BCGEU Long-Term Assisted PD leave Report 2020-2021

May 19, 2021

Graham White, Department Chair, Information Technology and Applied Systems

Table of Contents

Introduction

DevOps, or Development Operations, has become an important skill set in current IT operations. Nothing stays static for long in software which is why there are constantly new versions of our favourite apps, Operating Systems (Windows updates!), smart phone applications and so on. This is also common on websites as new features are constantly added or the look and feel changes to adapt to new technologies and end user requests.

Behind the scenes of all these changes is a team of software developers who spend a lot of time coding the new features, testing the code, and fixing bugs before they "deploy" the code files to the production servers that the end users around the world interact with. All this coding development needs to be managed – with a team of developers there must be certainty that everyone sees their teammates contributions, and that they are all working with the updated and correct version of the files. They must be careful that they do not overwrite or change code that someone else on the team is working on. The code then needs to be deployed to test servers for support staff to trial and indicate any bugs found. Ultimately new versions of the software will then be deployed "live" onto many production servers for clients and end users to use. In this process there needs to be assurances that all the servers have the same versions of the code files (no older versions on some) so that the end experience is the same for every user! This management of code integrity for the developers, ensuring that test platforms are available and functioning properly, and then facilitating the deployment in a consistent manner of new software versions to production servers is the role of the DevOps engineer.

The foundational technical skills required to be a DevOps engineer was the focus of my professional development. With the need for this skill set growing and the high demand for this career role, it is crucial that we have a knowledge foundation within our faculty to plan and build our ITAS curriculum for our students so that they are ready to enter this career path.

Research and Activities

I used a number of online learning sources and courses to research DevOps skill sets including Cloud Academy, Udemy and Microsoft Azure's DevOps learning system. My goal was to research and study more about the overall DevOps philosophies and methodologies and then build an applied environment so that I could experiment and simulate the workflow in action to reinforce the theoretical concepts. The two major areas of focus and study that were attached to my DevOps research are below – I will expand on the theoretical and applied aspects for both:

- Code versioning systems
- Continuous Integration and deployment of test and production builds

Code versioning systems

Code versioning is extremely important. As mentioned in the introduction, software developers usually work in teams which can be quite large and the amount of files that are joined together to make a piece

of software function properly can be well over a 100 files if not much more. It is important that an individual developer in the team can write or update their section of the software code and then have it tested in combination with the various "chunks" or grouping of code files worked on by their teammates. This merging of all the files is then automatically deployed to a testing platform daily to see how it all fits together and to test the software in its new state. This process is called continuous integration and it involves code versioning software/systems to manage it. My goals for this part of the research:

- What are the common code versioning systems being used.
- Setup a test system and install and work with a code versioning system to understand its fundamentals.
- Try some test deployments with different portions of the code files.

Results

I started by building a few Linux virtual machines that I could use as a testing platform. It became pretty clear from my research that the Git software from the company GitHub is the most common code versioning system in use today, so I built my test environment around its software. Using Git commands, I was able to build a main branch of my "test" software. The main branch is really just a grouping of code files that work together to run an application. From there I was able to pull down the main branch onto another virtual machine so that I could emulate a few developers working together on the software – developer A and developer B. I then made changes to various files in the offshoot branches each as developer A and B. Using Git, I could then push these changes in the files up into the main branch – Git would tell me what developer B had changed and vice versa with developer A. I could pull down the changes from each developer individually to "help" them with their code if necessary, as a separate branch before committing and merging any changes and improvements back into the main branch. The overall system would notify of any conflicting versions of files and would ensure that everything was merged properly so that we were not stepping on each other's work and creating issues with the files.

Practicing this code versioning skill set was ideal in anticipation of new DevOps curriculum we are adding in 2022 as a specific course – ITAS 276 DevOPs – and for updating current curriculum. I will be adding foundational pieces of version control into my ITAS 181 Introduction to Linux course in the next year. This will introduce the 1st year students to this process in anticipation of more advanced aspects of this in second year.

Continuous Integration

While it is great that we have the means for software developers to easily work together and collaborate their coding in an efficient and integral manner, it is important to test the software as the developers work on it. Software can get quite complex – you can imagine the amount of testing Microsoft does before releasing a new version of Word or Excel. Companies will often employ a team of people as quality assurance specialists to ensure that as many bugs and issues are resolved before the software is released for end users.

For developers, this testing process is often done on a daily basis, perhaps multiple times per day. This is ideal as the number of new errors or issues will be limited instead of a massive amount of errors that

could build over time if you tested less frequently. Continuous integration essentially means that as the developers commit their code changes (again, all in files), a set of software will automatically take the main branch with the current updated files and build the new version of the software in a testing environment. The quality assurance team as well as the developers themselves, can then trial the software in its current state to see if the changes produced the desired effect and to ensure that no errors are introduced that an end user or client might encounter.

Results

There are a number of tools that can integrate with code versioning systems such as Git to produce these automated builds. I chose one called Jenkins. It took a fair amount of time to install properly and then learn how to manage it. Using YouTube and other online articles I was able to get it functioning at a reasonable level. It was pretty cool how it worked with Git. I could make changes to code files, and then when I felt like it was ready, I could commit the code into the main branch. I then configured Jenkins to "notice" when new code changes were committed via Git. From there it would combine all of the code files together in a process called compiling, build the latest version of the software and then run it on a distinct Linux virtual machine that simulated what a production server would provide for end users. I could then test my very simple application by visiting it in a web browser as if I was an end user. All of this was automated saving me (and developers in real world scenarios) a lot of time from having to do this manually!

This was a very useful set of research as automated deployments have become the norm now in the IT industry. The obvious time saving benefits is a huge plus for companies as it has drastically cut down deployment and testing times allowing development and Q/A personnel more time for building/testing the software. This can obviously save a lot on expenses and allows the company to be rapidly adaptable to required changes and improvements. This was fairly complex process and not ideal for an introductory course in 1st year ITAS. However this automation process will be incorporated in the ITAS 276 DevOps course and other 2nd year Web and Mobile development courses which focus on software design and builds.

Conclusion

This professional development was beneficial in learning more about the modern IT development cycle. While these types of processes are constantly changing there does seem to be more of an accepted standard then there used to be. It was useful to work through this cycle from the initial code version phase through the deployment to a test platform. My understanding of it all is a lot stronger. This will help facilitate the curriculum building we need to do in order to stay current with not only employer expectations, but prospective student expectations as well as more and more post secondary IT programs will begin to provide this curriculum.

Summary of research and professional development:

Here are the key areas of development and knowledge that I gained while researching DevOps methodologies:

- Modern code versioning and organization techniques
- Working with versioning software and becoming more familiar with Git and GitHub
    - Prevent code file conflicts.
    - How to collaborate with another developer working on the same file or files
    - How to pull down test branches and experiment without interfering with the main branch of code files
    - How to commit code changes
- How to design and build a testing platform for new builds of software
    - What software was required for the Linux environment
- The concepts of continuous integration and advantages of frequent deploys.
    - What are the optimal times to try test builds – mechanisms for determining that frequency.
- Automating deployment of new code to the test platforms with Jenkins.
    - Installation, configuration and maintaining the Jenkins software