Singularity

Singurlaity is a platform that allows containers to run in HPC environments. Docker is the most popular tool for running applications in containers, but as it is designed, putting it into production with the possibility of the users themselves managing the containers is a major security risk. That is why alternatives such as Singularity and others were born.

Singularity has support for using MPI and GPU to run containers and can be integrated into a Slurm script without problems.

Use Singularity at TeideHPC

> **Info**
>
> We currently have version v3.11 available at TeideHPC.

Using the *modules* tool we can load the software:

```
$ module load Singularity/3.11.0
$ singularity -h

Linux container platform optimized for High Performance Computing (HPC) and
Enterprise Performance Computing (EPC)

Usage:
  singularity [global options...]

Description:
  Singularity containers provide an application virtualization layer enabling
  mobility of compute via both application and environment portability. With
  Singularity one is capable of building a root file system that runs on any
  other Linux system where Singularity is installed.

Options:
  -c, --config string   specify a configuration file (for root or
                        unprivileged installation only) (default
                        "/share/easybuild/software/common/software/Singularity/3.11.0/etc/
singularity/singularity.conf")
  -d, --debug           print debugging information (highest verbosity)
  -h, --help            help for singularity
      --nocolor         print without color output (default False)
  -q, --quiet           suppress normal output
  -s, --silent          only print errors
  -v, --verbose         print additional information
      --version         version for singularity
```

```
Available Commands:
  build       Build a Singularity image
  cache       Manage the local cache
  capability  Manage Linux capabilities for users and groups
  completion  Generate the autocompletion script for the specified shell
  config      Manage various singularity configuration (root user only)
  delete      Deletes requested image from the library
  exec        Run a command within a container
  help        Help about any command
  inspect     Show metadata for an image
  instance    Manage containers running as services
  key         Manage OpenPGP keys
  oci         Manage OCI containers
  overlay     Manage an EXT3 writable overlay image
  plugin      Manage Singularity plugins
  pull        Pull an image from a URI
  push        Upload image to the provided URI
  remote      Manage singularity remote endpoints, keyservers and OCI/Docker registry
credentials
  run         Run the user-defined default command within a container
  run-help    Show the user-defined help for an image
  search      Search a Container Library for images
  shell       Run a shell within a container
  sif         Manipulate Singularity Image Format (SIF) images
  sign        Add digital signature(s) to an image
  test        Run the user-defined tests within a container
  verify      Verify digital signature(s) within an image
  version     Show the version for Singularity

Examples:
  $ singularity help <command> [<subcommand>]
  $ singularity help build
  $ singularity help instance start


For additional help or support, please visit https://www.sylabs.io/docs/
```

Docker containers

Singularity uses its own container format, `.sif`, having to transform Docker containers so that they can be used, but this is done by the program itself without the need for user intervention.

Download containers from DockerHUB

> **Info**
>
> When downloading containers, always download them from the login nodes, which are the ones with Internet access, and not from the computation nodes.

In this example we will download an image from DockerHub to be used in the cluster:

```
singularity pull docker://hello-world
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
Getting image source signatures
Copying blob 8a49fdb3b6a5 done
Copying config 689808b082 done
Writing manifest to image destination
Storing signatures
2023/06/01 15:06:41  info unpack layer:
sha256:8a49fdb3b6a5ff2bd8ec6a86c05b2922a0f7454579ecc07637e94dfd1d0639b6
INFO:    Creating SIF file...
```

It will download the container image file to the current directory:

```
hello-world_latest.sif
```

If we want, as we do with docker, we can specify a specific version:

```
singularity pull docker://hello-world:latest
```

Once downloaded, we could try running the container:

> **Warning**
>
> We remind users that it is not possible to run software on login nodes and this includes containers. For this, the `salloc` slurm command is available to request a node interactively and work without problems.

```
singularity run hello-world_latest.sif
INFO:    Converting SIF file to temporary sandbox...
WARNING: passwd file doesn't exist in container, not updating
WARNING: group file doesn't exist in container, not updating

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
```

```
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

INFO:    Cleaning up image...
```

If needed, we also have the `build` command to download docker containers. The main utility of the build command is to be able to create your own containers from existing containers or from a definition file.

```
singularity build tutu.sif docker://hello-world
INFO:    Starting build...
2023/06/01 14:58:33  info unpack layer:
sha256:719385e32844401d57ecfd3eacab360bf551a1491c05b85806ed8f1b08d792f6
INFO:    Creating SIF file...
INFO:    Build complete: tutu.sif
```

And we would execute it in the same way:

```
singularity run tutu.sif
INFO:    Converting SIF file to temporary sandbox...
WARNING: passwd file doesn't exist in container, not updating
WARNING: group file doesn't exist in container, not updating

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

INFO:    Cleaning up image...
```

If for whatever reason our application is not available in DockerHub and we have to build the container from source, we can do it in our local computer, using

docker, and then upload that Docker image to our /home in TeideHPC and create the container with singularity.

Running a container

As we have seen, to run a container with singularity we have the run command:

```
singularity run mycontainer.sif <arg-1> <arg-2> ... <arg-N>
```

But we can also run a container by passing it a command to be executed inside the container and arguments:

```
singularity exec mycontainer.sif <command> <arg-1> <arg-2> ... <arg-N>
singularity exec tutu.sif python3 myscript.py 42
```

We can also work with the container interactively as we do with Docker containers. For this we have the shell command:

```
singularity shell alpine_latest.sif
INFO:    Converting SIF file to temporary sandbox...
Singularity> cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.18.0
PRETTY_NAME="Alpine Linux v3.18"
HOME_URL="https://alpinelinux.org/"
BUG_REPORT_URL="https://gitlab.alpinelinux.org/alpine/aports/-/issues"

Singularity> pwd
/home/vjuidias

Singularity> exit
INFO:    Cleaning up image...
```

As we can see, we can work with the container environment, but we are still in our directory, very useful if we want to work with files without copying them to the container.

Running a container in Slurm

To run singularity in slurm you run it like any other software, by loading the corresponding module and running it:

```
#!/bin/bash -l

# Job name
#SBATCH -J singularity_job
```

```
# Partitiion to run the job
#SBATCH -p batch

# Number of nodes
#SBATCH --nodes=1

# Output files
#SBATCH -o out.log
#SBATCH -e err.log

################################################################

module load Singularity/3.11.0

singularity run $HOME/hello-world_latest.sif
```

Slurm will treat singularity like any other software, i.e. it will apply the same resource constraints, in terms of cpu, memory and time, as the rest of the software.

MPI

> **Tip**
>
> Support for MPI will depend on the software we are going to run, not Singularity. We therefore ask users to read their software documentation carefully before running anything.

In order to use MPI with singularity we have to load the corresponding module and use the srun command:

```
#!/bin/bash -l

# Job name
#SBATCH -J singularity_mpi

# Partitiion to run the job
#SBATCH -p batch

# Number of nodes
#SBATCH --nodes=2

# Output files
#SBATCH -o out.log
#SBATCH -e err.log
################################################################

module purge
module load Singularity/3.11.0
```

```
module load GCC/12.2.0 OpenMPI/4.1.4

srun singularity run $HOME/hello-world_latest.sif
```

GPU

> **Tip**
>
> Support for GPU will depend on the software we are going to run, not Singularity. We therefore ask users to read their software documentation carefully before running anything.

To use GPU to run the software, the `--nv` parameter must be used:

```
singularity run --help
...
    --nv                    enable Nvidia support
```

And for the case of Slurm it would be:

```
#!/bin/bash -l

# Job name
#SBATCH -J singularity_gpu

# Partitiion to run the job
#SBATCH -p gpu

# Number of nodes
##SBATCH --nodes=1

# Number of task
#SBATCH --cpus-per-task=4
#SBATCH --gpus=a100:1

# Output files
#SBATCH -o out.log
#SBATCH -e err.log

##############################################################

module purge
module load Singularity/3.11.0

singularity run --nv $HOME/hello-world_latest.sif
```

Other options

As with docker, we have the `-B` option with which we can *bind a directory on the host machine to a directory in the container:

```
singularity run -B /usr/lib/locale/:/usr/lib/locale -B "${PWD}/input":"/input" mycontainer.sif
<command> <arg-1> <arg-2> ... <arg-N>
```