CSC3095:Web Platform for Digital Deployment of Virtual Servers

Plamen Kolev
Student number : 130221960
Supervisor : Neil Speirs

03.03.2017

1 Declaration

I declare that this dissertation represents my own work except where otherwise stated.

2 Acknowledgements

3 Abstract

Contents

List of Figures

4 Introduction

As of the year 2016, there are currently three billion people that have access to the internet [1]. Google handles between two and three billion search queries per day [5]. Dealing with so many requests on daily basis requires full utilisation of the available hardware in terms of resources. Part of Google's ability to scale and be efficient is due to the emergence of cloud infrastructure. The topic of the paper deals with one of the building blocks of cloud computing, which is virtualisation.

> "The quickest and cheapest method to providing the necessary level of abstraction in terms of server resource is currently virtualisation..."
>
> — Paul Robinson, Google Cloud Computing **(author?)**

The term virtualisation is defined as the ability of one piece of hardware to run multiple operating systems [3]. In this paper, a virtual machine, or an instance, is an operating system that runs on top of a "physical" operating system. The "physical" operating system is often referred to as the "host" system. Creating a platform that uses such technology enables an organisation to quickly set up any environment (operating system) that can be used in a variety of cases.

A company that wants to buy a high performing computer for each employee would requires physical access to perform repairs and maintenance and management. Physical systems are also more difficult to manage due to their non-central distribution. Another downside is non-scalable hardware utilisation, a case where one machine uses maximum resources but another one is idle. These are some problems that virtualisation technology can deal with. It is now a core part of most new desktop Intel processors and is integral part of all server-grade processors.

This helps with performance, as a virtual machine can be configured on the fly to use flexible amount of resources or even use shared pool of computing. This technology also allows for easy server migration, a physical machine cannot be moved within couple of minutes to a different continent. Physical infrastructure is also prone to hardware-related bugs, as a distributed software solution might not have been tested on all possible computing nodes that run it.

How does virtualisation allow for easy migrations? It achieves its task by abstracting away the available physical hard drives, on the virtualisation host, they appear as a large pool of partitions with the data ready for access. Having access to the virtualisation host allows the content of such system to be copied all at once to a different provider with the only requirement that it must have enough capacity to store the information. The alternative without using a virtualisation software would be to gain physical access to each hard drive individually and copying the data incrementally.

4.1 Purpouse

The project aims to utilise open-source virtualisation technology and make the process of managing and creating virtual machines automated through a web interface. A system manager should be able to open a website, fill in a web form with enough information about the desired operating system, click a button and create it.

The manager should also be able to obtain and generate credentials for that machine, as well as mark common packages for installation on it. The solution should also show performance statistics and allow for network port management. These features, alongside the benefits of virtualisation should create a strong and secure infrastructure for many applications, from virtual office workstation, to server testing and deployment.

Strong cryptography, virtual guest isolation, firewall rules and best practice credential and authentication methodologies and practices will be explored and will be the pillars of the system's security.

4.2 Aim and Objectives

4.2.1 Aim

Give developers a platform for easy deployment, management and monitoring of virtual servers

4.2.2 Objectives

1. Deploy a virtual machine of the user's choice through shell scripts
    - The main feature of the solution is the deployment of the virtual machine instances. The following will be achieved by using Oracle's virtualisation documentation for Virtualbox and the shell scripting language and the automation tool chef.
2. Configure firewall settings
    - Will be achieved through the virtualisation technology's API. Will add extra security layer to the guest operating system.
3. Allow console access and set up authentication credentials (SSH keys) for the instances
    - The main usage of the application is to obtain a shell access to the virtual machine
4. Monitor disk/CPU usage of the virtual instances
5. Allow the user to install software from a predefined list
6. Create a website that will manage and create the machines on the behalf of the user

4.3 Outline

5 Background

5.1 Similar in the field

For the duration of my dissertation, I have evaluated a variety of technologies, frameworks and utilities that will allow me to achieve the goal of automated system deployment. The trial and evaluation of these technologies allowed me to find the most suitable and relevant methodology for moving forward. I also experimented with different "cloud" virtual machine deployment services to understand and evaluate the key requirements that ought to be provided by the platform that I am building.

5.2 Virtualisation

5.3 Addoption

5.4 Application

6 Methodology

6.1 Overview

This chapter aims to give a more in-depth view into the process of scoping, developing, planning and building the application. This ranges from picking the tools to limiting the scope, making assumptions, as well as the scheduling of each task. Creating such a platform from nothing is a daunting task even for an experienced senior software engineer.

For the above reason, the initial research into building the right tool-chain was essential to the success of the project.

The development methodology can be broken into the following steps:

1. Research into virtualisation technology to understand the underlying architecture.

2. Select free and open-source software components that can be coupled together to orchestrate and utilise above mentioned technology. The public libraries used also allowed me to extend, modify and build on top of during the development process.

3. Build a set of core and advanced features. Core features are essential for a good operational and usable solution. Advanced features are the things that make this solution go above and beyond the scope of existing applications.

4. Integrate components into one final solution

5. Do sufficient testing to ensure that the platform behaves as expected.

6.2 Planning

When creating the project plan, two core stages were in mind. The first stage was semester one. During that period, I laid the ground work on which I based all the work that went into developing and testing. This was also a critical moment into the building process, as it allowed me to find materials created by the leading experts in the field (Amazon, Intel, oracle) and learn about the various aspects of building cloud-ready applications, infrastructure security and automation.

This research helped when rationing about the features to implement, helped me understand the scope and challenges and gave me an insight into why this problem requires solving, and it gave a detailed overview of how to solve it.

The first semester was not only about researching applications within the scope of what I was trying to achieve, but also about describing the constrains, limitations and infrastructure decisions I have to make in order to demonstrate a ready product on a production environment.

6.3 Constrains

Because of the limited duriation of the disseration and the immence work that has to be carried throughout the implementation (testing, documentation, development, debugging, planning, etc.) certain constrains had to be put in place to ensure that the project will be completed withing the appropariate period.

The initial constrain that I have defined is limiting the solution to a single host. This work is done as a proof-of-concept system that has the potential of being scaled up and extended to support "real-world" application. Creating a distributed and scalable implementation of this work goes beyond the scope. This is not to say that it is limited to single host, but testing and planning guarantee that it will be fully functional under such condition.

As a future task, it would be possible to deploy multiple instances of the underlying infrastructure that are managed and synchronised with a technology like puppet or chef. Then, creating virtual machines can be done via a manager machine that picks a host based on a round-robin algorithm or based on load. Additional implementation steps must be carried out to ensure that the web platform and the manager understands the physical location of the virtual machine. Physical location references to the real hardware (the host) running the virtual machine.

Another assumption and a constrain is the availability of a network interface. To simplify the development process and allow me to isolate different environments. A network environment is the space in which all the virtual machines have visibility of one another. It also makes it possible to allocate different network addresses dynamically and verify availability of resources. Another important bit is the fact that one must be connected to the network environment to communicate with the machines inside of it. I have taken advantage of VirtualBox's host-only networking interfaces, and have created multiple instances that allow me to create machines independent of each other when developing and testing.

When further extending the solution, additional work has to be carried out to ensure integration with a public DNS server, and some adjustments ought to be made to allow the production DNS to allocate DHCP ip addresses to the virtual machines.

The solution aims to orchestrate and automate generic virtual machine deployment, but creating a solution that enables every operating system to be deployable would require extensive amount of extra work. Future work can make it possible to deploy Windows, Mac OS and more linux variants as part of the infrastructure, but this is yet not the case.

Windows is not supported yet, as it has completely different architecture and has a separate interface for managing networking, users, installation and permissions. Doing windows deployment also requires licensing of their operating system. Another constrain in working with windows is that not many core components are configurable or accessable via a script making automation very difficult or impossible.

Mac OSX virtual deployment was also something left for future. It also has

similar issues to windows virtualisation (licensing, exposed API).

Finally, as it comes to linux virtual machine deployment, I have opted to cover

6.4 Functional requirements

6.5 Non-functional requirements

6.6 Tools

7 Development

A hypervisor is a computer software or hardware that creates the platform layer to a virtual machine and makes it possible to host multiple instances with different configurations [4].

Initially, I was planning on using a hypervisor virtualisation solution such as Virtualbox and VMware, and use the bash shell scripting language to interface directly with them. With this in mind, I decided to opt for Virtualbox, as it is an open-source software solution, has strong community, allows third party plug-ins There are other alternatives (gnome boxes, CHROOT), but what draw me to it is that it is configurable and command-line controllable straight out of the box. Another deciding factor was also the extensibility that is offered by third party modules developed by the community.

Another reason for not choosing a more unpopular or niche technology is due to the willingness of an adopter to switch their entire virtual infrastructure to it. The Virtualbox software is very popular, and as such, it would make the transition much more subtle and unnoticeable

As part of my research, I went through the Virtualbox command-line API. In the process of doing so, I discovered the variety of customisable parameters when building a virtual machine, like the different possibilities in setting up networks (private, network shared, host only, etc.), exposing ports, as well as different hardware configurations (RAM, hard drive, CPU count). After using the command line tools and writing a simple shell scripts for managing creation of different resources, I realised that a tool needs to be written to manage the different technical details of each virtual machine. Luckily, an open-source hypervisor provisioner software is available that helps with automation and management that satisfies the operational needs of the system.

7.1 Vagrant

The tool is called Vagrant by Hashicorp, a company specialising in easy virtual machine configuration management and set-up Their API provides the following features.

- Allow the administrator to create a vagrant configuration file (called Vagrantfile), which describes end-to-end each operation that will be per-

formed to the virtual machine instance (including the Linux distribution that will get installed on it).

- Create a system compatible with their platform starting from a bare configuration and incrementally building a bespoke instance. These instances can be managed and shared on the internet.
- Managing network interfaces
- Configure ports
- Allow ssh access

7.2 Architecture

When considering the architecture of the system, a list of factors were the driving factor when building the solution. I wanted to ensure that The solution has as little moving parts as possible, which will allow it to be flexible and extensible. For that purpose I chose the programming language ruby, as it is a suitable language for system automation. Its suitability comes from the available tools vagrant and puppet which allow end-to end integration when it comes to automation. The ruby programming language also draws inspiration from the Perl programming language One of the Perl's strengths is its ability to be a more powerful shell scripting language. This is the case as it has quick syntax for executing shell scripts directly and get STDOUT, STDERR and exit-codes quickly. It does also has native implementation of common UNIX tools, like grep, wget and awk.

Because the Ruby language draws inspiration and shares similar features with Perl, it makes it a great modern and powerful language to use in this domain-specific problem.

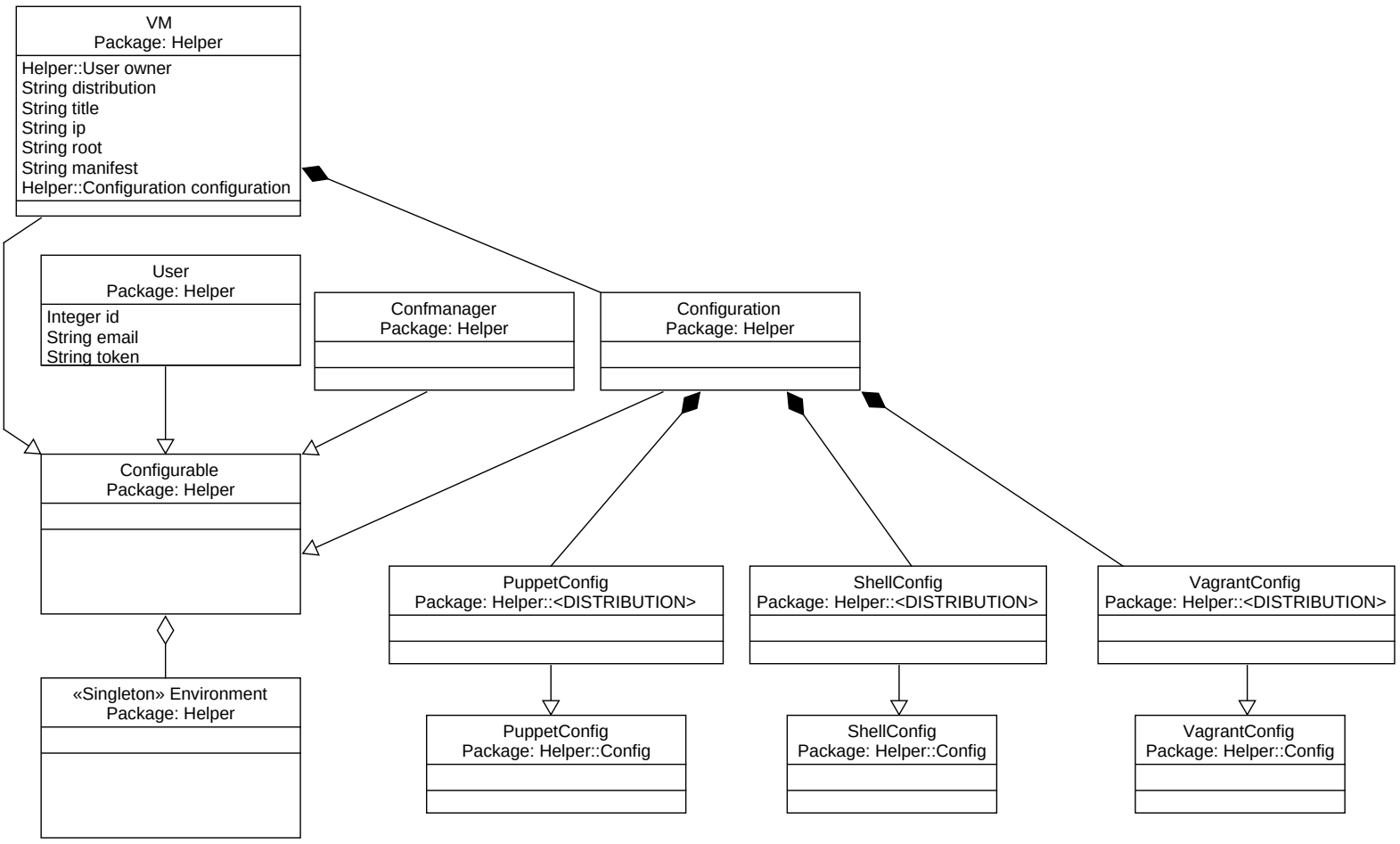
The framework I created uses a virtual machine instance object and authenticated user that owns it. When the user fills the web form with the relevant information, it will pass the configuration values directly to a virtual machine class. This class contains a dynamic configuration, which ensures that the solution is distribution agnostic. Here, the term distribution references to the different versions of Linux (ubuntu, fedora, Centos, arch Linux). Separating dependency specific configuration by sub-classing a configuration object allows for a quick way for the project to be extended to accommodate for new distributions.

For the purpose of this work, I have decided to limit the scope of allowed Linux distributions to Ubuntu, Centos and debian, but it is important to note that the architecture allows extending that list.

To do so, folder with the name of the new distribution must be placed in lib/configurations that describes the configuration specific settings that include updating and installing common software packages.

The framework also uses UNIX environmental variable to manage infrastructure state. For the purpose of developing and testing, I have created two main

isolated environments - testing and development. The most notable usage of such environment is the virtual host-only network that ensures that the virtual interfaces do not clash during testing and development. The two interfaces are `vboxnet0` and `vboxnet1`. One is for general purpose development, the another is for testing. A special host-only adapter is used to have an isolated ip range that is network agnostic, for example, on a Nat-network, some IP addresses might be reserved or used, this case will cause a clash and the virtual machine will not be reachable.


VM
Package: Helper
Helper::User owner
String distribution
String title
String ip
String root
String manifest
Helper::Configuration configuration
User
Package: Helper
Integer id
String email
String token
Confmanager
Package: Helper
Configuration
Package: Helper
Configurable
Package: Helper
«Singleton» Environment
Package: Helper
PuppetConfig
Package: Helper::<DISTRIBUTION>
ShellConfig
Package: Helper::<DISTRIBUTION>
VagrantConfig
Package: Helper::<DISTRIBUTION>
PuppetConfig
Package: Helper::Config
ShellConfig
Package: Helper::Config
VagrantConfig
Package: Helper::Config

8 Testing

9 Evaluation

10 Conclusion

11 References

References

[1] ITU ICT. Ict facts and figures 2016. Accessed: March 08, 2017.

[2] Javvin Technologies Inc. *Network Protocols Handbook*. Javvin Technologies, 2005.

[3] Techopedia Inc. Definition of virtualization. Accessed: November 26, 2016.

[4] Shannon Meier. *IBM Systems Virtualization: Servers, Storage, and Software*. IBM, April 2008.

[5] Internet Live Stats. Google search statistics. Accessed: November 26, 2016.

[6] Vic (J.R.) Winkler. *Securing the Cloud*. Elsevier/Syngress, 2011.

12 Glossary

Glossary

bash Bourn again shell, a UNIX command line virtual shell language. 5

Nat-network A technique where one IP address is mapped to more internal ip addresses, used for security and reduced number of ips. [2] . 7

open-source Software that makes its code openly available, allows derived work without restrictions and its distribution is done freely. 5

operating system The platform on which all user and system software is running. Examples are Windows, Linux, OSX, Android, iOSx. 1

ssh SSH: secure shell, a network protocol that allows a remote connection to a terminal . 6

13 Appendix