CS Capstone Problem Statement

May 31, 2019

Software product life cycle transformation with Docker container techonology.

Prepared for

HP

Bryan Crampton

Prepared by

Group 58

Brendand Byers
Daniel Grocki
David Jansen
Owen Loughran
Austin Sanders

CONTENTS

1 INTRODUCTION

HP must build and deploy a vast collection of software applications and websites for their commercial printing press. They are looking to optimize the process in order to maximize their efficiency, and in turn reduce costs. Specifically, they are curious about the possibility of containerization on a large scale for testing, development and deployment of their printing press software. This leaves our capstone project with a task: research the containerization solutions available, implement and test them on a small section of their utilities before ultimately bringing them to scale. Our goal will be to determine if it is worth HPs time to expand on containerized development, meaning we must show it is cost effective and will improve their efficiency.

In order to accomplish this our group will make use of many different technologies working together one with another to create an in depth and complete final product. I will be focusing my technical research and skills on what programming language we will use as well what tools we will use for package management and version control. Specifically I will be comparing between: Groovy, Java Spring, and C# for the programming language, Git, SVN, and Mercurial for our version control, and Gradle, NPM, and Pants for our package management.

2 Programming Languages

2.1 Groovy

Groovy is an object-oriented programming language based upon java, it has many modern day features and can be used for either compiled or scripted programs through the Java Virtual Machine. It is fully compatible with standard java syntax and can take advantage of any java library. You can often simply use straight Java code and still have compilation success. However Groovy is not just a clone of java, it can be built and ran with less dependencies than java itself, giving it the option of being more lightweight and compact. Being so strongly compatible with java itself will make this language easy to pick up for anyone who is familiar with the syntax of Java. Furthermore groovy also adds some features such as: static/dynamic typing, operator overloading, associative arrays, regex support, string interpolation and more. Most import to our project is the modularity of Groovy, being able to cut and groom the language down to do exactly what we need it to be and nothing more is vital to having an effective implementation of containerization. Additionally we may take advantage of the fact that Groovy can also be used as a scripting language in order to design some portions of our project. Sadly there is not very much of a community behind this language, as it seems to be dying out.

2.2 Java Spring

Java Spring is a framework for Java, created with the goal of being highly scalability for enterprise needs. Java Spring is even more compatible with standard java than Groovy as it just adds different modifiers and alternative ways to more efficiently complete tasks for web based applications. Java Spring is very common in the industry and thus has lots of documentation and support allowing for us to easily research ways to implement what we need. However; Java Spring is primarily marketed as being a tool for building API's with, and while we will be doing some API work this does not fall under the bulk of the project. The decorators in Java Spring can also get fairly complex, possibly making it difficult to fully utilize what Spring has to offer. Regardless Java Spring is incredibly prevalent in industry right now, and it may be worth using for the project solely so that we can become more familiar with it.

2.3 C#

C# is Microsoft's take on a modern backend language, differing from the other two options C# is actually based upon C, and follows the Common Language Infrastructure. Some features of C# include: strong typing, automatic garbage collection, and bounds checking. C# is meant to be portable and accessible to everyone in the world, applications built on it should be usable on any modern system. The language is efficient at memory management and fairly lightweight compared to Java Spring though not as efficient as C itself. C# has numerous libraries to pull in from in order to accomplish various tasks and has a large set of documentation and community behind it to show support.

2.4 Table

	JVM	Modular	Prevalent in Industry	Scripting Language	Common Language Infrastructure
Groovy	yes	yes	no	yes	no
Java Spring	yes	no	yes	no	no
C#	no	no	yes	no	yes

3 Version Control

3.1 GIT

Git is the prevalent version control software used today, it was originally developed by Linus Torvalds the creator of linux. Git is based upon distributed architecture, and is a Distributed Version Control System. This means that every developer using Git for a project has the whole project history on his locale repository, and that no single server holds all the project history. Git was designed to work quickly, securely, and with power. Git does its versioning based on file contents not file names, meaning renaming a file will not flag a change block. Being distributed, Git allows a user to be working on a feature update, but quickly switch back to a current version in order to do a bugfix, all on a locale machine.

3.2 SVN

SVN is the less popular centralized version control system developed by apache. Being centralized means that unlike Git it requires a connection to a central repo. Meaning that the user pulls down a copy of the project files onto their machine, and anytime they want to make a commit they must push it up to server. No connection to the server means no commits. SVN has no automatic merge conflict resolution when multiple users are commiting to a repository. Due to these reasons we will not be considering SVN for our version control.

3.3 Mercurial

Mercurial is much like git, it is distributed so we don't have to worry about the issues that come up with SVN's reliance on a central server. Mercurial was released within one month of Git, and has numerous features that are meant to make it easier to transition to from SVN. Compared to Git mercurial is very inflexible, but does this for the trade of a highly organized system that relies on only one binary file to function (git has 144). Otherwise the two softwares are very similar, Mercurial leaning towards the strict and slow, but organized and exact, with git being fast and flexible, but spread out.

4 Package Managers

4.1 Gradle

Gradle is an open source build automation system similar to apache maven, it was built with multi-project builds in mind. This means that it is capable of bringing together projects from multiple teams to form a compiled and built final product that can be extensively tested and deployed. It is able to build modularly, meaning that it can determine which parts of the build have changed and need to be rebuilt. Gradle was initially developed for Java, Groovy, and Scala, meaning it will support our choice of Groovy.

4.2 NPM

NPM is an open source javascript package manager, it comes preinstalled with Node.js. And is primarily built around a command line client. The client interacts with a remote registry of packages to pull them in and install them into a given directory or project. It also manages all local dependencies so that one can be sure there software will run the same on any developers machine. Sadly it is not powerful enough for our needs, nor does it function with our desired programming language.

4.3 Pants

Pants, similar to Gradle is designed for large codebases, it allows for very exact dependency management and is both scalable and modular. As opposed to Gradle which builds upon Maven, Pants was built from the ground up. Pants take advantage off a concept called a Monorepo, which allows for it's scalability through increased code reuse, easy collaboration, refactoring instead of working around, and powerful dependency management. Pants supports Java, Scala, Python, C/C++, Go, Thrift, Protobuf, and Android code, and while it does support Groovy through plugins it's support is just not enough for our project.

5 Conclusion

So after my research I have decided that we will be using Groovy as our primary programming language due to its modularity fitting well into the lightweight product that we desire, and it's ease for us to pick up coming from java backgrounds. For version control we will use git as we are all already well versed, and due to its flexibility and prevalence in industry. We will also be using Gradle for package management and building our application, it fits perfectly with our other choices and seems to be powerful enough to get the job done.