University of Brighton

Interim planning and Investigation report

Generation of Raspbian images

Adam Pietrzycki
14843569

supervised by
Dr. Aidan Delaney

November 30, 2016

Abstract

As of the 8th September 2016, in a blog post by raspberry pi founder Eben Upton [1], mentions that the ten millionth raspberry pi has just been sold. The official operating system for these devices is called Raspbian, it is a port of Debian which is available as a standalone image or in a 'New Out Of the Box Software' package for beginners. 'NOOBS' is pre-installed on SD cards that can be bought from many retailers, before the images can be burned to the cards they need to be somehow generated. The current method of generating these images can be found on GitHub [2], it is a set of shell scripts which from the commits look to be predominantly maintained by a single developer. The problem with this is that if the developer decided to move on and depart from the project it might take some time before someone else understands the code well enough to be able to carry it on; this can be simply described as the 'Bus Factor.' [3] Since the release of the raspberry pi in February 2012, new tools have been developed and standards decided; so it might be a nice idea to freshen up the current method.

Contents

Chapter 1

Introduction

1.1 Aims

The overall aim of the project is to show that the current code base, which generates the Raspbian images can be re-written and re-designed to make use of open source tools and a 'human friendly data serialization standard,' [4] called YAML.

1.2 Objectives

The objectives can be separated into two different categories; project and personal. These will provide an idea of the project structure and will help towards a successful completion.

1.2.1 Project

- Write code that utilizes open source tools and language standards.
- Generate Raspbian images at a faster speed.
- Give options to users for easy image customization.
- Get feedback from the Raspberry Pi foundation.

1.2.2 Personal

- Learn shell scripting.
- Learn a provision management tool.
- Understand Linux on a more complex level.
- Expand knowledge of virtualization.

Chapter 2

Background Research

2.1 Deployment Management Tools

There are many deployment management tools to choose from, all offer different benefits and are backed by various companies. A comparison of tools has been covered in detail at Openstack's Summit [5] of which are referenced throughout the deployment management sections below.

2.1.1 Puppet

Puppet [6] released 2005, is one of the earliest approaches to configuration management. It uses a mixture of Ruby and a custom language for writing manifests and templates. The puppet master provides commands to its agents over XML-RPC, it works over HTTP(S) and requires an agent to be installed on the host. Puppet makes use of a WebUI and has reporting tools built in but at the the same time has been described as having a steep learning curve for new users, requiring the user to learn a new language is not ideal; this rules out the use of Puppet for this project.

2.1.2 Chef

Chef [7] released 2008, takes many things from Puppet. They both use Ruby for writing templates and configuration, Chef seems to take on the idea that you should automate everything. The clients poll the Chef server by default every 30 minutes to pull the latest instructions, which are carried out over SSH. Chef still requires an agent to be installed on the server. Overall Chef seems to be most suitable for enterprise solutions due to the vast amount of features available, though it requires extensive understanding to set up and use. It is too feature packed for this kind of project.

2.1.3 Salt

Salt [8] released 2012, is quite different compared to the previous two. It is written in Python and instructions are done over SSH, this is to allow for parallel execution whilst keeping it highly scalable. You can have a centralized salt master control other masters who then control minions, essentially anything can control everything.

The difference now is that there is an optional GUI available. This could of been a suitable tool for the project but the documentation can be challenging to understand.

2.1.4 Ansible

Ansible [9] released 2013, is really similar to Salt. It is written in Python and uses SSH to push instructions to the servers, the only prerequisite server side is that python has to be installed; there is no need for any agents. There is no need for any central server and it is very easy to get started. A big factor that makes Ansible very suitable for this project is that the configuration is done in YAML, it is a standard that anyone can get their head around in no time at all. Having looked greatly at Ansible it is possible to use a chroot as a host, this can be integrated perfectly with parts of the current Raspbian generation code base.

2.2 Virtualization

When deploying code it is normally recommended to test on the same operating system as the current server to reduce the possibility of errors. There are many ways you can set up virtual systems in order to test on a local system or a local network.

2.2.1 ESXi

ESXi [10] is a bare-metal hypervisor by VMWare, it is installed directly on hardware which allows you to install virtual machines on top of it. This form of virtualization of normal within enterprise as it is more cost efficient than running multiple separate physical servers. VMWare do offer products which are designed to be ran on client devices rather than servers but most of them are paid products and there are free alternatives available.

2.2.2 VirtualBox

VirtualBox [11] is an open source solution by Oracle, it is supported on a large platform of systems and it is free to use for personal and enterprise reasons. Due to the nature of the licensing people have developed alternative front-ends and extended functionality in various scenarios.

2.2.3 Docker

Docker [12] is a really interesting approach to virtualization, instead of creating multiple virtual machines on a hypervisor, Docker can run multiple individual containers using one machine. This removes the need for having multiple guest operating systems making it lightweight and really scalable.

2.2.4 Vagrant

Vagrant [13] is built by HashiCorp, it is one of those tools that works on top of VirtualBox. Vagrant is a really simply way to set up a complete development environment in a productive manner. This is now becoming the most popular virtualization tool for developers as it works cross platform.

2.2.5 QEMU

QEMU [14] is also open source, it achieves near native performance but most importantly it can virtualize different CPU architectures. This is used in the current method Raspbian images are generated.

2.3 Current generation method

2.3.1 Analysis

The current method of generating Raspbian images makes use of QEMU-user, chroot and a set of shell scripts. To get it working you have to run the scripts as root on a Debian machine, make sure that the dependencies are installed or the build will fail. If running on a virtualized Debian machine you have to make sure the virtual disk it large enough, if you are using Vagrant then Debian bento boxes [15] work best. Overall the general structure of the code base is set up rather neatly but once you start looking at the actual code it can take some time to get your head around it. Stages are separated out and contain their own set of specific folders and scripts within them, though there is a scripts folder in the root directory that seems to be called at the very start.

2.3.2 Potential improvements

Having looked over the current code base and spoken to the original author a few things can be looked at to improve the overall performance.

- Don't copy each stage, use qcow and branch each stage instead
- Run zipping/tarring in parallel, look at lzop
- Export stages can be run at the same time as the next build stage

Chapter 3

Project Planning

3.1 Standards and quality checks

Ansible uses the YAML notation, which alone is a standard that can be used across multiple languages. There is in depth Ansible documentation [16] which provides you with examples of code and references on what the best code practise is, this ensures the quality of code is as best as possible.

3.2 Methodologies

There are many methodologies [17] available when it comes to project management. Different types of projects have their preferred project methodology.

3.2.1 Waterfall

Waterfall is a 'traditional, sequential methodology,' which has proven to work for projects over the years. The only problem with it is that it does not adapt well to software development as software is very likely to change; changes are disruptive to the overall project.

3.2.2 Agile

Agile is the preferred methodology when dealing with software development, the reason for it is that it allows for iterative cycles without affecting the overall progress of the project. Agile allows for testing to be integrated from the very beginning, this is essential for the success of the project.

3.3 Stakeholders

There are many stakeholders with various interests connected to this project, some direct and others indirect.

- Stakeholder - Dr. Aidan Delaney

  Interest - Project supervisor

  Communication - Meetings in person, Email

- Stakeholder - Raspberry Pi Foundation

  Interest - New method of generating Raspbian images

  Communication - Email, IRC, GitHub

- Stakeholder - Developers

  Interest - Able to easily customise Raspbian image

  Communication - Email, IRC, GitHub

- Stakeholder - Raspberry Pi users

  Interest - Unknowingly using images generated using this project

  Communication - None

3.4 Project schedule

An Gantt chart can be used to estimate the length of the project. This is helpful as it can be used to determine if a project will be complete on time.

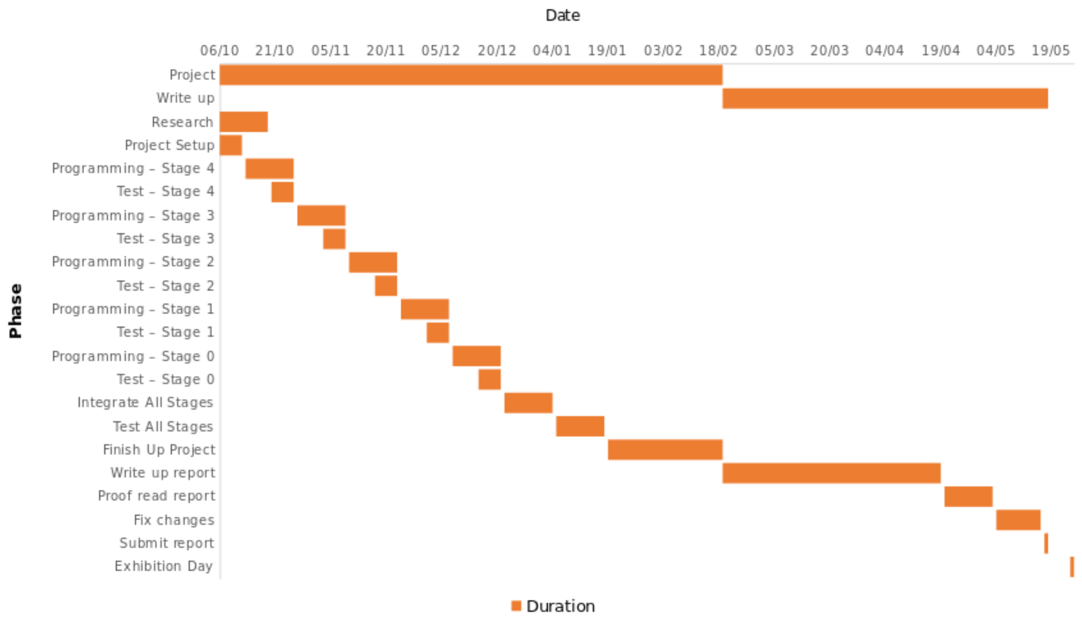

Figure 3.1: Project estimation Gantt chart

3.4.1 Risks

There are a few things that could risk this project, though you can not prevent everything.

- A programming stage takes longer than expected.

  This could potentially delay the project.

- Ansible does not support a native function.

    Code would have to be rewritten, might take longer than expected.

- GitHub could be down.

    People would not be able to clone the project.

- Current method of generation might break things if updated.

    Latest code would have to be integrated during the project.

Bibliography

[1] Raspberry Pi. (2016). Ten millionth Raspberry Pi, and a new kit.
`https://www.raspberrypi.org/blog/ten-millionth-raspberry-pi-new-kit/`
This is the official Raspberry Pi Foundation blog ran by the founder Even Upton, there are a few interesting statistics on this blog.

[2] Raspberry Pi. (2016). RPi-Distro/pi-gen.
`https://github.com/RPi-Distro/pi-gen`
The original code base for generating Raspbian images can be found here, the project pi-gen can be found here ***`https://github.com/adampie/pi-gen`****.*

[3] Wikipedia. (2016). Bus Factor.
`https://en.wikipedia.org/wiki/Bus_factor`
A Wikipedia page describing the 'Bus Factor'.

[4] YAML. (2009). %YAML 1.2.
`http://yaml.org/`
Main website for YAML, contains links to projects using various languages integrating YAML.

[5] Openstack. (2015). Chef vs. Puppet vs. Ansible vs. Salt - What's Best for Deploying and Managing OpenStack?.
`https://www.openstack.org/summit/tokyo-2015/videos/presentation/chef-vs-puppet-vs-ansible-vs-salt-whats-best-for-deploying-and-managing-openstack`
Video and slides for the Openstack Summit, information in the deployment management tools section was referenced from this talk. Really good and in depth comparison

[6] Puppet. (2016). Puppet - The shortest path to better software.
`https://puppet.com/`
Official site for Puppet

[7] Chef. (2016). Chef – Automate Your Infrastructure.
`https://www.chef.io/chef/`
Official site for Chef

[8] Glauser, R. (2016). SaltStack automation for CloudOps, ITOps & DevOps at scale.
`https://saltstack.com/`
Official site for Salt

[9] Red Hat. (2016). Ansible is Simple IT Automation.
`https://www.ansible.com/`
Official site for Ansible

[10] VMWare. (2016). vSphere ESXi Bare-Metal Hypervisor.
`http://www.vmware.com/products/esxi-and-esx.html`
Official site for ESXi

[11] Oracle. (2016). Oracle VM VirtualBox.
`https://www.virtualbox.org/`
Official site for VirtualBox

[12] Docker. (2016). Docker.
`https://www.docker.com/`
Official site for Docker

[13] HashiCorp. (2016). Vagrant by HashiCorp.
`https://www.vagrantup.com/`
Official site for Vagrant

[14] QEMU. (2016). QEMU - Open source processor emulator.
`http://wiki.qemu.org/Main_Page`
Official site for QEMU

[15] Chef. (2016). Bento by Chef.
`http://chef.github.io/bento/`
A list of Vagrant images that have a large virtual disk suitable for using with this project.

[16] Ansible. (2016). Ansible Documentation.
`http://docs.ansible.com/`
Official Ansible documentation, includes code examples useful for project.

[17] Wrike. (2016). Project Management Guide for Beginners.
`https://www.wrike.com/project-management-guide/methodologies/`
Explanation of choosing project management methodologies, helped get a quick overall idea.