Using Juju with a Local Provider with KVM and LXC in Ubuntu 14.04 LTS

A Dell and Canonical Technical White Paper

Mark Wenning
Canonical Field Engineer

Jose De la Rosa
Dell Software Engineer
## Contents

Introduction............................................................................................................................................... 4
  Purpose.................................................................................................................................................. 4
  A brief overview.................................................................................................................................. 4
Hardware Requirements......................................................................................................................... 5
Installation ................................................................................................................................................ 6
  KVM...................................................................................................................................................... 6
  Juju....................................................................................................................................................... 6
  Juju plug-ins....................................................................................................................................... 6
Configuration ............................................................................................................................................ 7
Start Up Juju............................................................................................................................................. 7
  Spectating.......................................................................................................................................... 7
  Bootstrapping..................................................................................................................................... 8
Container network connectivity ........................................................................................................... 8
Deploying charms ............................................................................................................................... 9
Adding charm relations....................................................................................................................... 10
Debugging ............................................................................................................................................ 10
Stopping ............................................................................................................................................... 11
Conclusion............................................................................................................................................ 12
References and further reading ........................................................................................................ 12
Introduction

Purpose

This whitepaper will show you how to easily deploy cloud services on a single Dell PowerEdge server or a laptop (if that’s all you have). It will illustrate how easily you can take the cloud with you using Linux containers, virtualization and Juju orchestration software from Canonical.

Though Juju is normally used with cloud providers (i.e. Openstack, Amazon AWS) or on bare metal in a MAAS environment, it can also be configured to run on a single machine, deploying charms and relations internally to that machine. This allows developers and users to:

- Experiment with a Juju environment without spending money on hardware or cloud providers.
- Develop Juju charms and applications locally and then deploy to a cloud environment with little or no modification.
- Demo small cloud applications from a laptop.

In this whitepaper, we will set up a small, scalable mediawiki cluster, modeled after the mediawiki:scalable bundle in the Juju charm store. Normally this application is lightweight enough to run as all containers, but we will use VM’s to illustrate the concept.

A brief overview

It is assumed that the reader is familiar with Ubuntu, Juju, virtualization and containers, so we will provide a very brief overview here.

Juju is cloud orchestration software from Canonical used to easily and quickly deploy and manage cloud services. Say you want to deploy a blog site using the open source Wordpress application. This can be done in 3 easy steps:

1. Deploy the Wordpress Juju charm
2. This charm requires a database backend, so deploy the MySQL Juju charm.
3. Associate both charms together.

Even though it’s technically possible to deploy several charms on the same machine, it’s not a good idea as it’s very likely that charms deployed this way will overwrite other charms’ configuration files and bind to the same network ports. A better way to deploy charms on a single machine is to isolate them using Linux containers (LXC) and virtual machines (VMs).

LXC containers are a lighter-weight method of isolating charms, but it must be kept in mind that the underlying kernel is the same as the host. For example, charms meant for a “precise” environment may not work correctly in a “trusty” container. Virtual machines use more system resources but provide excellent isolation. In this case, “precise” charms can actually run in a VM running a “precise” kernel and environment.

Juju has recently added the capability to run a VM as a container. In addition, you can run containers inside these VM’s. This affords a way to safely pack several smaller charms inside one VM, with larger, more compute-intensive charms running in their own VM.
When done, our finished deployment will look as depicted in Figure 1.

![Diagram of Juju deployment](Image)

**Host System**

`lxcb0:10.0.3.1`

**VM 1**

Bridged network

- `lxcb0<>eth0: 10.0.3.227`
- `1/lxc/0` - juju-gui
- `1/lxc/1` - memcached
- `1/lxc/2` - haproxy
- `1/lxc/3` - mysql-slave
- `1/lxc/4` - mysql

**VM 2**

- `eth0:10.0.3.13`
- mediawiki

**Figure 1 - Internal Configuration**

### Hardware Requirements

We ran our tests on several Dell laptops, including a Latitude E7400 laptop and a Precision M4700 workstation, but any Dell system will do as long as you meet the minimum requirements listed in Table 1.

<table>
<thead>
<tr>
<th>Component</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>Dual-core or better</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>8 GB</td>
</tr>
<tr>
<td><strong>Disk</strong></td>
<td>100 GB</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>1 GB NIC, external internet access for downloading media</td>
</tr>
<tr>
<td><strong>Operating System</strong></td>
<td>Ubuntu 14.04 LTS</td>
</tr>
</tbody>
</table>

*Table 1 – Minimum hardware requirements*
Installation

KVM
The first step is to install virtualization packages:

$ sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils
virt-manager qemu-system

virt-manager and qemu-system are nice if you want to watch the VMs be created as the juju charms are brought up.

After installing KVM, it is a good idea to reboot your system.

Juju
As of this writing, the "stock" Trusty install will install juju 1.18.1. In order to use some of the "advanced" features here, we will need at least juju 1.18.3. Therefore, we must install from the juju "stable" PPA (Personal Package Archive):

$ sudo add-apt-repository ppa:juju/stable
$ sudo apt-get update
$ sudo apt-get install juju-core charm-tools juju-local juju-quickstart
$ sudo apt-get install uvtool-libvirt uvtool

Juju plug-ins
Juju plug-ins are a set of scripts that add functionality to Juju. Though optional, they make it much easier to debug and observe what is going on.

Fetch the plug-in source:

$ cd ~/
$ sudo apt-get install git
$ git clone https://github.com/juju/plugins.git ~/.juju-plugins

Update your system path so that juju can find the new scripts:

$ echo 'PATH=$PATH:$HOME/.juju-plugins' >> ~/.bash_profile
$ source ~/.bash_profile

You can run "juju help plugins" to see the new plugin commands.
Configuration

The command ‘juju init’ will build a basic `juju/environments.yaml` file that you can then edit.

$ juju init

Now edit the file. Set the default line near the top to ‘local’:

default: local

Uncomment the line default-series: and set it to “precise”. Bringing up the VM’s as precise allows us access to more charms (as of this writing).

default-series: precise

Further down, uncomment the lines under local: and add the lines

lxc-use-clone: true
container: kvm

The finished local: section should look like this:

```
# https://juju.ubuntu.com/docs/config-local.html
local:
  type: local

# <Commented Section>

  # The default series to deploy the state-server and charms on.
  #
  # default-series: precise
  #
  ## ** add these lines to support KVM and LXC deployment **
  lxc-use-clone: true
  container: kvm
```

Start Up Juju

Spectating

- Start up a second terminal and run “watch juju status” to watch as the charms and machines get created and started. For a more concise status (only showing the services), run “watch juju pprint”.
- Run “virt-manager” to bring up the VM manager GUI. This will show the KVM sessions as juju creates and starts them.
- Once juju-gui is up, you can also log into that service and see the charms and relationships as they are created.
Bootstrapping

Bootstrap Juju and add the first VM. We set the default memory size for VM’s to 2 GB. If you have a machine with a large amount of memory, you can skip this step, or change it to a larger value. The default disk size for a VM is 8 GB.

Machine VM1 will be used to hold all the containers, so we will increase the default memory and disk size.

$ juju bootstrap --debug
$ juju set-constraints mem=2G
$ juju add-machine --constraints "root-disk=16G mem=3G"

If you are watching virt-manager at this point, you should see a VM named “ubuntu-local-machine-1” (or similar) start up. Run “juju status” to see the created VM.

$ juju status
environment: local
machines:
  "0":
    agent-state: started
    agent-version: 1.20.0.1
    dns-name: localhost
    instance-id: localhost
    series: utopic
    state-server-member-status: has-vote
  "1":
    agent-state: started
    agent-version: 1.20.0.1
    dns-name: 10.0.3.155
    instance-id: mwenning-local-machine-1
    series: precise
    hardware: arch=amd64 cpu-cores=1 mem=3072M root-disk=16384M
    services: {}

Container network connectivity

The containers in VM1 that will be created when we start deploying charms will communicate over an lxcbr0 bridge **inside** the VM. In order for these containers to communicate with the “external” network 10.0.3.0 on the host, we must create and configure the lxcbr0 bridge in VM1 **before** deploying our charms, otherwise the bridge will be configured with a different network (i.e. 10.0.4.0) and you will not be able to connect to the containers directly from the host.

Ssh to VM1 using juju. From the output of ‘juju status’ above, we notice that VM1 has machine ID “1”:

$ juju ssh 1

Inside the VM, create a new file **/etc/network/interfaces.d/bridge.cfg** file

$ cd /etc/network/interfaces.d/
$ sudo mv eth0.cfg eth0.cfg.bak
Using Juju with a Local Provider with KVM and LXC in Ubuntu 14.04 LTS

$ sudo nano bridge.cfg

Add the following:

auto eth0
iface eth0 inet manual

auto lxcbr0
iface lxcbr0 inet dhcp
  bridge_ports eth0

Restart the networking service to enable the new network setup.

$ sudo service networking restart
$ ifconfig

eth0 Link encap:Ethernet  HWaddr 52:54:00:dc:a2:5c
  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
  RX packets:137679 errors:0 dropped:0 overruns:0 frame:0
  TX packets:72198 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1000
  RX bytes:214095767 (214.0 MB)  TX bytes:4926208 (4.9 MB)

lo Link encap:Local Loopback
  inet addr:127.0.0.1  Mask:255.0.0.0
  inet6 addr: ::1/128 Scope:Host
  UP LOOPBACK RUNNING  MTU:16436  Metric:1
  RX packets:16 errors:0 dropped:0 overruns:0 frame:0
  TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:0
  RX bytes:1296 (1.2 KB)  TX bytes:1296 (1.2 KB)

lxcbr0 Link encap:Ethernet  HWaddr 52:54:00:dc:a2:5c
  inet addr:10.0.3.227  Bcast:10.0.3.255  Mask:255.255.255.0
  inet6 addr: fe80::7855:eff:fe21:64d5/64 Scope:Link
  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
  RX packets:420 errors:0 dropped:0 overruns:0 frame:0
  TX packets:234 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:0
  RX bytes:582073 (582.0 KB)  TX bytes:17240 (17.2 KB)

Note that lxcbr0 has an IP address in the 10.0.3.0 network and is visible to the “external” network. Now you should be ready to start deploying charms.

**Deploying charms**

Back on the host, start deploying charms to VM1:

$ juju deploy juju-gui --to lxc:1
Please note that deployment of the juju-gui charm can take from 5 to 15 minutes, depending on the speed of your network connectivity.

Deploy the rest of the charms to containers inside VM1:

```bash
$ juju deploy memcached --to lxc:1
$ juju deploy mysql --to lxc:1
$ juju deploy mysql mysql-slave --to lxc:1
$ juju deploy haproxy --to lxc:1
```

Note that we have renamed the slave mysql unit as "mysql-slave" to differentiate it from the master mysql.

VM2 will be used to run the mediawiki charm without using a container. Normally this would only be used for compute-intensive charms or those that need a specific environment. This isn’t really the case for mediawiki; we are doing it here to illustrate the idea.

```bash
$ juju deploy mediawiki
```

If you are monitoring using virt-manager, it should show a new VM starting up named “ubuntu-local-machine-2” (or similar).

**Adding charm relations**

Add the relations between charms to complete the configuration.

```bash
$ juju add-relation mediawiki:cache memcached:cache
$ juju add-relation mediawiki:db mysql:db
$ juju add-relation mysql:master mysql-slave:slave
$ juju add-relation haproxy:reverseproxy mediawiki:website
$ juju expose haproxy
```

Run “juju status haproxy” to get the IP address of the haproxy charm to view the content in a browser.

Run “juju status juju-gui” to get the IP address of the Juju gui, and view it in a browser to see the created charms and relations. The final charm deployment should look as show in Figure 2.

**Debugging**

If a charm fails to deploy (i.e. a hook fails), you can ssh into the VM or container where the charm is running and take a look at the files in `/var/log/` to get details on what happened.

When a charm is deployed to a VM, use:

```bash
$ juju ssh <machine>
```

Where `<machine>` is the ID of the machine where the charm was deployed to. You can use “juju status <service>” to get the machine ID. For example:

```bash
$ juju ssh 1
```
When a charm is deployed to a container inside a VM, use:

```
$ juju ssh <machine>/lxc/<container>
```

Where both `<machine>` and `<container>` can be obtained with “juju status”. For example:

```
$ juju ssh 1/lxc/0
```

Once the issue has been addressed, you need to tell Juju about it. You can ‘resolve’ it in one of two ways:

1) From the command line, run "juju resolved -r <unit>", i.e. if mysql is broken, "juju resolved -r mysql/0"

2) In Juju-gui, double-click on the charm where the error occurred. A screen will show up on the right; if you pull down the “1 unit running” section, a set of buttons/checkboxes will show up. You can select the unit (or select all) and hit the resolve button to clear it up.

![Diagram](image)

**Figure 2** - Finished configuration as shown in juju-gui.

**Stopping**

To stop and destroy all virtual machines and the containers inside of them, run:

```
$ juju destroy-environment local
```
Conclusion

As you can see, setting up a cloud environment on your laptop is fast and easy. Juju provides a complete cloud environment whether deploying to bare-metal, a cloud provider or your laptop.

References and further reading

<table>
<thead>
<tr>
<th>Guide</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam Stokes’ excellent article on deploying LXC’s and KVM in the local environment</td>
<td><a href="http://astokes.org/juju-deploy-to-lxc-and-kvm-in-the-local-provider/">http://astokes.org/juju-deploy-to-lxc-and-kvm-in-the-local-provider/</a></td>
</tr>
<tr>
<td>Installing Juju plugins, including juju pprint</td>
<td><a href="https://github.com/juju/plugins/blob/master/README.md">https://github.com/juju/plugins/blob/master/README.md</a></td>
</tr>
<tr>
<td>Deploying charms to specific machines with lxc’s</td>
<td><a href="https://juju.ubuntu.com/docs/charms-deploying.html">https://juju.ubuntu.com/docs/charms-deploying.html</a></td>
</tr>
</tbody>
</table>