Report on Second DevOps Educators' Workshop

Len Bass – lenbass@cmu.edu

Notes compiled by Stephen T. Frezza (Gannon University, USA) and Norha M. Villegas (Universidad Icesi, Colombia)

The second DevOps Educators' workshop was held Nov 7 as a portion of the 2017 Conference on Software Engineering Education and Training in Savannah, Georgia. About 25 educators attended from every continent except Australia and Antarctica. The workshop was divided in two portions. In the first portion, Len Bass presented information about the course he teaches at Carnegie Mellon University. This material can be found at https://github.com/devopseducator/2017workshop. This is also the location where a copy of this report can be found.

Questions asked during the presentations:

What resources are available for instruction?

LenBass Text: https://www.amazon.com/DevOps-Software-Architects-Perspective-Engineering/dp/0134049845

InfoQ magazine. https://www.infoq.com

Humble & Farely: Continuous-Delivery-Deployment Addison-Wesley

What are the responsibilities of a system operator?

See https://en.wikipedia.org/wiki/ITIL for a detailed description. DevOps practices mainly impact incident handling and monitoring of systems.

How does automated testing affect QA?

Automated testing is a pre-requisite for continuous deployment. Continuous development could integrate some limited human testing depending on the deployment schedule. The role of the QA team changes dramatically as a result of automated testing. Some organizations do away with the QA team and make the development of the tests a developer responsibility. Other organizations put the QA team in charge of the automated testing.

Under which conditions is 'rapid' deployment possible or even desirable?

Software as a service – rapid deployment is not just sensible, but required. For other types of delivery, it depends on the business model of the organization. The pressure for rapid deployment comes from time to market considerations. The extent to which time to market dominates other concerns depends on competitive and customer pressures.

Discussion questions

The second portion of the workshop was discussions centered on the topics suggested by the attendees. The topics and a brief summary of the discussions are:

How & where to teach DevOps? Separate course vs. integrated into other courses?

Some aspects of DevOps could be taught in foundational courses. Considering the operators as first class stakeholders could be taught in an introductory programming course. Automated testing could be introduced in a second or third programming course. Simple networking concepts – URL to IP address, ports – could be introduced in the course where students learn how to write web pages. The tool chain could be taught to seniors in their capstone course. DevOps is an application of many different concepts and mentioning DevOps practices when the relevant concepts are introduced is possible. The important thing is that the concepts are taught somewhere whether in a separate course or distributed.

How in an academic setting, do we establish Multi-team projects?

One course at RIT - Engineering enterprise software… Breaks the class up for multiple components, managing coordination among the teams. See www.se.rit.edu/~swen-343

Another idea – teams compete to propose various components of the project. The winner becomes the manager for executing the contract to have the teams collaborate to produce the various components of the broader system.

Teams with back-end , front-end collaborative projects. With the instructor as the project leader to help enforce the process and coordination.

Feedback loop in DevOps course?

Continuous deployment supports the idea of "fail fast". That is, errors, once detected, can be corrected and fixes deployed within hours rather than waiting for the next release. Getting students to appreciate this aspect of development can best be done in a capstone course where the students have a real project with real deliverables. The concepts of a tool chain, automatic testing, and automated deployment all are necessary to appreciate the feedback loop and without a real project it is very difficult to integrate all of these concepts.

Choice of tools and depth of understanding each tool.

A tool chain consists of a tool like Vagrant that sets up a uniform environment, a continuous integration server, automated testing tools such as Junit, and configuration management tools such as Chef. Deployment tools such as Spinnaker may be used for multi instance deployments. There are a wide variety of tools in the DevOps space but many of the leading tools are open source with extensive support materials available on line.

Having a student understand tools and their use is no different than having them understand compilers and their use. An assignment is given and it is the student's responsibility to figure out how to install and use the tool with some help from teaching assistants. The depth and pain of understanding each tool to

which the student is subjected is an age old question in computer science. It is comparable to the question how much does the student need to know about the compiled assembly language of a higher level programming language? There is no good answer to this question.

What are the implications of Continuous Integration… Continuous Delivery… and Continuous Deployment?

Each of these activities has implications on the process of producing a system. Continuous delivery has implications on the architecture of the system being built. An associated question is “what kind of instrumentation should be put into the system being built and who is responsible for interpreting the data produced by this instrumentation?”. DevOps practices are changing the answers to these questions. Treating operators as first class citizens brings error messages and logging into focus. Failing fast requires data be quickly available to someone who can recognize when a system is misbehaving in terms of performance or reliabity.

Key aspects in the discussion:

The attendees felt the following were the important take aways from the discussion:

- Operators must be seen as part of the stackeholders
- We teach more dev than ops, which makes difficult to teach students how to realize the DevOps feedback loop.
- For project-based courses it is fundamental to recreate the environment required to experience the whole loop