DevOps Drives Efficiency Across the Maturity Spectrum

STEVE CURTIS

1 Executive Summary

All enterprises, knowingly or not, are at some level of DevOps maturity. A highly mature organization using DevOps processes is able to deploy applications more quickly, in one or more locations, including those with different infrastructures, while maintaining high output across activities and supporting repeatable tasks. Less mature organizations spend time executing a highly manual, error-prone process, resulting in costly rework or Service Level Agreement failure. Wherever an organization resides along this spectrum, Devis and DevOps are poised to improve their performance by increasing their maturity.

The more mature an organization is, the more likely its DevOps process incorporates automation. The benefits of automated DevOps include greater time efficiency and accuracy, which translate into increased cost savings. Contrast that with an immature organization following inconsistent or non-existent and unrepeatable artisanal or stick-built processes, producing inconsistent and non-maintainable applications. Automated DevOps has the added benefit of a completely auditable path from inception to production.

DevOps is as much about culture and practice as it is about tools. By applying the practices and technologies that comprise DevOps incrementally and focusing on the goals, an organization incrementally reduces risk and time to deliver. They deliver solutions at higher velocity and higher quality with repeatable processes. ***By embracing this culture and adopting these practices and technologies, less mature organizations can realize significant benefits, including cost savings, improved quality, and shorter delivery cycles.***

Devis helps organizations achieve these benefits via easy-to-manage, incremental steps along the full spectrum, bringing them to higher levels of maturity at their own pace. We offer a dual-pronged approach, in which we help an organization create a corporate DevOps model, but also provide the development horsepower to implement and execute the model, thus helping them along the spectrum while doing excellent work.

2 What is DevOps?

First and foremost, DevOps is a holistic approach to applications development that integrates development and operations, thus the name. Its goal is to increase an organization's velocity in evolving and enhancing products for its user population, while improving quality, reducing cost, and maximizing repeatability.

DevOps is a combination of software development and information technology (IT) operations, comprised of multiple disciplines, including:

- planning,
- Agile development,
- automated testing,
- continuous integration (CI),
- configuration management,
- continuous deployment (CD), and
- automated monitoring

Each one of these disciplines can be executed individually but combining them and the staff supporting them is the goal of a DevOps process. The disciplines are important categorizations, as each one has one or more tools to support its specific tasks and goals. The combination of these tools is called a *toolchain*. The key to integrating the individual disciplines and the toolchain that supports them is a workflow tool that automates management of discipline activities while providing reporting to the proper stakeholders in a timely and meaningful way.

Figure 1 illustrates the elements of a DevOps environment and their interactions. Note that personnel are focused in specific, creativity-focused areas and the rest of the process flow falls to intelligent, purpose-built tools.

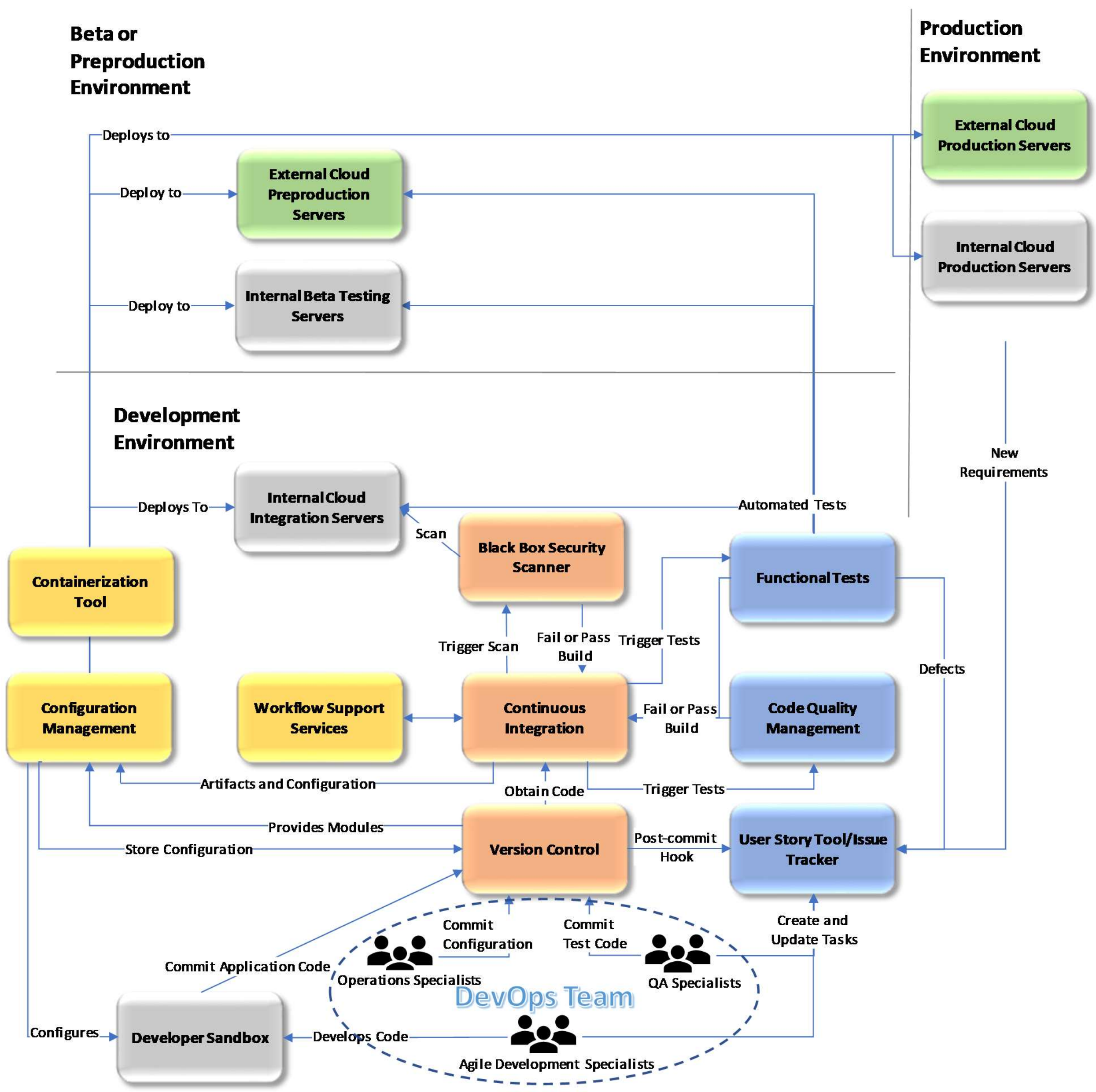

Figure 1 A Sample DevOps Tool Chain. Integrating the tools for each discipline maximizes the opportunity for automation and cost efficiency.

3 Integrating DevOps into an Organization

Once an organization makes the decision to explore implementing a DevOps process and gets management buy-in, the organization must analyze its structure, the software development and deployment life cycle, look for opportunities for teaming and collaboration, improvements and efficiencies in process, and identify opportunities and tools for automation. Some initial analysis that must be undertaken, including:

- **Organizational structure and culture**—Are teams siloed, is there any concept of joint ownership, are there competing priorities, are there teaming opportunities, is there open communication, and should there be reorganization?
- **What work is performed and why**—What is the full lifecycle of any application or component? Is this work a response to industry or regulatory requirements for the activity, or is it "just how we have always done it."
- **How is the work completed**—What processes and tools are used to accomplish each task?

DevOps requires members of an organization to hold a shared understanding and ownership of the complete lifecycle of delivering desired features to users and then applying consistent processes and automation to the problem. Coming to that common understanding involves bringing together staff from across the lifecycle: requirements stakeholders; designers; developers; testers; documentation; quality assurance; security and operations, production, and user support staff.

Knowing how and why work is completed is important because the organization will need to adapt or change these processes as they implement a DevOps methodology. For example, there will be tools put in place to do work that previously was performed by hand. One effect of this shift is that staff tasks also will morph. Mundane tasks, such as collection of metrics, will be performed by tools, allowing staff to give more attention to the important, creative work, such as design, testing, and the analysis of those metrics.

With this foundation laid down, the organization can move as quickly or deliberately as it finds comfortable, implementing processes and tools across the full lifecycle or focusing their efforts on individual disciplines, one or two at a time. In either approach, the organization will realize discernable, quantifiable improvements in their performance and results.

4 A Closer Look

The goal is to deliver high quality features to users in a timely, consistent, and supportable way. Increasingly mature DevOps environments seek to accomplish this goal with as little human intervention as possible to maximize consistency and quality while reducing cost and time to deliver.

4.1 Organizational Structure and Culture

In siloed organizations, system ownership passes completely from one team to the next. Each team "throws the code over the transom," as the system advances through the lifecycle. In this model, issues result in finger pointing and little system or process improvement is achieved beyond fixing the immediate problem. DevOps strives to combine development and operations staff into a single team or even a single organization; one that takes full ownership of systems throughout the lifecycle.

A team approach that integrates Operations staff early in the process enables a better understanding of the features that will require support as a system is deployed. Development staff benefit from knowing Operations' capabilities and limitations. Pairing developers and operators builds trust and understanding across the disciplines. As engineers work through the entire application lifecycle, they expand their range of skills, enabling them to support the system from development to test to operations.

The most mature teams also integrate security experts to achieve DevSecOps. Identifying and addressing security risks early in the process greatly improve system quality and reliability.

Creating a culture of joint discipline teams is a simple, inexpensive step less mature DevOps organizations can implement that pays big dividends in system quality, timeliness, and cost.

4.2 What Work is Performed and Why?

Typically, organizations lack a compressive view of the activities involved in supporting a system throughout the delivery pipeline. Most have SOPs or "tribal" knowledge of the steps and procedures needed to deploy a system in their environments. But many lack a full view of the chain of events and activities need to plan, code, build, test, release, deploy, operate, and monitor systems.

By exploring, documenting, and diagraming their workflows and understanding the full lifecycle of any application or component, an organization can begin to identify bottlenecks and redundant or non-value-added processes. As a result of this examination, individual processes can be confirmed as best practice, verified as necessary to meet regulatory requirements, streamlined, or removed. Organizations can improve the interplay between processes and optimize their workflows. Finally, they can identify performance metrics by manually mapping and tracking the delivery pipeline. These metrics will enable the organization to measure, in a uniform, repeatable way, the effects of changes they implement and know for certain they are improving the system. All these activities can provide value to organizations across the DevOps maturity spectrum and are a prerequisite for less mature organizations before implementing automation and creating toolchains.

4.3 How is the Work Completed?

Once an organization understands what processes are performed, it can examine how by asking detailed questions of themselves, such as:

- What tools are used to accomplish each task?

- Can these tools or processes be automated?
- Should they be replaced?
- Can they be standardized?

Through automation and the use of standardized tools, teams can increase velocity and issue new releases more frequently with fewer problems. Automated test and review cycles keep releases from being blocked as incident response times improve. Organizations that are less mature on the DevOps spectrum benefit from incremental automation and adoption of integration toolchains.

5 Continuous Delivery Example

By converting defined processes into actionable software and validating the resultant code on a continuous basis, as depicted in **Figure 2**, the achievable goal is to always have code or features in a production-ready state.

A critical success driver in this process is user feedback that is integrated into the loop as new requirements are identified. In this way, these key stakeholders' voices are heard and accounted for in each delivery cycle, improving communication and collaboration, and ultimately, the end results.

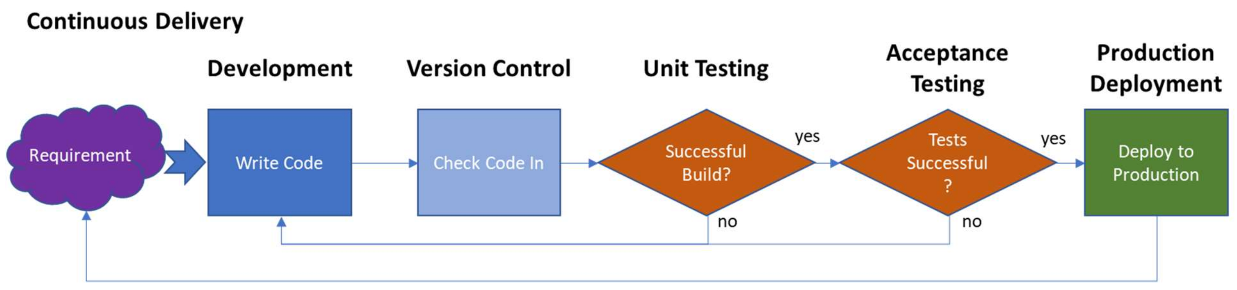

Figure 2 The Continuous Delivery Process Loop. Short, quick cycles bring incremental feature enhancements to production smoothly

As mentioned earlier, another strength of DevOps is its well-defined and flexible workflow. All enterprises have different structures and environments in which they deploy code or features. One enterprise may operate completely physically on premises ("on-prem"), another may access multiple clouds on demand, and a third may integrate on-prem and cloud resources. While the architecture may be different, the commonality is that each environment requires certain tasks, tests, and approvals along the journey from a feature request to a functioning feature in the production system. One key step to orchestrating that matriculation is the use of CD/CI pipelines or delivery pipelines.

The delivery pipeline is the part of the DevOps toolset that integrates code, initiates builds, executes tests, and deploys or packages code for deployment. A mature automated delivery pipeline identifies tasks in a traceable, reportable, and auditable manner. It also provides feedback to the organization as to how long each step takes and how long the aggregate of all steps takes to complete. The delivery pipeline has the capability to run tasks in serial, parallel or mixed fashion and the completion of one task can initiate or stop another. The delivery pipeline tools are normally API-driven and can accommodate new tests, scans, or approval gates as required.

Figure 3 depicts a simple delivery pipeline that accomplishes several tasks in serial and parallel fashion. This particular workflow is:

- Checking out code from a versioning repository,
- Running tests and code analysis,
- Build the deployment package,
- Deploying it to a User Acceptance Test environment for functional testing

There are details within each action and logs that describe facts about the task such as build number, elapsed time and status. The tools that were used within this workflow are:

- Version repository,
- Configuration Management tool, and
- Continuous Integration tool.

There are a number of well-constructed commercial and open source tools that can fulfill the functions and each organization should use the tools that are most appropriate for them. Devis takes a "vendor-agnostic" approach to tool selection, preferring to focus on each client's specific environment, system requirements and configuration, and DevOps maturity to recommend individual tools or suites of tools to address that organization's needs most effectively.

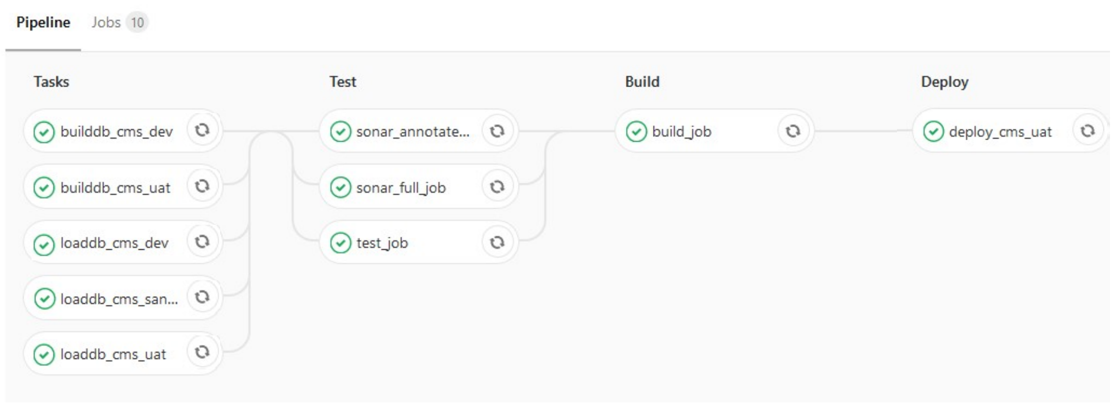

Figure 3 Sample delivery pipeline. Providing automation and a simple-to-follow audit trail for each step in a process.

Documenting the delivery pipeline in real time is an important element of a DevOps methodology. Accordingly, most automation tools created to support DevOps include the capability to produce artifacts that document and support the process as a byproduct of the work. For example, log files that are created throughout the process are sent to the versioning repository and stored as artifacts of any build. These artifacts are valuable to any auditing entity to support an assertion of the status of an environment at any point in time. An organization also can use these artifacts to quickly recreate an environment if necessary.

6 Conclusion

There are many benefits to a properly implemented DevOps environment and it does not require an all-or-nothing decision to commit. Each organization may have different needs but the basic framework is flexible enough to support them. It is important to keep in mind that while each discipline can be used in an independent manner, the end goal is to increase the capability, flexibility, accuracy, precision and efficiency with which features, enhancements, or entire applications go from inception to production. Implementing a DevOps methodology will manifest itself as cost savings, higher quality, and greater user satisfaction. Devis is the partner that will smooth the implementation process and support the organization as developers and mentors along the entire maturity spectrum.

About Steve Curtis

Steve Curtis is Devis's Vice President of Development Services. His experience in the management, delivery and optimization of delivery teams is an invaluable resource to Devis and our clients.

Mr. Curtis' experience, coupled with his PMP, ACP, CSM, and CSPO certifications make him a well rounded expert in management and delivery in Agile organizations.

About Devis

Devis is a minority, woman-owned small business (WOSB) with more than 25 years as a leading provider of IT solutions to the Federal Government and international development community. We have built our practice focused on solving the information sharing problems faced by public and private organizations with dispersed stakeholders. Our core areas of expertise include:

- Agile Application Development
- Tier 2 & 3 Help Desk Support
- Secure, Cloud-Based Managed Services
- Worldwide IT Deployment
- Systems Integration
- Knowledge Management
- IT Consulting
- Section 508 Accessibility
- Software and Business Process Training

Our accomplished staff has an average of more than 15 years of experience in the IT field, and more than six at Devis. They have travelled on hundreds of TDYs to over 70 counties. Our proven success is a direct result of our certified staff and our ability to develop, deploy, and maintain systems while understanding and supporting our clients' goals.

For More Information

Chris Kagy

VP Business Development

ckagy@devis.com

703-525-6485 ext 126