Docker in Production

Reality, Not Hype
Bridget Kromhout
@bridgetkromhout
bridgetkromhout.com
Operations Engineer
@arresteddevops
@devopsdays
DramaFever.com
Streaming international content starting in 2009
Docker in production since October 2013
THE UNRAVELING MIND

Losing your grip on reality? Join the club. This collection features mixed up men and women on the verge of a nasty nervous breakdown, trying desperately to keep from going over the edge. When that final meltdown begins - watch out. Our subjects can’t be held responsible for their actions...

shudder.com launched Summer 2015

@bridgetkromhout
Peak load: tens of thousands of requests per second
Traffic variance: swings 10-20x throughout the week
autoscaling in AWS

streaming delivery via Akamai

@bridgetkromhout
Architecture

Python/Django
Celery/SQS for async tasks
API endpoints for native clients
Go microservices
Upstreams routed via nginx
state in RDS, DynamoDB, Elasticache

@bridgetkromhout
Why Docker?

consistent development

repeatable deployment

@bridgetkromhout
one year ago...

Vagrant for local development

chef-solo provisioner

17 minutes to install everything

@bridgetkromhout
a year of boot2docker

images built on jenkins
mysql image built with fixtures

can run master or qa image (or even prod)

can build new local images from Dockerfiles

@bridgetkromhout
docker in production: in theory

@bridgetkromhout
docker in production: in practice

@bridgetkromhout
Distributed private S3-backed Docker registry:

registry container on each ec2 instance

more effective scaling

Post by Tim Gross: http://0x74696d.com/posts/host-local-docker-registry/

@bridgetkromhout
docker options

# goes in /etc/default/docker to control docker's upstart
DOCKER_OPTS="--graph=/mnt/docker --insecure-registry=localhost-alias.com:5000"

localhost-alias.com in DNS with A record to 127.0.0.1
OS X /etc/hosts: use the boot2docker host-only network IP
registry upstart

docker pull public_registry_image
docker run -p 5000:5000 --name registry \
-v /etc/docker-reg:/registry-conf \
-e DOCKER_REGISTRY_REGISTRY_CONFIG=/registry-conf/config.yml \
public_registry_image

@bridgetkromhout
config.yml

s3_region: us-east-1
s3_access_key: <aws-accesskey>
aws-secretkey
s3_secret_key: <aws-secretkey>
s3_bucket: <bucketname>
standalone: true
storage: s3
storage_path: /registry
docker run \
-d \n-p 5000:5000 \n--name docker-reg \n-v ${DFHOME}:${DFHOME} \n-e DOCKER_REGISTRY_CONFIG=${DFHOME}/config/registry/config.yml \npublic_registry_image
S3 requires clock sync

$ docker pull local-repo-alias.com:5000/mysql
Pulling repository local-repo-alias.com:5000/mysql
2014/11/24 19:44:31 HTTP code: 500

$ boot2docker ssh sudo date --set "$(env TZ=UTC date '+%F %H:%M:%S')"
Jenkins-driven image builds
weekly base builds

FROM local-repo-alias.com:5000/www-base

- include infrequently-changing dependencies
  - ubuntu packages
  - pip requirements
  - wheels

- other builds can start from these images (so they’re faster):

@bridgetkromhout
sudo docker build -t="a12fbdc" .
sudo docker run -i -t -w /var/www -e DJANGO_TEST=1 --name test.a12fbdc a12fbdc py.test -s

sudo docker tag a12fbdc local-repo-alias.com:5000/www:'dev'

sudo docker push local-repo-alias.com:5000/www:'dev'

@bridgetkromhout
container-building containers

easier with statically linked binaries

Go microservices
Android APK

@bridgetkromhout
$ docker images

<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
<th>IMAGE ID</th>
<th>CREATED</th>
<th>VIRTUAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>local-repo-alias.com:5000/mysql</td>
<td>dev</td>
<td>b0dc5885f767</td>
<td>2 days ago</td>
<td>905.9 MB</td>
</tr>
<tr>
<td>local-repo-alias.com:5000/www</td>
<td>dev</td>
<td>82cda604a4f1</td>
<td>2 days ago</td>
<td>1.092 GB</td>
</tr>
<tr>
<td>local-repo-alias.com:5000/micro</td>
<td>local</td>
<td>bed20dc84ea1</td>
<td>4 days ago</td>
<td><strong>10.08 MB</strong></td>
</tr>
<tr>
<td>google/golang</td>
<td>1.3</td>
<td>e3934c44b8e4</td>
<td>2 weeks ago</td>
<td>514.3 MB</td>
</tr>
<tr>
<td>public_registry_image</td>
<td>0.6.9</td>
<td>11299d377a9e</td>
<td>6 months ago</td>
<td>454.5 MB</td>
</tr>
<tr>
<td>scratch</td>
<td>latest</td>
<td>511136ea3c5a</td>
<td>18 months ago</td>
<td>0 B</td>
</tr>
</tbody>
</table>

$
a cautionary word on storage drivers

@bridgetkromhout
2014/10/30 21:35:31 Error getting container init rootfs b528d54a0458a8cd8a798309930adb45cb5e1a7430e981e0f3108f86386aab67 from driver devicemapper: open /dev/mapper/docker-9:127-14024705-b528d54a0458a8cd8a798309930adb45cb5e1a7430e981e0f3108f86386aab67-init: no such file or directory
make: *** [build-django] Error 1
Build step 'Execute shell' marked build as failure
Naginator Plugin

- **Retry build after failure**
  - Rerun build for unstable builds as well as failures
  - Rerun build only for failed parts on the matrix
  - Delay before retrying build
    - Fixed delay: 30
  - Progressive

- **Maximum number of successive failed builds**: 3

- **Only rerun build if regular expression is found in output**: checked

- **Regular expression to search for**:
  - (Cannot destroy container!Error getting container init rootfs)
useful for unattended base builds, but...

...seeing this in Slack got old
replace storage driver for jenkins instance

DOCKER_OPTS="--graph=/mnt/docker
--insecure-registry=local-repo-alias.com:5000
--storage-driver=aufs"
ubuntu 14.04: aufs in kernel extras
bash 'install kernel extras for aufs' do
  code <<-EOH
    apt-get -y install linux-image-extra-$\{(uname -r)\}
EOH
end

@bridgetkromhout
Which docker storagedriver should I use?

3:16:50 PM <crosbymichael> ahmetalpbalkan: easy, overlay

(yes, modulo what’s available for your kernel)
for persistent instances

# remove stopped containers
@daily docker rm `docker ps -aq`

# remove images tagged "none"
@daily docker rmi `sudo docker images | grep none | awk -F' ' '{print $3}'`
@bridgetkromhout

deploy using fabric

tag for staging
tag for prod

out of ELB

restart upstart

back in ELB
Autoscaling

Jenkins

Chef

EC2 Instances

AMI factory

Packer

AMI factory

GitHub

@bridgetkromhout
replacing 100s of lines of userdata...

```bash
#!/bin/bash

cat <<EOF > /etc/init/django.conf
description "Run Django containers for www"

start on started docker-reg
stop on runlevel [!2345] or stopped docker
respawn limit 5 30

[...]

@bridgetkromhout
```
...with a chef-client run & packer build.

#!/bin/bash

# upstart configs are now created by chef
rm /etc/chef/client.pem
mkdir -p /var/log/chef
chef-client -r 'role[rolename]' -E 'environment' -L /var/log/chef/chef-client.log

@bridgetkromhout
```bash
docker run \
  -e DJANGO_ENVIRON=PROD \ 
  -e HAPROXY=df/haproxy-prod.cfg \ 
  -p 8000:8000 \ 
  -v /var/log/containers:/var/log \ 
  --name django \ 
  localhost-alias.com:5000/www:prod \ 
  /var/www/bin/start-django
```
docker run \
  <% if @docker_rm == true %>
    --rm \
  <% end %> 

  <% @docker_env.each do |k, v| %>
    -e <%= k %>=<%= v %> \
  <% end %> 

  <% @docker_port.each do |p| %>
    -p <%= p %> \
  <% end %>

  @bridgetkromhout
<% @docker_volume.each do |v| -%>
  -v <%= v %> \
<% end %>

--name <%= @application_name %> \
localhost-alias.com: <%= @registry_port %>:/<%= @docker_image %>:/<%= @docker_tag %> \
<%= @docker_command %>

@bridgetkromhout
using attributes

attribute :command, :kind_of => String, :required => true
attribute :env, :kind_of => Hash, :default => {}
attribute :port, :kind_of => Array, :default => []
attribute :volume, :kind_of => Array, :default => ['/var/log/containers:/var/log']
attribute :rm, :kind_of => [TrueClass, FalseClass], :default => false
attribute :image, :kind_of => String, :required => true
attribute :tag, :kind_of => String, :required => true
attribute :type, :kind_of => String, :required => true
attribute :cron, :kind_of => [TrueClass, FalseClass], :default => false
recipe using LWRP

```ruby
base_docker node['www']['django']['name'] do
  command node['www']['django']['command']
  env node['www'][service]['django'][env]['env']
  image node['www']['django']['image']
  port node['www'][service]['django'][env]['port']
  tag node['www'][service]['django'][env]['tag']
  type node['www']['django']['type']
end
```

@bridgetkromhout
packer for ami building

{
    "type": "chef-client",
    "server_url": "https://api.opscode.com/organizations/dramafever",
    "run_list": [ "base::ami" ],
    "validation_key_path": "\{\{user `chef_validation`\}\}",
    "validation_client_name": "dramafever-validator",
    "node_name": "packer-ami"
}

@bridgetkromhout
packer run
$HOME/packer/packer build \
  -var "account_id=$AWS_ACCOUNT_ID" \
  -var "aws_access_key_id=$AWS_ACCESS_KEY_ID" \
  -var "aws_secret_key=$AWS_SECRET_ACCESS_KEY" \
  -var "x509_cert_path=$AWS_X509_CERT_PATH" \
  -var "x509_key_path=$AWS_X509_KEY_PATH" \
  -var "s3_bucket=bucketname" \
  -var "ami_name=$AMI_NAME" \
  -var "source_ami=$SOURCE_AMI" \
  -var "chef_validation=$CHEF_VAL" \
  -var "chef_client=$HOME/packer/client.rb" \
  -only=amazon-instance \
$HOME/packer/prod.json
limiting packer IAM permissions

```
"Action": [
  "ec2:TerminateInstances",
  "ec2:StopInstances",
  "ec2:DeleteSnapshot",
  "ec2:DetachVolume",
  "ec2:DeleteVolume",
  "ec2:ModifyImageAttribute"
],

"Effect": "Allow",

"Resource": "*",

"Condition": {
  "StringEquals": {
    "ec2:ResourceTag/name": "Packer Builder"
  }
}
```
and now you have a new problem...

@bridgetkromhout
container clustering

evaluating Mesos/Marathon

+/- autoscaling

+/- discovery

@bridgetkromhout
obligatory container disaster

protip: does not represent reality

tl;dr: containers aren’t going to solve all your problems...

...but they aren’t actually all that hard to use, either.

@bridgetkromhout
security

(we focus on host-level security, not isolation...
...and we don’t run arbitrary images from the internets.)
-v /var/log/containers:/var/log

<Input containers_in>
Module im_file
Recursive False
File '/var/log/containers/*.log'
Exec $FileName = file_name();
Exec $raw_event = $FileName + ' ' + $raw_event;
Exec $Message = $raw_event;
</Input>

@bridgetkromhout
monitoring & alerting
Docker in production:
honestly, it’s pretty awesome.

@bridgetkromhout
Thank you!
(and we’re hiring!)
dramafever.com/company/careers.html