



Installing containerized software on Alpine

Installing containerized software on Alpine

Date: February 27, 2024

Instructor: Andrew Monaghan

Contributors: Layla Freeborn, Trevor Hall, Brandon Reyes

- Website: www.rc.colorado.edu/rc
- Documentation: <https://curc.readthedocs.io>
- Helpdesk: rc-help@colorado.edu
- Survey: <http://tinyurl.com/curc-survey18>

Slides

https://github.com/ResearchComputing/containerized_software_alpine_primer



Learning Objectives

- What is a software container?
- How can I find existing software containers?
- How can I use a container on Alpine?
- How can I build a container on Alpine?

Session Overview

Introduction

- Installing software on CURC systems
- Defining containers
- Description of Apptainer

Running containers

- Logging into CURC
- Executing our first container!
- Container commands
- Shelling into containers

Building containers

- Container definition file
- Building our first container
- Pulling a pre-existing containers

Complex topics (if time allows)



Building Software on Alpine

- There are numerous ways to install software on Alpine:
 - grab pre-compiled binaries
 - compile from source
 - within virtual environments (via Conda, Miniconda, or Mamba)
 - **using containers (Apptainer)**
 - using a package manager for HPC systems (Spack)



Additional information:

<https://github.com/ResearchComputing/research-software-curc>

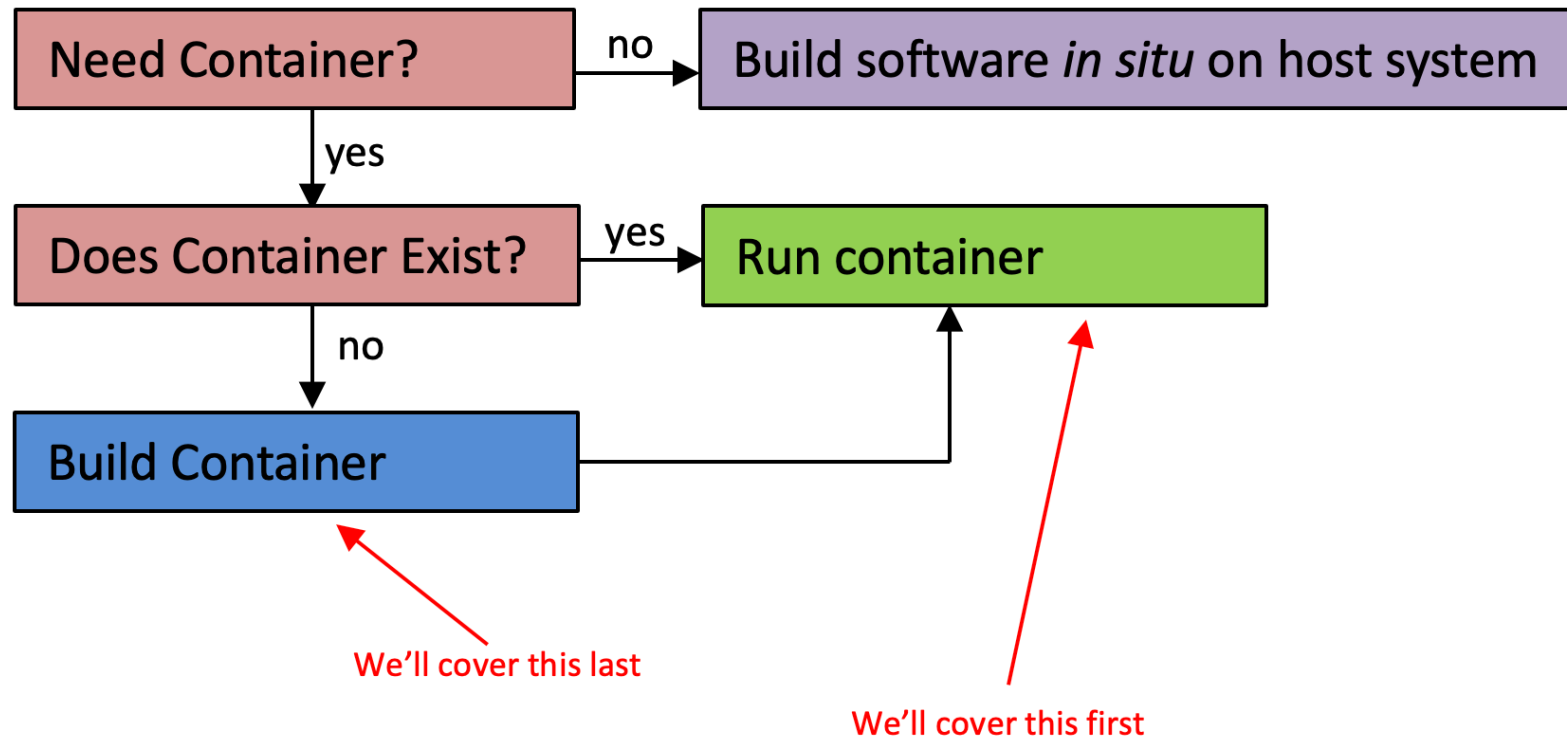
What is a Container?

- A container is a portable environment that packages some or all of the following: an operating system, software, libraries, compilers, data, and workflows. Containers enable:
 - Mobility of compute
 - Reproducibility (software and data)
 - User Freedom

Containerization Software

- Docker 
 - Well-established
 - Most widely used
 - Millions of containers already available on [DockerHub](https://hub.docker.com/)
- **Apptainer** 
 - Formerly “Singularity”
 - HPC-safe
- Others

Making the decision to containerize



Logging into CU Research Computing

login to CURC via your terminal:

```
ssh monaghaa@login.rc.colorado.edu
```

...or login to CURC via your browser:

<https://ondemand-rmacc.rc.colorado.edu>

(once logged in, navigate to **Clusters** -> **Alpine shell**)

Additional information:

<https://curc.readthedocs.io/en/latest/access/logging-in.html>

<https://curc.readthedocs.io/en/latest/gateways/OnDemand.html>

Start a compute session

Start a session on an Alpine compute node with **acompile**:

```
[monaghaa@login11 ~]$ acompile --help
[monaghaa@login11 ~]$ acompile --ntasks=4 --time=90:00
...
[monaghaa@c3cpu-a5-u28-1 ~]$ apptainer --help
```

Note: when you login to CURC you'll be on a **login** node. You'll need to be on a **compute** node to use apptainer. The **acompile** command allows you to quickly start an interactive job on a compute node.

Additional information:

<https://curc.readthedocs.io/en/latest/clusters/alpine/alpine-hardware.html#partitions>

Let's run our first container!

Invoke “R” statistical software in a pre-existing container:

```
[monaghaa@c3cpu-a5-u28-1 ~]$ apptainer exec $CURC_CONTAINER_DIR/debian_r_4_2_2.sif R  
> 1+2  
> quit ()
```

```
[monaghaa@c3cpu-a5-u28-1 ~]$ ls $CURC_CONTAINER_DIR/    # ← what the heck is in this directory?
```

Let's run an external R script with the container

```
[monaghaa@c3cpu-a5-u28-1 ~]$ cd /scratch/alpine/$USER  
[monaghaa@c3cpu-a5-u28-1 ~]$ git clone https://github.com/ResearchComputing/containerized\_software\_alpine\_primer  
[monaghaa@c3cpu-a5-u28-1 ~]$ cd containerized_software_alpine_primer/examples  
[monaghaa@c3cpu-a5-u28-1 ~]$ apptainer exec $CURC_CONTAINER_DIR/debian_r_4_2_2.sif Rscript prime_functions.R
```

Key Apptainer Commands

[exec](#): Execute a command to your container

[run](#): Run your image as an executable (behavior must be predefined)

[build](#): Build a container on your user endpoint or build environment

[pull](#): pull an image from Docker or Singularity Hub

[inspect](#): See labels, run and test scripts, and environment variables

[shell](#): Shell into your image

Now let's explore a container

Shell into the container:

```
[monaghaa@c3cpu-a5-u28-1 ~]$ apptainer shell $CURC_CONTAINER_DIR/gmtsar_v6.2.sif
Apptainer> cat /etc/os-release # what operating system is in the container?
Apptainer> ls /usr/local # what do you see?
Apptainer> exit
```

Inspect how the container was built:

```
[monaghaa@c3cpu-a5-u28-1 ~]$ apptainer inspect -d $CURC_CONTAINER_DIR/gmtsar_v6.2.sif
```

...now let's look more closely at a container definition file

Container definition file

lolcow.def

```
Bootstrap: docker
From: ubuntu:20.04

%post
    apt-get -y update
    apt-get -y install cowsay lolcat

%environment
    export LC_ALL=C
    export PATH=/usr/games:$PATH

%runscript
    date | cowsay | lolcat
```

Example source: https://apptainer.org/docs/user/latest/definition_files.html



Now let's build and run a container

```
[monaghaa@c3cpu-a5-u28-1 ~]$ apptainer build lolcow.sif lolcow.def
...will take a couple of minutes

# run default container behavior
[monaghaa@c3cpu-a5-u28-1 ~]$ apptainer run lolcow.sif

# run your own script from container
[monaghaa@c3cpu-a5-u28-1 ~]$ apptainer exec lolcow.sif /bin/sh lolcow.sh
```

Now let's pull and run a Docker container

```
[monaghaa@c3cpu-a5-u28-1 ~]$ apptainer pull cp2k.sif docker://cp2k/cp2k:latest
...will take a couple of minutes

# run at command line:
[monaghaa@c3cpu-a5-u28-1 ~]$ apptainer run cp2k.sif cp2k -o md300.out md300.inp

# schedule a job
[monaghaa@c3cpu-a5-u28-1 ~]$ sbatch run_cp2k.sh
```

More complex topics (if time allows)

- Bind mounting external directories
- Using containers on GPUs
- Using containers with MPI

Thank you!

Survey and feedback

<http://tinyurl.com/curc-survey18>

