ubuntu

Juju Charm Testing and Debugging

How Juju provides observability in debugging

Matthew Bruzek - matthew.bruzek@canonical.com

CANONICAL

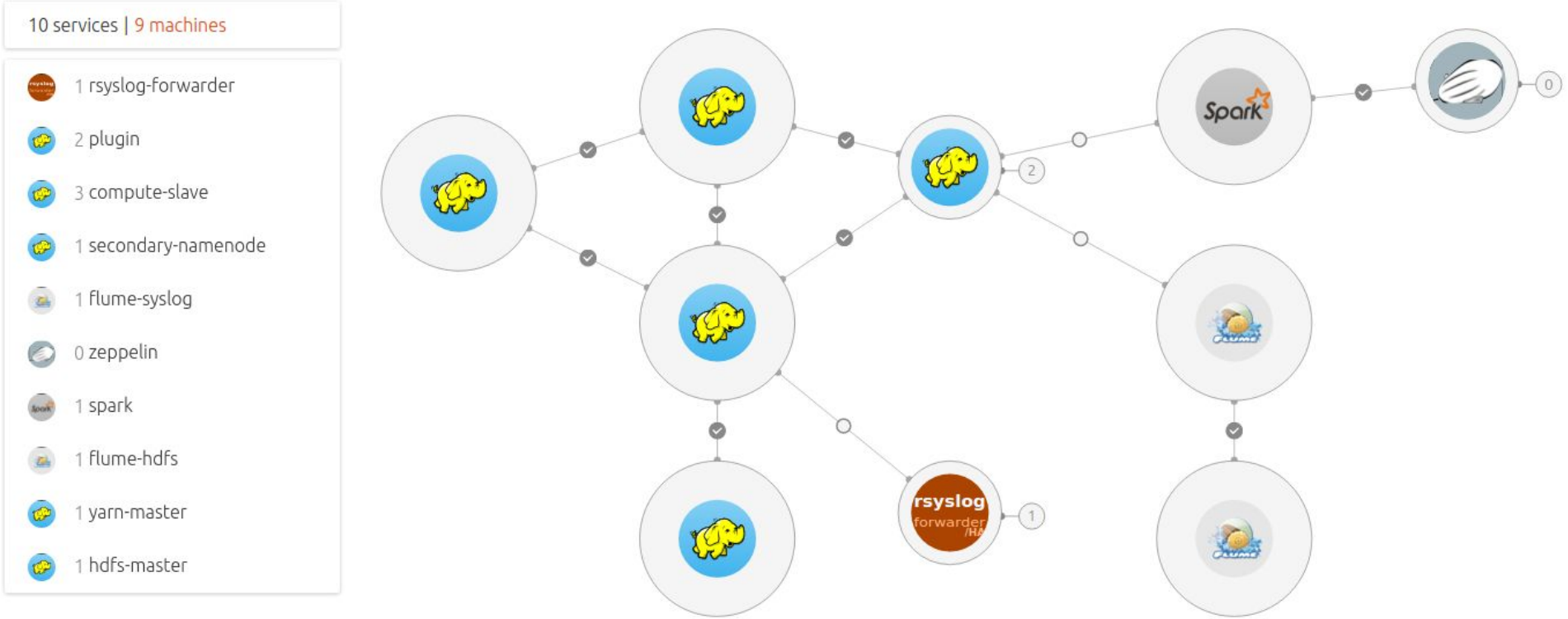
Testing is hard!

but it doesn’t have to be

Modelling makes testing easier

fast, repeatable deployments of your solution

This is what a model looks like:



Juju is application modelling

how applications are managed on public + private clouds and bare metal

https://jujucharms.com/get-started

Charms

application lifecycle management

How do we use test charm with Jenkins?

Here is how we do it

Every time someone submits a charm or update to the charm store:

- A Jenkins job is initiated
- Which fire off test against all clouds
- We get results back
- A charmer reviews the changes to the charm
- The charm is either accepted or the process starts over

Amulet

an integration testing framework for charms

Amulet

A testing framework written in Python.

```
#!/usr/bin/python3
import amulet
deployment = amulet.Deployment(series='trusty')
deployment.add('tomcat', units=2)
deployment.add('memcached')
deployment.add('terracotta')
deployment.relate('tomcat:jndi-memcached', 'memcached:cache')
deployment.relate('tomcat:jndi-terracotta', 'terracotta:dso')
deployment.setup(timeout=1100)
```

Show me the docs: https://jujucharms.com/docs/devel/tools-amulet

Create a “tests/” directory in the charm with executable tests.

Bundletester

Runs tests the charms and models

- To ensure quality Canonical uses bundletester to test charms and models on clouds
- Python source code: https://github.com/juju-solutions/bundletester
- It runs lint checks, unit tests, and functional tests for the charm or model.
- `# bundletester -Fvl DEBUG`

But wait!
You don’t need to install bundletester!

Tests are messy, have lots of requirements and dependencies.
Use disposable containers instead of your own system.

Charmbox

A collection of Juju tools in a disposable, **isolated** container.

- Charmbox is a container that has Juju, charm-tools, bundletester and more installed.
- The container isolates your system from the requirements of the testing tools.
- The tests will no longer install requirements on your system; they get installed in the container which is reset after each use.
- Source: https://github.com/juju-solutions/charmbox

Charmbox

Uses docker for isolation and portability

```
$ docker run --rm \
-v $HOME/.juju:/home/ubuntu/.juju \
-v $JUJU_REPOSITORY/trusty:/home/ubuntu/trusty \
-ti jujusolutions/charmbox
```

--rm	Automatically remove the container when the process exits
-v	Bind mount a volume inside the container
-ti	Associate a tty session and use interactive mode

All together now

Jenkins + Charmbox + Bundletester

…

```
sudo docker pull jujusolutions/charmbox:latest
sudo docker run --rm \
  -u ubuntu \
  -e "JUJU_HOME=/home/ubuntu/.juju" \
  -t jujusolutions/charmbox:latest \
  sudo bundletester -Fvl DEBUG $ARGS
…
# The json results from bundletester get copied to s3 on Amazon for reference.
```

What should I do when it isn’t working?

How modelling with Juju makes it easy to debug

There are lots of debugging tools

That makes it easy to debug charms, and iterate on development

Debugging tools

and other tips and tricks

juju status

The command shows the runtime state of the Juju charms, and machines.

```
$ juju status
```

```
$ watch juju status --format=tabular
```

The output format is yaml in Juju 1.x and tabular on 2.0.

juju debug-log

The debug-log command tails the logs on all units in real time.

Useful as a filtered tail for all Juju messages.

```
$ juju debug-log -n 100 -x 0
```

`juju ssh charm-name/#`

The command gives access to the deployed system.

Useful to debug and diagnose problems on the unit.

`$ juju ssh consul/0`

Gives you access to the system as the “ubuntu” user.

ProTip: Use `**`sudo su`**` to get root authority.

Common directories:

/var/log/juju/unit-[charm-name]-[#].log (The log file for charm-name/#)

/var/lib/juju/agents/unit-[charm-name]-[#]/charm/ (the Charm directory on a running unit)

Pairing

Everyone needs a little help once in a while

`juju debug-hook charm-name/#`

The command puts you in a hook context.

Useful to fix problems with failing hooks.

`$ juju debug-hooks etcd/0`

Gives you access to the system as the “root” user.

ProTip: Once debug-hooks is run. Use `` `juju resolved --retry etcd/0` `` to re-run failed hooks and you get a charm execution environment (hook context).

`juju resolved --retry charm-name/#`

The command re-runs the hook in error state.

Useful to rerun failed hooks.

```
$ juju resolved --retry etcd/0
```

Marks units resolved and retry the failed hooks.

`juju dhx --sync charm-name/#`

The **dhx** plugin enables paired debugging sessions

- Uploads customized preferences of an editor such as **vim**, **emacs**, etc.
- Allows multiple people to join the session, using their ssh keys imported from launchpad or github.
- Synchronizes modified code from the remote unit to your computer after the pairing session is over.

Need to install juju plugins: https://github.com/juju/plugins

Remember

use `juju dhx` instead of `juju debug-hooks`

Demo time

It is debugging; What could go wrong?

Questions

summary : run CI on all clouds now!

https://jujucharms.com/get-started

Thanks for coming