

Singularity Containers

eScience 2021

Sept/20/2021

Eric Coulter

Indiana University, XCRI Engineer



Supported by OAC 15-48562.

XSEDE

Singularity Container Runtime Environment

- Designed specifically to work with HPC environments
- Integrates with scheduler, MPI and module systems
- Does NOT allow processes to run as root user (unless started as root)
- Containers generated as simple files in 'SIF' format (Singularity Image Format) - no layering



XSEDE

Singularity User Environment

- Environment variables set by module
- Root required for building images
 - non-root builds can be done, *if* allowed on the system
- Caching in:
 - `${HOME}/.singularity/`
- Configuration available via environment variables



XSEDE

Singularity User Environment

- User storage:
 - Store remote configuration in `${HOME}/.singularity/remotes.yml`
 - ``singularity remote add``
 - Store PGP keys in `${HOME}/.singularity/sypgp`
 - ``singularity key newpair``
 - ``singularity key list``
 - ``singularity key {push,pull,export,import}``



XSEDE

Singularity User Environment

- Cached images in `${HOME}/.singularity/cache`
 - Change via “SINGULARITY_CACHEDIR”
 - Remember ‘-E’ with sudo when building if you change this!
 - ``singularity cache {list,clean}``
- Separate directories for:
 - “library” - docker layers
 - “oci” - singularity cloud images
 - “oci-tmp” - image metadata



XSEDE

Singularity User Environment

- Why is caching important?
 - Default on all systems with Singularity installed
 - Allows easy portability of runscripts and Gateway App definitions!
 - Point to a registry URL with your run/exec commands
 - Singularity checks the hash, compares against the cache, and only updates if necessary
- Image tagging and use of local files can bypass auto-updating if needed



XSEDE

Definition or Recipe Files

- Same concept as Dockerfiles, with some changes
- Multi-stage builds instead of layering
- Based on sections, not commands
 - order is not important for different sections
 - BUT can have multiple copies of each section, will be appended to each other



XSEDE

Definition or Recipe Files

- Header section
 - Defines base container source via ``bootstrap:`` directive
 - Most options require only:
 - ``Bootstrap: $agent_name``
 - ``From: $image_source``
 - Docker, Singularity, and OCI images are supported, in addition to some OS-specific bootstrap agents which rely on OS package managers.



XSEDE

Bootstrap Agents

- `Bootstrap: library`
 - `From: \$entity/\$collection/\$container:\$tag`
 - entity defaults to `Library`
 - `Library: \$library_name` defaults to <https://cloud.sylabs.io>
 - Customizable to other endpoints!



XSEDE

Bootstrap Agents

- `Bootstrap: docker`
 - `From:\$registry/\$namespace/\$container:\$tag@\$digest`
 - \$registry defaults to <https://index.docker.io> (customizable via keyword `Registry: `)
 - \$namespace defaults to `library` (customizable via keyword `Namespace: `)
 - Triggers automatic conversion from a docker image, which may not be successful for all images!
 - Base OS images are generally safe.



XSEDE

Definition File Sections

- ``%setup``

- Dangerous, takes actions as ``root`` on the build host - AVOID

```
%setup
touch /file1
touch ${SINGULARITY_ROOTFS}/file2
```

- ``%files``

- Safe way to copy files into the container FS - use!
- Does not require all files to exist under the current directory (convenient)

```
%files
/file1
/file1 /opt
```



XSEDE

Definition File Sections

- `%post``
 - Install dependencies, download files, change config files, create directories, etc.
- `%test``
 - Optional, use to validate your build with custom tests

```
%post
apt-get update && apt-get install -y netcat
NOW=`date`
echo "export NOW=\"${NOW}\"" >> $SINGULARITY_ENVIRONMENT
```

```
%test
grep -q NAME=\"Ubuntu\" /etc/os-release
if [ $? -eq 0 ]; then
    echo "Container base is Ubuntu as expected."
else
    echo "Container base is not Ubuntu."
fi
```



XSEDE

Definition File Sections

- ``%environment``
 - Provide variables to the container at RUNTIME - not available during build (not defined in `%post`)
- ``%runscript``
 - Default commands to execute when 'singularity run \$container' is invoked
 - ``%startscript`` is the service equivalent

```
%environment
export LISTEN_PORT=12345
export LC_ALL=C
```

```
%runscript
echo "Container was created $NOW"
echo "Arguments received: $*"
exec echo "$@"
```



XSEDE

Definition File Sections

- ``%help``
 - Provide help to your users - provide text here describing your container, available via ``singularity run-help $container``
- ``%app``
 - Allows for packaging multiple apps in separate sections
 - https://sylabs.io/guides/3.4/user-guide/definition_files.html#apps
 - Separate `%post`, `%environment`, `%runscript` sections for your apps



XSEDE

Definition File Sections

- ``%labels``
 - key-value metadata (delimited on 1st space)

`%labels`

```
Author d@sylabs.io  
Version v0.0.1  
MyLabel Hello World
```



XSEDE

Types of Images

- Default format is immutable, in the Singularity Image File (SIF) format
- Writable “sandbox” for development, testing - NOT Reproducible
 - ``singularity build --sandbox containername/ library://container-image``
 - Creates a local ``containername`` directory
 - ``singularity shell containername``
 - Typical ``singularity {exec,run} containername/`` commands as well
 - convert to SIF via ``singularity build sif-name containername``



XSEDE

Running Services

- It's also possible to run “instances” of Singularity containers
- ``singularity instance {start,stop,list}``
- Additional ``%startscript`` section available for service-oriented containers



XSEDE

Bind Mounting

- Defaults:
 - \$HOME, /sys:/sys , /proc:/proc, /tmp:/tmp, /var/tmp:/var/tmp, /etc/resolv.conf:/etc/resolv.conf, /etc/passwd:/etc/passwd, and \$PWD.
- User-configurable:
 - ``singularity run --bind /mnt/data:/usr/local/data $container``
 - ``singularity exec --bind /opt,/data:/usr/local/data $container``
 - Without ':', mounts to same location in container environment
- Not often necessary!



XSEDE

MPI

- Hybrid model
 - Use the host mpi version: ``mpirun singularity run ...``
 - Requires version compatibility between internal and external MPIS!
 - Internal MPI must be configured for the hardware if performance is critical
- Bind model
 - Bind-mount local MPI implementation at runtime
 - Same caveats apply, requires access to build host with compatible MPI



XSEDE

Up next:

Hands-on Exercise: Parts C & D



XSEDE