Athey Lab Tech Hour - Containers

Zhaolei (Henry) Shi

Stanford University

February 19, 2019

Overview

Tech Hour

Motivation

Background
- What is a Container?
- History of Container Technology
- Containers vs. Virtual Machines
- Docker and Singularity

Using Containers

Setting Up a Singularity Container

References

Tech Hour

Goals

- Get Athey Lab to move together in the technology space.
- Soften learning curve for new tools.

Operation

- Every two weeks
- Presentation + Demo
- Tuesday 4:30 PM
- Topics and speakers wanted.

Motivation

Branch: **master** ▾ **atheylab.utils** / R / **utils.R**

cwalker4 on_the_yens fix for new yens

2 contributors

98 lines (85 sloc) | 2.13 KB

```r
#' Running on the yens?
#'
#' returns TRUE if called on one of the yens
#'
#' @export
#
on_the_yens <- function() {
  grepl('^yen[0-9]', Sys.info()['nodename'])
}

```

Figure 1: Portability and reproducibility are the primary motivations for using containers

Motivation

You've been working on a project on your laptop. You need to produce a new table which demands more RAM than the 8GB you have on your laptop. To overcome this, you decide to use Stanford's excellent Sherlock clusters to run your code. More likely than not, you would need to transfer these things:

- Paths
- R/Python package package dependencies
- OS-level libraries (LaTex-related, gdal, GNU scientific libraries)
- Configuration files (.gitconfig, .vimrc, setup files for cloud services such as AWS Redshift)

Containers solves these problems, and helps to get at:

- Unified clusters and local work flow
- Push-button reproducibility (picking up old projects)
- Better code review practices

What is a Container?

Containers are operating-system-level virtualizations, meaning that different containers on the host computer share the host OS kernel space but keeps its own user space. Kernel space is runs privileged operating system kernel, kernel extensions, and most device drivers. User space executes applications.

Practically, users within a container “feel” like they are using an isolated computer (paths, applications etc.).

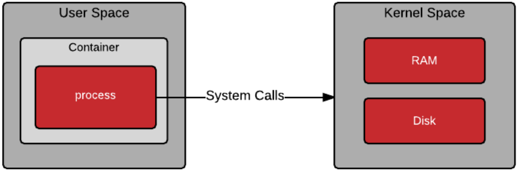

Figure 2: Containers live in the user space (Source `https://www.backblaze.com/blog/vm-vs-containers/`)

History of Container Technology

- Container technology date back to 1979. Unix V7 implemented the the `chroot` system call, creating early process isolation.
- `chroot` simply changes the apparent root directory forA chroot on Unix operating systems is an operation that changes the apparent root directory for the current running process and its children.
- In 1991, Cheswick (1991) of AT&T labs made a container to study a malicious intruder as he/she attempt to hack their system.
- Docker was released as an open-source project in 2013. Docker allowed users to package containers so that they could be moved around. Docker Hub was borne.
- Kubernetes was released in 2014 to automate container deployment.
- Singularity was released in 2015 to bring Docker-like capabilities to High Performance Computing.

Docker Hub

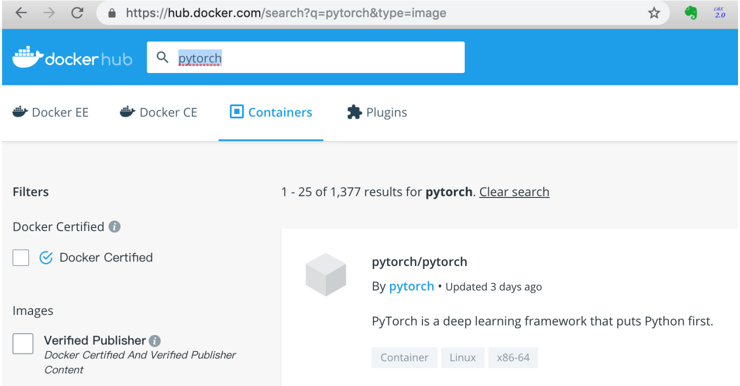

Figure 3: Docker Hub hosts ready-to-use containers

Containers vs. Virtual Machines

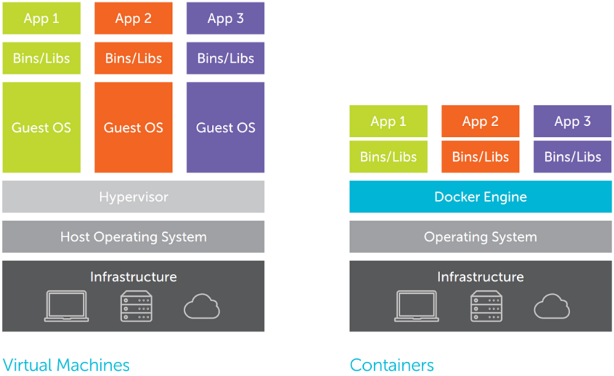

Figure 4: Containers vs. Virtual Machines (Source: https://www.softserveinc.com/en-us/blogs/security-containers-vs-virtual-machines/)

Containers generally have less overhead than virtual machines, see Zhang et al. (2018) for a comparison.

Docker and Singularity

Singularity can be thought of as Docker for shared clusters.

- Docker gives superuser privileges, Singularity does not.
- Singularity is designed to work within the resource allocated by schedulers.
- Docker images can be used in Singularity out of the box.

Singularity on Sherlock and Yen

- Sherlock (Stanford-wide high performance computing cluster) supports Singularity: `https://www.sherlock.stanford.edu/docs/software/using/singularity/#why-singularity`
- The new Yen (GSB cluster) does not have singularity yet, but will likely have it in the future (I'll be writing to them).

Recipe

Singularity (Docker) can build images automatically by reading the instructions from a Recipe (Dockerfile). The recipe is a text document that contains all the commands a user could call on the command line to assemble an image.

```
Bootstrap: docker
From: r-base:latest

%help

    Athey Lab Tech Hour Demo 2019-02-19

%setup

%files

%post
    # manually installing the libsodium library
    cd ~
    wget "https://download.libsodium.org/libsodium/releases/LATEST.tar.gz"
    tar -xzvf LATEST.tar.gz
    cd libsodium*
    ./configure
    make && make check
    make install
    ldconfig
    echo "check if libsodium is linked"
    ldconfig -p | grep libsodium
    cd /
    rm ~/LATEST.tar.gz
    rm -r ~/libsodium*

    # R
    apt-get install -y --allow-unauthenticated r-base-core  r-recommended
    R --slave -e 'install.packages("sodium", dependencies = T, repos="https://cloud.r-project.org/")'

    mkdir /atheylab_demo

%environment

```

Figure 5: Singularity recipe sample

Toward a Container-Based Work Flow

- Mostly run code on Sherlock/Yen using shared Singularity containers.
- Use a Docker version of the container for local debugging.
- Shared folders for inputs, intermediates, and outputs on Sherlock/Yen. Keep shell scripts to bind these directories to the container.
- Keep recipe as part of Git repository.
- If additional packages are needed, install in shared container directly and commit changes to recipe.

Install Singularity

We will follow the instructions here:
`https://gsbdbi.github.io/ra_manual/main/container.html`

Making a Docker Equivalent

- On a mac, using Docker containers on your local computer has less overhead than running a container with Singularity which runs in a Linux virtual machine.
- Singularity Recipe to Dockerfile translation is fairly straight forward.
- The same trade off may not apply to Windows machines.

Quick Tips (Demo)

- Mounting folders
- Working with LaTex
- Back compatibility problems with Singularity.
- Modifiable containers.
- Port forwarding in Singularity.

References I

Cheswick, B. (1991). An evening with berferd in which a cracker is lured, endured, and studied.

Zhang, Q., Liu, L., Pu, C., Dou, Q., Wu, L., and Zhou, W. (2018). A Comparative Study of Containers and Virtual Machines in Big Data Environment. *arXiv e-prints*, page arXiv:1807.01842.