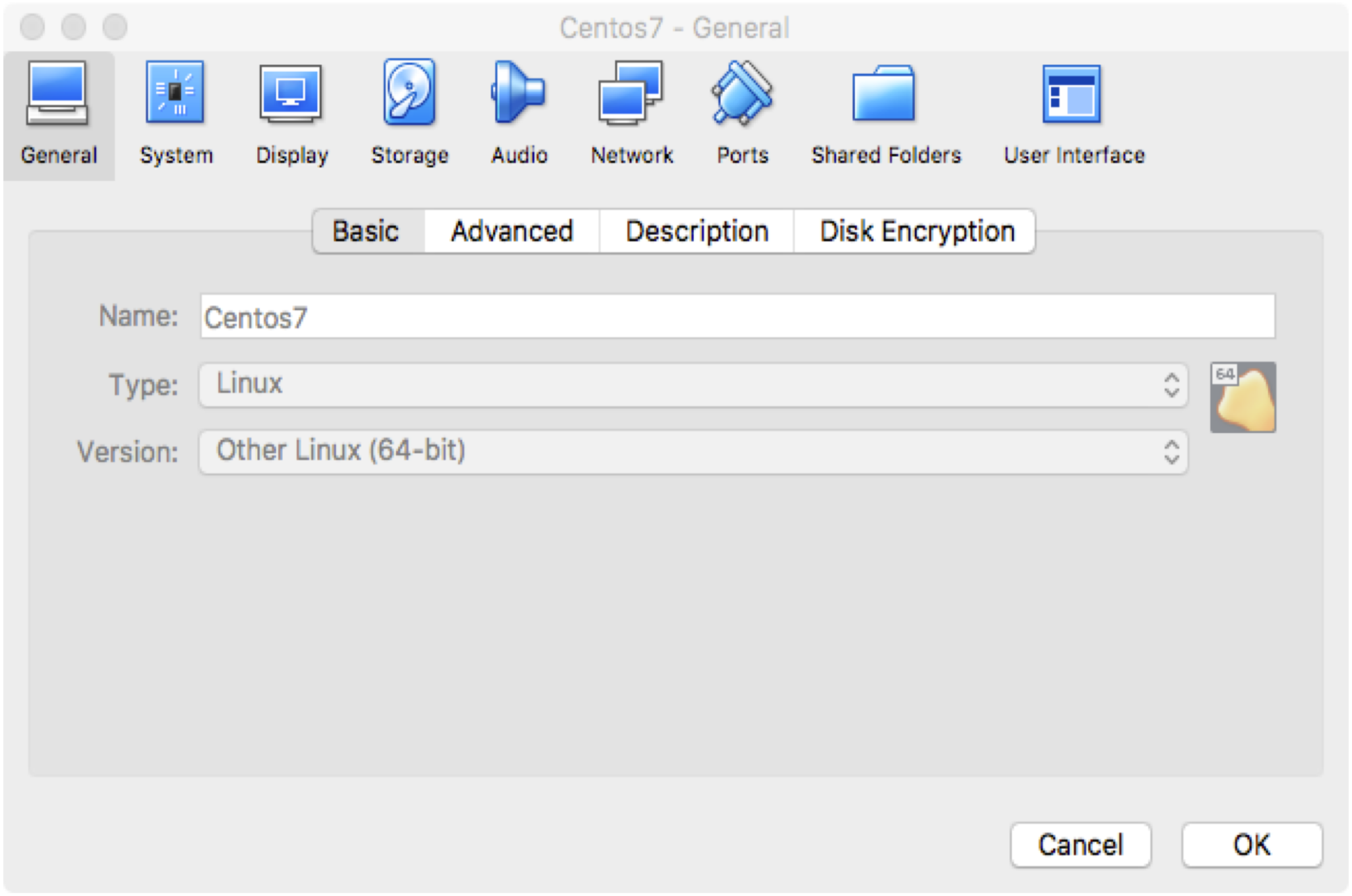
Left ⌘



✓ Enable Network Adapter

Attached to:	NAT Network
Name:	NatNetwork
Adapter Type:	Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode:	Allow All
MAC Address:	0800271DCA4B

✓ Cable Connected



Name: Centos7

Type: Linux

Version: Other Linux (64-bit)

Making your own Singularity Containers

- The Workflow – Step 2 – Getting and Building Singularity
  - Start your VM from VirtualBox, login as root
  - https://github.com/sylabs/singularity/releases - the .tar.gz are fine
    - wget https://github.com/sylabs/singularity/releases/download/2.5.2/singularity-2.5.2.tar.gz
    - gunzip that file, untar that file
    - cd singularity-2.5.2
    - ./configure –prefix=/singularity   (prefix not necessary) Note if configure fails with missing packages – you might need to   yum install somepackagelikegcc
    - make   - if there are no errors….
    - make install
    - yum install epel-release   ,   yum install debootstrap
  - Voila – you now have singularity in your VM and can create singularity containers

Making your own Singularity Containers

- The Workflow – Step 3 – Creating an Ubuntu container
  - Build an empty container
    - export PATH=$PATH:/singularity/bin
    - singularity create ubuntu.img
    - singularity image.expand –size 4000 ubuntu.img
    - singularity build ubuntu.img createdeb.def where createdeb.def:

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

%runscript
apt-get install python
```

    - singularity shell ubuntu.img ← you're now in the container (--writable)
    - apt-get install python sudo ← and anything else you want to install (might need sudo for other apt-gets like sudo apt-get install somepackageoranother – so you need sudo)
    - Exit ← gets you out of the container back into CentOS 7

Next Steps

- Now that you have a container- customize it to work with your workflow. Install whatever packages you need.
- When you use a container on the clusters, it automatically mounts your home directory.
- The container sees all of the system's memory and CPUs, but none of the other filesystems/directories unless you explicitly mount them – and then they're generally readonly unless it's /scratch.
    - singularity shell -B /scratch:/mnt /sw/singularity/SingularityImages-knot/ubuntu_croco.img
      Here, the /scratch directory is mounted in your container at /mnt.
- From your CentOS 7 install, scp myubuntu.img username@pod.cnsi.ucsb.edu
- Note that once your image is on the clusters, it is immutable (unless you ask us to alter something)

- Example job submission file on pod.cnsi.ucsb.edu – test-croco.job

```
#!/bin/bash -l
#Serial (1 core on one node) job...
#SBATCH --nodes=1 --ntasks-per-node=1
cd $SLURM_SUBMIT_DIR
source .bashrc
singularity exec -B /scratch:/mnt /sw/singularity/SingularityImages-knot/ubuntu_croco.img /home/fuz/test-croco.in
```

- Example run file for the container – test-croco.in

```
export PATH=/home/fuz/anaconda2/bin:$PATH
apt list --installed
echo ""
echo "Which python am I using:"
which python
echo ""
Echo "Determine whether a number is prime or not"
python primeornot.py
```

- Submit the job
  - sbatch test-croco.job

What in the world is croco? (Just an FYI)

- Optimized parallel implementation of local sequence alignment algorithms
    - Local sequence alignment is a cornerstone of bioinformatics, allowing to compare the amino-acid sequences of different proteins, or the nucleotide sequences of different pieces of DNA. The Basic Local Alignment Search Tool (BLAST) has revolutionized the field of bioinformatics, and is currently implemented in all free and commercial bioinformatics packages. However, with the advent of Next Generation Sequencing (NGS) and the development of new sequencing techniques, the utility of traditional BLAST implementations is limited. CrocoBLAST combines the accuracy and general applicability of BLAST with computational efficiency, accessibility, and user experience, so that NGS data can be analyzed efficiently even when only modest computational resources are available.

User currently using Singularity

CroCo was created to identify instances of Cross-Contamination in Next Generation Sequencing runs. When NGS reads first came out, researchers realized that we rarely need as much data is generated from a single sample, and that it is cost-effective samples concurrently on the same run. However, years later it was found that doing so often results in 'bleed through', where the sequence of one protein from one sample can be mislabeled as also being native to another sample in the same run. Obviously, the impacts of this finding can be disastrous! CroCo is designed to check each and every read for these types of errors. I now run each of the sequencing runs through CroCo prior to using these data in downstream analyses so that I can be sure of which proteins are native to the 100-some species that we are working with to understand their evolutionary relatedness. Being confident in the species relationships means being confident that the protein sequences are correctly identified.

Singularity Tips

- Singularity can have environment variables in its containers

```
Singularity ubuntu-tf17-knot-gpu2.img:/.singularity.d/env> more 90-environment.sh
# Custom environment shell code should follow
export PATH=/usr/local/nvidia:$PATH
export LD_LIBRARY_PATH=/usr/local/nvidia:$LD_LIBRARY_PATH
```

- Can also pipe that script through the container and into the Python binary which exists inside the container using the following command:

```
cat hello.py | singularity /sw/singularity/SingularityImages-knot/ubuntu_croco.img
```

- Notice how the runscript has bash followed by $@? This is good practice to include in a runscript, as any arguments passed by the user will be given to the container.

```
singularity exec /sw/singularity/SingularityImages-knot/ubuntu_croco.img  cat /.singularity.d/runscript
#!/bin/sh
exec /bin/bash "$@"
```

There is so much more

- Obviously Machine Learning / AI can use Singularity extensively.
  - However we have yet to get it working consistently on pod.cnsi.ucsb.edu but we're working on it and hope to have it up and running in the coming month (or two depending on workload)

- The LBL docs are still online and we've PDF'd them for future use. Once available they'll be online on the http://csc.cnsi.ucsb.edu documentation section. http://singularity.lbl.gov/

- More to come – if you're interested in something specific please let us know and we'll inform you of future enhancements and how they can be used (i.e. docker images)