Containers

Slides from our workshop: Singularity and GitLab CI

Singularity from Syslabs.io is available on the DCC for users who would like to run a containerized environment. Singularity is a secure HPC alternative to Docker and all Docker images can be imported into Singularity. Singularity was designed with high performance computing in mind and can securely run on a HPC cluster and fully support MPI, GPUs, and other specialty high performance computing resources.

Why use containers?

Containers are used to get software to run reliably across multiple computing platforms. They can also be used to simplify software installations across different groups or even members of the same group by packing all dependencies of an application within a single image. Since the entire user space portion of the Linux environment, including programs, custom configurations, and environment, are bundled into a single file, providing the following benefits:

- **portability** - the single container file is easy to transport and can be used on a variety of stand-alone and cluster platforms.
- **reproducibility** - as software dependencies continue to grow, bundling software together with its dependencies, data, scripts, adn documentation help make results easier to reproduce.
- **shareability** - you can host singularity images on web sites and in common repositories to make it easy for colleagues to download and use.
- **usability** - you can bring a singularity container to the DCC and run it without the need for support from a systems administrator to install something for you.

Getting and building Singularity containers

Pre-built Docker and Singularity containers are often available within scientific repositories and on GitHub. Images can be moved to the DCC using the same methods as any other file. Because many images are quite large, we recommend storing them in `/hpc/group`.

Docker containers

Most Docker containers are fully supported with singularity and can be run using Singularity.

Useful Singularity Container repositories

Pre-built containers are also available in external repositories and can be loaded to the DCC using `singularity pull` or `singularity build` commands.

- Sylab's Container Library
- Docker Hub
- Red Hat's Quay
- NVIDIA GPU Cloud

Creating your own Singularity containers

Singularity containers cannot be created on the DCC directly because you need root access to the build system. Generally to build a container, you will need root access to a Linux machine. If you do not have a Linux VM, you can obtain one through Duke Virtual Computing Manager, or lab groups may have access to a RAPID VM through their Faculty/PIs allocation. Helpful documentation:

- Install Singularity on Linux
- Singularity Definition Files
- Instructions to Build a Container

Once you build your container image, you can move it to the Duke Compute Cluster for use.

Using OIT GitLab CI

Users may also create their own Singularity containers by using the OIT Gitlab CI process. Using GitLab CI, your build process for the Linux environment is automated through GitLab by providing a singularity definition file and a .gitlab-ci.yml file. Using this process, the basic steps are:

1. Create a project in GitLab. Each singularity container should have its own project and your Container image will be named after the project.
2. Create a singularity.def file in your project. You can create from scratch or start from a copy from a repository.
3. Copy the .gitlab-ci.yml file from the sample project into your project. Once this file is added, every time you commit to your project, the GitLab Continuous Integration pipeline will be initiated and a singularity image is created for your project.
4. Pull your image down to the DCC from GitLab.

For more details see these sample projects: General use or in PACE , DCC sample project

Running Singularity on the DCC

Singularity is available on the DCC and can be used as part of SLURM interactive or batch jobs. Note: you may need to bind in file system mount points (such as /datacommons) using the -B option.

Sample interactive session

In this session, jmnewton requests an interactive session through SLURM and executes a shell using the singularity image for Detectron2. Using singularity in this way lets you interact with the environment like it is a virtual machine.

```
jmnewton@dcc-login-03  /hpc/group/oit/jmnewton $ srun -p common --pty bash
srun: job 1518869 queued and waiting for resources
srun: job 1518869 has been allocated resources
jmnewton@dcc-core-315  /hpc/group/oit/jmnewton $ singularity shell detectron2.sif
Singularity> cat /etc/os-release
NAME="Ubuntu"
VERSION="18.04.3 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.3 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
```

```
Singularity> exit
exit
jmnewton@dcc-core-315  /hpc/group/oit/jmnewton $ exit
exit
jmnewton@dcc-login-03  /hpc/group/oit/jmnewton $
```

Sample batch session

The sample below illustrates how to use singularity as part of a batch session.

Sample batch script(`slurm_singularity.sh`):

```
#!/bin/bash
# Submit to a random node with a command e.g.
#   sbatch slurm_singularity.sh
#SBATCH --job-name=slurm_singularity
#SBATCH --partition=common
singularity exec /hpc/group/oit/jmnewton/detectron2.sif cat /etc/os-release
```

Submitting the batch script:

```
jmnewton@dcc-login-03  /hpc/group/oit/jmnewton $ sbatch slurm_singularity.sh
Submitted batch job 1518992
jmnewton@dcc-login-03  /hpc/group/oit/jmnewton $ cat slurm-1518992.out
NAME="Ubuntu"
VERSION="18.04.3 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.3 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
jmnewton@dcc-login-03  /hpc/group/oit/jmnewton $
```