CIS 602-01: Computational Reproducibility

Containers and Reproducibility

Dr. David Koop
## The Problem Matrix

<table>
<thead>
<tr>
<th></th>
<th>Developmen&lt;sup&gt;t&lt;/sup&gt; VM</th>
<th>QA Server</th>
<th>Single Prod Server</th>
<th>Onsite Cluster</th>
<th>Public Cloud</th>
<th>Contribution&lt;sup&gt;er&lt;/sup&gt;'s laptop</th>
<th>Customer Servers</th>
</tr>
</thead>
</table>

[Docker, Inc., 2016]
Shipping Analogy

A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.

...in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another.

[Docker, Inc., 2016]
The Solution: Containers

An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container...

...that can be manipulated using standard operations and run consistently on virtually any hardware platform

[Docker, Inc., 2016]
Containers vs. Virtual Machines

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

...result is significantly faster deployment, much less overhead, easier migration, faster restart

[Dockers, Inc., 2016]
Dockerfile and Docker image

• Dockerfile is a set of rules to create a Docker image
• Docker images are layered binaries
• Docker images used to run container in a sandboxed environment
  - docker run [options] <image> [args]
  - Can be non-interactive (default) or interactive
• Can save the image (copy-on-write) after interactions
Docker container lifecycle

- `run`
- `create`
- `start`
- `stop`
- `kill`
- `pause`
- `unpause`
## Docker images

The `docker tag` is made of two parts: `REPOSITORY:TAG`. If not provided, the default is `:latest`.

```
$ docker images

<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
<th>IMAGE ID</th>
<th>CREATED</th>
<th>VIRTUAL SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>debian</td>
<td>8</td>
<td>835c4d274060</td>
<td>2 weeks ago</td>
<td>122.6 MB</td>
</tr>
<tr>
<td>debian</td>
<td>8.0</td>
<td>835c4d274060</td>
<td>2 weeks ago</td>
<td>122.6 MB</td>
</tr>
<tr>
<td>debian</td>
<td>jessie</td>
<td>835c4d274060</td>
<td>2 weeks ago</td>
<td>122.6 MB</td>
</tr>
<tr>
<td>debian</td>
<td>rc-buggy</td>
<td>350a74df81b1</td>
<td>7 months ago</td>
<td>159.9 MB</td>
</tr>
<tr>
<td>debian</td>
<td>experimental</td>
<td>36d6c9c7df4c</td>
<td>7 months ago</td>
<td>159.9 MB</td>
</tr>
<tr>
<td>debian</td>
<td>6.0.9</td>
<td>3b36e4176538</td>
<td>7 months ago</td>
<td>112.4 MB</td>
</tr>
<tr>
<td>debian</td>
<td>squeeze</td>
<td>3b36e4176538</td>
<td>7 months ago</td>
<td>112.4 MB</td>
</tr>
<tr>
<td>debian</td>
<td>wheezy</td>
<td>667250f9a437</td>
<td>7 months ago</td>
<td>115 MB</td>
</tr>
<tr>
<td>debian</td>
<td>latest</td>
<td>667250f9a437</td>
<td>7 months ago</td>
<td>115 MB</td>
</tr>
<tr>
<td>debian</td>
<td>7.5</td>
<td>667250f9a437</td>
<td>7 months ago</td>
<td>115 MB</td>
</tr>
<tr>
<td>debian</td>
<td>unstable</td>
<td>24a4621560e4</td>
<td>7 months ago</td>
<td>123.6 MB</td>
</tr>
<tr>
<td>debian</td>
<td>testing</td>
<td>7f5d8ca9fdcf</td>
<td>7 months ago</td>
<td>121.8 MB</td>
</tr>
<tr>
<td>debian</td>
<td>stable</td>
<td>caa04aa09d69</td>
<td>7 months ago</td>
<td>115 MB</td>
</tr>
<tr>
<td>debian</td>
<td>sid</td>
<td>f3d4759f77a7</td>
<td>7 months ago</td>
<td>123.6 MB</td>
</tr>
<tr>
<td>debian</td>
<td>7.4</td>
<td>e565fbbc6033</td>
<td>9 months ago</td>
<td>115 MB</td>
</tr>
<tr>
<td>debian</td>
<td>7.3</td>
<td>b5fe16f2ccba</td>
<td>11 months ago</td>
<td>117.8 MB</td>
</tr>
</tbody>
</table>
```

[A. Baire, 2016]
A Dockerfile example:

```Dockerfile
# base image: last debian release
FROM debian:wheezy

# name of the maintainer of this image
MAINTAINER Anthony.Baire@irisa.fr

# install the latest upgrades
RUN apt-get update && apt-get -y dist-upgrade

# install nginx
RUN apt-get -y install nginx

# set the default container command
# -> run nginx in the foreground
CMD ["nginx", "-g", "daemon off;"

# Tell the docker engine that there will be something listening on the tcp port 80
EXPOSE 80
```

[A. Baire, 2016]
Docker Compose

- Have multiple containers with links between them!
- Process as a Service… (PaaS)
Related: Package Management & Deployment

• Examples:
  - Anaconda for Python
  - Gems for Ruby
  - apt-get, yum, etc. for Linux distributions

• Help to build images but can introduce issues?
Project

• Find some papers that you may be interested in reproducing
• Do a survey of the material that is available for each paper:
  - Code?
    • Is the code under version control?
  - Data?
    • Is it clear how to process or understand the data?
    • Is there metadata?
  - Virtual machine or container?
    • Does the hardware/software that deals with these still work?
  - Provenance?
    • Do we have a record of the steps taken in producing a result?
    • How complete is it?
Project

• If you are interested in a topic that aligns with reproducibility, please email me/talk to me about your ideas

• For example, if you are working on a research project that could incorporate reproducibility

• Formal Specification Online
• [http://www.cis.umassd.edu/~dkoop/cis602/project.html](http://www.cis.umassd.edu/~dkoop/cis602/project.html)
• Due Monday, November 7
Containers and Reproducibility

- What are the benefits of containers over virtual machines with respect to reproducibility?
- Do containers address all of the problems we are concerned with?
- What issues remain?
An introduction to Docker for reproducible research

C. Boettiger
Two audiences

• Domain scientist: How does Docker make research more reproducible, extensible, and portable?
• Systems Researcher: How do Linux kernel changes impact reproducibility?
Cultural problems still dominate

- Code availability
- Data availability
- Many journals don't have requirements
Technical Challenges

• Dependency Hell
• Imprecise Documentation
• Code rot
• Barriers to adoption and reuse in existing solutions
Other solutions and their problems

• Scientific Workflows
  - nice environment to abstract underlying computations
  - constrained environments

• Virtual machines
  - reproduce most everything including system environment vars
  - black box: cannot examine or easily build on
Docker Solutions

- Dependency Hell: Docker images
- Imprecise documentation: Dockerfiles document how the image was created
- Code Rot: versions of images (latest, v2.2, etc.)
- Adoption/reuse barriers: ease of performing tasks in Docker
Docker Features for Reproducibility

- Integrating into local development environments
- Modular reuse (Docker compose)
- Portable environments (snapshots)
- Public repository for sharing (DockerHub)
- Versioning
Best Practices

• Use Docker containers during development
• Write Dockerfiles instead of installing interactive sessions
• Adding tests or checks to the Dockerfile
• Use and provide appropriate base images
• Share Docker images and Dockerfiles
• Archive tarball snapshots
Limitations

- Docker does not provide complete virtualization (reliant on kernel)
- Limited to 64-bit machines (https://github.com/docker-32bit)
- Mac and Windows support (improving)
- Security issues? signed images?
- Adoption???
Discussion
Other

- Combining virtualization with other reproducible-research tools
- Impacting cultural norms
Docker Reproducibility Example

- [http://bioinfoblog.it/2015/03/reproducible-bioinformatics-pipelines-with-docker/](http://bioinfoblog.it/2015/03/reproducible-bioinformatics-pipelines-with-docker/)
More Docker Feedback

• Titus Brown:
Other Ideas

- nix(os)
- rabix
- bioboxes
nix

R. Garbas
rabix

Kaushik et al.