Using Containers on the Cannon Cluster : Singularity

Objectives

- What are containers and why we care? (overview)
- Singularity container system
- How to run Singularity containers on Cannon:
  - running simple containers
  - pulling from docker registry or sylab library
  - Using GPUs
- How to build your own containers
  - docker
  - bind mount
- How do I get help?

What problems are we trying to solve?

Deploying Applications:

Building software is often a complicated business, particularly on HPC and other multi-tenant systems:

- HPC clusters have typically very specialized software stacks which might not adapt well to general purpose applications.
- OS installations are streamlined.
  Some applications might need dependencies that are not readily available and complex to build from source.
- End users use Ubuntu or Arch, cluster typically use RHEL, or SLES, or other specialized OS.
  (... “>$ sudo apt-get install “ will not work)

What problems are we trying to solve?

Portability and Reproducibility:

- Running applications on multiple systems typically needs replicating the installations multiple times making it hard to keep consistency.
- It would be useful to publish the exact application used to run a calculation for reproducibility or documentation purpose.
- As a user can I minimize the part of the software stack I have no control on, to maximize reproducibility without sacrificing performance?

What problems are we trying to solve?

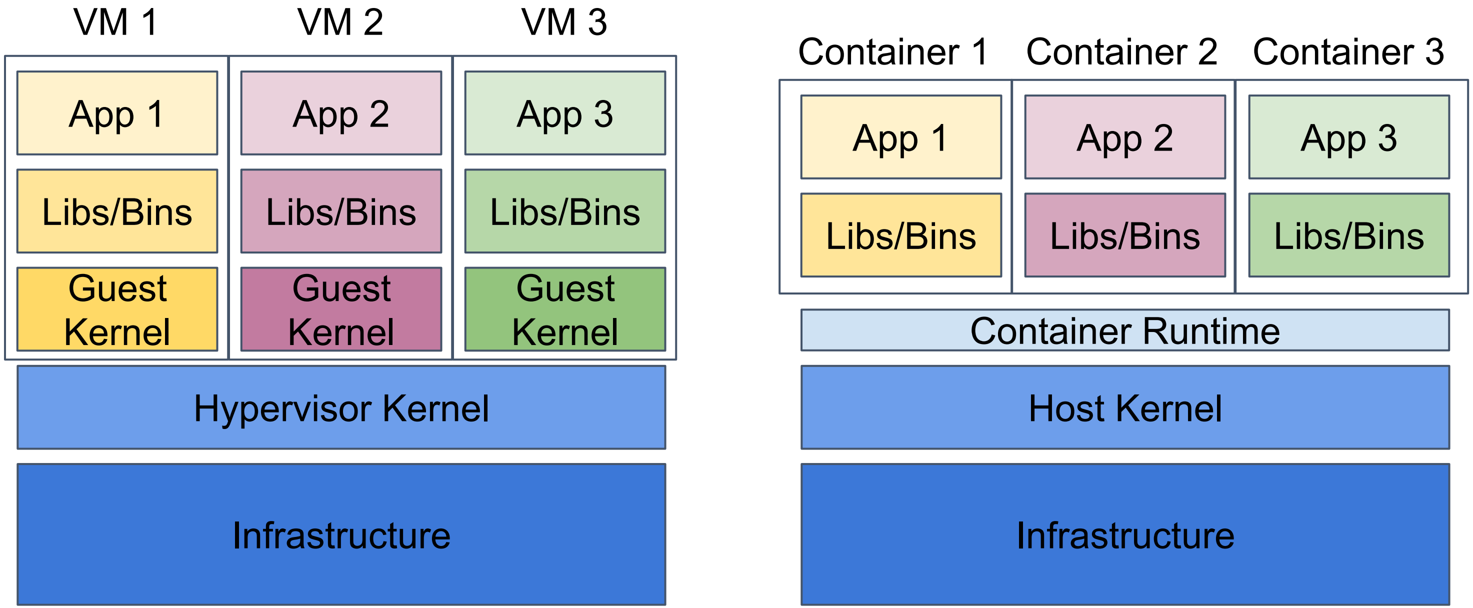
Resource Contention and Security:

- Tasks on a normal OS float between cores and memory space.
- Want to set a cap on usage for multiple tenants.
- Ensure users cannot see other users applications and satck

Types of Containers

- cgroups
- python/conda environment
- Docker-like containers
- Virtual Machine (VM)

VMs or Containers



VMs:
hardware virtualization + OS

Containers:
User defined software stack

Containers: easi"er" software deployment

Containers provide a potential solution…. or at the very least can help.

- Easier software deployment:
  Users can leverage on installation tools that do not need to be available natively on the runtime host
  (e.g. package managers of various linux distributions).
- Software can be built on a platform different from the exec hosts.
- they package in one single object all necessary dependencies.
- easy to publish and sign
- they are portable **
  - … provided you run on a compatible architecture)
  - access to special hardware needs special libraries also inside the container , which at the moment limits portability

Types of Containers

docker

Linux LXC

OpenVZ
Linux Containers

podman

rkt

Charliecloud

SHIFTER

General purpose / microservice Oriented.

HPC Oriented :

- Compatible with WLM
- No privilege escalation needed

Singularity (https://www.sylabs.io)

Singularity provides a container runtime and an ecosystem for managing images that is suitable for multi-tenant systems and HPC environments.
Important aspects :

- no need to have elevated privileges at runtime, although root privileges are needed to build the images.
- each applications will have its own container
- containers are not fully isolated (e.g. host network is available)
- users have the same uid and gid when running an application
- containers can be executed from local image files, or pulling images from a docker registry, a singularity hub or from sylab libraries (see https://cloud.sylabs.io … N.B. service is still in alpha)

For basic usage refer to
https://www.rc.fas.harvard.edu/resources/documentation/software/singularity-on-odyssey/
https://www.sylabs.io/docs/

Example : running from a docker registry

Running tensorflow on a cpu node

```
login-node      ] >$ srun  --pty --mem=4000 -p test -N 1 -t 60  /bin/bash
compute-node ] >$ cd $SCRATCH/your_lab/your_user/
compute-node ] >$ git clone https://github.com/tensorflow/models.git
compute-node ] >$  singularity exec docker://tensorflow/tensorflow:latest-py3 python \
                  ./models/tutorials/image/mnist/convolutional.py
```

Running tensorflow on a gpu node

```
login-node      ] >$ srun  --pty --mem=4000 -p gpu --gres=gpu  -N 1 -t 60  /bin/bash
compute-node ] >$ cd $SCRATCH/your_lab/your_user/
compute-node ] >$ git clone https://github.com/tensorflow/models.git
compute-node ] >$  singularity exec  --nv docker://tensorflow/tensorflow:latest-gpu-py3 python \
                  ./models/tutorials/image/mnist/convolutional.py
```

Example : pulling images from a repositories

pulling from docker

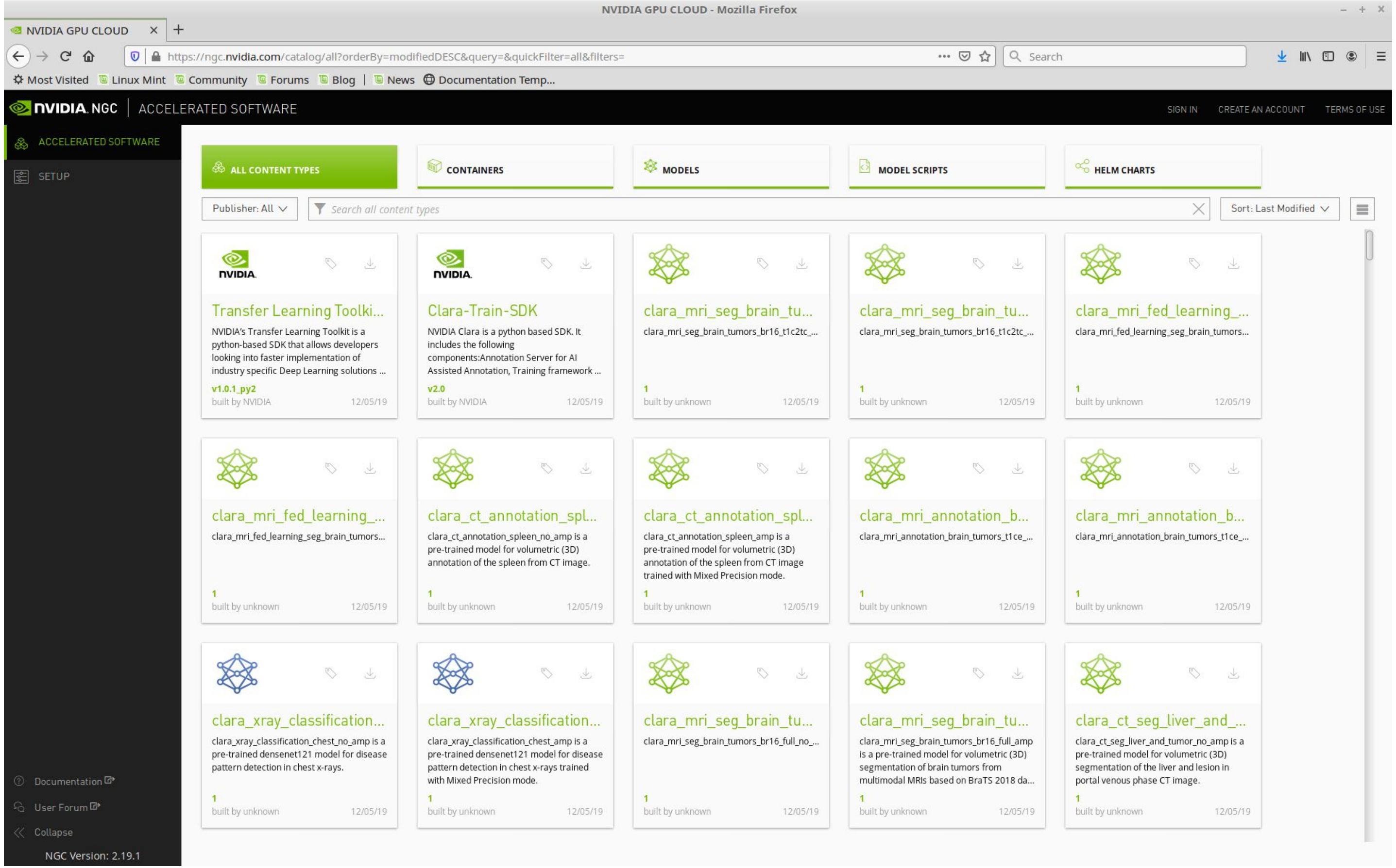
```
login-node      ] >$ srun  --pty --mem=4000 -p test -N 1 -t 60  /bin/bash
compute-node ] >$ cd $SCRATCH/your_lab/your_user/
compute-node ] >$ singularity pull docker://tensorflow/tensorflow:latest-gpu-py3
compute-node ] >$
```

pulling from shub

```
login-node      ] >$ srun  --pty --mem=4000   -p test -N 1 -t 60  /bin/bash
compute-node ] >$ cd $SCRATCH/your_lab/your_user/
compute-node ] >$ singularity pull shub://vsoch/hello-world
```

pulling from library

```
login-node      ] >$ srun  --pty --mem=4000   -p test -N 1 -t 60  /bin/bash
compute-node ] >$ cd $SCRATCH/your_lab/your_user/
compute-node ] >$ singularity pull mylolcow.sif   \
                          library://francesco/examples/lolcow_odyssey:latest
```


NVIDIA GPU CLOUD - Mozilla Firefox
NVIDIA GPU CLOUD
https://ngc.nvidia.com/catalog/all?orderBy=modifiedDESC&query=&quickFilter=all&filters=
Search
Most Visited
Linux Mint
Community
Forums
Blog
News
Documentation Temp...
NVIDIA NGC
ACCELERATED SOFTWARE
SIGN IN
CREATE AN ACCOUNT
TERMS OF USE
ACCELERATED SOFTWARE
SETUP
ALL CONTENT TYPES
CONTAINERS
MODELS
MODEL SCRIPTS
HELM CHARTS
Publisher: All
Search all content types
Sort: Last Modified
Transfer Learning Toolki...
NVIDIA's Transfer Learning Toolkit is a python-based SDK that allows developers looking into faster implementation of industry specific Deep Learning solutions ...
v1.0.1_py2
built by NVIDIA
12/05/19
Clara-Train-SDK
NVIDIA Clara is a python based SDK. It includes the following components:Annotation Server for AI Assisted Annotation, Training framework ...
v2.0
built by NVIDIA
12/05/19
clara_mri_seg_brain_tu...
clara_mri_seg_brain_tumors_br16_t1c2tc_...
1
built by unknown
12/05/19
clara_mri_seg_brain_tu...
clara_mri_seg_brain_tumors_br16_t1c2tc_...
1
built by unknown
12/05/19
clara_mri_fed_learning_...
clara_mri_fed_learning_seg_brain_tumors...
1
built by unknown
12/05/19
clara_mri_fed_learning_...
clara_mri_fed_learning_seg_brain_tumors...
1
built by unknown
12/05/19
clara_ct_annotation_spl...
clara_ct_annotation_spleen_no_amp is a pre-trained model for volumetric (3D) annotation of the spleen from CT image.
1
built by unknown
12/05/19
clara_ct_annotation_spl...
clara_ct_annotation_spleen_amp is a pre-trained model for volumetric (3D) annotation of the spleen from CT image trained with Mixed Precision mode.
1
built by unknown
12/05/19
clara_mri_annotation_b...
clara_mri_annotation_brain_tumors_t1ce_...
1
built by unknown
12/05/19
clara_mri_annotation_b...
clara_mri_annotation_brain_tumors_t1ce_...
1
built by unknown
12/05/19
clara_xray_classification...
clara_xray_classification_chest_no_amp is a pre-trained densenet121 model for disease pattern detection in chest x-rays.
1
built by unknown
12/05/19
clara_xray_classification...
clara_xray_classification_chest_amp is a pre-trained densenet121 model for disease pattern detection in chest x-rays trained with Mixed Precision mode.
1
built by unknown
12/05/19
clara_mri_seg_brain_tu...
clara_mri_seg_brain_tumors_br16_full_no_...
1
built by unknown
12/05/19
clara_mri_seg_brain_tu...
clara_mri_seg_brain_tumors_br16_full_amp is a pre-trained model for volumetric (3D) segmentation of brain tumors from multimodal MRIs based on BraTS 2018 da...
1
built by unknown
12/05/19
clara_ct_seg_liver_and_...
clara_ct_seg_liver_and_tumor_no_amp is a pre-trained model for volumetric (3D) segmentation of the liver and lesion in portal venous phase CT image.
1
built by unknown
12/05/19
Documentation
User Forum
Collapse
NGC Version: 2.19.1

Example : running from a local image

```
## Running matlab

login-node      ] >$ srun  --pty --mem=4000 -p test -N 1 -t 60  /bin/bash
compute-node ] >$ cd $SCRATCH/your_lab/your_user/
compute-node ] >$ myimage=/n/helmod/apps/centos7/Singularity/matlab/matlab-2018-el7-7.4.1708.img
compute-node ] >$  singularity exec $myimage matlab -nodesktop -nosplash
```

RC Portal - Mozilla Firefox

RC Portal

https://portal.rc.fas.harvard.edu/p3/build-reports/

Most Visited | Linux Mint | Community | Forums | Blog | News | Documentation Temp...

Portal

Applications

Search

Search

Select from the available application types

Singularity 3

Cactus

Cactus is a reference-free whole-genome multiple alignment program

cactus 2019-11-29

Please see detailed instructions for the use of this cactus image on the FAS Informatics website.

To activate this build:

```
singularity exec --cleanenv /n/singularity_images/informatics/cactus/cactus:2019-11-29.sif --binariesMode local jobStore "${SEQFILE}" "${OUTPUTHAL}"
```

Cell Ranger ATAC

Cell Ranger ATAC is a set of analysis pipelines that process Chromium Single Cell ATAC data

Running cluster job

```
#!/bin/bash
#SBATCH -n 1
#SBATCH -p test
#SBATCH --mem=4G
#SBATCH -t 1-00:00:00

singularity run my_image.sif
```

or

```
singularity exec my_image.sif mycommand
```

Build your first container

Container images can be built using a file that specifies the recipe.

For example :

```
>$ cat Singularity
BootStrap: debootstrap
OSVersion: trusty
MirrorURL: http://us.archive.ubuntu.com/ubuntu/

%runscript
    echo "This is what happens when you run the container..."

%post
    echo "Hello from inside the container"
    sed -i 's/$/ universe/' /etc/apt/sources.list
    apt-get update
    apt-get -y install vim
    apt-get clean
```

complete description of the def files at

https://www.sylabs.io/guides/3.4/user-guide/definition_files.html

Build your first container

Once I have my singularity file I have three options to build my image:

1. Build Locally

To do this you need to be on your own development environment where you have admin privileges.

```
mycomputer ]> sudo /usr/local/bin/singularity build some_imagename.sif  Singularity.def
```

2. Build remotely

You can do it on Cannon, but you need to have an account on https://cloud.sylabs.io , get a token and store it in ~/.singularity/sylabs-token

```
login-node      ] >$ srun  --pty --mem=4000   -p test -N 1 -t 60  /bin/bash
compute-node ] >$ cd $SCRATCH/your_lab/your_user/
compute-node ] >$ singularity build --remote  some_imagename.sif Singularity.def
```

this will create your def file, build the image and download it to the local folder.

Build your first container (3): Docker

- You can build an image in Docker on your local machine
  - Has the advantage of quicker iteration

- Export to dockerhub or use docker2singularity (https://github.com/singularityhub/docker2singularity)

- Pull image to cluster in singularity, or scp it and use.

Bind Mount

- By default all directories in the Singularity image are read only.
  - **Note:** When building from Docker, sometimes Docker expects something to be writable that may not be in Singularity.

- In addition system directories are not available, only those defined in the Singularity image.

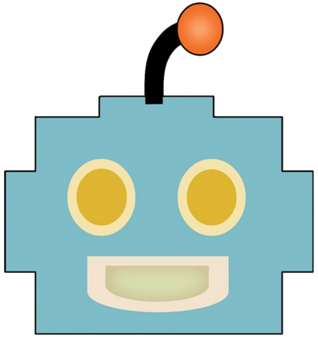
- You can bind external mounts into singularity using the -B option
  - -B hostdir:containerdir
  - -B hostdir <- Maps it to same path inside the container

```
singularity exec -B /scratch:/var/log ls /var/log
```

- On Cannon, we automatically map /n, /net, and /scratch into the image using bind mount.

Request Help - Resources

- https://rc.fas.harvard.edu/resources/support/
  - Documentation
    - https://rc.fas.harvard.edu/resources/documentation/
  - Portal
    - http://portal.rc.fas.harvard.edu/rcrt/submit_ticket
  - Email
    - rchelp@rc.fas.harvard.edu
  - Office Hours
    - Wednesday noon-3pm 38 Oxford - 206
  - Training
    - https://www.rc.fas.harvard.edu/upcoming-training/



- RC Staff are here to help you and your colleagues effectively and efficiently use Cannon resources to expedite your research endeavors.
- Please acknowledge our efforts:
  - "The computations in this paper were run on the FASRC Cannon cluster supported by the FAS Division of Science Research Computing Group at Harvard University."
  - https://rc.fas.harvard.edu/about/attribution/