POWER8 Continuous Integration

Final Report

Leon Leighton, Thomas Olson, Derek Wong

Project 35

Abstract

This document contains an overview of the progress of the POWER8 Continuous Integration project. It includes the goals and purpose of the project, the current status, items remaining to be done, and a discussion of problems we have encountered.

CONTENTS

I. Introduction

IBM requested a project that would create an easy to use, public, Cloud-based, POWER8 Continuous Integration service in order to increase the ability of Open Source projects to build and test their software on POWER8. Open Source software projects are often designed to work on as broad a range of hardware architectures as possible. The users of Open Source software want the flexibility to consider a variety of hardware solutions in order to take advantage of different capabilities and to avoid vendor lock-in in the future. To support their users, Open Source software projects spend their time and resources on adding support for new architectures as they become available. The introduction of IBM's POWER8 architecture broadened the range of possible solutions available to the end users of Open Source software. This creates a need for Open Source projects to ensure that their software builds and functions correctly on POWER8. Open Source software projects often use Continuous Integration (CI) systems to build and test their software. CI enables these projects to validate their software after each change, ensuring that it continues to work as expected. Public Cloud-based solutions already exist to enable Open Source software developers to build and test their software on Intel's x86 architecture. Thus, while much of the work needed to enable software to compile on POWER8 has already been done at the compiler level, Open Source software developers still need an easy way to build and test their software on POWER8.

II. Requirements Document

A. Introduction

1) Purpose: The goal of this Software Requirements Specification is to set out the requirements for the POWER8 Build and Test Infrastructure project to help coordinate between the multiple stakeholders involved in bringing the project to fruition. This document is intended to be read by technically adept persons, with an emphasis on being read by the Oregon State University Open Source Lab, employees of IBM and faculty of the Oregon State University College of Engineering.

2) Scope: The POWER8 Build and Test Infrastructure shall automatically pull source code from online software repositories, build the required binaries for the POWER8 architecture, run required test suites on the binaries and make the binaries available for use. It shall allow interested parties to register their project's repository through an online interface to maximize ease of use. The system will vastly simplify the ability for open-source projects to test their source code on POWER8. This benefits the developers by allowing such projects to operate on a wider variety of platforms and benefits IBM by greatly increasing the accessibility of the POWER8 architecture to a enormous magnitude of open-source projects.

3) Overview: The rest of this Software Requirements Specification consists of the overall description of the system and the specific requirements for it as laid out in IEEE 830-1998, followed by the table of contents at the end of the document.

B. Overall description

1) Product perspective: The POWER8 Build and Test Infrastructure will interface with software code repositories such as GitHub to automatically pull source code from those repositories to be built. It will run natively on POWER8 architecture hosted by the Oregon State University Open Source Lab.

2) Product functions: The system will allow people to register projects hosted on GitHub through a web interface to be automatically built, tested and released. Once a project is registered, the system will pull the updated version of the source code for a given project when it is updated. It will build the binaries for the POWER8 architecture in either a VM or a container and test them with tests provided by the user for regressions and errors. The user will be notified if the build

and/or test suites failed. If everything succeeds, the system will upload the tested and verified binaries such that the user may access them.

3) User characteristics: This system will be intended to be used by people interested in expanding the scope of what architectures are supported by their platform. It will be targeted towards open source software developers with some experience in building and releasing software.

4) Apportioning of requirements: The fundamental base-line goal is for the system to be able to automatically do a pull-build-deploy cycle for a given project. Once that is complete, adding a test step to the cycle and a web interface are the next two goals. If time permits, further stretch goals may be considered.

C. Specific requirements

1) External interfaces:

1) GitHub repository
2) External authentication sources such as GitHub and Google.
3) Test suites
4) Test results
5) Produced binaries

2) Functional requirements:

1) The system shall allow people to register projects hosted on GitHub through a web interface
2) The system shall automatically build the software.
3) The system shall run tests specified by the project.
4) The system shall report the results of the test to the project.
5) The system shall make the resulting binaries available to the project.

3) Design constraints: The software must compile and run on Linux running on POWER8.

4) Software system attributes:

a) Security: All project builds should be isolated from both the host system and other project builds.

b) Portability: The CI system should be able to be run on other POWER8 systems with a minimal amount of configuration.

III. Design Document

A. Introduction

The goal of this document is to communicate the requirements and design of the POWER8 Continuous Integration project. This information is conveyed in the form of user stories which contain three parts. The 'CARD' contains the requirement and the role of the stakeholder. The 'CONVERSATION' conveys how we intend to meet that requirement. Finally, the 'CONFIRMATION' indicates a way that the meeting of the requirement can be verified. As can be seen from the user stories, the overall design of this project is one of integrating a number of existing resources and open source projects into a system which will provide a continuous integration system for IBM's POWER8 hardware.

Below is a list of the user stories and the author.

POWER8 Cluster	POWER8 servers that can be used for Continuous Integration. Authored by Leon Leighton.
Deployment/Configuration Management	Manage installing the POWER8 CI system in a reproducible way. Authored by Leon Leighton.
Isolation	Isolate builds from each other and other users of the cluster. Authored by Leon Leighton.
Resource Management	Manage resources by automating the creation and removal of virtual machines and containers. Authored by Leon Leighton.
Login	Use an existing login. Authored by Derek Wong.
Build Status	Informing user of build status. Authored by Derek Wong.
Persistant VMs	Persistent virtual machines for our website and Jenkins. Authored by Derek Wong.
Build Artifacts	Access the built and tested files. Authored by Thomas Olson.
Build/Environment Configuration	Configure the build and testing environment and methods. Authored by Thomas Olson.
Automation of Builds	Automatically run builds when a commit is made to the repository. Authored by Thomas Olson.
Pre-build VM images	The system should use pre-built images for the base system of VMs. Authored by Thomas Olson.

B. User Stories

1) POWER8 Cluster:

a) CARD: As an open source software developer, I want to build and test my software on POWER8, so that I can ensure proper functioning on that architecture.

b) CONVERSATION: The goal of this project is to enable open source software projects to build and test their software on POWER8. The OSU Open Source Lab has a POWER8 cluster that will serve as the hardware basis for this project. The OSL POWER8 cluster runs OpenStack [1] and is managed via the Chef [2] configuration management system. We will be working closely with the OSL to ensure that we have access to the POWER8 cluster for both testing and deployment of our Continuous Integration system. We will also work with the OSL to ensure that any needed OpenStack components are installed.

c) CONFIRMATION: The POWER8 CI team members, in coordination with the OSL, are able to deploy the CI system on the OSL's POWER8 OpenStack cluster.

2) Deployment/Configuration Management:

a) CARD: As a system administrator, I want to centralize the management of the POWER8 CI system, so that there is a single source of truth for the configuration of the system.

b) CONVERSATION: We will need to deploy and configure multiple components to create the POWER8 CI system. In addition, one of the goals of our project is to enable others to recreate our POWER8 CI system. We can meet these needs and goals by using a configuration management system. Ansible [3] is a IT automation solution that will allow us to manage the configuration and deployment of our CI system. Specifically, we will be using Ansible playbooks to install and configure Jenkins and Jenkins plugins and any additional components of the CI system. Also, we will be keeping the Ansible playbooks in version control to allow us to keep track of changes and rollback changes when necessary. This will give us the flexibility we need to test changes while being able to return to a known good configuration.

c) CONFIRMATION: Ansible playbooks exists that will deploy our CI system on a POWER8 OpenStack cluster.

3) Isolation:

a) CARD: As an open source software developer, I want my project builds to be isolated from other projects, so that my builds and test complete without outside interference.

b) CONVERSATION: Our project will be hosted on the OSL's POWER8 OpenStack cluster and we want to ensure that our usage of the cluster does not negatively impact other users of the system. Additionally, we do not want to run builds and tests in such a way that our users could interfere with each other. We can achieve these ends by isolating builds and tests in virtual machines or Linux containers. We will work with the OSL to determine appropriate quotas for usage of CPU, memory, and storage resources. Our primary means of launching and removing virtual machines and containers will be through OpenStack's Nova and Magnum components. By using these components we can leverage OpenStack's ability to manage virtual machines and Linux containers to provide isolation between different projects' builds and tests.

c) CONFIRMATION: Projects are built in virtual machines or containers.

4) Resource Management:

a) CARD: As an open source software developer, I want my project builds to finish in a timely manner, so that I can receive build and test feedback as quickly as possible.

b) CONVERSATION: The users of our project want their builds to run as soon as possible. However, our project has a finite amount of resources available on the POWER8 OpenStack cluster. This may cause a delay in starting a build if the necessary resources are not available. Builds should be queued to run in the order received if the resources are not available to run them immediately. In order to ensure that resources are available for new builds, we will need to automatically remove old VMs and containers. As mentioned in another user story, OpenStack has the ability to manage virtual machines and containers through the Nova and Magnum components. In order to automate this, we will utilize Jenkins [4] which has the ability to interact with OpenStack by using a plugin [5]. This will give us the ability to automate both the creation and removal of VMs and containers.

c) CONFIRMATION: Builds are triggered immediately if resources are available, and queued in the order recieved otherwise.

5) Login:

a) CARD: As an open source software developer, I want to be able to use my github account credentials to login, so that I don't have to create another account, to access Jenkins.

b) CONVERSATION: We will use pluggable authentication from Jenkins to accomplish this task. Specifically, we will use the GitHub OAuth Plugin to allow users to login with their GitHub credentials.

c) CONFIRMATION: Our registration web site will allow developers to simply use their GitHub credentials to login into our services.

6) Build Status:

a) CARD: As an open source software developer, I want to see my build status, so I know if my build passed or failed.

b) CONVERSATION: We will use a plugin from Jenkins to display the status of our developer's build. More specifically, we intend to use the Build Monitor Plugin [6] to display the information to our users. The Build Monitor Plugin supports many features such as: display the status and progress of selected jobs, display the names of people who might be responsible for 'breaking the build', display who is fixing a broken build, and display what broke the build. The plugin can be installed through Jenkins.

c) CONFIRMATION: Our web site will allow developers to see their build status (pass/fail).

7) Persistent VMs:

a) CARD: As a system administrator, I want persistent virtual machines (VMs) for our website and Jenkins, so that I have a platform to host my service.

b) CONVERSATION: We will use OpenStack to deploy our virtual machines. Then we will use those virtual machines to host our website and Jenkins. OpenStack is a set of software tools for building and managing cloud computing platforms, more specifically it is used to deploy virtual machines to handle different tasks. For our website, we will be using Apache HTTP Server.

c) CONFIRMATION: Our website and Jenkins will be hosted on virtual machines.

8) Build Artifacts:

a) CARD: As an open source software developer, I want to access the built and tested files, so that I can distribute them to users.

b) CONVERSATION: Developers need to be able to access the created binaries and other files from building the source code, so that they may be able to distribute them as needed on their own.

c) CONFIRMATION: The files can be downloaded from the website.

9) Build/Environment Configuration:

a) CARD: As an open source software developer, I want to configure the build and testing environment and methods, so that I can be sure it matches my standard build process.

b) CONVERSATION: Each repository needs a way to configure things such as required packages for building and command-line arguments. To that end, the system will require a configuration file in YAML format. This file will contain information such as necessary libraries and packages, programs and command-line arguments for building, and how to test the built code.

c) CONFIRMATION: The system checks for a file called '.powerci.yml' in registered repositories and configures the build environment with it.

10) Automation of Builds:

a) CARD: As an open source software developer, I want the system to automatically run builds when a commit is made to the repository, so that I can determine if a bug has been introduced as early as possible.

b) CONVERSATION: Developers require a system that will automatically pull, build and test so they can always have the most current binaries available and so they can quickly know if a recent change causes the build to fail. Jenkins has plugin support for polling Git repositories for changes and then executing a pull-build-test-deploy sequence when a commit is detected to have been pushed since the last build was ran.

c) CONFIRMATION: The system automatically executes a pull-build-test-deploy sequence when a commit is pushed to the Git repository.

11) Pre-build VM images:

a) CARD: As an open source software developer, I want to be able to choose from pre-built virtual machine, so that my builds and tests can target a known platform.

b) CONVERSATION: Pre-built images that are used for spinning up VMs makes the entire process of deploying and configuring a VM for running a pull-build-test sequence much faster and smoother. Ubuntu will probably be the OS of choice for the base images, however we may look into providing images for other Linux distributions.

c) CONFIRMATION: The system uses disk images with Linux installed for starting VMs.

IV. Tech Review Document

A. Introduction

This document contains an evaluation of nine pieces of technology that we will be using in the IBM POWER8 Continuous Integration project. We define evaluation criteria based on the goal we are attempting to achieve by using that piece. For each piece we have selected three options, and we choose the best option based on the evaluation.

The nine pieces are:

Cluster Management	The software that will be used to manage a cluster of POWER8 nodes. Authored by Leon Leighton.
Continuous Integration Software	The software that will perform the building and testing of projects. Authored by Leon Leighton.
Configuration Management	System that will be used to manage the configuration of the system. Authored by Leon Leighton.
Linux Distribution Support	The Linux distribution that projects will be built and tested on. Authored by Thomas Olson.
Platform For Running Builds	The type of platform used for building and testing. Authored by Thomas Olson.
Configuration File Format	The file format that will be used by projects for configuring builds. Authored by Thomas Olson.
Login/Authentication	Method for users to login to the system. Authored by Derek Wong.
Frontend/Web Frameworks	Method of creating a user interface for interacting with the system. Authored by Derek Wong.
Tracing State of Builds/Tests	Functionality to show users the status of their builds and tests. Authored by Derek Wong.

B. Cluster Management

1) Options:

a) OpenStack: OpenStack [1] is a cluster management solution originally created by Rackspace and NASA, and now managed by the OpenStack Foundation. It contains a number of projects that provide services such as object and block storage, identity management, networking and compute resource management, bare metal provisioning, and DNS services [7]. This modular approach allows users to select which parts they need to accomplish their goals without having to install components they will not be using.

b) CloudStack: CloudStack [8] is an Apache Software Foundation cluster management project which aims to provide many of the same services as OpenStack. It follows a more monolithic approach where most components are distributed as part of a single binary [9].

c) Ganeti: Ganeti [10] is a cluster management solution from Google that focuses on managing virtual machines. It can be used to create new virtual machines, manage disks, and manage failover of virtual machine instance.

2) Goals for use: The primary goal for using cluster management software is to manage the creation and destruction of temporary virtual machines and containers that will be used for building and testing projects.

3) Criteria for Evaluation:

1) Community Support: Our chosen solution should have a community that provides support in the form of documentation, bug fixes, and assistance.
2) Linux Support: Linux must be supported as both host and guest.
3) Container Support: Optional, but we would like to have support for building and testing in containers.
4) Jenkins Plugin Support: Jenkins is a likely piece of our solution, therefore having a Jenkins plugin that interacts with our cluster management solution is highly desired.

4) Table:

Criteria	OpenStack	CloudStack	Ganeti
Community Support	Yes	Yes	Yes
Linux Support	Yes	Yes	Yes
Container Support	Yes	Partial	No
Jenkins Plugin Support	Yes	Yes	No

5) Discussion: OpenStack, CloudStack, and Ganeti can all run on Linux and can host Linux virtual machines. Ganeti does little more than this, however, and will not be considered further. Both OpenStack and CloudStack can interact with Jenkins through a plugin [5] [11]. OpenStack appears to have a larger community and a more mature container solution than CloudStack.

6) Selection of best option: OpenStack is our chosen solution for cluster management. It meets all of our criteria for evaluation.

C. Continuous Integration Software

1) Options:

a) Jenkins: Jenkins [4] is an automation server that can be used to automate builds, tests, and deployments. It is extensible and has a large number of plugins that enable it to interact with other systems.

b) Buildbot: Buildbot [12] is another system that enables the automation of builds, tests, and deployments. It can be extended through the use of Python configuration files [13].

c) Strider: Strider [14] is a continuous integration server written in Node.js. Like Jenkins it uses plugins to extend its functionality.

2) Goals for use: Automate building and testing Open Source projects.

3) Criteria for Evaluation:

1) Open Source: As our target audience is Open Source software developers, we should use an Open Source solution to be consistent with the community.
2) Language Support: We need to be able to build and test projects written in a variety of languages.
3) Notifications: We need some way of providing the user with feedback on the status of their builds and tests.
4) Extensible: If we find that there are missing features we will need some way of extending the functionality of the software.
5) Number of Existing Plugins: A larger number of plugins increases the probability that someone may have already provided functionality that we need that is not already in the software.

4) Table:

Criteria	Jenkins	Buildbot	Strider
Open Source	Yes	Yes	Yes
Language Support	Any	Any	Any
Notifications	Yes	Yes	Yes
Extensible	Yes	Yes	Yes
Number of Plugins	High	Low	Medium

5) Discussion: All three of our options meet our basic evaluation criteria. However, there is quite a large difference in the number of plugins available, with Jenkins [15] having a much larger number than Buildbot [16] or Strider [17]. There is also a large variety in the type of plugins for Jenkins, covering functionality such as source code management, reports and notifications, and UI elements [15].

6) Selection of best option: With its large number and variety of plugins, Jenkins is our chosen solution for continuous integration software.

D. Configuration Management

1) Options:

a) Ansible: Ansible [3] is a configuration management and infrastructure automation solution. It is agentless, using SSH to connect to and run commands on nodes. Ansible uses YAML files for configuration [18].

b) Chef: Chef [2] is a configuration management solution that uses agents running on each node to poll a central server to access configuration 'cookbooks' that are written in Ruby [19].

c) Puppet: Puppet [20] is also a configuration management solution. Like Chef, it uses agents on each node and a central 'puppet master' server [21]. Configuration is done in a language specific to Puppet that is meant to be accessible to system administrators [22].

2) Goals for use: Using a configuration management system will allow us to create a solution that can be replicated by others who may wish to setup their own POWER8 Continuous Integration system. Used in conjunction with a version control system, it will also give us the ability to roll back changes that have been made, giving us flexibility in experimenting with different options as we progress through the project. At a higher level, a configuration management system will allow us to specify what we want the state of the system to be, and it will also give us an ultimate 'source of truth' for the project.

3) Criteria for Evaluation:

1) Support for installing Jenkins: Our chosen solution should have the ability to manage our Continuous Integration software.
2) Overhead: Like most projects, we wish to reduce the amount of overhead in our system.
3) Difficulty: We should choose simpler solutions when possible.

4) Table:

Criteria	Ansible	Chef	Puppet
Installs Jenkins	Yes	Yes	Yes
Overhead	Low	Moderate	Moderate
Difficulty	Low	Moderate	Moderate

5) Discussion: As we are likely to use Jenkins for our continuous integration software we would like there to be an already existing method to install and configure it with the option we choose for configuration management. All three options under consideration meet this criteria through user created modules. We would also like to reduce the overhead involved in using a configuration management system. Both Puppet and Chef require the use of a master server and an agent installed on any node that will be managed. While this is not a high level of overhead, we do contrast that with Ansible which uses SSH which is already included on most Linux systems. It should be noted, however, that since Ansible uses SSH, we will need to create and secure the SSH keys that will be used for authentication. Ansible is also noted to be easier to setup and use. [23]

6) Selection of best option: While all three options would allow us to accomplish our goals, we have selected Ansible as our best option.

E. Linux Distribution Support

1) Options:

a) RedHat Enterprise Linux: RedHat Enterprise Linux is widely considered to be one of the standard Linux distributions for server environments, having been around for a number of years and being very stable. It is supported by the Red Hat corporation and has a ten-year supported production cycle for each major version, followed by an extended support period[24]. Unlike most Linux distributions, it has a paid subscription requirement for support and the extended support has a higher price. However, as a paid product, Red Hat provides dedicated support and certifications which can be important if such certifications are required for a project or if the project requires high availability and dedicated support in case issues arise.

b) CentOS: CentOS is an unbranded version of RedHat Enterprise Linux, providing the same functionality and stability with only very minor functional differences. It is supported by the CentOS Project and is free for all use. As a result, it does not have the certifications nor the dedicated support of RedHat Enterprise Linux. As it tries to track the development

of RedHat Enterprise Linux very closely, CentOS has approximately the same ten-year support cycle[25], but it does not have the same extended support period provided by Red Hat.

c) Ubuntu Server: Ubuntu Server is a version of the Ubuntu distribution that differs from the desktop version only by the installer and the life cycle[26]. It is supported by Canonical and has a five-year life cycle per major version. Like CentOS, it is free to use, but Canonical optionally offers a contract for dedicated support. Ubuntu tends to update application packages sooner, trading some amount of stability for newer functionality. Finally, Ubuntu has better support for containers compared to CentOS.

2) Goals for use: The operating system selection will have a major effect on the stability and ease-of-use of the entire system. Our goal is to select an operating system that is stable, widely supported and easy to work with.

3) Criteria for Evaluation:

1) Stability: It is important that the operating system used for running builds has as few issues as possible so as to minimize the need for manual intervention in case there's a failure.
2) Support: The operating system should have a strong support base to maximize ease of maintenance and help ensure longevity.
3) Usability: The greater the usability and ease of configuration, the easier it becomes to maintain and improve the reliability of the system.
4) Cost: The less the entire system costs, the better.

4) Table:

Criteria	RedHat	CentOS	Ubuntu Server
Stability	Highest	High	Moderate
Support	Highest	Moderate	Moderate
Usability	High	High	High
Cost	High	None	None

5) Discussion: As this project does not require high availability or dedicated support, the benefits of buying a subscription for Red Hat Enterprise Linux are negligible, especially being that CentOS has much of the same functionality and the same length support cycle. This project also does not require an operating system that provides newer functionalities sooner like Ubuntu does, stability being a more important requirement. However, the client has expressed an interest in using containers and using Ubuntu as a result of that.

6) Selection of best option: We have opted for using Ubuntu as the operating system that builds will be ran on and tested against as per the client's request.

F. Platform for Running Builds

1) Options:

a) Virtual Machine: Virtual Machines perform hardware-level emulation and do not allow the guests any access to the underlying operating system or hardware. They're the standard for situations where multiple OS's need to be deployed. Because they emulate all the hardware, there's nearly no risk of a malicious guest system affecting the host system. However,

running a virtual machine requires a full operating system to be installed on each individual instance, causing them to take up a relatively significant amount of space per virtual machine compared to to size of the applications running on them. Additionally, their performance suffers due to having to emulate assembly-level instructions instead of executing them directly.

b) Container: Containers perform operating system virtualization, allowing for multiple independent user-spaces to exist at the same time. Because they emulate parts of the operating system and not the physical hardware hardware, containers still need to allow some level of direct hardware access for applications to run properly. They also use some libraries provided by the host operating system instead of providing their own. As a result, they tend to be much smaller than a full-fledged virtual machine and run somewhat faster. A relatively new technology compared to virtual machines, they're rapidly gaining popularity in situations where hardware emulation is not required and restricting access to the host operating system is not a concern.

c) Bare metal: Bare metal servers would have no virtualization or emulation; builds and tests would be ran directly on the hardware itself. The benefits of bare metal are speed and security; no emulation layer exists between the applications and the hardware and there is no host operating system for a malicious guest to interfere with. Builds would also be unable to hoard system resources, preventing a situation where one build slows down another. The single major downside is the requirement for a full physical server per each build that would be ran, greatly restricting the ability for multiple builds to be ran in parallel

2) Goals for use: The goal of selecting the platform the builds would interact with is to allow builds a host system to be ran and tested on while providing reliability and security for the underlying system.

3) Criteria for Evaluation:

1) Scalability: Ideally, it should be easy to scale the system up to allow a large number of simultaneous builds.
2) Performance: Our selection should not significantly decrease the performance of the build-test process.
3) Security: Being that we will compiling and testing untrusted and unverified code, it is important for the selected platform to minimize the ability of the build to negatively affect the host.
4) Reliability: It is preferable that one build does not use up so many resources as to significantly reduce the speed of other builds being ran.

4) Table:

Criteria	Virtual Machines	Containers	Bare metal
Scalability	High	Highest	Low
Performance	Moderate	High	Highest
Security	High	Moderate	Highest
Reliability	High	High	Highest

5) Discussion: Although bare metal servers have undeniably the best performance, security and reliability, they simply do not have the scalability required for a project like this. Depending on storage space and resources, virtual machines and containers could potentially run upwards of one hundred builds simultaneously on a single server, negating the performance benefits. Furthermore, virtual machines provide nearly the same level of security and safety that a bare metal server provides,

while containers can nearly match them in terms of performance.

Between virtual machines and containers, virtual machines provide better security and have years of support behind them, while containers provide better performance; virtual machines can have a performance penalty upwards of ten percent due to overhead.[27] Both virtual machines and containers allow for limiting the resources provided to a running instance, meeting the reliability requirements. They are also both exceedingly easy to configure for automated deployment, making them highly scalable.

6) Selection of best option: Although virtual machines can have a noticeable performance penalty due to overhead, this can be reduced with proper tuning. Furthermore, we believe that the better isolation and security provided by virtual machines can easily outweigh the performance penalties. Therefore, we have decided that virtual machines are the ideal platform for running the builds and tests on. However, as per our client's request, the system should also support the use of containers if at all possible for running builds in. As such, we will support both the use of virtual machines and containers for builds to be ran in.

G. Configuration File Formats

1) Options:

a) YAML: YAML is a data-oriented markup language that is very concise while still being highly expressive. Notably, Ansible already uses YAML for its configuration files.

b) JSON: JSON is a data-interchange format that is very concise, focused around key-value pairs. Its readability can suffer at times due to being designed with automated usage in mind instead of being human-generated.

c) XML: XML is a document markup language that is incredibly powerful and expressive. However, it tends to being excessively verbose, making it somewhat difficult to read.

2) Goals for use: The configuration file will be included in a Git repository and will be how the system determines builds and tests are ran and interpreted. It should use a format powerful enough to accurately express all the information that the system will require without being unnecessarily complicated.

3) Criteria for Evaluation:

1) Expressiveness: The file format needs to be able to sufficiently express the users needs for building and testing their project.
2) Readability: The ideal file format will be easy for a user to create and modify to suit the needs of their project.

4) Table:

Criteria	YAML	JSON	XML
Expressiveness	High	Moderate	Highest
Readability	Highest	High	Moderate

5) Discussion: XML is the most expressive of the three options, but it is unlikely we would need the power provided by it. JSON is on the opposing end of the spectrum, being so simple that it might not be powerful enough for what we need. In the middle lies YAML, powerful and highly readable due to it conciseness. Aditionally, Anisble using YAML gives YAML the bonus of already being used in this project.

6) Selection of best option: We have selected YAML as the file format for configuration files due to it being powerful, concise and consistent with what Ansible uses.

H. Login/Authentication

1) Options:

a) GitHub OAuth Plugin: The GitHub OAuth Plugin uses a github account for authentication and it also has the ability to set permission/authorization to users. The authorization includes: ability to create a job, build a project, configure a project, delete a project, etc...

b) Active Directory Plugin: The Active Directory Plugin is a plugin for Jenkins and it uses Active Directory to authenticate the username and password.

c) Jenkins Login: The Jenkins login is simply a Jenkins account and users are able to create it on the Jenkins website.

2) Goals for use: The goal for having an authentication is to provide our users a safe and secure place to build their project and to use our continuous integration service. It will help prevent hackers from obtaining private information that could be detrimental to our users. This is a security measure that is absolutely necessary for our project.

3) Criteria for Evaluation:

1) Security: Security is what keeps users confidential information such as source code from being leaked.

4) Table:

Criteria	GitHub OAuth Plugin	Active Directory Plugin	Jenkins Login
Security	Yes	Yes	Yes

5) Discussion: The github OAuth plugin is a really straight forward method for authentication. Users can simply use their github accounts to login. The second option is to use the active directory plugin. This plugin is used with Jenkins to authenticate the username and password through active directory [28]. The third option is to use Jenkins login. Users can simply create a Jenkins account and have access to its functionalities. They will also have the ability to give authorizations to different accounts for their projects.

6) Selection of best option: The best option is to use GitHub OAuth plugin. Most of our users will be using github for their projects, so it is most resonable to use the github OAuth plugin for authentication.

I. Frontend/Web frameworks

1) Options:

a) Tomcat: Tomcat is an open source implementation of Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies.

b) Glassfish: GlassFish is an open source application server; more specifically it is for the Java EE platform (Java Enterprise Edition).

c) Jetty: Jetty provides a Web server and javax.servlet container, and it supports many components such as HTTP/2, WebSocket, OSGi (Open Service Gateway Initiative), JMX (Java Management Extensions), JNDI (Java Naming and Directory Interface), JAAS (Java Authentication and Authorization Service) and many other integrations.

2) Goals for use: The goal for having a web framework is so that our users have a platform to use our continuous integration service. We will provide a simple and easy to use user interface (UI) so that our users would not have a complicated time using our services.

3) Criteria for Evaluation:

1) Usability: Having an easy-to-use user interface can give our users an easier time to use our service and it will help reduce confusion.
2) Security: Security is important for our web framework because we don't want hackers to gain access to our services and obtain confidential information.
3) Stability: Stability is needed because our service need to maintain at a functional state at all time so that it doesn't hinder our users from working
4) Load time: The page load time should load at a reasonable speed so that our users would not have to wait a long time using our service.

4) Table:

Criteria	Tomcat	Glassfish	Jetty
Usability	High	High	Moderate
Security	Moderate	High	High
Stability	Highest	High	High
Load time	High	High	High

5) Discussion: Tomcat is very popular and it is known as a lightweight application that offers all the basic features required running a server. It is an open source project so it is cost free. It is highly flexible because it allows users to tweak their code as they see fit, and it has many built-in features [29]. The second option is Glassfish and it is also an open source project, which means that it is free of cost. It has high performance and it was considered the fastest open source application server, according to SPECjAppServer2004 benchmark results [30]. Jetty provides a Web server and javax.servlet container. They have a heavy focus on multi-connection HTTP and features such as SPYDY, which can significantly reduce page load latencies [31].

6) Selection of best option: Overall, all three of the options are great, but the best option to use is Tomcat. Glassfish has a lot of features and is the biggest out of the three options, but we are only using it for our frontend, so most features are unnecessary. It has a big memory footprint, which means that it would consume more resources. Jetty is a very light application and it consumes the least amount of resources but it is too small for our project. Tomcat is ideal because it sits in between these two options and it offers a good amount of features to use.

J. Tracing State of builds/tests

1) Options:

a) Build Monitor Plugin: The Build Monitor Plugin is a plugin in Jenkins that provides a detailed view of the status of selected Jenkins jobs.

b) Lava Lamp Notifier: The Lava Lamp Notifier is a notifier that indicate job status using a Lava Lamp.

c) Radiator View Plugin: The Radiator View Plugin is a plugin in Jenkins and Hudson that provides a job view, which displays project status.

2) Goals for use: The goal to having the functionality to trace the build and test state is so that our users can see the status of their projects. This will reveal the success or failure of projects that were build by our service and it will show where the failures have occurred.

3) Criteria for Evaluation:

1) Usability: Having an easy-to-use user interface (UI) allows our users to operate our service with minimal issues.
2) Feature to display pass or fail builds:The ability to display pass or fail build is the main purpose for this technology so this is a must have
3) Feature to display progress of builds: The feature to display progress when building a project is helpful for our users to see how much longer it will take until their build is complete
4) Feature to display where an error has occurred for build fails: The feature to see where an error has occurred in the build process is another important criteria; it will help developers ave time from manually tracking down the problem.

4) Table:

Criteria	Build Monitor Plugin	Lava Lamp Notifier	Radiator View Plugin
Usability	Yes	None	Yes
Display pass or fail builds	Yes	Yes	Yes
Display progress of builds	Yes	None	None
Display where an error has occurred if build fails	Yes	None	None

5) Discussion: The build monitor plugin supports many features and incorporates other plugins as well. The features in this plugin include:

- Displays the status and progress of selected jobs, the view is updated automatically every couple of seconds using AJAX. No ‘Enable Auto Refresh’ needed.
- Displays the names of people who might be responsible for ‘breaking the build’.
- Supports the Claim plugin, so that you can see who’s fixing a broken build.
- Supports View Job Filters, so that you can easily create Build Monitors for ‘slow builds’, ‘only failing’, etc.
- Supports Build Failure Analyzer, so that you know not only who, but also what broke the build.
- Supports CloudBees Folders Plugin, so that you can have project- and team-specific nested Build Monitors.
- The number of columns and size of the font used is easily customizable, making it trivial to accommodate screens of different sizes.
- UI configuration is stored in a cookie, making it possible to display different number of columns and using different font size on each of the screens at your office.
- Can work in a colour-blind-friendly mode
- [6]

The lava lamp notifier is a really simple design created to indicate if a build passed or fail. The only necessary component

to use this type of method to track our build status is a USB LED light. The light will simply light up green for pass and red for fail [32]. This is not ideal for our type of project because our users would like to know more information than just a pass or fail notification. The third option is the radiator view plugin. This plugin is somewhat similar to the build monitor plugin but with less features. It will display all the project that are building and indicate pass or fails on screen [33].

6) Selection of best option: The best option to use is the build monitor plugin. This technology has the most features out of all the other options I listed, and all the features are very useful for our users.

K. Conclusion

The following is a summary of the choices we have made for the nine pieces that we have evaluated.

1) Cluster Management: OpenStack
2) Continuous Integration Software: Jenkins
3) Configuration Management: Ansible
4) Linux Distribution Support: CentOS
5) Platform For Running Builds: Virtual Machines
6) Configuration File Format: YAML
7) Login/Authentication: GitHub Login Plugin
8) Frontend/Web Frameworks: Tomcat
9) Tracing State of Builds/Tests: Build Monitor Plugin

V. Weekly Blog Posts

A. Fall Week 6

1) Leon Leighton:

a) This week: Had conference call with client. Discussed using user stories as basis of requirements/design document.

b) Next week: Focus on creating user stories for various roles.

2) Thomas Olson:

a) This week: Had conference call with IBM folks. Established the use of user stories as the basis of our requirements and design document. Discussed with Kevin on how to use user stories instead of the IEEE 830-1998 standard for a requirements document. Informed us to do what the customer requires.

b) Next week: Work on user stories and start work on the technical review.

3) Derek Wong:

a) This week: Went over requirement doc with group. Meeting with IBM client on Thursday.

b) Next week: Create user stories for requirement doc./design doc.

B. Fall Week 7

1) Leon Leighton:

a) This week: Worked with group to identify nine pieces for technology review and divide tasks. Began work on tech review.

b) Next week: Finish tech review. Work on user stories and design document.

2) Thomas Olson:

a) This week: Established nine separate pieces for the technology review. I chose to work on reviewing the operating system choices, the platform choices (VM vs Container vs Baremetal) and formats of configuration files.

b) Next week: Finish the tech review. Work on user stories and the design document.

3) Derek Wong:

a) This week: Had meetings with group members to discuss about the technology document. Brainstormed 9 different piece of technology for our project and divided the work.

b) Next week: Edit/Put together technology document with group. Meeting with IBM on Thursday. Work on Design Document with Group.

C. Fall Week 8

1) Leon Leighton:

a) This week: Finished work on tech review. Had conference call with IBM.

b) Next week: Continue work on user stories and design document.

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week: Completed my 3 piece of technology for the tech document. Had a meeting with IBM on Thursday. Met with group to compile all of our parts together into one document. Discuss with group about design document.

b) Next week: Work on my part for the design document. Meet with group to merge our parts together.

D. Fall Week 9

1) Leon Leighton:

a) This week: Review requirements for design document.

b) Next week: Write design document. Start on Progress Report and presentation.

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week: Go over design documents with group.

b) Next week: Split design document work with group. Start on design document. Start on progress report. Meeting with IBM.

E. Fall Week 10

1) Leon Leighton:

a) This week: Finished work on design document.

b) Next week: Finish and turn in Fall Progress Report.

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week: Finish up design document.

b) Next week: Start on Progress Report.

F. Winter Week 1

1) Leon Leighton:

a) This week: Worked on scheduling. Cancelled weekly call with IBM due to time conflict with a class. Continued to read about Ansible.

b) Next week: Work with OSL to setup testing environment. Begin work on Ansible playbook for Jenkins installation.

2) Thomas Olson:

a) This week: First week of term; getting readjusted with everything.

b) Next week: Meet with group; sort out scheduling for the regular meetings for the rest of term.

3) Derek Wong:

a) This week: Canceled Meeting with IBM. Review documents.

b) Next week: Have a meeting with group and talk over the project.

G. Winter Week 2

1) Leon Leighton:

a) This week: Scheduled new conference call date/times with IBM for winter term. Continued to read about Ansible.

b) Next week: Work with OSL to setup testing environment. Continue work on Ansible playbook for Jenkins.

2) Thomas Olson:

a) This week: Established time and day for bi-weekly meetings with IBM and OSL.

b) Next week: Bi-weekly meeting with IBM and OSL; work with group on documents.

3) Derek Wong:

a) This week: Set up meeting times with TA and IBM. Discussed about how we are beginning our project.

b) Next week: Plan out our project and schedules. Split up Group work.

H. Winter Week 3

1) Leon Leighton:

a) This week: Created Ubuntu VM on OSL's OpenPower OpenStack cluster. Did proof of concept install of Jenkins with Ansible. Had conference call with IBM. Discussed CIAAS and travis file compatability

b) Next week: Work on updating Design Document. Clarify with IBM which parts of previous work (such as CIAAS) that they may want us to use.

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week: Meeting with TA to discuss about alpha stage of our project. Meeting with IBM discussing where we are at with our project. Set up group meeting time for next week.

b) Next week: Revise design document to better fit requirements. Work on alpha stage of project.

I. Winter Week 4

1) Leon Leighton:

a) This week: Received clarification from client on direction of project. There was some confusion on whether we were to implement our own solution or try to put in to production a solution that was already being worked on. Client indicated that we could pursue our own solution.

b) Next week: Finish configuring Jenkins with basic plugins and go over configuration options.

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week: Group meeting. Sent email to our client to clarify some information.

b) Next week: Take a look at jenkins plugin and go over configuration for jenkins.

J. Winter Week 5

1) Leon Leighton:

a) This week: Finished initial work on getting Ansible to install Jenkins and plugins. Configured Jenkins to allow the creation of jobs. Had conference call with IBM. We discussed the need for more communication from us to students to our client.

b) Next week: Finish revisions of all documents. Updating as needed. Finish progress report.

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week: Have VM set up with jenkin install with group. Added plugins to jenkins. Conference call with IBM discussing about progress and what needs to be done.

b) Next week: Finish updating document with group. Meet up with group to complete progress report.

K. Winter Week 6

1) Leon Leighton:

a) This week: Primarily worked on midterm progress report and presentation.

b) Next week: Start work on configuring Jenkins to do builds in separte VMs/containers. Work on putting configurations done through Jenkins interface into Ansible.

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week:

b) Next week:

L. Winter Week 7

1) Leon Leighton:

a) This week: Enabled building Jenkins jobs in Docker containers.

- Used docker-plugin in Jenkinsi

- Used ppc64le/ubuntu as base image for Docker container

Had conference call with IBM. Received clarifications on suggestion to create a backlog of issues that we pull from for weekly focus. Also received some ideas about where to look for Ansible configurations.

b) Next week: Will focus on getting configurations that have been done in Jenkins interface into Docker.

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week: Conference call with IBM. They gave us suggestions on what to do next, such as creating a backlog of issues, which is essentially small tasks for us to do

b) Next week: Create ”issues” and start working on them.

M. Winter Week 8

1) Leon Leighton:

a) This week: Researched ways to add configuration to Ansible.

b) Next week: Continue to work to put configuration in Ansible that is currently only in Jenkins

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week:

b) Next week:

N. Winter Week 9

1) Leon Leighton:

a) This week: Moved more of configuration that had been done directly in Jenkins to Ansible. Had conference call with client.

b) Next week: Based on conference call I am going to focus on the security of our project. Also, work on end of term progress report, presentation, and draft of poster.

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week: Conference call with IBM. Research how to integrate TravisCI yml file into our service.

b) Next week: Work on progress report, presentation. Meet with group to create a draft of our poster.

O. Winter Week 10

1) Leon Leighton:

a) This week:

b) Next week:

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week:

b) Next week:

P. Spring Week 1

1) Leon Leighton:

a) This week: Conference call with IBM. Travis compatibility no longer a priority. Created private network on OpenStack for Docker and build VMs.

b) Next week: Create Docker and VM images for use in builds. Configure Matrix authorization.

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week: Conference call with IBM. Installed Embeddable Build Status Plugin and Build Monitor Plugin to Jenkins. Set-up Email-Notification Plugin.

b) Next week: Look for a new task to work on in the back log.

Q. Spring Week 2

1) Leon Leighton:

a) This week: Configured Matrix Authorization for Admin users. Created virtual machine image for use in builds. Test build now works in VM.

b) Next week: Configured Matrix Authorization for normal users. Additional VM images and test builds.

2) *Thomas Olson:*

a) *This week:*

b) *Next week:*

3) *Derek Wong:*

a) *This week:*

b) *Next week:*

R. Spring Week 3

1) *Leon Leighton:*

a) *This week:*

b) *Next week:*

2) *Thomas Olson:*

a) *This week:*

b) *Next week:*

3) *Derek Wong:*

a) *This week:*

b) *Next week:*

S. Spring Week 4

1) *Leon Leighton:*

a) *This week:* Worked on poster. Began review of Ansible for code pull on May 1.

b) *Next week:* (Weekend) Make sure all code is in repo and updated. Begin work on landing page for project. Intall TLS cert.

2) *Thomas Olson:*

a) *This week:*

b) *Next week:*

3) *Derek Wong:*

a) *This week:*

b) *Next week:*

T. Spring Week 5

1) *Leon Leighton:*

a) *This week:*

b) *Next week:*

2) *Thomas Olson:*

a) *This week:*

b) *Next week:*

3) *Derek Wong:*

a) This week:

b) Next week:

U. Spring Week 6

1) Leon Leighton:

a) This week: Installed TLS certificate for web page. Rebuilt Docker container with additional options to resolve build issues. Completed first draft of Demo for Expo. Began work on Midterm Progress Report.

b) Next week: Finish Midterm Progress Report. Proxy connection to Jenkins interface through Apache for TLS support. Continue to troubleshoot build issues in Docker. EXPO!!! (Friday)

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week:

b) Next week:

V. Spring Week 7

1) Leon Leighton:

a) This week: Finished and turned in our Spring Midterm Progress Report. Attended the Engineering Expo and explained our project to both industry representatives and members of the OSU and Corvallis community.

b) Next week: Apache to Jenkins reverse proxy. PR to change OSL's powerdev request form to include information about project and allow access to be requested.

2) Thomas Olson:

a) This week:

b) Next week:

3) Derek Wong:

a) This week:

b) Next week:

W. Post Engineering Expo

1) Leon Leighton:

If you were to redo the project from Fall term, what would you tell yourself?

Focus on the project workflow. Set deadlines and meet them. Use a task management system from the beginning. Set aside time in the Fall to make sure everyone on the project has a basic understanding of all the technical areas. Always communicate with team members and client.

What's the biggest skill you've learned?

Something that we didn't do terribly well in the beginning, but I think got better at as we went. Being able to see what tasks need to be done from the goals that are set, and then breaking up tasks into individual assignments to meet those goals.

What skills do you see yourself using in the future?

Technical: Ansible and configuration management in general.

Non-technical: Project coordination. Working with teammates.

What did you like about the project, and what did you not?

I enjoyed being part of a project that will continue after I am done with it. I liked that I got to use Ansible as a configuration management system since I had not used it before.

I did not like the way that Jenkins is configured. I also did not enjoy the near constant coordination problems.

What did you learn from your teammates?

Everyone has strengths and weaknesses. Understanding what those are and how to plan with those in mind is an important part of working together.

If you were the client for this project, would you be satisfied with the work done?

I would be satisfied with the work that was done while feeling that more could have been accomplished.

If your project were to be continued next year, what do you think needs to be working on?

There are a variety of ways that the build containers and virtual machines could be configured. A more thorough understanding of the needs of projects with regard to building and testing could guide a group to making a number of preconfigured containers and VMs that projects could take advantage of rather than having to specify what needs to be installed in the job configuration.

2) Thomas Olson:

If you were to redo the project from Fall term, what would you tell yourself?

Start early; communicate often. Because we needed to use Agile, but the class was designed for Waterfall, we couldn't do the necessary testing for Agile until Winter, putting our documentation a good bit behind where it should've been. We probably would've done better if we utilized IBM's willingness to help to its fullest potential.

What's the biggest skill you've learned?

How to communicate directly and effectively. This includes knowing when and how to ask for assistance and how to ask questions in a simple and lucid manner.

What skills do you see yourself using in the future?

Probably the project management and communication skills, more than anything else.

What did you like about the project, and what did you not?

I liked that it was different from what the other projects were, as we were developing a service more than a product, and how it was quite different from what we had experienced in classes. On the flip-side, I didn't like how the project had almost nothing in common with the majority of the technical classes I had taken, leaving me without a foundation of

relevant knowledge.

What did you learn from your teammates?

The best way to have a project succeed is to leverage the strengths of each teammate to account for their weaknesses, instead of trying to arbitrarily divide the work up without regards to who is good at what.

If you were the client for this project, would you be satisfied with the work done?

I would consider it sufficient for a service that is going to be transitioned to another team for further development and support.

If your project were to be continued next year, what do you think needs to be working on?

Testing, testing, testing. The system could use some stress and scalability testing. It could also do for testing more and more intense builds with more complicated build procedures.

3) Derek Wong:

If you were to redo the project from Fall term, what would you tell yourself? If I could redo the project from Fall term I would start my project earlier and have a detailed plan of what we need to do. One of the key problems we ran into was splitting up work between my team members. We didn't have a good workflow and things got done slow because of it. During winter term, our client suggested us to come up with a list of task to accomplish and that workflow was much better than the one we previously had.

What's the biggest skill you've learned? What skills do you see yourself using in the future? I learned a lot about communication and teamwork. It is very valuable life skill that I can apply to anywhere in the future. In a development team, having good communication is what gets work done and having a good workflow. Having a well defined plan of goals and milestone can greatly improve efficiency. I will be applying all the skills I learned in my future job as a software developer. I also learned a lot of Jenkins as well. Jenkins is used by many development team because continuous integration is a very useful tool in developing software.

What did you like about the project, and what did you not? I liked that our project involves Jenkins because my previous job was using this and I could easily relate to it. I also got to learn more about what it does. I didn't like that our project didn't involve much coding because that was what I was hoping to do from the start.

What did you learn from your teammates? I learned new technologies from my teammates that I didn't know existed, such as ansible and openstack.

If you were the client for this project, would you be satisfied with the work done? I would be satisfied with the work done. We have a functional continuous integration system which is what the goal of this project is. There can be more improvements made, but what we have now is sufficient.

If your project were to be continued next year, what do you think needs to be working on? There can be more testing done to our system and just general refinement. We could add more features or just have a more secure system.

REFERENCES

[1] OpenStack Foundation. Openstack open source cloud computing software. [Online]. Available: http://www.openstack.org/

[2] Chef Software. Chef – embrace devops. [Online]. Available: https://www.chef.io/

[3] Red Hat. Ansible is simple it automation. [Online]. Available: https://www.ansible.com/

[4] Jenkins Project. Jenkins. [Online]. Available: https://jenkins.io/

[5] ——. Jclouds plugin – jenkins – jenkins wiki. [Online]. Available: https://wiki.jenkins-ci.org/display/JENKINS/JClouds+Plugin

[6] J. Molak. (2016, Nov) Build monitor plugin. [Online]. Available: https://wiki.jenkins-ci.org/display/JENKINS/Build+Monitor+Plugin

[7] OpenStack Foundation. Software – openstack open source cloud computing software. [Online]. Available: http://www.openstack.org/software/

[8] Apache Software Foundation. Apache cloudstack: Open source cloud computing. [Online]. Available: https://cloudstack.apache.org/

[9] F. Gaudreault. (2015, Jul) Cloudstack vs. openstack: Follow-up. [Online]. Available: https://www.cloudops.com/2015/07/cloudstack-vs-openstack-is-there-really-a-winner/

[10] Google. Ganeti. [Online]. Available: http://www.ganeti.org/

[11] Apache Software Foundation. Apache jclouds :: Providers. [Online]. Available: http://jclouds.apache.org/reference/providers/#compute-apis

[12] Buildbot. Buildbot. [Online]. Available: http://buildbot.net/

[13] ——. 2.4.1 configuring buildbot. [Online]. Available: https://docs.buildbot.net/current/manual/cfg-intro.html

[14] Strider. Strider. [Online]. Available: https://strider-cd.github.io/

[15] Jenkins Project. Plugins – jenkins – jenkins wiki. [Online]. Available: https://wiki.jenkins-ci.org/display/JENKINS/Plugins

[16] Buildbot. Plugins – buildbot. [Online]. Available: http://trac.buildbot.net/wiki/Plugins

[17] Strider. Strider continuous deployment. [Online]. Available: https://github.com/Strider-CD?query=strider-

[18] Red Hat. (2016) How ansible works. [Online]. Available: https://www.ansible.com/how-ansible-works

[19] Chef Software. An overview of chef – chef docs. [Online]. Available: https://docs.chef.io/chef_overview.html

[20] Puppet. Puppet – the shortest path to better software. [Online]. Available: https://puppet.com/

[21] ——. (2016) Overview of puppet's architecture – documentation – puppet. [Online]. Available: https://docs.puppet.com/puppet/4.8/reference/architecture.html

[22] ——. (2016) Language: Basics – documentation – puppet. [Online]. Available: https://docs.puppet.com/puppet/4.8/reference/lang_summary.html

[23] P. Venezia. (2013, Nov) Review: Puppet vs. chef vs. ansible vs. salt. [Online]. Available: http://www.infoworld.com/article/2609482/data-center/data-center-review-puppet-vs-chef-vs-ansible-vs-salt.html?page=4

[24] Red Hat. (2016, November) Red hat enterprise linux life cycle. [Online]. Available: https://access.redhat.com/support/policy/updates/errata

[25] CentOS Project. (2016, November) Centos product specifications. [Online]. Available: https://wiki.centos.org/About/Product

[26] Canonical. (2012, November) What's the difference between desktop and server? [Online]. Available: https://help.ubuntu.com/community/ServerFaq#What.27s_the_difference_between_desktop_and_server.3F

[27] F. W., F. A., R. R., and R. J., "An updated performance comparison of virtual machines and linux containers," IBM Research Division, Tech. Rep. RC25482, July 2014. [Online]. Available: http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B

[28] K. Kawaguchi. (2016, Oct) Active directory plugin. [Online]. Available: https://wiki.jenkins-ci.org/display/JENKINS/Active+Directory+plugin

[29] A. Mangat. (2010, Sept) Top 7 features in tomcat 7: The new and the improved. [Online]. Available: http://www.developer.com/java/web/article.php/3904871/Top-7-Features-in-Tomcat-7-The-New-and-the-Improved.htm

[30] Oracle. (2010, Jun) The oracle glassfish server advantage for small businesses. [Online]. Available: http://www.oracle.com/us/products/middleware/application-server/glassfish-server-adv-sbiz-wp-080573.pdf

[31] webtide. (2016) Why choose jetty. [Online]. Available: https://webtide.com/why-choose-jetty/

[32] E. Randall. (2010, Nov) Lava lamp notifier. [Online]. Available: https://wiki.jenkins-ci.org/display/JENKINS/Lava+Lamp+Notifier

[33] M. Howard. (2011, May) Radiator view plugin. [Online]. Available: http://wiki.hudson-ci.org/display/HUDSON/Radiator+View+Plugin