The first DevOps Educators Workshop was held on November 19, 2016 at Northeastern University in Seattle, WA. The goal of the workshop was to share ideas and materials for teaching DevOps and to plan ways to create a sustaining community of DevOps educators. Nine faculty and one industry developer from the US and Canada participated.

The workshop focused on the skills students need in order to understand DevOps. The skill set was segmented into prerequisite andadvanced. A spiral approach for introducing them, where topics are introduced early in the curriculum, then repeated with more complexity in later courses, would likely be the easiest way to push the topics into courses already crowded with topics. The workshop attendees used a constructive approach to generate the list of advanced DevOps skills: activities and exercises would bring students up to speed in each area. The list is not exhaustive and is intended as a starting point for further discussion. While it may be possible for a standalone DevOps course to accomplish most of the following, again an iterative spiral approach would likely be better. Also, specific tools are mentioned below, but any set of equivalent tools would work.

The set of prerequisite skills include the following: basic networking – knowledge of addressing, connecting to remote machines, and protocols used in distributed computing; scripting – shell scripting and some interpreted language like Python or Ruby; simple version control; command line interfaces – using an editor, basic Unix command, and running scripts; and the basics of troubleshooting runtime problems in applications. While some computer science programs provide these skills, even graduate students often lack one or more of them. One suggestion is that curricula recommendations, such as the ACM and SWEBOK, be updated to include both these skills (and possibly the more advanced concepts below).

Networking and distributed computing skills and related exercises include the following. Virtualization concepts are key here: scripting to launch VMs locally and on the cloud, using exercises from Docker and Amazon Web Services; automatic deployment with Vagrant; replication and coordination with ZooKeeper.

Version control and repositories skills are a progression from basic commits to branching, merging, and rebasing. There are many online tutorials for git. More advanced topics include analysis of file revision histories, bug repositories, and crash repositories.

Scripting and automation skills are required to handle Infrastructure as Code concepts. This includes scripting in several languages (Python, Ruby, bash), for which there are online tutorials; applying software engineering discipline to IaC scripts for maintainability; and developing a workflow philosophy to handle infrastructure changes and regeneration after crashes.

Monitoring and log file processing skills are somewhat of an art, but there are skills that can be easily taught. These include locating and manual scanning of log files for relevant error messages; processing logs into graphs using Splunk; and performance monitoring using JMeter. More advanced monitoring requires knowledge of statistics and data science, so may only be possible for graduate students with those skills.

Continuous integration can be introduced by building an open source project from scratch and then doing incremental builds as changes are made. Larger projects can use a tool like Travis CI.

Communication and culture can be facilitated by requiring HipChat or Slack for team projects. While academic projects can only simulate industry process, skills such as code reviews and peer evaluation should be introduced.

Automated testing beyond unit and integration testing include smoke testing, performance testing, and security testing. Security was recognized as a complex topic, so some simplification is required here – for example, simple intrusion testing and database log tracing.

Designing for deployment should cover several deployment models, including blue-green deployment, AB testing, and canary testing. Design should cover feature toggles, forward compatibility, and backward compatibility. Container/VM orchestration can be handled by Ansible or Netflix Spinnaker.

Dependency management issues include version dependencies, microservice dependencies, and operating system interdependencies. These should include both developer and sysAdmin perspectives.

Workshop participants were: Len Bass (organizer), Bram Adams (organizer), Ian Gorton (organizer), Jim Knapik, Theo Skoteiniotis, Hanieh Alipour, Bonita Sharif, Andrew Neitsch, Shane McIntosh, and Marty Barrett. Material related to the workshop can be found at https://github.com/devopseducator/2016workshop. Tentative plans are to hold another workshop in conjunction with CSEET in November 2017.