

Politecnico di Milano
Facoltà di Ingegneria dell'Informazione



Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e
Bioingegneria

Towards Virtual Machine Consolidation in
OpenStack

Advisor: Sam Jesus Alejandro Guinea Montalvo

Master thesis by:

Giacomo Bresciani matr. 804979

Lorenzo Affetti matr. 799284

Academic Year 2013-2014

dedica...

Ringraziamenti

Milano, 1 Aprile 2005

Affear

Table of Contents

List of Figures	ix
List of Tables	1
1 Introduction	3
2 State of the art	5
2.1 Introduction	5
2.2 Virtual Machine consolidation	6
2.3 Cloud test environments	6
2.3.1 Vagrant	6
2.3.2 Chef	7
2.3.3 Puppet	10
2.3.4 Docker	10
2.3.5 Dockentack	10
3 aDock	11
4 FakeStack	13
5 Simulation tools	15
6 Nova Consolidator	17
7 Evaluations	19
8 Conclusions and future works	21

TABLE OF CONTENTS

Appendices	23
Bibliography	25

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 2

State of the art

2.1 Introduction

At the beginning of the development of the thesis we were mainly focused on the implementation of a module for OpenStack that would allow to implement different consolidation algorithms and test them to see their real impact on a cloud system in terms of resource allocation. During the first phases we faced with the problem of running, testing and benchmarking our code in an OpenStack environment: to deal with aspects like scheduling, VM placement, and consolidation we needed an highly configurable system that would allow us to run simulations and benchmark to evaluate the goodness of our solution. We wanted it to be fully customizable to match different requirements and let the user customize a lot of aspect such as the structure of the environments, the number of compute nodes (the nodes that host the Virtual Machines), their fake characteristics or the OpenStack services to run. Secondly we needed a way to automatically simulate, in a repeatable way, the workload generated from user applications that normally run on an OpenStack installation. At last we realized that it would be very useful to show the real time data of the simulations to analyze the behavior of the system in different configurations. Therefore we decided to develop aDock, a suite of tools for creating performant, sandboxed, and configurable cloud infrastructure experimentation environments that developer, sysadmins and researchers can exploit to access a fully functional cloud installation of OpenStack. For that reason this chapter is divided into two sections that describe the state of the arts of Virtual Machine

consolidation and of Cloud test environments.

2.2 Virtual Machine consolidation

In the cloud world is fundamental, as in any engineering field, to be able to test environment configurations and algorithms, both to analyze the behavior of tested code that integrates with a real environment and to benchmark and collect data for researches and experimentations. Unfortunately it can be expensive and complex to create and manage a cloud test environment in terms of time, resources and expertises, especially if the hardware resources like server machines or network infrastructures are limited. Fully understand and handle an OpenStack installation is not easy, especially for non sysadmins (Find a better way to say it) like developers or researchers, it has infact an high learning curve and often it is necessary a lot of time to achieve a good and desired result. To reduce the impact of these complications are available some tools that make the process of setting up a cloud infrastructure experimentation environment more easy and manageable. The three main ones are Chef, Puppet, and Docketstack. In order to achieve the goal of running the desired sandboxed environment on limited hardware resources the three solution are used together with two core technologies for the cloud world, Vagrant and Containers (Docker), both outlined below.

2.3 Cloud test environments

2.3.1 Vagrant

Vagrant¹ is a virtualization framework for creating, configuring and managing development environments, written in Ruby, that allows to virtual development environments. It is a wrapper around virtualization software such as VirtualBox, KVM, VMware and could be used together with configuration management tools such as Chef and Puppet. Thanks to an online repository ² it is possible to automatically download a Vagrant Box and run it with a single

¹www.vagrantup.com

²www.vagrantcloud.com

2.3 Cloud test environments

command: `vagrant up vagrant-box-name`. It is also possible to create and configure custom Vagrant Box by simply writing a Vagrantfile:

```
1 box      = 'trusty64'
2 url      = 'http://files.vagrantup.com/precise32.box'
3 hostname = 'customtrustybox'
4 domain   = 'example.com'
5 ip       = '192.168.0.42'
6 ram      = '2048'
7
8 Vagrant::Config.run do |config|
9   config.vm.box = box
10  config.vm.box_url = url
11  config.vm.host_name = hostname + '.' + domain
12  config.vm.network :hostonly, ip
13
14  config.vm.customize [
15    'modifyvm', :id,
16    '--name', hostname,
17    '--memory', ram
18  ]
19 end
```

Provisioners in Vagrant allows to automatically install and configure software in a Vagrant Box as part of the “vagrant up” process, therefore you can start with a base Vagrant Box, adapt it to your needs and eventually share it with other developers who can reproduce the same virtual development environment. Vagrant in combination with configuration management software such as Chef and Puppet is used to create repeatable and easy to setup development and test environments that rely on Virtual Machines.

2.3.2 Chef

Description Chef³ is a configuration management tool used to streamline the task of configuring and maintaining servers in a cloud environment and can be integrated with cloud-based platforms such as Rackspace, Amazon EC2, Google Cloud Platform, OpenStack and others. It is written in Ruby and

³www.chef.io

Erlang and uses a domain-specific language (DSL)⁴ for writing configuration files called “recipes”. “Recipes” are used to define in a declarative way the state of certain resources and define everything that is required to configure different parts of the system: they can contain the definition of software that should be installed and all the required dependencies, services that should be running or files that should be written. Given a “recipe” Chef ensures that all the software is installed in the right order and that each resource state is reached, eventually correcting those resources in a undesired state; “recipes” can be collected into “coobooks” to be more maintainable and powerful.

A base installation of Chef is composed by three main components, a `chef-server` that orchestrates all the Chef processes, multiple `chef-client` found on all the servers, and the user workstation that communicates with the Chef Server to launch commands.

To simplify the communication with the `chef-server` Chef provides a command-line tool called Knife that helps users to manage nodes, “cookbook” and “recipes”, and the majority of possible operations.

Chef and OpenStack Chef and OpenStack can be combined and used together in different ways, many of which have a different goal compared to our thesis. Is it possible in fact to deploy and manage a production OpenStack installation running on multiple servers and supervised by a Chef Server using the subcommand `knife openstack` to control the OpenStack APIs through Chef and thus instantiate new physical servers with a chef-client installed or turn them off (`knife openstack server create / delete`). In this situation you can achieve a “1 + N” configuration that is one OpenStack Controller and N OpenStack Nodes, and the OpenStack services are predefined and you cannot configure an ad hoc configuration. Therefore an “All-in-One Compute” configuration can be chosen where all the OpenStack services (both of Controller and Compute) are installed on a single node.

These configurations can be achieved with the help of Vagrant that will cover all the steps to install OpenStack on a virtual machine and configure all its services (excluded Block Storage, Object Storage, Metering, and Orchestration). Within the OpenStack chef-repo⁵ there is a Vagrantfile that configure a

⁴A programming language specialized to a particular application domain.

⁵<https://github.com/stackforge/openstack-chef-repo>

2.3 Cloud test environments

VirtualBox virtual machine that will host and All-in-One installation. Here is a part of it:

```
1 machine 'controller' do
2   add_machine_options vagrant_config: controller_config
3   role 'allinone-compute'
4   role 'os-image-upload'
5
6   chef_environment 'vagrant-aio-nova'
7   file('/etc/chef/openstack_data_bag_secret',
8        "#{File.dirname(__FILE__)}/.chef/encrypted_data_bag_secret")
9   converge true
```

Of course it is possible to setup a “1 + N” configuration using different Vagrantfile to create and configure one VM to host the Controller and N VMs to host the Compute nodes. However it is unlikely to succeed in running a lot of VMs on the same host especially if they will contain a fully functional OpenStack installation as a Virtual Machine typically require a significant amount of resources to operate.

...Insert timing to install OpenStack with Chef...

Pro and Cons Chef, as seen, is therefore a very powerful tool to create, manage and configure cloud environments and offers a lot of functionalities to structure the desired architecture. In combination with Vagrant is also useful to setup test environments for development or research purposes.

However, with regard to this last aspect, it has several limitations:

- *Heaviness*: due the greed of resources of a Virtual Machine is very difficult to achieve a “1 + N” configuration for development or research purpose on a single machine. On the other hand the “All-in-One Compute” solution that allows a full OpenStack on a single Virtual Machine is very simplistic and doesn’t represent a real environment setting as it runs all the OpenStack services on a single node.
- *Lack of customization*: at the state of the art all of the described solutions install both the Controller node and the Compute node with a predefined set of installed service (in particular are installed all the OpenStack service excluded Object Storage, Metering, and Orchestration) so it is

State of the art

not possible to setup the environment with more or less services or new ones (as in our case).

- think others...

2.3.3 Puppet

Description Similarly to Chef (described in section 2.3.2)

2.3.4 Docker

2.3.5 Dockentack

Chapter 3

aDock

Chapter 4

FakeStack

Chapter 5

Simulation tools

Chapter 6

Nova Consolidator

Chapter 7

Evaluations

Chapter 8

Conclusions and future works

Appendices

Bibliography