Development of a Configuration Management Course for Computing Operations Students*

Charles Border[1], Author[1]
[1]I School
Rochester Institute of Technology
Rochester, NY 14623
{cbbics,author}@rit.edu

Abstract

The Operations side of deploying a modern computing application necessarily involves multiple groups working in concert to develop the application and the server side configuration that will support that application. This paper reports on efforts to develop a course that encourages students to dig into issues related to configuration management, security policy development, application auditing, business control issues, and most importantly, team work. While the course is entitled "Configuration Management" it is much more about students creating a process for secure iterative application deployment that borrows extensively from the DevOps movement.

Ansible, our chosen configuration management tool, is relatively easy to work with at the level of complexity that can be reached in an undergraduate class. What made this class different was the attempt made to create a process that would more closely mimic the Operations side of a DevOps workflow. Initial results from the class were encouraging and many lessons were learned.

1 Introduction

In the recent past only a few of the "Unicorn" companies such as Facebook, Amazon, Netflix and Google were concerned about being able to deploy software into production on distributed computing architectures multiple times per day. To facilitate their ability to do this the "Unicorns" developed a series of practices such as Continuous Integration/Continuous Deployment (CI/CD), Test Driven Development, Agile Development and DevOps. The goal of these techniques was to be able to get the work of their thousands of developers into production as quickly, safely, reliably and consistently as possible. Most other large-scale companies have since adopted at least some of the practices of the "Unicorns" and these practices are now considered standard procedure for most large companies. This paper reports on an attempt to integrate the tools and techniques of the "Unicorns" into a Computing and Information Technology curriculum at a regional university.

The perspective taken in this paper is that the need is not so much for our students to work with the technology; it is to understand the mindset and the ideas behind the practices of the large organizations and to experience a version of this through the adoption of a process reminiscent of that of their practices. The fundamental problem that we face in attempting to model the practices of the large organizations is the problem of scale. Our students can get away with avoiding the issue of process because the teams that they work in and the individual projects they accomplish can be encapsulated in small-scale architectures over which they have complete administrative control. They do not have to learn to work together because they control the entire architecture as a single individual or as part of a team that is so small that it can quickly meet and make any changes that are required. Another issue that we have to deal with is that we tell our students too much and do not require them to develop the rules for their work. This is particularly the case with security policies.

In this class, I attempted to empower the students to develop their own security policies and write scripts to determine if the security policies were met. I provided scaffolding for this effort by providing examples that they could use and by discussing potential security policies in class on a regular basis. When you empower someone to make a decision, you have to live with the decisions that they make. Not doing so can create a trust issue that can be hard to overcome.

The class outlined in this paper is a reaction to these ideas and an attempt to constrain and facilitate at the same time student groups to better model the ideas and practices of the large organizations where many of these students will be employed. The paper is organized as follows: The paper begins with a literature review that attempts to build context around these ideas and their relevance in higher education. It then moves on to a discussion of the mechanics of the course and the methodology of organizing and running the course. After this, it discusses the results experienced in offering the course and concludes with ideas related to future offerings.

2 Literature Review

The "Unicorns" have been very forthright about the practices that they have developed to facilitate their operations. In books (7, 8, 9, 13), Facebook and Google groups (3, 5), Youtube and other videos (2) and presentations at well attended conferences and other events (14) the developers of these practices have carefully explained the ideas and techniques that they feel make them able to practice their particular craft. For academics who were listening, the call to change our curriculum was very clear.

Unfortunately, this call has not been well received. Artec, et al. (1) attempted to bring the ideas surrounding configuration management and DevOps (Infrastructure as code and Topology and Orchestration Specification for Cloud Applications (TOSCA), in their terms) into their curriculum. A problem that they ran into is they became so enamored with the rules of TOSCA that they lost the fluidity and experimentation inherent in the requirements to implement DevOps in a real world application deployment.

Olagunju, (10) in the abstract of a Lightening Talks presentation discusses the differences and similarities of Agile and DevOps and makes a call for Information Technology students to understand how to work within a DevOps culture, but in the abstract does not discuss how this might be carried out.

Jiang and Kamali (6) present a more coherent view of the integration of Configuration Management (CM) in the IT curriculum in response to the changes in the IT Curriculum 2005. They discuss a 400 level course that they included in their curriculum called Application Configuration and Management whose goal was to expose students to the practices associated with CM largely from a developer's perspective. They spend a considerable amount of time in their article presenting the proposition that CM is a "Soft" skill that students should be exposed to after they have taken a sufficient amount of courses in more traditional "Hard" skills. They also discuss their surprise when this elective course became much more popular with their students in their networking and security tracks rather than their application development track.

2.1 Class Goals

This class was far from revolutionary. It had a small set of familiar outcomes associated with it and had the following goals for the students:

Operations as a process: the operational support of applications, or Operations is not a dramatic field. To be a competent practioner in Operations requires that you understand and follow a process that reliably deploys applications in a safe and secure fashion. Operations is about process.

Configurations as code treated like all other code that we work with: The title of the course is configuration management and one of the secondary goals of the course is for students to understand how to work with Ansible as our chosen configuration management tool. However, a more important goal of the course is for Operations students to learn how to work with code. How to write code in a manner that allows others to understand and modify their code over time, how to utilize version control systems as a means to communicate with others as part of a team, and to create an auditable archive of configurations that can be controlled over time.

Web scale resilient architectures and processes: When the cloud first came on the scene as a viable architecture for application deployment many organizations looked to it as a way to save money while hosting traditional applications. This has not worked out. What has worked out well is organizations have found that deploying portions of applications to public and private clouds gives them resilience against regional outages and elasticity to respond to highly variable workloads. The problem is that the processes that they used to rely on to support local applications have needed to be adapted to support the new hybrid architectures.

Micro-service architectures (small, loosely coupled services connected via APIs): Workloads for internet-based applications are highly variable and users are notoriously impatient with perceived to be slow applications. Monolithic applications have a role in this field, but primarily as test beds for new ideas. The predominant architecture used to support web scale applications is the distributed micro-service architecture. This is the only architecture that provides a rapidly evolving migration path from small-scale new idea to support for a global user base spread across multiple public and private clouds. The labs in this class force the students to confront the messiness of the migration path from monolithic new idea to a single application deployed across multiple tiers.

Teamwork necessary to support large architectures: Large, distributed architectures require a large number of small, empowered teams to support them. Highly centralized decision-making paradigms cannot keep up with the rapid pace of change in the marketplace. Architectures need to be organized in such a fashion that teams are empowered to make changes they feel are necessary for their small corner of a large application to function effectively.

Testing: How do we know that our architecture will respond in the way that we need to changes in workload?

2.2 Class Format:

The class met on Tuesdays and Thursdays for an hour and a half. The class sessions were divided into lab classes and lectures. In lectures, I covered the concepts associated with DevOps and Continuous Integration and Continuous Deployment (CI/CD) as well as an overview of Ansible and its use. The class had five labs, each completed over two weeks. The class was about process so I developed a predictable rhythm for the class and tried to stick to it.

Labs were due on Mondays, the following Tuesday was used as a project planning session at the end of which the groups were to hand in a project plan with students assigned to either Operations or Security, a list of deliverables with due dates, and a Gantt chart that outlined the dependencies across the deliverables. To facilitate the completion of this and other deliverables I gave students a form to be completed and handed in at the end of the Tuesday class (all forms are available upon request).

The general responsibilities and workflow of the Operations and Security teams were as follows:

Operations Team

- Goal: enhance production environment by creating a more resilient architecture that supports new functionality.
- Create a branch off of main in the versioning system.
- Make necessary modifications.
- Run modified scripts.
- Test functionality of newly created release candidate in the test environment.
- Merge modified script into the main branch.
- Run script to create new production environment.

Security Team

- Goal: ensure that the newly enhanced architecture lives up to the security and audit requirements of the organization.
- Based on required functionality for a lab, develop a list of security policies that the application should live up to.
- Create audit scripts that can be run against the Operations team's release candidate architecture to check to see if it lives up to the security policies.
- Meet with the Operations team to let them know the results of your audit and any changes that need to be made.

To help the students conceptualize the process I wanted them to follow and the points at which the two groups responsibilities intersected I developed the following flowchart:

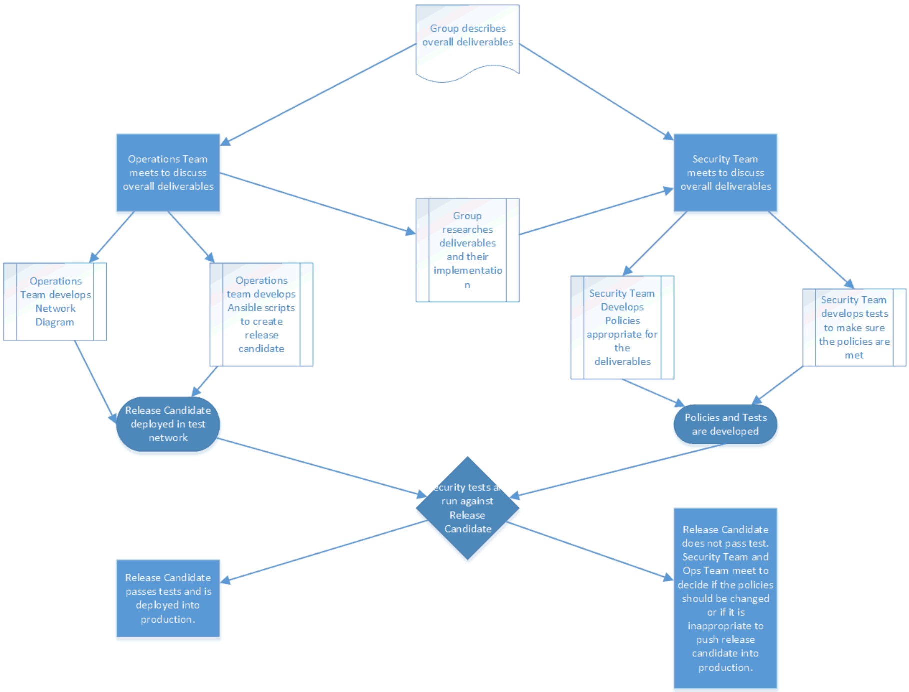

Figure 1: Interactions between the Ops team and the Sec team.

3 Methodology

As part of the introduction to the class, I carefully presented and reviewed the workflow and goals of the methodology I was asking them to pursue. I then divided the class into five groups of six. Because I did not know the students background or abilities I allowed the groups to self-select.

The students were assigned five labs for the course. Each lab built on the previous lab and reflected real issues in the iterative deployment of a web application. Because this class is about operations, and not application development, I based the labs around a basic PHP web site called "Explore California" that included multiple pages with hyperlinks, etc. As part of the fourth lab, students were asked to add a form generation component to the web site that used MariaDB as the backend database. Besides a general overview of the functionality I expected at the completion of each lab, I did not lecture about the steps involved in completing the labs. Instead, I expected them to do their own research to figure out what the operational and security issues were surrounding the increased functionality that came with each lab. I also very carefully laid out my justification for each of the labs. My goal in this was to help the students to understand that the series of labs they were working on represented some of the typical evolutionary paths that organizations pursue as an application becomes more popular and experiences increased workload.

3.1 Labs

The labs were developed to attempt to help students try the processes that we discussed in class.

Lab 1: The Golden Image: In this lab the students were to become more familiar with the Linux image that would be the basis for all their subsequent work. The Operations team was expected to write an Ansible script that would strip out all the unnecessary components of a CentOS image and prepare the image for their later work. The Security team was expected to develop a set of basic security policies and the scripts that they could run to make sure that the golden image satisfied those policies.

In an effort to help students to get started and also to give them an example of what my expectations were the following set of security policies that could serve as a starting point for their work was provided:

- All code must be stored in a versioning system that records changes made to the code and preserves a longitudinal records of who made the changes and why.
- All scripts must clearly be labeled as follows: the author of the script, the date it was written, the version number of the script, the date it passed

the Operations test suite, and the date it passed the security test suite, the problem the script is addressing, and the system that it impacts.

- Only packages needed to satisfy operational requirements may be installed on an operating system.
- No unnecessary user accounts may be installed on an operating system.
- Only file systems, directories and files needed to satisfy operational requirements may be installed on an operating system.
- Only open ports needed to satisfy operational requirements may be installed on an operating system.
- Only those SSH keys needed to satisfy operational requirements may be installed on an operating system.
- The Host file may not include any other entries than those required to satisfy operational requirements may be installed on an operating system.
- The Firewalld status must satisfy operational requirements of an operating system.

I also went over the types of Bash scripts that could be developed to create an audit script to illustrate that the security policies were satisfied.

Lab 2: Deploy a Monolithic (Single Server) Web Site ("Explore California"): Often when an organization has a new idea for an application their goal is to develop a minimum viable product version of the application to quickly test if there is sufficient user interest in the application to justify continued development. This is often done as a monolithic deployment where all the components of the application are deployed on a single server. This idea was the basis for the second lab.

Lab 3: Load Balancing across Multiple Versions of the Monolithic Application: In this lab students were tasked with deploying multiple versions of the monolithic application and a load balancer to spread traffic across the multiple versions.

Lab 4: Create a Loosely Coupled SOA by Adding a PHP Based Form Service Using a Single Instance of MariaDB: This lab required that students modify the "Explore California" web site to allow users to request information by filling out a form with their name and address. This form service was to be developed on a separate instance of MariaDB that would be accessed by all the instances of the web servers. While this is not a particularly difficult architecture to build, for this class it created a significant change to what the students were used to.

Lab 5: Release Engineering: Create a process where you can quickly switch from one version to the next. Adoption of DevOps and CI/CD methodologies are driven by the need to respond quickly, accurately and consistently to changes in the market. In this lab I try to get the students to think about how they could create a reusable, and consistent process for substituting a new version of an application for an old one.

3.2 Group Project:

When students are working on the labs that I assign them they are working on what I think is most important. In the group project students are encouraged to think about how the technologies we have discussed in class can be applied on issues they think are important. I discuss this group project every week during class and spend a lot of class time helping students to understand its importance.

The process that I use for the group project has four steps. In the first step I require each group to develop what amounts to a project plan or proposal for what they intend to do. I try to respond to each proposal within a day or two. Based on my feedback students are tasked with actually doing the work they have proposed to do. I usually give students at least a week or two to do this work. The next step involves students presenting their work to the class. After this they write a formal report outlining their successes and challenges.

4 Results

I ran this class for the first time in Fall 2018 with 30 students divided into six groups. The class was run a second time in Fall 2019. Most of the students in both iterations were from our Computing Information Technology program and had taken scripting and system administration courses prior to this course. I allowed the students to form their own groups and most of the students chose group members they knew from past classes. Based on their scripting and other programming background none of the students reported significant problems working with Ansible. Students were allowed to choose between three architectures to support their lab work, a university owned VMWare private cloud, VMWare workstation on lab machines, and any virtualization environment on their own laptops. The prime determinant of which architecture most groups chose was the degree to which the architecture was remotely accessible with the private cloud being the most easily accessed by all members of the group. VMWare on the lab computers required that anyone who wanted to access the architecture had to physically be in the lab (which has a fair amount of availability), and have shared access to the group's code repository. Relying on a student owned laptop to host a virtualization system was the worst from many perspectives as the group would have to assemble to do their work. While I made this option available to the groups, I warned them of the problems of actually using it to host labs.

4.1 Student Performance:

While all the students in the class understood why the topics we were discussing were important, the groups who had participated in multiple Internships/co-operative work experiences seemed to come to the material more quickly. Students did not have much difficulty learning how to use Ansible. Some groups did have trouble deciding what needed to be done and then allocating the work across group members.

Dividing the groups into Operations and Security teams created a new dynamic that some groups had a difficult time getting used to. The role of the Security team in developing security policies which were to be included in the Operations team's configurations made it seem as though the Security team was creating work for the Operations team or, at least, making their work more difficult. This is a dynamic that I chose to set up and represented the work of real DevOps teams. My only effort to assuage this dynamic was to ask that group members switch from Operations to Security and vice versa throughout the class.

4.2 Student Feedback:

This was the first two times that I have run this class and student feedback was not altogether positive. Some students felt that the labs were too easy and some felt that they were too hard. Other students wanted me to spend more time telling them what to do and how to do it. I am happy with this kind of feedback and feel that I must have been doing something right.

5 Future Work

5.1 Labs:

I feel that I made a fundamental mistake in the way that I structured the class. One of the key ideas I was trying to stress was that in a DevOps workflow changes are made often and each change is small. By having only five labs, each of which made a dramatic change to the architecture I violated that idea. In the future, I hope to have many more labs each of which represents a small change in the architecture. I expect this to have the following benefits: The groups will learn to work together faster and more consistently. Since one of the goals of the course is to get students used to working through a process, the more they can work the process the better they should learn it.

5.2 Group Project:

I am continually impressed by the projects that our students come up with and their ability to do interesting and relevant work. The projects were the best part of the class and allowed students to display both their command of the technology and their creativity.

5.3 Project planning:

Project planning has been a continual weakness for our students. In most of our classes students are told to complete a lab and they immediately start hammering on the keyboard trying to get it to work. In this class, the required project plan was my attempt to force the students to think more about developing a plan to complete their work before they started hammering on the keyboard. To a certain extent, I think that it worked. In a class about process the more planning we can have, the better.

5.4 Complexity of the Ansible scripts compared to industry based issues:

Some of the comments that I got from students dealt with the idea that the scripts that they wrote were either too hard or not hard enough. Below is a typical workflow from an industry based rolling update of a cluster of servers:

- Consult configuration/settings repository for information on involved server.
- Configure base OS on all machines and enforce desired state
- Identify a subset of the web application servers to update.
- Signal the monitoring system of an outage window prior to taking the servers off line.
- Signal load balancers to take application servers out of the load balanced pool.
- Stop the web application servers.
- Deploy the desired update in code, data, and/or content.
- Start the effected web application servers.
- Run appropriate tests of functionality on the renewed servers and code.

- Signal the load balancers to put the application servers back into the load balanced pool.
- Signal the monitoring system to resume alerts on any detected issues.
- Apply the above process to any other groups of application servers that need to be updated.
- Apply the above process to any other types of servers (such as database servers).
- Send email reports and audit reports regarding the status of the servers and the update.

While the above may reflect the needs of an industry application, I am not sure that for my class I add value to the student's abilities by forcing them to deal with all of this complexity. My question is, if I were to ask my students to do all of this work, how many of the weaker students would I lose and how much of this would just be busy work? Students today have very complicated lives and often have other issues that require their time other than classes.

5.5 Alternative Architectures:

Several student groups used the class project to implement one or more of our labs using Docker containers. Some even included Kubernetes as a container manager. The use of Docker containers represents real opportunity for this class as the small size and speed of implementation for Docker containers cuts back on the operational overhead of using virtual machines on our VMWare private cloud. There is an increased level of complexity around important issues such as persistent storage and networking with Docker. The question is does the increased complexity add value to the class given that we have access to our private cloud?

6 References

[1] Artec, M. et. al. DevOps, Introducing Infrastructure-as-Code. In proceedings of the 2017 IEEE/ACM 39th IEEE International Conference on Software Engineering Companion., IEEE 2017.

[2]Dickson, C A Working Theory of Monitoring. USENIX LISA'13 November 3-8 2013, Washington, DC.

[3] Facebook Code: https://code.fb.com/web/rapid-release-at-massive-scale/

[4] Forsgren, N, et. al. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. IT Revolution Press, Portland, OR 2018.

[5] Google Tech Dev Guide: https://techdevguide.withgoogle.com/

[6] Jiang, K and Kamali, R. Integration of Configuration Management into the IT Curriculum. In Proceedings of SIGITE'08 October, 16-18, 2008 Cincinnati, OH.

[7] Kim, Gene, et al., The DevOps Handbook: How to Create World-class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press, Portland, OR, 2016.

[8] Kim, G. et. al. The Phoenix Project: A Novel about IT, DevOps and Helping Your Business Win. IT Revolution Press, Portland, OR, 2016.

[9] Limoncelli, Thomas. The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems, Vol. 2. Addison-Wesley, 2015.

[10] Olagunju, A, Revamping the IT Curriculum with Agile and DevOps Methodology. In Proceedings of SIGITE'18 October 3-6, 2018, Fort Lauderdale, FL.

[11] O'Reilly Ideas: https://www.oreilly.com/ideas

[12] Pan, Y, Security Auditing Course Development. In Proceedings of SIGITE'07 October 18-20, 2007. Destin, FL.

[13] Schlossnagle, T. Scalable Internet Architectures, Sam's Publishing Indianapolis, IN, 2007.

[14] USENIX Conferences: SRE Con: https://www.usenix.org/srecon, LISA: https://www.usenix.org/conference/lisa19