Singularity

Singurlaity es una plataforma que permite la ejecución de contenedores en entornos HPC. Docker es la herramienta más popular para ejecutar aplicaciones en contenedores, pero tal y como está diseñado, ponerlo en producción con la posibilidad de que sean los propios usuarios quiénes gestionen los contenedores, supone un riesgo de seguridad muy importante. Es por eso que nacieron alternativas como Singularity y otras.

Singularity cuenta con soporte para utilizar MPI y GPU para ejecutar contendores y se pueden integrar en un script de Slurm sin problemas.

Utilizar Singularity en TeideHPC

> **Info**
>
> Actualmente disponemos de la versión v3.11 en TeideHPC.

Utilizando la herramienta de *modules* podemos cargar el 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/
```

Contenedores Docker

Singularity utiliza su propio formato de contenedores, `.sif` , teniendo que transformar los contenedores de Docker para que puedan ser utilizados, pero es algo que hace el propio programa sin necesidad de que el usuario tenga que intervenir.

Descargar un contenedor desde DockerHUB

Info

A la hora de descargar contenedores hay que hacerlo siempre desde los nodos de login que son los que tienen acceso a Internet y no desde los de cómputo.

En este ejemplo descargaremos una imagen desde DockerHub para ser utilizada en el clúster:

```
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...
```

Nos descargará el archivo de la imagen del contenedor en el directorio actual:

```
hello-world_latest.sif
```

Si queremos, al igual que hacemos con docker, podemos especificar una versión en concreto:

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

Una vez descargado, podríamos probar a ejecutar el contenedor:

> **Warning**
>
> Recordamos a los usuarios que no se puede ejecutar software en los nodos de login y esto inluye contendores. Para ello, tienen disponible el comando de slurm `salloc` para solicitar un nodo de manera interactiva y poder trabajar sin problemas.

```
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...
```

En caso de necesitarlo, también disponemos del comando `build` para descargar contenedores de docker. La utilidad principal del comando build está en poder crear nuestros propios contenedores a partir de otros ya existentes o a partir de un fichero de definición.

```
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
```

Y lo ejecutaríamos de la misma manera:

```
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...
```

Si por lo que sea nuestra apliación no está disponible en DockerHub y tenemos que construir el contenedor desde la fuente, lo podemos hacer en nuestro ordenador local, utilizando docker, y luego subir esa imange de Docker a nuestro /home en TeideHPC y crear el contenedor con singularity.

Ejecución de un contenedor

Como se ha visto, para ejecutar un contenedor con singularity tenemos el comando run:

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

Pero también podemos ejecutar un contenedor pasándole un comando que se ejecute dentro de éste y argumentos:

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

También podemos trabajar con el contenedor de manera interactiva al igual que hacemos con los contenedores de Docker. Para esto tenemos el comando shell:

```
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...
```

Como vemos, podemos trabajar con el entorno del contenedor, pero seguimos en nuestro directorio, muy útil si queremos para trabajar con ficheros sin necesidad de copiarlos al contenedor.

Ejecutar un contenedor en Slurm

Para ejecutar singularity en slurm se ejecuta como cualquier otro software, cargando el módulo correspondiente y ejecutándolo:

```
#!/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 tratará singularity como un software más, es decir, que le aplicará las mismas restricciones de recursos, en cuanto a cpu, memoria y tiempo que el resto del software.

MPI

> **Tip**
>
> El soporte para MPI dependerá del software que vayamos a ejecutar, no de Singularity. Por tanto, pedimos a los usuarios que lean la documentación de su software detenidamente antes de ejecutar cualquier cosa.

Para poder utilizar MPI con singularity tenemos que cargar el módulo correspondiente y utilizar el comando de srun:

```
#!/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**
>
> El soporte para GPU dependerá del software que vayamos a ejecutar, no de Singularity. Por tanto, pedimos a los usuarios que lean la documentación de su software detenidamente antes de ejecutar cualquier cosa.

Para utilizar GPU en la ejecución del software hay que utilizar el parámetro `--nv` :

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

Y para el caso de Slurm sería:

```
#!/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
```

Otras opciones

Al igual que con docker, tenemos la opción `-B` con la que podemos hacer un *bind* de un directorio de la máquina host en un directorio del contenedor:

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