
Exercise Introduction

This is the exercise on using Apptainer (`https://apptainer.org`) from the Singularity/Apptainer ecosystem.

Contents

Task 1: Cleaning up the cache (3 min)

While Singularity/Apptainer do not have a local container image storage in the same way that Podman/Docker do, they still do cache downloaded container images for faster re-use. This can cause the disk space to steadily increase over time. This can be cleaned by Apptainer's `cache clean` command. Quickly look at its documentation and determine how you would wipe anything more than a year old.

Further Reading

- `apptainer cache clean` – `https://apptainer.org/docs/user/main/cli/apptainer_cache_clean.html`

Task 2: Reading and writing files (7 min)

Singularity/Apptainer containers are very different than Docker/OCI in that they are by default read only while also by default mount some external directories inside that one can read and write files in.

`$HOME` and `/tmp`

Take one of the SIF containers you made before and launch a shell inside it. Go to your `$HOME` directory inside it. What files and directories do you see? Create or edit a file you see. Exit the container. Does the file you created there or the file you editted have the same changes?

Do the same for `/tmp`, but make a file rather than edit any files there (editting files in `/tmp` can cause various consequences for other programs).

Further Reading

- `apptainer shell` – `https://apptainer.org/docs/user/main/cli/apptainer_shell.html`

Trying to write inside a SIF file

Now, you will try modifying the contents of the container itself rather than stuff in its mounts. Launch a shell in one of your SIF containers and go to the `/` directory. Try to make a file/directory, remove a file/directory, or edit a file. What happened?

Try launching the shell again, but using the `--writable` option. Did that do any better?

Writable overlay

Both of those failed. The first failed because Singularity/Apptainer containers are by default readonly. But the `--writable` flag failed since squashfs images are always readonly, unlike say a sandbox directory.

To solve this, you can use an overlay. An overlay is where two mounts are used, the bottom readonly one (in this case, the SIF image) and the top one where all modifications are written to. The overlay system handles the reads and writes transparently so that programs don't need to care about low level details.

The simplest is a temporary overlay, which is lost once the container is exited. This is done with a `tmpfs` filesystem. Try launching a shell in your SIF container again, but add the `--writable-tmpfs` option. Try to modify something in the `/` directory, which should succeed. Exit the shell. Relaunch the container again. Your previous modification should not be present, since you made a change in a temporary overlay.

But what if you want persistent changes. Then, you would use a persistent overlay. This can either be done with a directory or an `ext3` filesystem image. The latter can be created with the Apptainer's `overlay create` command, but we won't do that here. When using Apptainer's `exec`, `run`, or `shell` commands, you use the `--overlay OVERLAY` option where `OVERLAY` is the directory to use or the overlay image file made by Apptainer's `overlay create` command.

Make your `OVERLAY` directory and launch a shell in your SIF container again, but add the `--overlay OVERLAY` option. Make some change in the `/` directory, which will succeed. Exit the shell. Relaunch the shell using the same container and overlay. Is your change still there? Exit the shell and relaunch without the overlay. The change should not be present.

Further Reading

- Persistent Overlays – `https://apptainer.org/docs/user/main/persistent_overlays.html`

Task 3: Changing default mounts (7 min)

In an HPC environment, it is common that various shared filesystems are provided for users to store their data, results, temporaries etc. beyond just their `$HOME` directory. These often have names like `/scratch`, `/data`, `/archive`, etc.

Make just such a directory in your VM and make sure you have permissions to enter the directory and either make changes or have a subdirectory where you can (e.g. `/data/$USER`). Go into Apptainer's main configuration file `/etc/apptainer/apptainer.conf` and make this directory (the parent, not the `$USER` part) one of the default bind mounts. Then, launch one a shell in one of your SIF containers and check that you can access that directory inside the container.

How would you make it so that `$HOME` is not mounted by default? Make it so it would not be mounted by default, and then test that you succeeded. Then change the setting back.

Further Reading

- `/etc/apptainer/apptainer.conf` documentation – `https://apptainer.org/docs/admin/main/configfiles.html#apptainer-conf`
- `chmod` – `https://www.man7.org/linux/man-pages/man1/chmod.1.html`
- `chown` – `https://www.man7.org/linux/man-pages/man1/chown.1.html`

Task 4: Locking down a setUID build (8 min)

Now, imagine for a moment that you had installed a setUID build of Apptainer, say with the `apptainer-suid` package or built it yourself that way from source. Then the setUID helper program it includes runs as root even when Apptainer is run by an unprivileged user. The setUID part of Apptainer (just like SingularityCE/PRO and old Singularity) does a number of checks and has some safety mechanisms to reduce the danger. Some of these are exposed in the main configuration file `/etc/apptainer/apptainer.conf`.

One way to improve safety is to only allow a select group of users to use a setUID build, say users who convinced the admins that they need it and that they are careful and trustworthy enough to be granted access (if you don't trust the design and programming quality of the Apptainer team). One way to do this would be to make a group that users who are trusted are added to. How would you change `/etc/apptainer/apptainer.conf` so that only members of that group are allowed to use setUID Apptainer? Note, it is also possible to do the same thing, but probably more reliably, using selinux and other tools to restrict who can even run the Apptainer executables.

Another potential lockdown method would be to use a curated list of SIF containers that users are allowed to use that have been checked and verified in some way (or even possibly made) by the admins. If, suppose, the admins put all of these containers in the `/software/sif_containers` directory, how would you change `/etc/apptainer/apptainer.conf` so that only those containers can be used?

Further Reading

- `/etc/apptainer/apptainer.conf` documentation – `https://apptainer.org/docs/admin/main/configfiles.html#apptainer-conf`