CANONICAL

DevOps at Canonical

SREcon15 Europe - Tom Haddon

Managing service orchestration with Juju and Mojo

About Me

- At Canonical for 8+ years
- Started as the first member of what became our DevOps team
- Currently manage a squad of 6 SREs

What this talk is about

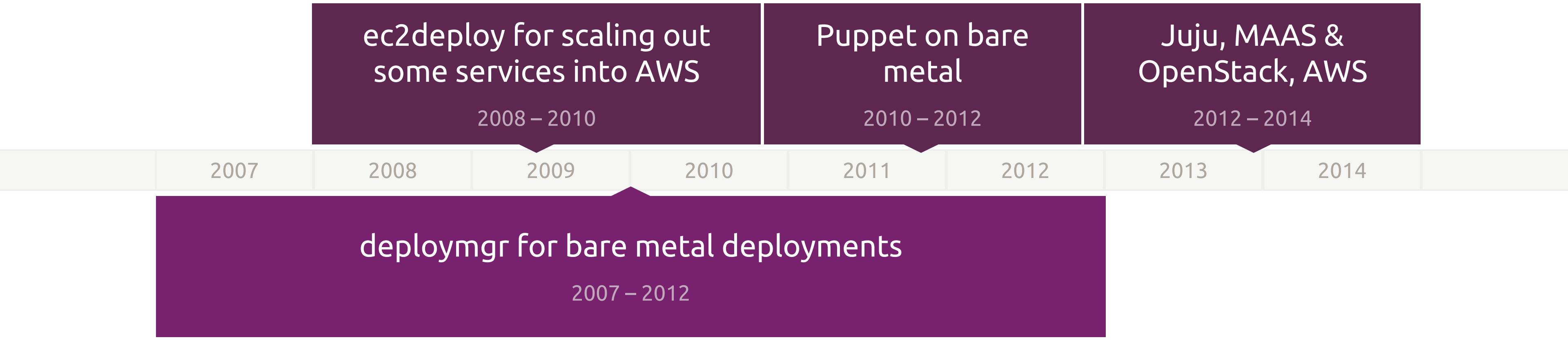
- Brief history of DevOps at Canonical
- What we’re doing now in DevOps
- Intro to Juju & Mojo

Services We Run For Canonical

- 13 development teams
  - 80 developers
  - Supported by 6 SREs
- 240 distinct services
- IT Services for Canonical and Ubuntu Community

Deployment Tools/CM/Orchestration

For new deployments:


ec2deploy for scaling out some services into AWS
2008 – 2010
Puppet on bare metal
2010 – 2012
Juju, MAAS & OpenStack, AWS
2012 – 2014
2007
2008
2009
2010
2011
2012
2013
2014
deploymgr for bare metal deployments
2007 – 2012

Issues We Were Seeing

- Differences between tools developers and SREs using to deploy
- Lack of developer visibility into problems with deployments
- Differences between staging and production services
- Overloaded SREs & poor SRE/developer relations

Where Are We At Today?

Juju & Mojo

(& MAAS, OpenStack, AWS, etc.)

Juju

- Tool allowing modelling of services
- Charms encapsulate service definitions
  - Reusability/shared fixes
- Multiple substrates
  - Baremetal
    - x86, Power, ARM
  - Cloud
    - Private or public clouds (geo-specific services)

Mojo

- Layer on top of Juju providing structure for deployments
- Started life as a CI tool
- As of 2015 also doing full service deployments, service upgrades and scaling of services

Juju

```
juju deploy apache2 --num-units 2
juju deploy content-fetcher
juju deploy nrpe
juju set apache2 servername=mojo.canonical.com enable_modules=ssl
nagios_check_http_params=...
./build-and-upload-content.sh
juju add-relation apache2 content-fetcher
juju add-relation apache2 nrpe
nova floating-ip-associate <server1> <address1>
nova floating-ip-associate <server2> <address2>
```

Mojo

mojo run

Live Demo

Kill mojo.canonical.com environment

Re-deploy from scratch using Mojo

Mojo: specifications & manifests

- Specification for each service
- Specification is a VCS branch
    - can have multiple services in one branch
- Manifest files define what “mojo run” will do
    - deploy ops-ready service
    - verify environment status
    - perform other operations (service upgrade, scaling)

```
# We need the markdown package to be able to generate the docs for Mojo
builddeps packages=make,markdown
# Run the collect step
collect
# Run the build step
build
# Pull in any secrets - this is only used in the production stage
secrets
# Deploy services only
deploy config=services local=services-secrets delay=0
# Copy our built resources to the instances
script config=upload-built-content
# And now deploy relations as well
deploy config=relations
# Run verify steps
include config=manifest-verify
# Run post deploy steps
script config=post-deploy
```

Mojo: phases

- Phases are specific steps within a manifest
  - builddeps
  - collect
  - build
    - inside LXC with no network access
  - script
  - deploy
  - verify

```
# We need the markdown package to be able to generate the docs for Mojo
builddeps packages=make,markdown
# Run the collect step
collect
# Run the build step
build
# Pull in any secrets - this is only used in the production stage
secrets
# Deploy services only
deploy config=services local=services-secrets delay=0
# Copy our built resources to the instances
script config=upload-built-content
# And now deploy relations as well
deploy config=relations
# Run verify steps
include config=manifest-verify
# Run post deploy steps
script config=post-deploy
```

```
#!/bin/bash
# Script to generate docs from Mojo source tree
set -e
set -u

cd ${MOJO_BUILD_DIR}/mojo
make generate-docs
tar cvpf ${MOJO_LOCAL_DIR}/mojo.tar --directory=docs/www .

if [ ${MOJO_STAGE##*/} != "production" ]; then
      # We don't deploy landscape in non-production environments, but we need an
      # dummy secrets file
      echo "mojo-how-to:
      services:
            nrpe:
                        charm: nrpe-external-master" > ${MOJO_LOCAL_DIR}/services-secrets
fi
```

Mojo: secrets

- Secrets kept outside of the specification so it can be shared widely
- Secrets copied into working directory during “mojo run” to be used by Mojo

```
# We need the markdown package to be able to generate the docs for Mojo
builddeps packages=make,markdown
# Run the collect step
collect
# Run the build step
build
# Pull in any secrets - this is only used in the production stage
secrets
# Deploy services only
deploy config=services local=services-secrets delay=0
# Copy our built resources to the instances
script config=upload-built-content
# And now deploy relations as well
deploy config=relations
# Run verify steps
include config=manifest-verify
# Run post deploy steps
script config=post-deploy
```

Mojo: stages

- Stages define differences between how to deploy the same service in different environments e.g:
    - numbers of units
    - instance constraints (“mem=4G”)
    - ops services for production
- Example:
    - `export MOJO_STAGE=mojo-how-to/production && mojo run`

```
# We need the markdown package to be able to generate the docs for Mojo
builddeps packages=make,markdown
# Run the collect step
collect
# Run the build step
build
# Pull in any secrets - this is only used in the production stage
secrets
# Deploy services only
deploy config=services local=services-secrets delay=0
# Copy our built resources to the instances
script config=upload-built-content
# And now deploy relations as well
deploy config=relations
# Run verify steps
include config=manifest-verify
# Run post deploy steps
script config=post-deploy
```

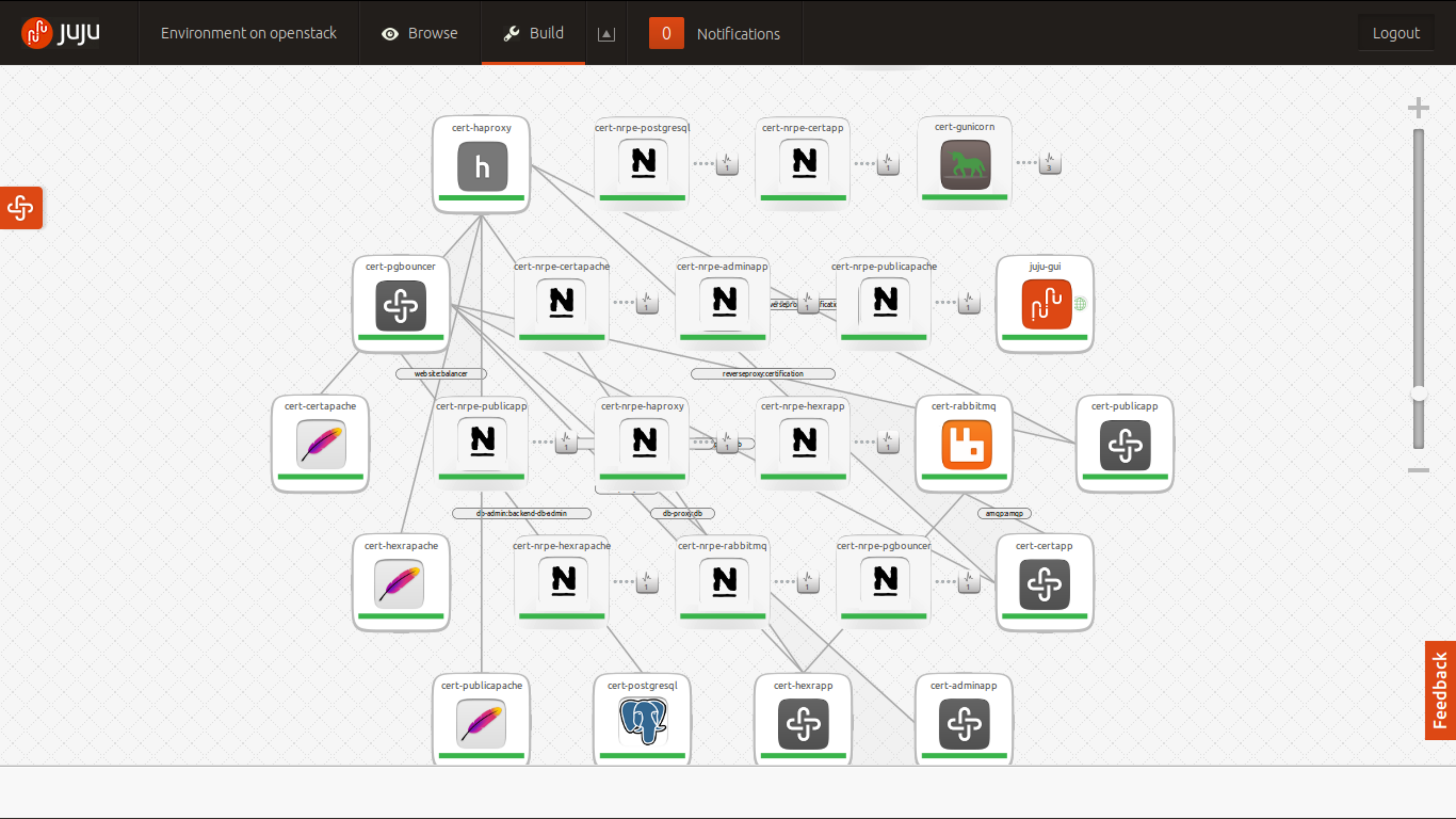
```
--- mojo-how-to/devel/services    2015-05-07 15:01:55.434547845 +0100
+++ mojo-how-to/production/services    2015-05-07 15:01:39.194472327 +0100
@@ -4,17 +4,27 @@
      apache2:
            charm: apache2
            expose: true
-           num_units: 1
+           num_units: 2
            options:
-                 servername: mojo-how-to.example.com
+                 servername: mojo.canonical.com
                        enable_modules: "ssl"
```

```
--- mojo-how-to/devel/services    2015-05-07 15:01:55.434547845 +0100
+++ mojo-how-to/production/services    2015-05-07 15:01:39.194472327 +0100
                  enable_modules: "ssl"
-                 nagios_check_http_params: "-I 127.0.0.1 -H mojo-how-to-example.com -e '200' -s 'Mojo'"
-                 vhost_http_template: 'include-base64://{{spec_dir}}/{{stage}}/../configs/mojo-how-to-vhost-http.template'
-                 ssl_cert: SELFSIGNED
+                 nagios_check_http_params: -I 127.0.0.1 -H mojo.canonical.com -S -e '200' -s 'Mojo'
+                 vhost_http_template: 'include-base64://{{spec_dir}}/{{stage}}/../configs/mojo-how-to-production-vhost-http.template'
+                 vhost_https_template: 'include-base64://{{spec_dir}}/{{stage}}/../configs/mojo-how-to-production-vhost-https.template'
+                 ssl_key: include-base64://{{local_dir}}/mojo.canonical.com.key
+                 ssl_keylocation: mojo.canonical.com.key
+                 ssl_cert: include-base64://{{local_dir}}/mojo.canonical.com.crt
+                 ssl_certlocation: mojo.canonical.com.crt
+                 ssl_chain: include-base64://{{local_dir}}/ssl_chain.crt
+                 ssl_chainlocation: ssl_chain.crt
```

```
--- mojo-how-to/devel/services    2015-05-07 15:01:55.434547845 +0100
+++ mojo-how-to/production/services    2015-05-07 15:01:39.194472327 +0100
      content-fetcher:
                  charm: content-fetcher
                  options:
                              archive_location: file:///home/ubuntu/mojo.tar
                              dest_dir: /srv/mojo
+     landscape:
+                 charm: landscape-client
      nrpe:
                  charm: nrpe-external-master
+     ksplice:
+                 charm: ksplice
```

What Have I Just Seen?

- You can run this yourself against any Juju environment
- Repeatable network-isolated builds
- “Stages” for different versions of services
- Secrets handling
- Scales up to much more complex services
  - www.ubuntu.com/certification


juju
Environment on openstack
Browse
Build
0
Notifications
Logout
cert-haproxy
cert-nrpe-postgresql
cert-nrpe-certapp
cert-gunicorn
cert-pgbouncer
cert-nrpe-certapache
cert-nrpe-adminapp
cert-nrpe-publicapache
juju-gui
website:balancer
reverseproxy:certification
cert-certapache
cert-nrpe-publicapp
cert-nrpe-haproxy
cert-nrpe-hexrapp
cert-rabbitmq
cert-publicapp
db-admin:backend-db-admin
db-proxy:db
amqp:amqp
cert-hexrapache
cert-nrpe-hexrapache
cert-nrpe-rabbitmq
cert-nrpe-pgbouncer
cert-certapp
cert-publicapache
cert-postgresql
cert-hexrapp
cert-adminapp
Feedback

DevOps at Canonical

- Mojo
  - CI env driven by jenkins
  - Development: local provider, AWS, company internal cloud
  - Staging and production: production internal cloud, MAAS, AWS, etc.
- Developers can run staging (and some production) services themselves in our production cloud
  - SREs run service and receive alerts or devs run service and receive alerts

DevOps at Canonical (continued...)

- Read-only access to production services
  - User accounts via our LDAP
  - Apparmor profile to restrict access as role account
- Push-button/triggered deployments
  - For most fixes
  - Deploy from a blessed branch, gated on CI

The Good

- Repeatable service deployments and updates
  - Devs and SREs using same deployment tools
  - Shorten feedback loop for developers
  - Full stack deployment for developers
- Speed of bringing up new services vastly increased
- Scaling out and back in is trivial
  - www.ubuntu.com at release time
  - Prodstack nova-compute

The Good (continued...)

- Quick adoption by developers
  - Had to add compute capacity to our production OpenStack instance twice in first three months of “DevOps solution”
- DevOps ticket queue under control

The Bad

- New tools for developers and SREs to learn
- Writing good Juju charms and Mojo specs is the hard part
- Some parts of our infrastructure still not self-service
    - Firewall
    - DNS
    - SSL certs

The Future

- Ongoing improvements for Mojo and Juju
- Better infrastructure and tools around our deployment story
  - Provide monitoring & trending services
  - Better surfacing of problems with services
- Fixing parts of our infrastructure to be self-service

CANONICAL

Any Questions?

tom.haddon@canonical.com

juju.ubuntu.com
mojo.canonical.com