Singularity: What, Why, & How

Nathaniel R. Stickley

Caltech/IPAC Software Containerization Workshop

September 12, 2019

Overview

- Personal Introduction
- Strengths and Weaknesses of Docker
- Introduction to Singularity
- A brief tutorial
- How Singularity is used at IPAC

Professional Background

- M.S. in Applied & Engineering Physics (2008)
  - Developed a magnetosphere modeling code

Professional Background

- M.S. in Applied & Engineering Physics (2008)
  - Developed a magnetosphere modeling code
- Ph.D. in Physics (2013)
  - Performed galaxy merger simulations using HPC facilities
  - Developed snapshot visualization & analysis software

Professional Background

- M.S. in Applied & Engineering Physics (2008)
  - Developed a magnetosphere modeling code
- Ph.D. in Physics (2013)
  - Performed galaxy merger simulations using HPC facilities
  - Developed snapshot visualization & analysis software
- Project Scientist at UC Riverside (2014 – 2016)
  - Developed a Big Data framework based on Mesos & HDFS
  - Built computing cluster (and virtual clusters in EC2)
  - Improved SPHEREx redshift estimation pipeline prototype

Professional Background

- M.S. in Applied & Engineering Physics (2008)
  - Developed a magnetosphere modeling code
- Ph.D. in Physics (2013)
  - Performed galaxy merger simulations using HPC facilities
  - Developed snapshot visualization & analysis software
- Project Scientist at UC Riverside (2014 – 2016)
  - Developed a Big Data framework based on Mesos & HDFS
  - Built computing cluster (and virtual clusters in EC2)
  - Improved SPHEREx redshift estimation pipeline prototype
- Applications Developer at IPAC (2016 – present)
  - Euclid spectral decontamination module
  - Joint Survey Processing science platform

Docker seems awesome!

During 2014, Docker popularity began increasing rapidly

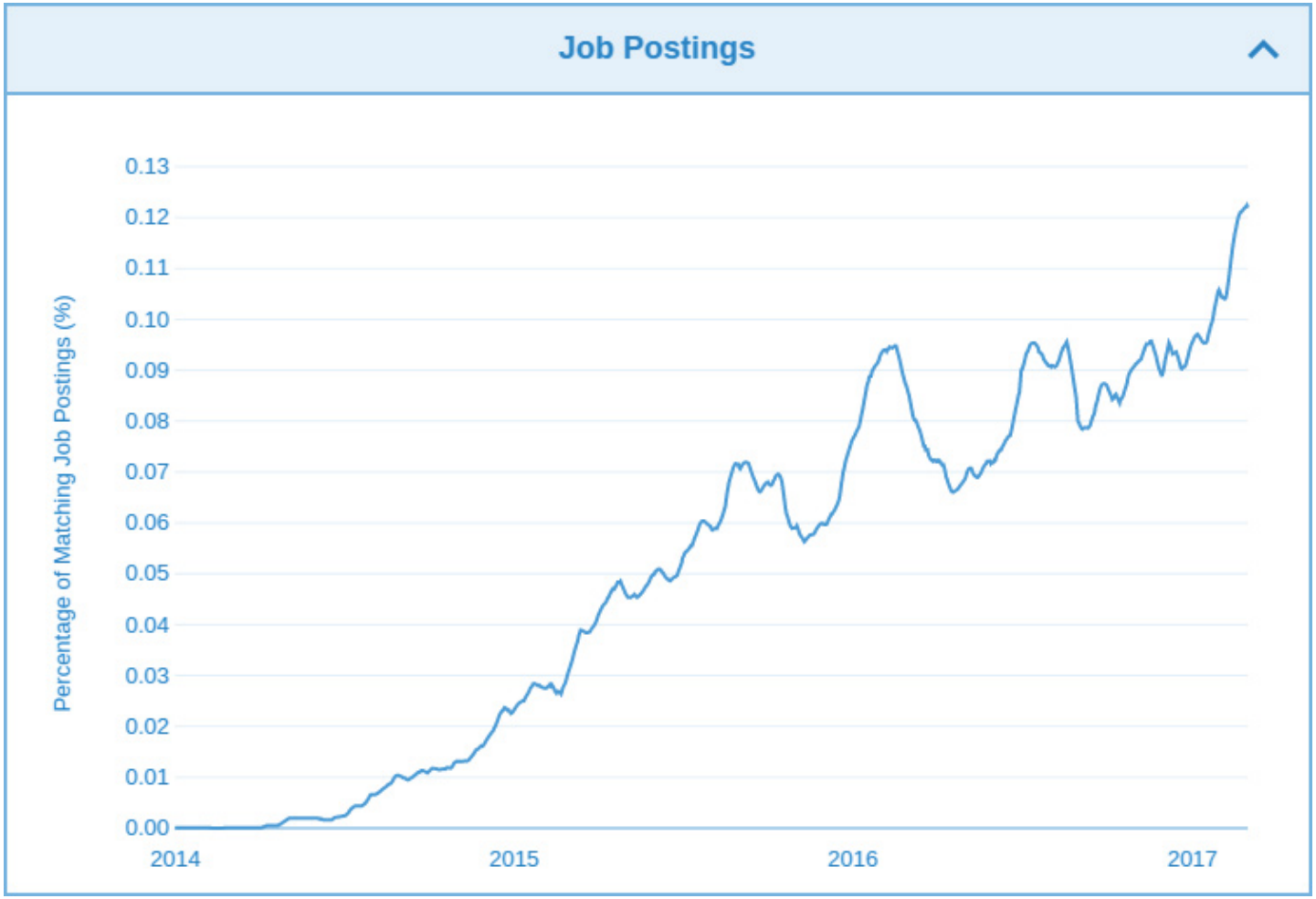

image source: https://diveintodocker.com

Dabbling with Docker

Benefits of Docker and containerization, in general

- No hypervisor / virtualization overhead
- No overhead from running an extra OS kernel
- Zero boot time
- The ability to easily pull and share images (via Docker Hub)
- Easy, reproducible creation of development environment
- Relatively easy software deployment

Dabbling with Docker

The Good:

- Docker is really nice for deploying scalable (distributed) applications, like web services and databases.

Dabbling with Docker

The Good:

- Docker is really nice for deploying scalable (distributed) applications, like web services and databases.

The not-so-Good:

- The Docker daemon runs as root (superuser)
- The daemon is in charge of managing everything, including image placement and naming
- Running containerized applications as a regular user is not straightforward (not intended for batch processing systems)
- Moving container images around is not straightforward
- Using with GPU-accelerated code is not straightforward (until recently)
- Deploying apps with MPI is not straightforward...

The Not-so-Good (continued):

- The CLI is not particularly intuitive (for me, anyway).

Singularity to the rescue!

Development of Singularity began in 2015 at Lawrence Berkeley National Laboratory.

Goal: Create a containerization solution for HPC systems.

Singularity to the rescue!

Development of Singularity began in 2015 at Lawrence Berkeley National Laboratory.

Goal: Create a containerization solution for HPC systems.

Key properties:

- it is compatible with existing HPC environments (MPI, GPUs)

Singularity to the rescue!

Development of Singularity began in 2015 at Lawrence Berkeley National Laboratory.

Goal: Create a containerization solution for HPC systems.

Key properties:

- it is compatible with existing HPC environments (MPI, GPUs)
- it has no daemon; containers run as regular user processes (isolated or not isolated)
  - user 'impersonation' is automatic / not needed
  - works with existing batch processing systems

Singularity to the rescue!

Development of Singularity began in 2015 at Lawrence Berkeley National Laboratory.

Goal: Create a containerization solution for HPC systems.

Key properties:

- it is compatible with existing HPC environments (MPI, GPUs)
- it has no daemon; containers run as regular user processes (isolated or not isolated)
  - user 'impersonation' is automatic / not needed
  - works with existing batch processing systems
- it uses compressed images which can be signed and encrypted

Singularity to the rescue!

Development of Singularity began in 2015 at Lawrence Berkeley National Laboratory.

Goal: Create a containerization solution for HPC systems.

Key properties:

- it is compatible with existing HPC environments (MPI, GPUs)
- it has no daemon; containers run as regular user processes (isolated or not isolated)
  - user 'impersonation' is automatic / not needed
  - works with existing batch processing systems
- it uses compressed images which can be signed and encrypted
- its images are regular files which can be named anything you wish, copied, and moved as usual—no need to register the images.

Singularity to the rescue!

Extra properties:

- Singularity images can act as regular executable files

Singularity to the rescue!

Extra properties:

- Singularity images can act as regular executable files
- Images can be pushed and pulled from Singularity Hub, analogous with Docker Hub

Singularity to the rescue!

Extra properties:

- Singularity images can act as regular executable files
- Images can be pushed and pulled from Singularity Hub, analogous with Docker Hub
- Singularity can pull containers from Docker Hub!

Singularity to the rescue!

Extra properties:

- Singularity images can act as regular executable files
- Images can be pushed and pulled from Singularity Hub, analogous with Docker Hub
- Singularity can pull containers from Docker Hub!
- Singularity allows you to use a 'sandbox' directory as a mutable / writable image

The definition file

Images are created by following instructions specified in a plain text `definition` file, (`.def`)

The definition file

Images are created by following instructions specified in a plain text `definition` file, (`.def`)

The header must begin with a `BootStrap` agent specification:

```
BootStrap: library
From: ubuntu:18.04
```

Other bootstrap agents include:

- Hubs: `docker`, `shub`
- Linux distros: `yum`, `debootstrap`, `arch`, `zypper`
- Local singularity images: `localimage`
- From scratch: `scratch`

The definition file

The header is followed by several %sections, many of which are optional. For example:

```
BootStrap: localimage
From: ./jsp_apps-base.sif

%post

    # commands here are executed at build-time
    # install software
    # modify configuration files
    # and clean up temporary files

%files

    source_file /dest/path/
    another_source_file /another/dest/
```

```
%environment

    export LC_ALL=C.UTF-8
    export LANG=C.UTF-8

%test

    # the name of a test script
    # or several test scripts

%runscript

    # the default executable to run

%startscript

    # instance start executable
```

Building an Image

There are two ways to build a Singularity Image Format (SIF) file:

```
sudo singularity build img.sif source.def
```

```
singularity build --fakeroot img.sif source.def
```

Building an Image

There are two ways to build a Singularity Image Format (SIF) file:

```
sudo singularity build img.sif source.def
```

```
singularity build --fakeroot img.sif source.def
```

Similarly, you can build a sandbox directory:

```
sudo singularity build --sandbox dir source.def
```

```
singularity build --fakeroot \
                  --sandbox dir source.def
```

Building an Image

You can also convert an image to a sandbox directory:

```
sudo singularity build --sandbox dir image.sif
```

and do the inverse:

```
sudo singularity build image.sif dir
```

Running Applications in Singularity

There are several ways to start applications in a Singularity container...

To run the `runscript`:

```
/path/to/image_name.sif

singularity run /path/to/image.sif
```

Running Applications in Singularity

There are several ways to start applications in a Singularity container...

To run the `runscript`:

```
/path/to/image_name.sif

singularity run /path/to/image.sif
```

To execute an arbitrary command:

```
singularity exec /path/to/image.sif command
```

Running Applications in Singularity

There are several ways to start applications in a Singularity container...

To run the `runscript`:

```
/path/to/image_name.sif

singularity run /path/to/image.sif
```

To execute an arbitrary command:

```
singularity exec /path/to/image.sif command
```

To start a shell session in the container:

```
singularity shell /path/to/image.sif

# OR:

singularity exec /path/to/image.sif bash
```

Container Instances

You can also start a container as an **instance**, like this:

```
singularity instance start image.sif inst_name
```

The `startscript` will run in the container, which will detach from the shell.

Container Instances

You can also start a container as an **instance**, like this:

```
singularity instance start image.sif inst_name
```

The `startscript` will run in the container, which will detach from the shell.

More commands related to instances:

```
singularity instance list

singularity instance stop inst_name

singularity shell instance://inst_name
```

A few useful options

Singularity commands have many options. Some very useful ones are `--bind`, `--writable`, `--containall`, `--pwd`, and `--home`:

`--bind`:

```
singularity cmd --bind /path image.sif ...
```

where `cmd` is one of `shell`, `run`, `exec`, `instance start`

For example:

```
singularity shell --bind /run image.sif
```

Now, the host's `/run` directory is available from inside of the container.

A few useful options

`--writable`:

```
singularity shell --fakeroot --writable /sandbox
```

Now the sandbox can be modified; you can add and remove software.

A few useful options

`--writable`:

```
singularity shell --fakeroot --writable /sandbox
```

Now the sandbox can be modified; you can add and remove software.

`--containall`:

```
singularity cmd --containall image.sif ...
```

The container is isolated in a separate namespace and cannot communicate with external processes (`htop` only shows processes running in the container).

A few useful options

`--pwd`:

```
singularity cmd --pwd /work/dir image.sif ...
```

Sets the working directory in the container to `/work/dir` instead of the current working directory of the parent shell.

A few useful options

`--pwd`:

```
singularity cmd --pwd /work/dir image.sif ...
```

Sets the working directory in the container to `/work/dir` instead of the current working directory of the parent shell.

`--home`:

```
singularity cmd --home /alt/home image.sif ...
```

Sets the `$HOME` directory to `/alt/home` instead of the user's actual home directory.

IPAC use-cases

Euclid The Euclid development environment is available as a Singularity image augmented with a Cern VM File System (CVMFS) mount. This is used on the Euclid cluster at IPAC.

JSP The Joint Survey Processing science platform (in early development) uses Singularity, since Singularity is installed at XSEDE supercomputing centers.

WFIRST Early-stage WFIRST grism simulations are being carried out using a version of aXeSIM that was patched and containerized using Singularity.

Thank You!

To learn more:

SyLabs Home: `sylabs.io`

Singularity GitHub: `github.com/sylabs/singularity`

RPM for Singularity 3.4.0: `bit.ly/2mbG2XZ`

DEB for Singularity 3.4.0: `bit.ly/2kFFQQ7`