# 400,000 Ephemeral Containers

## Testing entire ecosystems with Docker

Daniel Axtens

linux.conf.au, Hobart, 2017

# Scope

- 125417 Ruby gems
- 369326 npm packages
- 95852 PyPI packages
- 9323 R packages

# Test all the packages

1. Simple
2. Fast
3. Accurate

# Test all the packages

1. Simple
2. Fast
3. Accurate

Does it install?

# Let's do it

**Live Demo!**

# Where we're going

1. Introduction and Motivation
2. Methodology
   - Testing a package
   - Managing a test
   - Managing all the tests
   - Going fast
   - Display and post-processing
3. Review and results
   - Results
   - Interesting bugs
4. Future directions
   - Other applications
5. Conclusion
   - Final Thoughts

1. Introduction and Motivation
2. Methodology
   - **Testing a package**
   - Managing a test
   - Managing all the tests
   - Going fast
   - Display and post-processing
3. Review and results
   - Results
   - Interesting bugs
4. Future directions
   - Other applications
5. Conclusion
   - Final Thoughts

# Testing a package

- Tests shouldn't influence each other.
- Tests should be ephemeral.

# Isolation Technology

- Isolation
- Low overhead
- Easy to script

# Isolation Technologies

- Virtual machines
- Chroots
- Containers

# Interaction with Docker

**Live demo**

# Let's build a container...

# Let's build a container...

```
FROM ppc64le/ubuntu:16.04
```

# Let's build a container...

```
FROM ppc64le/ubuntu:16.04

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get upgrade -y && \
    apt-get install -y build-essential curl libcurl4-openssl-dev \
                       libxml2-dev libpq-dev libmysqlclient-dev \
                       libpcre3-dev libxslt-dev libmagickcore-dev \
                       libmagickwand-dev libglib2.0-dev libv8-dev \
                       libpango1.0-dev default-jdk git cmake \
                       libmagic-dev libatk1.0-dev libsdl2-dev \
                       libsdl2-ttf-dev libpango1.0-dev libgsl0-dev \
                       libgl1-mesa-dev libfreeimage-dev \
                       libopenal-dev libsndfile-dev libgtk2.0-dev \
                       libgirepository1.0-dev libsasl2-dev \
                       libtag1-dev libavahi-compat-libdnssd-dev \
                       libmemcached-dev uuid-dev libsdl2-dev
```

# Let's build a container...

```
RUN curl -sSL https://rvm.io/mpapis.asc | gpg --import - && \
    curl -sSL https://get.rvm.io | bash -s stable --ruby=2.2.6 && \
    rm -rf /var/cache/apt/* /usr/local/rvm/src && \
    useradd -m -G rvm tester
```

# Let's build a container...

```
RUN curl -sSL https://rvm.io/mpapis.asc | gpg --import - && \
    curl -sSL https://get.rvm.io | bash -s stable --ruby=2.2.6 && \
    rm -rf /var/cache/apt/* /usr/local/rvm/src && \
    useradd -m -G rvm tester

USER tester
```

# Let's build a container...

```
RUN curl -sSL https://rvm.io/mpapis.asc | gpg --import - && \
    curl -sSL https://get.rvm.io | bash -s stable --ruby=2.2.6 && \
    rm -rf /var/cache/apt/* /usr/local/rvm/src && \
    useradd -m -G rvm tester

USER tester

RUN bash -c "source /usr/local/rvm/scripts/rvm && \
             gem install --no-rdoc --no-ri nokogiri"
```

# Let’s build a container...

```
RUN curl -sSL https://rvm.io/mpapis.asc | gpg --import - && \
    curl -sSL https://get.rvm.io | bash -s stable --ruby=2.2.6 && \
    rm -rf /var/cache/apt/* /usr/local/rvm/src && \
    useradd -m -G rvm tester

USER tester

RUN bash -c "source /usr/local/rvm/scripts/rvm && \
             gem install --no-rdoc --no-ri nokogiri"

COPY gemrc /home/tester/.gemrc
ENV http_proxy=http://localhost:3128/ \
    https_proxy=http://localhost:3128/
```

# Let’s build a container...

```
RUN curl -sSL https://rvm.io/mpapis.asc | gpg --import - && \
    curl -sSL https://get.rvm.io | bash -s stable --ruby=2.2.6 && \
    rm -rf /var/cache/apt/* /usr/local/rvm/src && \
    useradd -m -G rvm tester

USER tester

RUN bash -c "source /usr/local/rvm/scripts/rvm && \
             gem install --no-rdoc --no-ri nokogiri"

COPY gemrc /home/tester/.gemrc
ENV http_proxy=http://localhost:3128/ \
    https_proxy=http://localhost:3128/

ENTRYPOINT ["/bin/bash", "-c"]
CMD ["source /usr/local/rvm/scripts/rvm && \
      gem install --no-rdoc --no-ri $GEM"]
```

# Let's test a package

**Live Demo!**

1. Introduction and Motivation
2. Methodology
   - Testing a package
   - Managing a test
   - Managing all the tests
   - Going fast
   - Display and post-processing
3. Review and results
   - Results
   - Interesting bugs
4. Future directions
   - Other applications
5. Conclusion
   - Final Thoughts

# Managing tests...

```
#!/bin/bash
GEM=$1
TIMEOUT=240s

http_proxy=

echo "Testing $GEM"
cont=$(docker run -d --env GEM=$GEM --net=host \
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/gems:rw,mode=777,gid=1000,exec \
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/cache:rw,mode=777,gid=1000,exec \
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/wrappers:rw,mode=777,gid=1000,exec \
--tmpfs=/tmp/:rw,mode=777,exec \
       rubytest)
```

# Managing tests...

```
#!/bin/bash
GEM=$1
TIMEOUT=240s

http_proxy=

echo "Testing $GEM"
cont=$(docker run -d --env GEM=$GEM --net=host \
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/gems:rw,mode=777,gid=1000,exec \
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/cache:rw,mode=777,gid=1000,exec \
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/wrappers:rw,mode=777,gid=1000,exec \
--tmpfs=/tmp/:rw,mode=777,exec \
       rubytest)

code=$(timeout $TIMEOUT docker wait "$cont")
```

# Managing tests...

```
if [ -z "$code" ]; then
    echo "TIMEOUT on $GEM"
    docker kill $cont &> /dev/null
    docker logs $cont 2>&1 | curl -X PUT --data-binary @- \
        http://127.0.0.1/blaster/submit/ruby/${GEM}/T/
else
    if [ "$code" != "0" ]; then
        echo "FAILURE on $GEM"
        docker logs $cont 2>&1 | curl -X PUT --data-binary @- \
           http://127.0.0.1/blaster/submit/ruby/${GEM}/F/
    else
        echo "SUCCESS on $GEM"
        docker logs $cont 2>&1 | curl -X PUT --data-binary @- \
           http://127.0.0.1/blaster/submit/ruby/${GEM}/S/
    fi
fi
```

# Managing tests...

```
if [ -z "$code" ]; then
    echo "TIMEOUT on $GEM"
    docker kill $cont &> /dev/null
    docker logs $cont 2>&1 | curl -X PUT --data-binary @- \
        http://127.0.0.1/blaster/submit/ruby/${GEM}/T/
else
    if [ "$code" != "0" ]; then
        echo "FAILURE on $GEM"
        docker logs $cont 2>&1 | curl -X PUT --data-binary @- \
           http://127.0.0.1/blaster/submit/ruby/${GEM}/F/
    else
        echo "SUCCESS on $GEM"
        docker logs $cont 2>&1 | curl -X PUT --data-binary @- \
           http://127.0.0.1/blaster/submit/ruby/${GEM}/S/
    fi
fi

docker rm $cont &> /dev/null
```

# Let’s test a package - with timeout

**Live Demo!**

1. Introduction and Motivation
2. Methodology
   - Testing a package
   - Managing a test
   - Managing all the tests
   - Going fast
   - Display and post-processing
3. Review and results
   - Results
   - Interesting bugs
4. Future directions
   - Other applications
5. Conclusion
   - Final Thoughts

# #winning

- `test.sh`

# #winning

- test.sh

```
#!/bin/bash
parallel --joblog ruby.log --resume -j30 ’./testit.sh {}’ :::: gems
```

# #winning

- `test.sh`

  ```
  #!/bin/bash
  parallel --joblog ruby.log --resume -j30 ’./testit.sh {}’ :::: gems
  ```

- `gems`

# #winning

- `test.sh`

```
#!/bin/bash
parallel --joblog ruby.log --resume -j30 ’./testit.sh {}’ :::: gems
```

- `gems`

```
gem list --remote -q | awk ’{print $1}’ > gems
```

1. Introduction and Motivation
2. Methodology
    - Testing a package
    - Managing a test
    - Managing all the tests
    - Going fast
    - Display and post-processing
3. Review and results
    - Results
    - Interesting bugs
4. Future directions
    - Other applications
5. Conclusion
    - Final Thoughts

# Caching

```
Cache information for squid:
        Hits as % of all requests:      5min: 65.8%, 60min: 66.7%
        Hits as % of bytes sent:        5min: 64.5%, 60min: 72.0%
```

# Other limiting factors

- Ruby — Disk IO:

```
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/gems:rw,mode=777,gid=1000,exec \
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/cache:rw,mode=777,gid=1000,exec \
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/wrappers:rw,mode=777,gid=1000,exec \
--tmpfs=/tmp/:rw,mode=777,exec \
```

- NodeJS/npm

# Other limiting factors

- Ruby — Disk IO:

```
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/gems:rw,mode=777,gid=1000,exec \
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/cache:rw,mode=777,gid=1000,exec \
--tmpfs=/usr/local/rvm/gems/ruby-2.2.6/wrappers:rw,mode=777,gid=1000,exec \
--tmpfs=/tmp/:rw,mode=777,exec \
```

- NodeJS/npm
- Diminishing returns...

1. Introduction and Motivation
2. Methodology
   - Testing a package
   - Managing a test
   - Managing all the tests
   - Going fast
   - Display and post-processing
3. Review and results
   - Results
   - Interesting bugs
4. Future directions
   - Other applications
5. Conclusion
   - Final Thoughts

# Excluding ‘dumb stuff’

- Broken dependencies

```
ERROR:  While executing gem ... (Gem::DependencyResolutionError)
    conflicting dependencies activesupport (= 3.0.0) \
                         and activesupport (= 4.2.0)
```

- Out of date use of the C API

```
../fbuffer/fbuffer.h:175:47: error: macro "rb_str_new" requires 2
arguments, but only 1 given
     VALUE result = rb_str_new(FBUFFER_PAIR(fb));
```

- Other dumb errors/outright breakage

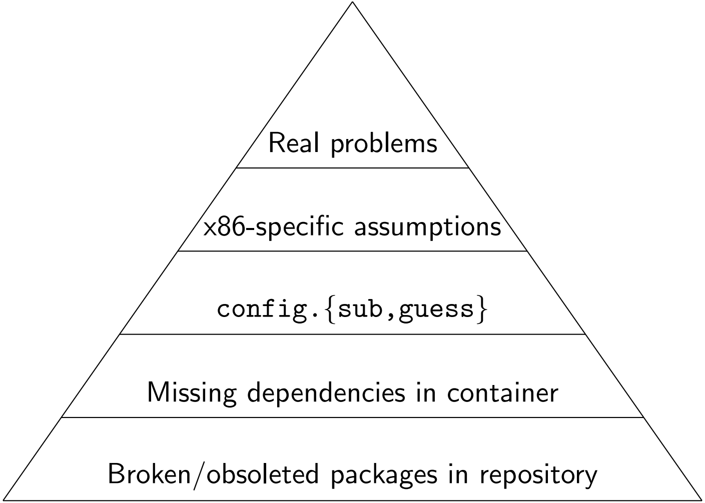
```
npm ERR! enoent ENOENT: no such file or directory, chmod \
    ’/home/tester/node_modules/abn-validator/abn-validator.js’
npm ERR! enoent This is most likely not a problem with npm itself
npm ERR! enoent and is related to npm not being able to find a file.
npm ERR! enoent
```

1. Introduction and Motivation
2. Methodology
   - Testing a package
   - Managing a test
   - Managing all the tests
   - Going fast
   - Display and post-processing
3. Review and results
   - Results
   - Interesting bugs
4. Future directions
   - Other applications
5. Conclusion
   - Final Thoughts

# Summary of Results

$$\begin{aligned}
\text{Ruby} &> 96\%\ \text{success}\\
\text{NodeJS} &> 97\%\ \text{success}\\
\text{R} &\approx 98\%\ \text{success}
\end{aligned}$$

# Bugs: Packages



Real problems

x86-specific assumptions

`config.{sub,guess}`

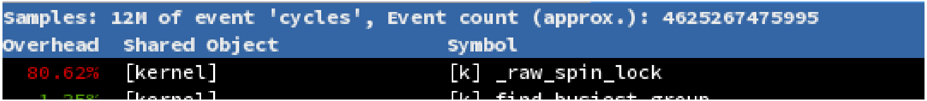
Missing dependencies in container

Broken/obsoleted packages in repository

# Bugs: Docker

### Docker Storage Drivers

- AUFS
  - Overheads:



```
Samples: 12M of event 'cycles', Event count (approx.): 4625267475995
Overhead  Shared Object                Symbol
  80.62%  [kernel]                     [k] _raw_spin_lock
   1.25%  [kernel]                     [k] find_busiest_group
```

  - Random hangs
  - Out-of-tree module
- Use Overlay

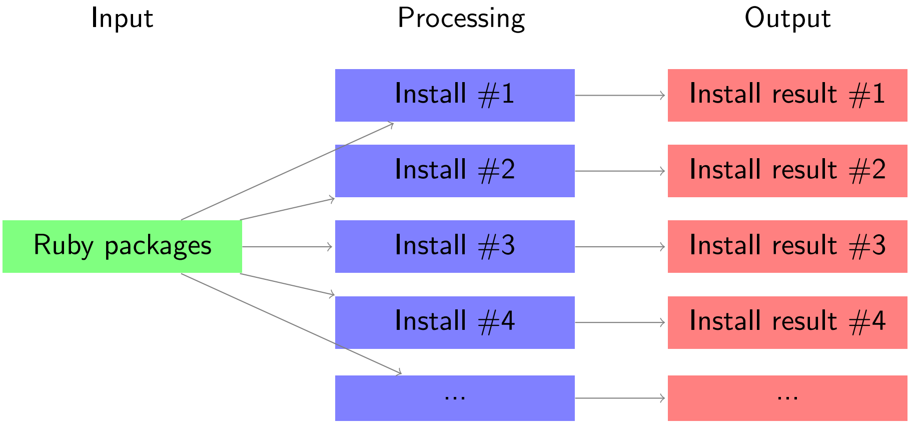
# Bugs: Kernel (https://lkml.org/lkml/2016/3/10/682)

| | |
|---|---|
| From | Daniel Axtens <dja@axtens.net> |
| Subject | ext4_file_open: Inconsistent encryption contexts (commit ff978b09f973) breaking Docker |
| Date | Fri, 11 Mar 2016 11:44:54 +1100 |

```
Hi,

Trying to run a Docker container on a mainline kernel is failing
intermittently, in interesting and exciting ways, such as:

$ docker run -it --rm --env PACKAGE=sinatra npmtest
operation not permitted
docker: Error response from daemon: Cannot start container \
4fc0120a6389f25241f84527a0d31854806f6fe4fd98d019f790cea0ae7e230b: \
[10] System error: operation not permitted.

...
```

1. Introduction and Motivation
2. Methodology
   - Testing a package
   - Managing a test
   - Managing all the tests
   - Going fast
   - Display and post-processing
3. Review and results
   - Results
   - Interesting bugs
4. Future directions
   - Other applications
5. Conclusion
   - Final Thoughts

# Currently...



Input

Processing

Output

Ruby packages

Install #1

Install #2

Install #3

Install #4

...

Install result #1

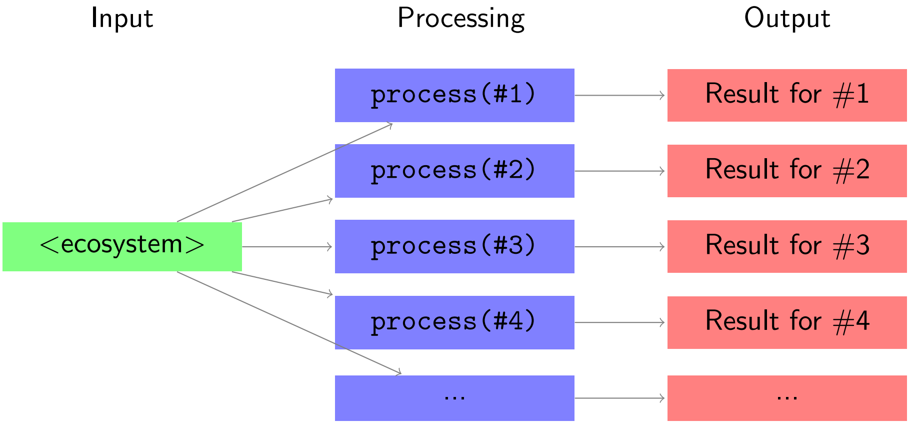
Install result #2

Install result #3

Install result #4

...

# In general...


Input
Processing
Output
<ecosystem>
process(#1)
process(#2)
process(#3)
process(#4)
...
Result for #1
Result for #2
Result for #3
Result for #4
...

# cppcheck

```
#include <stdlib.h>

int do_work(int value) {
    char * mem = malloc(1024);
    if (value < 0) {
        // we can’t operate on negative values!
        return -1;
    }
    //do stuff with mem and value
    free(mem);
    return 42;
}
```

# cppcheck

```c
#include <stdlib.h>

int do_work(int value) {
    char * mem = malloc(1024);
    if (value < 0) {
        // we can’t operate on negative values!
        return -1;
    }
    //do stuff with mem and value
    free(mem);
    return 42;
}
```

```
> cppcheck foo.c
Checking foo.c...
[foo.c:7]: (error) Memory leak: mem
```

# s/gem install/cppcheck/

```
#!/bin/bash
cd /tmp
source /usr/local/rvm/scripts/rvm

gem fetch $GEM
gem unpack ${GEM}*.gem
cd ${GEM}*
if [ -d ext ]; then
    cppcheck --quiet --force ext/ || true
else
    echo "no source found for $GEM"
    exit 1
fi
```

# s/gem install/cppcheck/

```
#!/bin/bash
cd /tmp
source /usr/local/rvm/scripts/rvm

gem fetch $GEM
gem unpack ${GEM}*.gem
cd ${GEM}*
if [ -d ext ]; then
    cppcheck --quiet --force ext/ || true
else
    echo "no source found for $GEM"
    exit 1
fi
```

The results are . . . well, lets just do a live demo.

# Wat? (https://www.destroyallsoftware.com/talks/wat)

The offending code is:

```
static VALUE rb_gsl_blas_zgemm(int argc, VALUE *argv,
                               VALUE obj)
{
  gsl_matrix_complex *A = NULL, *B = NULL, *C = NULL;
  gsl_complex *pa = NULL, *pb = NULL;
  CBLAS_TRANSPOSE_t TransA, TransB;
  int flag = 0;
  (*pa).dat[0] = 1.0; (*pa).dat[1] = 0.0;
  (*pb).dat[0] = 0.0; (*pb).dat[1] = 0.0;
```

# Recap

1. Introduction and Motivation
2. Methodology
   - Testing a package
   - Managing a test
   - Managing all the tests
   - Going fast
   - Display and post-processing
3. Review and results
   - Results
   - Interesting bugs
4. Future directions
   - Other applications
5. Conclusion
   - Final Thoughts

# Don't do it yourself

This project is a clumsy combination of other people's tools.

- GNU Parallel
- `timeout`
- `squid`
- Docker

# Future of Open Source

The future of open source is...

Cool new stuff

# Future of Open Source

The future of open source is...

~~Cool new stuff~~

# Future of Open Source

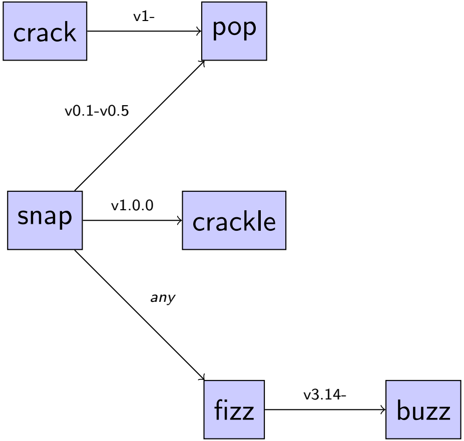
The future of open source is...

**Boring stuff at scale**

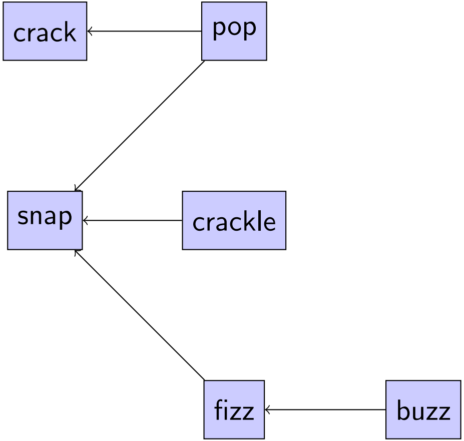
# Legal

- This work represents the view of the author and does not necessarily represent the view of IBM.
- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.

# CI: Dependency Graph



# CI: Reverse Dependency Graph



# CI: Reverse Dependency Mapping

- Snap: []
- Crack: []
- Crackle: [Snap]
- Pop: [Snap, Crack]
- Fizz: [Snap]
- Buzz: [Fizz, Snap]