12-15-2022

DevOps: Course Development

James Lee VanderZouwen

DevOps: Course Development

James Lee VanderZouwen

A Project Submitted to

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Applied Computer Science

School of Computing and Information Systems

December 2022

The signatures of the individuals below indicate that they have read and approved the project of James Lee VanderZouwen in partial fulfillment of the requirements for the degree of Master of Science in Applied Computer Science.

_____

Robert Adams, Project Advisor Date

_____

Robert Adams, Graduate Program Director Date

_____

Paul Leidig, Unit head Date

Abstract

DevOps has become somewhat of a buzzword amongst software engineers in the industry. Often developers do not have a dedicated DevOps engineer let alone a DevOps team. Developers benefit when they know what happens between ‘works on my machine’ and production. Making sure those steps make sense and are safe benefits the operations team. From compliance to code review to regression testing, understanding the full SDLC, employing DevOps concepts, and minimizing overhead from dependencies is quickly becoming a pre-requisite for the modern software engineer. This project attempts to bridge the gap between buzzword and best practice by developing a college-level course on DevOps. The course covers time-tested DevOps concepts in a stack-agnostic, discussion-based approach. The course materials include a syllabus with detailed student objectives, weekly slides, and hands-on activities paired with discussions that give the learner the experience they need to confidently identify gaps in the SDLC and recommend solutions.

Motivation

The DevOps course delivered is intended to be given as a college-level class in a discussion setting. Included in the deliverable are:

- Course catalog listing, description, and prerequisites
- Syllabus
- Detailed course outline
    - Learning objectives
    - Actions / homework prompts
    - Content examples
- Midterm and final assessment examples
- Class session examples with notes

Throughout the MS Applied Computer Science program, DevOps concepts have appeared, but often learning stops at exactly that – the concept. Similarly, my career as an engineer showed me that operationalizing the concept is the most difficult part of the problem.

As a part of a DevOps team for my job, I'm able to speak as the subject matter expert on a subset of DevOps topics in context of my job. I approached my master's degree with the hope that I could extend and update my knowledge of DevOps to more generalized in my approaches as well as be a better trainer. One of the benefits of this is that I can use this course at my organization for our engineers to help them understand DevOps concepts. Often DevOps is not the responsibility of a dedicated resource. Small teams often leave these concerns to the developers. For me, being able to synthesize this information into a concise guide would have been very helpful, so I'm trying to pay it forward by explaining some of the hurdles I've come across at my office, in this program, and on this project.

The program path chosen also allowed me to conduct further deep dives into applying course topics to existing and legacy systems. I tried to use my project time for class to address low-priority but high-value challenges at work. Projects included adding SAST scans and enforcement in my organization that led to more secure code. Mapping our document management system dependencies so that we could modernize them was also an effort that sparked a large change at my office. Conducting research for this project with peers at work helped me learn what gaps there are in understanding of what my team does at the office. It seems that DevOps is considered a post-development concern to most and my aim is to change that.

This project is important to me because I believe no team is complete without DevOps. It's so much easier to consider operationalization concerns at the start and with the whole picture in mind. An ounce of DevOps saves a pound of production support any day. I'm trying to evangelize DevOps to my team, my peers, and any student who wants to learn.

Project Management

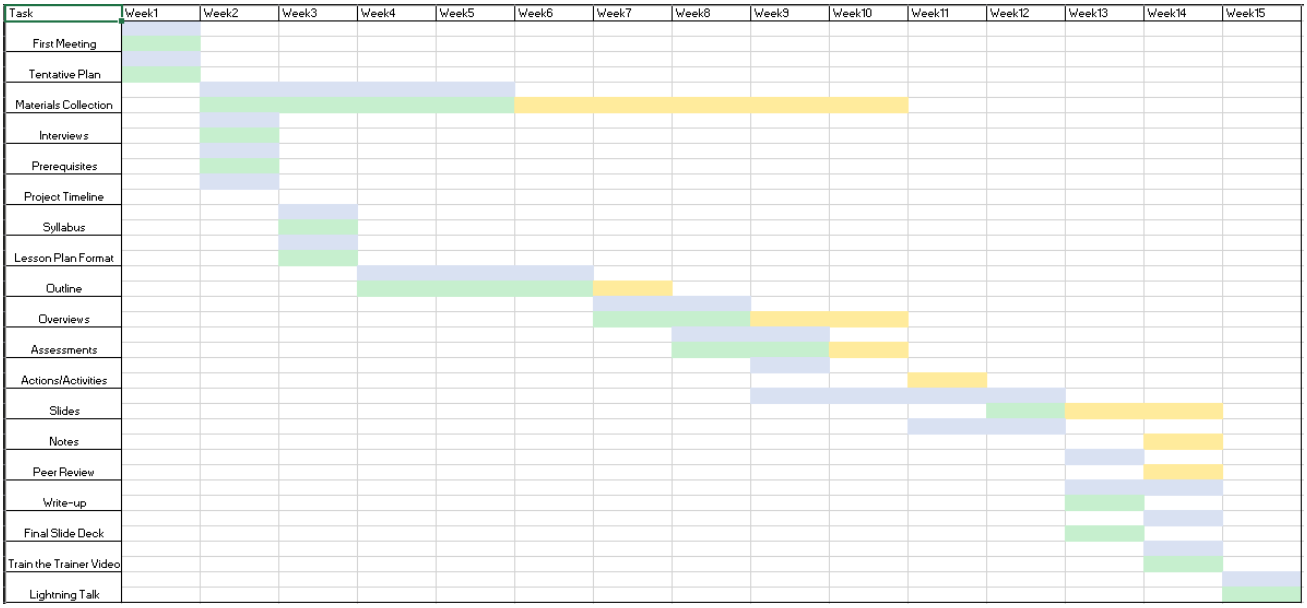
My goal for this project was to provide a full-featured college-level course on DevOps. The initial project management was done in the initial meeting with backward planning using deliverable dates. This led to a project timeline dividing each of the 15 weeks in the semester into blocks of tasks.

Project Timeline

Date	Working on
31-Aug	First Meeting, Tentative Plan
7-Sep	Materials Collection, Begin Interviews, Prerequisites, Project Timeline
14-Sep	Syllabus, Lesson Plan Format, Sources
21-Sep	5 weeks of slides + Course Intro and base modules
28-Sep	
5-Oct	Feedback / Adjust
12-Oct	5 weeks of slides + Midterm Assessment Plan
19-Oct	
26-Oct	Feedback / Adjust
2-Nov	5 weeks of slides + Final Assessment Plan
9-Nov	
16-Nov	Feedback / Adjust
23-Nov	Peer Review
30-Nov	Final Slide Deck
7-Dec	Train the Trainer' Video
14-Dec	Final Write-up / Lightning Talk on 12/15

Deliverables for the capstone project impacted the dates at the end of the timeline and initial preparation was necessary to determine the work to be done in the middle of the semester, so the timeline was intentionally left vague. As the planning and investigation portion of the project wrapped up and implementation began, a Gannt chart seemed appropriate for tracking plan versus progress. At this point the project evolved and the idea of providing completed slides up front was thrown out. An iterative approach was used instead to continuously improve the outline of topics into finer and finer detail so as to create better continuity through the course

as well as to be able to complete all sections to the same granularity by the end of the project time.

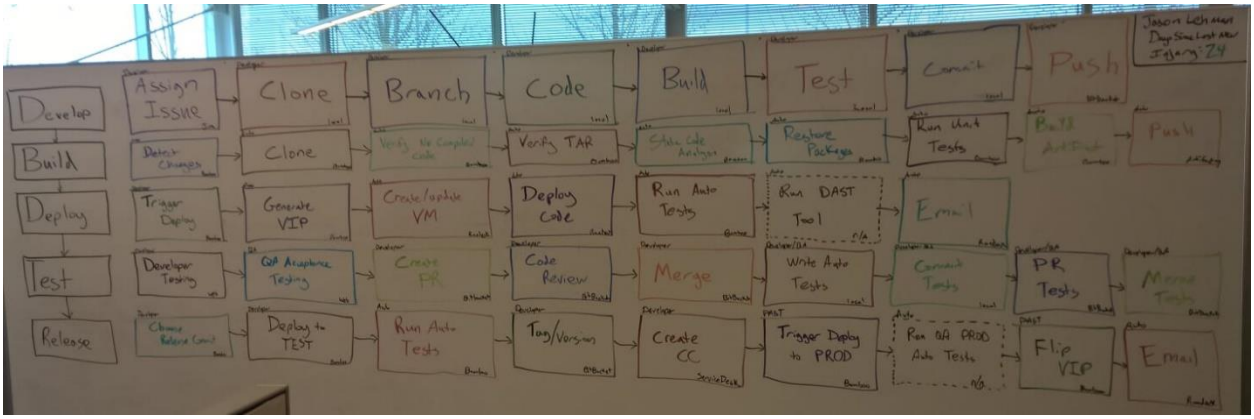


The Gannt chart above shows that materials collection ended up being an ongoing task throughout the project. The yellow shows tasks that went beyond the intended week (green blocks are time spent in the planned week). It seems logical to assume that I became quite busy around midterms with family, holidays, school, but I was able keep my tasks in check still using this chart and seeing that I was behind schedule.

In terms of individual tasks completed during each week, these were discussed during the weekly call and agreed upon using the current information each week. These were basically run like a stand-up meeting where we discussed what was done, what will be done, and if anything needs to be changed (blockers). This worked well because if the plan had been finalized up front, it would not have been completed. Overall, I would call this project management style 'agile-lite' in that this project benefited from minimal artifacts and less stringent project management to maximize time spent on the actual project.

Course Organization and Topics

The course is organized into 15 weeks that include three 45-minute topics. This format is the standard college course format (45 class periods minus intro, assessment review, and assessment times ~ 33 class periods of new content). To determine how to sequence the course, I used a bakery-style approach to building a continuous delivery pipeline if each step needed explaining from an operationalization context. This pipeline I documented for my former employer was a good starting point. I created a list of topics from here:



In 2014, Microsoft had a DevOps simulator that had the following categories that I like to use as top-level grouping for DevOps concepts as well:

- Automated Testing
- Continuous Integration
- Infrastructure as Code
- Application Performance Management
- Continuous Deployment
- Release Management
- Configuration Management

Of course, times have changed since 2014, so I added the following topics based on my own experiences as well.

- Dependency Management
- Training
- Security

Much of the first portion of the course is ensuring good developer habits that are necessary to ensure time-consuming code project activities can be easily automated. Development is the first part of DevOps, so much of the focus is on creating operationalizable code in the first place. The sequencing of the course is based a Gartner CI/CD course from 2017. It basically said {Develop, then Continuously Integrate, then configure environments automatically, then deploy automatically, then test automatically, then do fancy stuff after that…}. Using that as a base, I worked with the semester schedule to ensure like concepts were grouped and set such that they would be on the same assessment.

- Week 1 – Intro/housekeeping
- Week 2-4 – Develop
- Week 5-7 – Continuous Integration
- Week 8 – Security
- Week 9 – Environment Management
- Week 10-12 – Automatic Deployment and Testing
- Week 13-15 – Training and Culture

These concepts were further broken down over the course of the project into learning objectives. Each session has a Terminal Learning Objective (TLO) that should be considered the primary goal of the class session. If one thing is learned, this is the main thing. Extended

Learning Objectives (ELOs) generally support or are related to the TLO but aren't necessarily sub-topics. Assessments are based specifically on the bullets in the learning objectives.

Actions are also defined for in-class actions other than discussion or slides. Conditions for these activities are provided if necessary.

Asset links are provided to current supporting assets, documentation, or information. The class is meant to be an active discussion. The topics and information are high-level, so sharing of experiences and understanding between students is important.

Assessments include multiple choice, matching, true/false, and short answer questions. Assessment review should be interactive to ensure students are comfortable answering questions about DevOps directly.

Reflection

This project is close to what I was hoping to achieve, but I would have liked to finish all the PowerPoints for the class sessions. I also wanted an a la cart 'choose your own adventure' type class that allowed the teacher or student to pick a sub-set of topics they want to focus on. However, I learned that collecting, sequencing, and making sense of all this information in a teachable/testable way is a lot of work. Instead of getting all the PowerPoints done, I ended up continuously improving the outline instead. I found it impressively difficult to sequence topics. I'm happy with the amount of work I was able to get done, but I believe it would have been more effective in a completely teachable form.

This work is highly dependent on my experiences in the DevOps field, so I hope it's useful outside my stack and situation. This is valuable information to the right audience, so I'm glad that I was able to get this down into a concise informational document that I can use professionally. I look forward to sharing with my team at work.

My concern was covering DevOps in a 'not trying to sell you a magic bullet' way. As I stated, DevOps has become a sort of a buzzword in the industry and clearly on the search engine. Simply typing 'DevOps' and searching will result an many promises of success upon subscription. This is not DevOps. If someone says I have a tool that can help you with DevOps, they don't know what DevOps is. I tried to stay away from specifically defining DevOps in favor of sharing tips with developers on how to operationalize their processes.

I've attached my course outline as an appendix, and I tried to focus on making the outline itself as helpful and full of 'pro-tips' as possible.

Conclusions

This project broke down DevOps concepts into undergrad-level blocks of instruction ideally spread across 15 weeks of 3 session/week, along with 2 assessments. Getting these ideas on paper in a readable format was important to me.

Open issues are creating the slides for classes from the objective/bullet-point outline. Train the trainer clips for each week would be nice to have as well. I would like to see this used for DevOps education either online, at a college, or at an organization.

I am planning on using the assessments because many of the questions will work as interview questions in the future. I would like to perhaps convert parts of this into a talk for the BeerCityCode conference to get some experience delivering talks as well as feedback on the content. I've enjoyed the idea of sharing my experiences with others and look forward to trying it out with content I created. Part of the reason I pursued a master's degree was to enable a potential future in which I can do more mentoring. DevOps is a growing field, so I expect these skills will be in demand and I'm excited to share.

Appendices

Course Description:

DevOps

Development Operations: Automated Testing, Continuous Integration, Infrastructure as Code, Application Performance Management, Continuous Deployment, Release Management, Configuration Management, Dependency Management, Training, Security. Introduction to Continuous Delivery Pipelines. Prerequisites: Either (CIS163/CIS500);

Course Objectives

- Explain fundamental DevOps Concepts
- Identify and Determine requirements for CI/VCS
- Model SDLC of an application
- Understand Continuous Delivery Pipeline Architecture
- Apply DevOps Concepts to Operationalize a Proof of Concept

Course Asset Example:

See Attached PowerPoint Slides

Lesson Plan:

See Attached Course Outline

Assessments:

See Attached Midterm and Final Assessments with Rubrics