

Continuous Delivery

In An AutoScaling Environment

Who am I?

David McKay

Technical Director / Team Frustrator

TeamRock

Code slinger since 1997

Professionally since 2004

Who are TeamRock?

TeamRock would like to claim they're a forwarding thinking 21st century media company...

... and we're getting there, one day at a time.

Show of hands

Who's already in a team utilising:

- Continuous integration or delivery
 - AutoScaling

Why continuous integration / delivery?

“Continuous Integration doesn’t get rid of bugs, but it does make them dramatically easier to find and remove”

Martin Fowler

Why continuous integration / delivery?

For me, the main reasons we adopted CI/D:

1. Confidence (Deployments, Merges)
2. Short feedback loop (When it's built, it's deployed)

Don't make CI/D harder than it has to

Build a single-distributable-runnable artifact

The Build



docker

What is Docker?

Docker is a tool that allows us to provide a single runnable artifact that encapsulates the configuration and dependencies of our applications

Why Docker?

- Isolation
- Lightweight
- Simplicity
- Composable

Docker: Dev, Staging, Production

```
## Dockerfile

FROM teamrock/apache2:development

RUN apt-get -y update && apt-get -y install libapache2-mod-php5

RUN ln -s /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-enabled/ &&
    sed -i 's/html/web/' /etc/apache2/sites-available/000-default.conf

ADD run.sh /usr/local/bin/start.sh

ENTRYPOINT [ "/usr/local/bin/start.sh" ]


## run.sh

#!/bin/sh

cd /var/www

chmod -R 777 /var/www/var/cache /var/www/var/log

/usr/sbin/apache2 -DFOREGROUND
```

Docker: Dev, Staging, Production

```
## docker-compose.yml
database:
  image: mysql
  environment:
    MYSQL_ROOT_PASSWORD: mysql
application:
  build: .
  ports:
    - "6601:80"
  volumes:
    - "/home/docker/symfony2:/var/www"
```

Docker: Dev, Staging, Production

How we used to do it:



How we do it now:



Docker: Dev, Staging, Production

This does leave us with a problem though ...

Vagrant was handling our nfs mounts. How do we get our code on to our Docker container that's living inside a virtual machine?!

Docker: Dev, Staging, Production

```
## bin/sync
#!/bin/sh
ssh-add ~/.ssh/id_boot2docker
chmod 777 var/cache var/log
boot2docker ssh "tce-load -wi rsync > /dev/null 2>&1"
watch -n1 rsync --exclude '.git' --exclude 'var/cache/*'
--exclude 'var/log/*' -rlptDv ./ docker@$(boot2docker ip
2>/dev/null):~/symfony2
```

Docker: Dev, Staging, Production

Live Demo

Done With Development

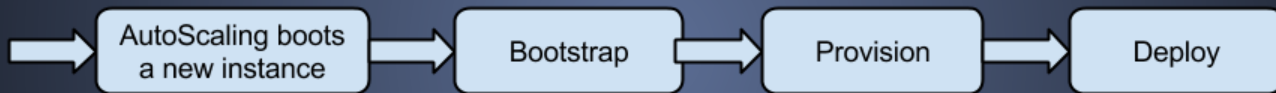
Where's the AutoScaling?

Auto-scaling

AutoScaling is hard, but it promotes good practice (Cattle > Pets)

- New instances need bootstrapped and provisioned before they can be used
- Deploying to a dynamic inventory

Auto-scaling pipeline



Bootstrapping

Bootstrapping takes a fresh machine and gets it prepared for provisioning.

Some people do bake this stage into their AMIs / etc. If you've ever had to update an AMI, you know this sucks

Bootstrapping & Provisioning



<http://saltstack.com/>

Bootstrapping

```
#!/bin/sh
```

```
sudo mkdir /etc/salt
sudo echo -n '{{ vm['instance_id'] }}' > /etc/salt/minion_id
sudo mkdir -p /etc/salt/pki
sudo echo -n '{{ vm['priv_key'] }}' > /etc/salt/pki/minion.pem
sudo echo -n '{{ vm['pub_key'] }}' > /etc/salt/pki/minion.pub
sudo mkdir -p /etc/salt/minion.d
sudo cat > /etc/salt/minion.d/master.conf <<EOF
master: 172.16.224.160
EOF
sudo cat > /etc/salt/minion.d/reauth_delay.conf <<EOF
random_reauth_delay: 5
EOF
sudo locale-gen en_GB.UTF-8
sudo sh -c "sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 36A1D7869245C8950F966E92D8576A8BA88D21E9"
sudo sh -c "echo deb http://get.docker.io/ubuntu docker main > /etc/apt/sources.list.d/docker.list"
sudo apt-get -y update
sudo apt-get install -y python-pip
sudo pip install docker-py
```

```
sudo curl -L https://bootstrap.saltstack.com | sudo sh -s -- -p python-boto
```

SaltStack Vocabulary (For humans)

Top File

The top file is used to map instances to states

Grain

A grain is an attribute of an instance

State

A state describes how a part of the system should be

Pillar

A pillar is a piece of data that can be injected to the machine

Reactor

A scripted reaction to an event

Top File

```
base:
  '*':
    - users.teamrock-developers
    - setup
    {% if salt['grains.get']('ec2_tags:Project', '') %}
    - deploy
    {% endif %}
  'ec2_tags:Environment:Production':
    - match: grain
    - datadog
```

Grains

i-e83cca42:

Environment:

Production

LaunchedBy:

rawkode

Name:

i-e83cca42

aws:autoscaling:groupName:

rawkode-monolith-AutoScalingGroup-1SHLILSAYXJ50

States

```
# Timezone!
```

```
Europe/London:
```

```
  timezone.system:
```

```
    - utc: True
```

```
# mysql.sls
```

```
install-mysql-server:
```

```
  - name: mysql-server
```

```
  pkg:
```

```
    - installed
```

State Modules:

SaltStack already works with your favourite tools:

1. composer
2. docker
3. cron
4. supervisor
5. hipchat
6. more!

<http://docs.saltstack.com/en/latest/ref/modules/all/>

Pillars

```
## environment_variables
```

```
datadog:
```

```
    api_key: i-actually-remembered-to-change-this
```

```
blackbird:
```

```
    port: 6601
```

```
syro:
```

```
    port: 6701
```

Reactors

deploy:

cmd.state.sls:

- tgt: "G@ec2_tags:Project:{{ data['post']['project']
}} and G@ec2_tags:Environment:{{ data['post']
['environment'] }}"
- expr_form: compound
- arg:
 - deploy
 - "pillar={ 'project': '{{ data['post']['project']
}}', 'buildNumber': {{ data['post']['buildNumber'] }} }"

Stitching Everything Together

Live Demo:
AutoScaling Provisioning

Live Demo:
Remote Execution

Questions?

Ask away!

Want to read more, with extra code samples?

<https://medium.com/teamrock-devtech/continuous-delivery-in-an-autoscaling-environment-97e77c4e624e>