Lessons learnt from "Shipping" containers.

Abejide Ayodele (Ayo)
bjhaid (twitter, github, stackoverflow...)

- ~ 30811 containers daily
- Average of 11 containers per build
- ~ 2801 builds on average daily

Problems

- Builds were too slow (feedback cycle for developers was too long)
- Builds sometimes were not easily reproduceable between dev environment and Jenkins
- Host where builds ran were mutated by builds and their dependencies

I haven't had a chance to look into it yet, but it looks like the java 7 CL builds are hard down due to some version problems with the java environment

```
Incorrect version of java. Expected 1.7 but got openjdk version "1.8.0_66-internal"
```

2:09 PM

jenkins09 seems to be  with postgres 😞

I'll bounce it

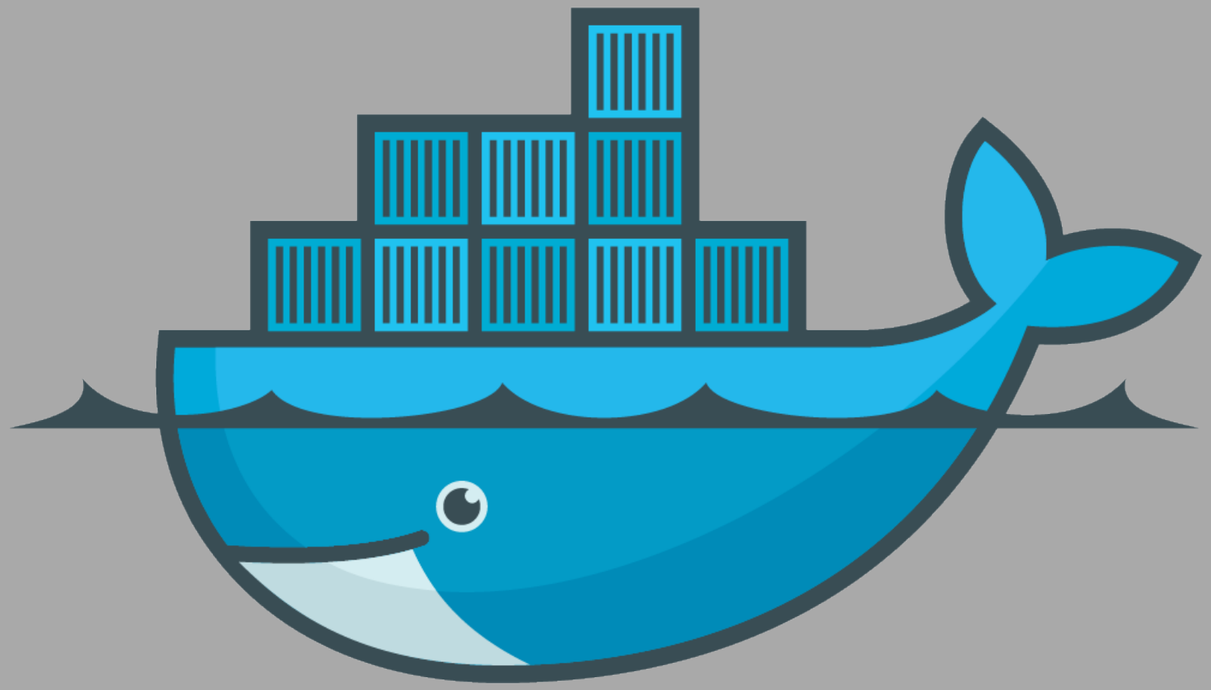
regarding that last Gateway Master failure. Looks like something went wrong during `db:do_schema`

```
PG::ConnectionBad: could not connect to server: Connection refused
    Is the server running on host "localhost" (127.0.0.1) and accepting
    TCP/IP connections on port 5433?
```

Core Objectives

- Make builds faster
- Make builds reproduceable
- Make builds *host agnostic*


I'm going on an adventure!



docker

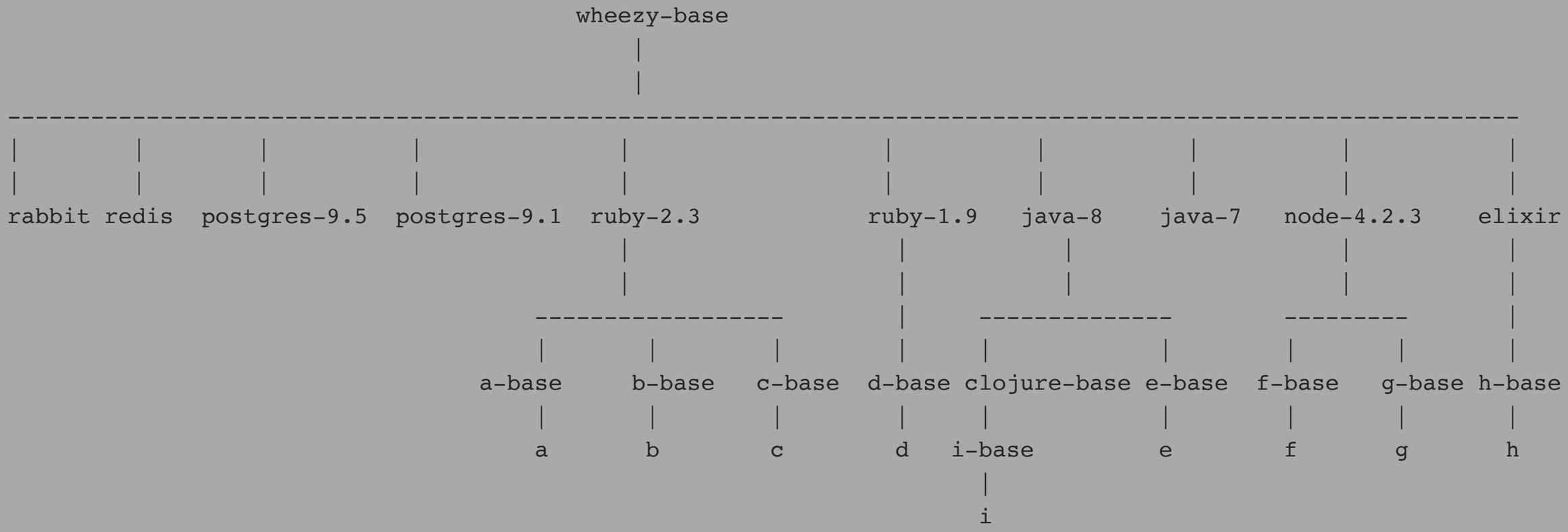
A brief about Docker niceties

- Docker Image/Dockerfile inheritance
- Docker Image caching
- Layer reuse
- Portability across machines
- Widely adopted (big community)

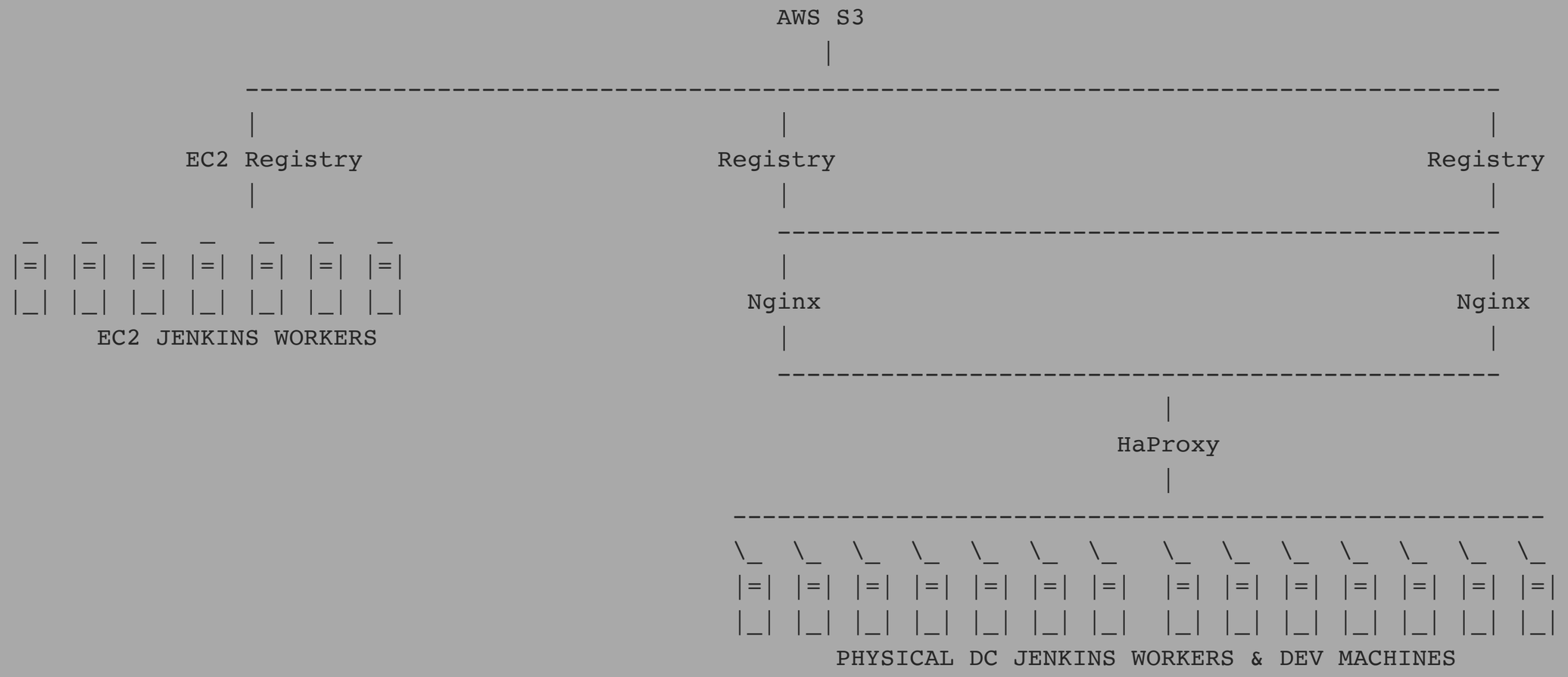
Putting docker to use

- We heavily exploited docker's image/dockerfile inheritance.

Minimal dockerfile/image tree



Docker Registry Architecture



Flow

- Build an image
- Run tests
- Seed database
- Commit project and database image
- Tag image with project name, current git sha and an extra branch tag
  e.g: **hub.braintree.com/bt/sample_app:18155b6bb4384cccd8acf796ecdcd7698b9c7f3c**

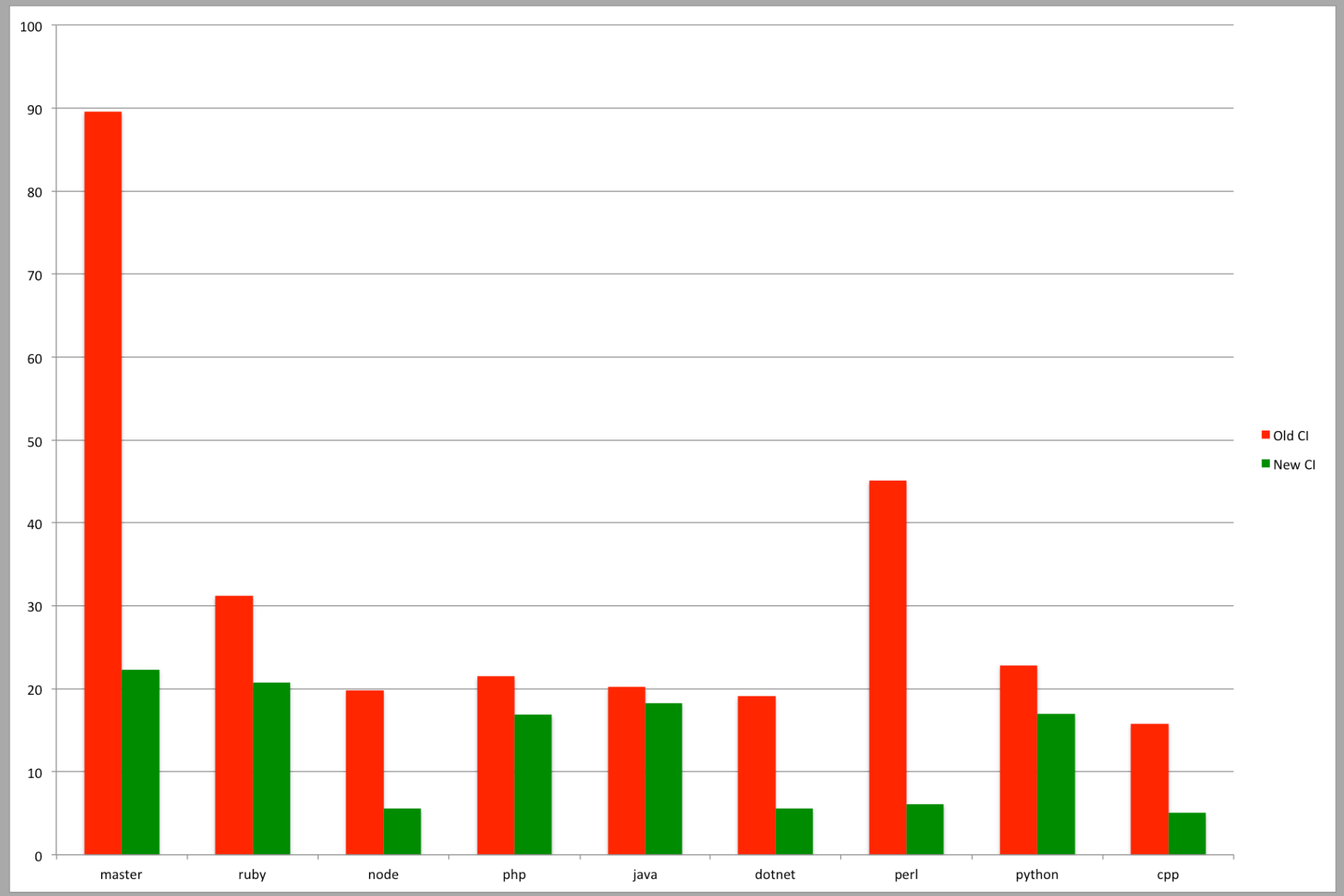
Flow

- Tag database image with project name, md5sum of migration and seed files and an extra branch tag
  e.g: **hub.braintree.com/bt/sample_app/data:master**
- Push project and database image
- Trigger parameterized build of downstream builds passing the SHA through to them

Early results

- Builds were reproduceable
- Builds were faster

Successful builds on New(green) vs Old(red) CI (small is better) in minutes



Objectives achieved but...

It came with other problems

- We were daisy-chaining Makefiles, bash files and docker-compose files
- Duplicating things across docker compose/ Makefiles
- Which was a bad UX


PixyStix
Tootsie Roll
DOTS
Tootsie
Lemon Flavored

New Objective

- Replace the daisy chain with something better

A DSL to replace compose files and Makefiles

- Written in a language familiar to Braintree Developers (ruby)
- Heavily inspired by rake
- Extendable in Ruby
- Reduced/*Removed* duplication

```
service :postgres do
  image "hub.braintree.com/bt/postgres:custom"
end
service :sample_app do
  image hub.braintree.com/bt/sample_app:master
end
job :test => [:sample_app, :postgres] do
  sample_app
    .link(:sample_app_db, postgres)
    .env("POSTGRES_HOST", "sample_app_db")
    .env("POSTGRES_PORT", "5433")
    .command("rake")
    .compose
    .run
end
```

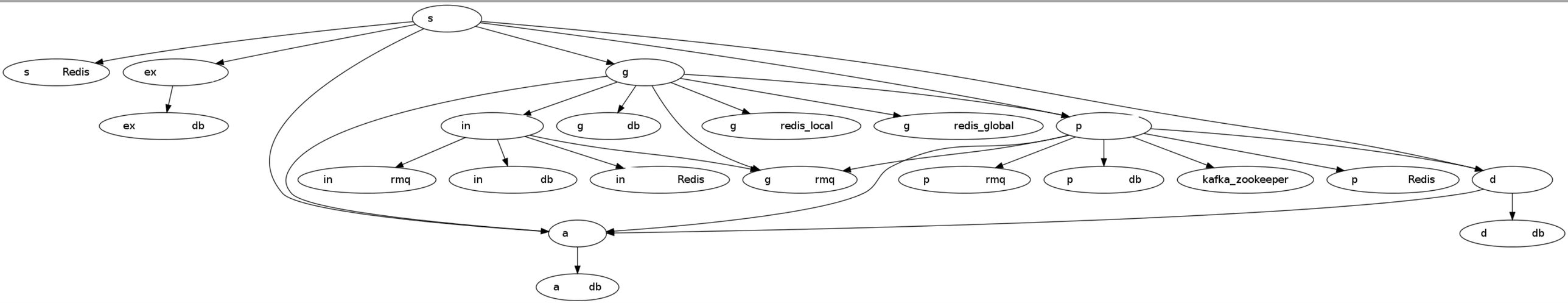
```
sample_app:
  command: "rake"
  environment:
    - DRAKE_HOST_USER_ID=1000
    - POSTGRES_HOST=sample_app_db
    - POSTGRES_PORT=5433
  image: hub.braintree.com/bt/sample_app:master
  links:
    - postgres:sample_app_db
postgres:
  environment:
    - DRAKE_HOST_USER_ID=1000
  image: hub.braintree.com/bt/postgres:custom
```

```
application :sample_app do
  local_path "../sample_app"
  remote "github.com:braintree/sample_app.git"
  revision "final"
end
service :foo do
  image "hub.braintree.com/bt/foo:master"
end
job :test => ["sample_app:sample_app", "sample_app:postgres", :foo]
  foo
    .link(:sample_app, sample_app[:sample_app])
    .link(:postgres, sample_app[:postgres])
    .command("rake")
    .compose
    .run
end
```

```
foo:
  command: "rake"
  environment:
    - DRAKE_HOST_USER_ID=1000
  image: hub.braintree.com/bt/foo:master
  links:
    - sample_app
    - postgres
sample_app:
  environment:
    - DRAKE_HOST_USER_ID=1000
  image: dockerhub.braintree.tools/bt/jenkins-grove-sample:87fbed05576406a2098fe8b173e62d578c6e7329
postgres:
  environment:
    - DRAKE_HOST_USER_ID=1000
  image: dockerhub.braintree.tools/bt/postgres:custom
```

```
job :integration => [
  :s,
  "g:g",
  "g:g_rmq",
  "a:a",
  "d:d",
  "ex:ex",
  "i:i",
  "p:p",
  :kafka_zookeeper,
] do
  d[:d]
    .link(:a, a[:a])

  p[:p]
    .link(:a, a[:a])
    .link(:d, d[:d])
    .link(:p_rmq, g[:g_rmq])
    .link(:p_kafka_zookeeper, kafka_zookeeper)

  i[:i]
    .link(:i_rmq, g[:g_rmq])

  g[:g]
    .link(:a, a[:a])
    .link(:p, p[:p])
    .link(:i, i[:i])

  compose = s
    .include_drake_scripts
    .env("g_URI", "http://g:3000")
    .env("g_PORT", "3000")
    .link(:a, a[:a])
    .link(:d, d[:d])
    .link("ex.docker.dev", ex[:ex])
    .link(:g, g[:g])
    .link(:p, p[:p])
    .command("./docker_integration.sh")
    .compose

  compose.pull
  compose.run
end
```

```
s:
  command: "./docker_integration.sh"
  dns:
    - 8.8.8.8
  environment:
    - DRAKE_HOST_USER_ID=1000
    - g_URI=http://g:3000
    - g_PORT=3000
  image: dockerhub.braintree.com/bt/s:cee77c064ba81ea7b6b8be2d394815d53637c
  links:
    - sRedis
    - a
    - d
    - ex:ex.docker.dev
    - g
    - p
  volumes:
    - /home/pair/bt/s/log:/home/bt/log
    - /var/lib/gems/1.9.1/gems/drake-0.12.1/script:/drake
sRedis:
  environment:
    - DRAKE_HOST_USER_ID=1000
  image: dockerhub.braintree.com/bt/redis:2.8
a:
  command: "bash -c 'true && foreman start'"
  dns:
    - 8.8.8.8
  environment:
    - DRAKE_HOST_USER_ID=1000
  image: dockerhub.braintree.com/bt/a:4cb9906801a0e8911f84b7c80ddc2ae6064
  links:
    - a_db
  volumes:
    - /home/pair/bt/a/log:/home/bt/log
a_db:
  environment:
    - DRAKE_HOST_USER_ID=1000
  image: dockerhub.braintree.com/bt/postgres-a/data:4cb9906801a0e8be911f84b7c80ddc2ae6064
d:
  command: "bash -c 'lein trampoline run -s -p 8080'"
  dns:
    - 8.8.8.8
  environment:
    - DRAKE_HOST_USER_ID=1000
  image: dockerhub.braintree.com/bt/d:8028123d64e769668eb922855a1169978cb
  links:
    - d_db
    - a
  volumes:
    - /home/pair/bt/d/log:/home/bt/log
d_db:
```


s
s Redis
ex
ex db
g
in
g db
g redis_local
g redis_global
p
in rmq
in db
in Redis
g rmq
p rmq
p db
kafka_zookeeper
p Redis
d
d db
a
a db

- Docker will create a directory as root if you specify a host directory as volume and the directory doesn't exist

```
pair88:~/docker_volume% ls
pair88:~/docker_volume%
```

Tip

If you will use volumes ensure UID/GID of user running docker daemon (docker-compose) is consistent with the user in the container

OR

```
groupadd docker -g 918 \
&& sed -i "s#root:x:0:0:root:/root:/bin/bash#root:x:0:918:root:/root:/bin/bash#g" /etc/passwd
```

```
command: "/bin/bash -c 'umask 0002 && mix do deps.get, clean, compile, ecto.create, ecto.migrate, test'"
```

docker-compose rm or **docker-compose stop** does not remove or stop primary service

```
 1 foo:
 2   command: "/bin/bash -c 'sleep 1h'"
 3   environment:
 4     - REDIS_HOST=redis
 5   image: debian:wheezy
 6   links:
 7     - redis
 8   volumes:
 9     - /home/pair/bt/foo:/home/bt
10 redis:
11   image: redis:2.8
~
~
~
~
~
~
~
~
[1] compose.yml                                [yaml]      L(1/11) C(1/125) All
```

Transient docker networking problems correlating with preloading our VMs/AMIs with loads of docker images.

Random kernel crashes (xen/docker/linux related)

Thanks!

Questions?