Containerization for R&D Applications (Part 1!)

Daniel Trahan
Daniel.Trahan@colorado.edu

Slides and exercises available for download at:
https://github.com/ResearchComputing/Containers-Spring-2021

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Outline

Part 1: Container fundamentals and Docker (2/23/21)

- Introduction to containers
- Docker commands and options
- Hands-on: Running Docker containers on your personal Machine
- Hands-on: Building Docker images
- Hands-on: Practical Application

Part 2: Containers for HPC with Singularity (3/2/21)

- Singularity commands and options
- Hands-on: Running containers
- Building containers
- Special cases: Running containers for MPI and GPU jobs

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Introduction to Containers

Seacontainersales.com

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

What is a container?

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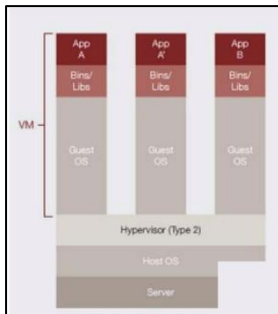
- Mobility of Compute
- Reproducibility (software and data)
- User Freedom

Virtualization (1)

Hardware virtualization (not used by containers!)

- Can run many OS's on same hardware (machine)
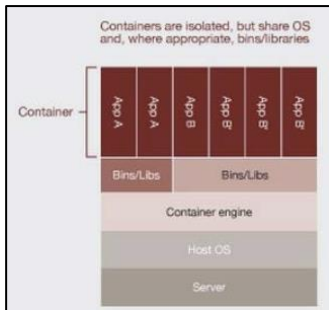- E.g., VirtualBox, VMWare

Material courtesy: M. Cuma, U. Utah

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Virtualization (2)

OS-level virtualization (used by containers!)

- Can run many isolated OS instances (guests) under a server OS (host)
- Also called containers
- E.g., Docker, Singularity

Best of both worlds: isolated environment that user wants, but can leverage host OS resources (network, I/O partitions, etc.)

Containers are isolated, but share OS and, where appropriate, bins/libraries

Container — | App A | App A | App B | App B | App B | App B |

Bins/Libs | Bins/Libs

Container engine

Host OS

Server

Material courtesy: M. Cuma, U. Utah

Containerization software

- Docker
    - Part 1 focus of today's tutorial
    - Well established – largest user base
    - Has Docker Hub for container sharing
    - Problematic with HPC
- Singularity
    - Part 2 focus of next week tutorial
    - Designed for HPC
- Charliecloud; Shifter
    - Designed for HPC
    - Based on Docker
    - Less user-friendly

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Installing Docker

- Docker Community Edition
    - Windows: Windows 10 Professional or Enterprise
    - Mac: OS X El Capitan 10.11 or later
    - Linux
- Docker toolbox
    - Legacy solution for Windows and Mac for versions that do not meet the version requirements.
    - Utilizes the Virtual Box hypervisor for virtualization
    - For this tutorial, Windows users regardless of version will use Docker toolbox

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Why Docker?

- Probably the most popular containerization software
- Offers a variety of prebuilt images including:
  - Python
  - Perl
  - NodeJS
  - Ubuntu
- Very well documented with a large community creating and supporting docker images.
- DockerHub

Docker Nuts and Bolts

- Docker runs on a concept of images and containers.
  - ***Images***: Saved snapshots of a container environment.
    - Made from Dockerfile or pulled from Docker Hub
    - Stored in the Docker cache on your disk
  - ***Containers***: Instances of images that are generated by Docker when an image is 'run'
    - Instance of image running in memory
    - Ephemeral Instances that cannot be continued
    - Can be run interactively

Docker Commands

- Docker Commands are usually in the form of:

  `docker <sub-command> <flags> <target/command>`

- Examples:

  `docker run -it myimage`

  `docker container ls`

  `docker image prune`

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Running Docker Containers

- Run a docker image as a container:

  ```
  docker run <image-name>
  ```

- Run a docker image interactively:

  ```
  docker run -it <image-name>
  ```

- If an image is not on the system, then Docker will search Dockerhub to see if the image exists.

- Specify commands after your image to execute specific software in your container.

  ```
  docker run <image-name> <program>
  ```

11/12/20 Containers

Be Boulder.

Containerized Hello World

- Let's start with something simple:
  - Docker "Hello, World!"
    - Relatively small image
    - No dependencies
    - Built as a general test case
- Command:

```
docker run hello-world
```

Docker Image and Container Commands

Image Commands	
`docker image ls`	List docker images stored in cache:
`docker image rm <image>` `docker rmi <image>`	Remove (an) image(s):
`docker image prune`	Remove unused images
Container Commands	
`docker container ls`	List docker containers currently running:
`docker container rm <container>` `docker rm <container>`	Remove (an) container(s):
`docker container prune`	Remove all stopped containers

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Docker Utility Commands

Commands	
`docker info`	Shows Docker system-wide information
`docker inspect <docker-object>`	Shows low-level information about an object
`docker config <sub-command>`	Manage docker configurations
`docker stats <container>`	Shows container resource usage
`docker top <container>`	Shows running processes of a container
`docker version`	Shows docker version information

- More details and commands can be found _on the docker documentation page_

Demo 1: Running Containerized Python

Be Boulder.

Demo 1: Python

- Running the Python docker container will pull Python from Docker Hub:

  ```
  docker run python:3.7.2-slim
  ```

- ...did it work?
- Run your python image interactively:

  ```
  docker run -it python:3.7.2-slim
  ```

- This puts us into a python interpreter, where you can run python code containerized in its own environment.

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Building Docker Containers

- To build a docker container, we need a set of instructions Docker can use to set up the environment.
  - Dockerfile
- Once we set up our dockerfile we can use the command

  `docker build -t <image-name> .`
- Then we can run the image with our `docker run` command

  `docker run <image-name>`

11/12/20 Containers

Be Boulder.

Demo 2: Building an Ubuntu Container

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Demo 2: Setup

Dockerfiles and test files are provided for this workshop. We can pull the files from a github repository as such:

1. Navigate to your home directory

   ```
   cd ~
   ```

2. Clone the repository

   ```
   git clone https://github.com/ResearchComputing/Containers-Spring-2021
   ```

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Demo 2: Ubuntu w/ GCC

- For this first example we will be building a custom Ubuntu image that will provide a location to run the GNU Compiler Collection.

- Dockerfile provided

- Need to build:

  1. Navigate to the directory:

     `cd ~/Containers_Spring_2021/dockerdemo/ubuntu-gcc`

  2. Build the docker image with:
     `docker build –t happy-gcc .`

  3. Run the docker image as a container:
     `docker run –it happy-gcc`

Editing Docker Images

- Suppose you have an existing docker image and want to make changes…
  - Rebuild Dockerfile!
    - Usually a bit cumbersome
    - No Dockerfile?
- Use docker commit!

```
docker run -it <image-name> bash #or any shell...
```

- Then commit it to the image

```
docker commit <image-id> <image>
```

Mounting and Accessing files

- So now that we have a working container, how can we access the test files we downloaded?
  - Mounting directories: Bind Mount
    - Allows the docker container to access files on the host OS
    - Choose host's **source directory**, files in the directory will be moved to the container's **target directory**
      - ***Source Directory***: Directory on the host system. Never within a container.
      - ***Target Directory***: Directory in the Docker Container. Never on the host system.
    - A flag set within the `docker run` command:

```
docker run --mount type=bind,source=<source>,target=<target> <image>
```

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Mounting and Accessing files

- Mounting directories: Volume Mount
  - Same concept, but volumes are stored within docker cache.
  - Create Docker volumes in your terminal and link your volume directory
  - Similarly linked through the `docker run` command.

```
docker run --mount type=volume,source=<volume>,target=<target> <image>
```

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

Be Boulder.

Demo 2 (Cont.): Mounting

- Returning to our demo, can we give our container access to our test files?
- Let's use a bind mount!
- In the directory where our Dockerfile lives... use this command (all on one line):

```
docker run –it --mount
type=bind,source=$(pwd)/source,target=/target happy-gcc
```

- We can `cd /target` and run our test files!
- Command:

```
gcc hello.c -o hello.exe
./hello.exe
```

Demo 3: NCL container

- For this next example we will building a Docker image that will run the NCAR Command Language (NCL).
- Dockerfile provided
- Same process as before:
    1. Navigate to the Dockerfile found at:
        ~/ Containers_Spring_2021/dockerdemo/ncl
    2. Build your docker file as an image titled "bright-ncl"
    3. Run your docker image as a container
- Can we test a sample script?

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Dockerhub

- The place where containers live!
- Dockerhub is a Docker hosted library of public and private Docker images.
    - Free and unlimited public images
    - 1 free private repository
- Great for hosting images for fellow researchers
- Commands similar to git

Dockerhub Commands

- Download and upload docker images with ease.
    - docker run <image>
    - docker pull <image>
- Uploading a little more complicated...
    - Sign in with:
      docker login
    - List docker images with:
      docker image ls
    - Tag your image:
      docker tag <image-id> <your-username>/<image-name>:<tag>
    - Push!
      docker push <your-username>/<image-name>

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Docker-Compose

- Utility that can create and install docker images.
- Builds docker images based on a docker-compose.yml file.
  - YAML: *YAML Ain't Markup Language*
  - Data serialization language
  - Describes containers you wish to build with what features.
- Not a docker command!

Docker-Compose Commands

- Build all containers in YAML file:
  docker-compose build
- Build and run all containers in YAML file:
  docker-compose up
- List all containers in YAML file:
  docker-compose images
- Run a one-off command from a container:
  docker-compose run <container-name> <command>

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

11/12/20 Containers

Be Boulder.

Thank you!

Please fill out the survey:
http://tinyurl.com/curc-survey18

Contact information:
Daniel.Trahan@Colorado.edu

Additional learning resources:

Slides and Examples from this course:
https://github.com/ResearchComputing/CONTAINER_TUTORIAL_FALL_2020

Web resources:
https://training.play-with-docker.com (docker online training materials)
https://hub.docker.com (Docker Hub)

Be Boulder.