DevOps

Imagine a world where product owners, Development, QA, IT Operations, and Infosec work together, not only to help each other, but also to ensure that the overall organization succeeds. By working toward a common goal, they enable the fast flow of planned work into production, while achieving world-class stability, reliability, availability, and security.
—The DevOps Handbook [1]

DevOps is a mindset, a culture, and a set of technical practices. It provides communication, integration, automation, and close cooperation among all the people needed to plan, develop, test, deploy, release, and maintain a Solution.

SAFe enterprises implement DevOps to break down silos and empower each Agile Release Train (ART) and Solution Train to continuously deliver new features to their end users. Over time, the separation between development and operations is significantly reduced and trains operate with an automated, continuous delivery pipeline. This mechanism seamlessly defines, implements and delivers solution elements to the end user, without handoffs or excessive external production or operations support.

The goal is simple: Deliver value more frequently. This is indeed achievable, as “high-performing IT organizations deploy 30x more frequently with 200x shorter lead times. … 60x fewer failures and recover 168x faster.” [1]

Details

DevOps is a combination of two words, ‘development’ and ‘operations.’ Without a DevOps approach, there’s often significant tension between those who create new features and those maintaining the stability of the production environment. The ‘development team’ is measured on the business value they deliver to end-users, while ‘IT service management’ is measured on the health and stability of the production environment. When each group has seemingly opposing business objectives, delivery inefficiency and organizational friction may rule the day.

But DevOps ends the silo approach, providing an enterprise with the ability to develop and release small batches of functionality to the business or customer in a flow process called the Continuous Delivery Pipeline. DevOps is integral to every Value Stream, and, by definition, is integral to SAFe.

Many SAFe concepts and principles—systems thinking, small batch sizes, short iterations, fast feedback, and more—directly support DevOps principles. In addition, the SAFe practices of Continuous Exploration, Continuous Integration, Continuous Deployment, and Release on Demand directly support this business need.

The Goal of DevOps

From planning through delivery, the goal of DevOps is to improve collaboration between Development and IT Operations by developing and automating a continuous delivery pipeline. In doing so, DevOps:

- Increases the frequency and quality of deployments
- Improves innovation and risk taking by making it safer to experiment
- Realizes faster time to market
- Improves solution quality and shortens the lead time for fixes
- Reduces the severity and frequency of release failures
- Improves the Mean Time to Recovery (MTTR)

SAFe's 'CALMeR' approach to DevOps covers the five main aspects, as illustrated in Figure 1.

Figure 1. SAFe's CALMeR approach to DevOps

Each aspect is described in the sections below.

Culture of Shared Responsibility

In SAFe, DevOps leverages the culture created by adopting the Lean-Agile values, principles and practices of the entire framework. Just about every principle of SAFe, from "#1 Take an Economic View" to "#9 Decentralize Decision Making," applies to DevOps. It enables shifting some operating responsibilities upstream, while following development work downstream into deployment, and operating and monitoring the solution in production. Such a culture includes:

Collaboration and organization – DevOps relies on the ability of Agile Teams and IT Operations teams to collaborate effectively in an ongoing manner, ensuring that solutions are developed and delivered faster and more reliably. This is implemented, in part, by including operations personnel and capabilities on every ART.

Risk tolerance – DevOps requires a tolerance for failure and rapid recovery, and rewards risk taking.

Self-service infrastructures – Infrastructure empowers development and operations to act independently without blocking each other.

Knowledge sharing – Sharing discoveries, practices, tools, and learning across silos is encouraged.

"Automate everything" mindset – DevOps relies heavily on automation to provide speed, consistency, and repeatable processes and environment creation, as we describe below.

Automate Everything

DevOps simply recognizes that manual processes are the enemy of fast value delivery, high productivity and safety. But automation is not just about saving time. It also enables the creation of repeatable environments and processes, which are self-documenting and, therefore, easier to understand, improve, secure, and audit. The entire continuous delivery pipeline is automated to achieve a fast, Lean flow.

Automation facilitates faster learning and response to market demand and customer feedback. Builds, testing, deployments, and packaging that are automated improve the reliability of processes that can be made routine.

This is accomplished, in part, by building and applying an integrated and automated 'tool chain,' shown in Figure 2, which typically contains the following categories of tools:

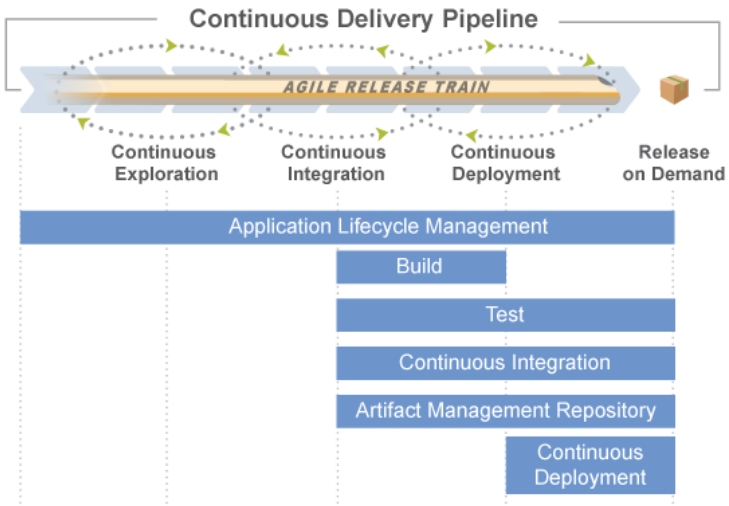

Figure 2. DevOps tool chain within the CD Pipeline

Application Lifecycle Management – Application and Agile Lifecycle Management tools (ALM) create a standardized environment for communication and collaboration between software development teams and related groups. (CA Agile Central, Version One, Agile Craft, tools for Model-Based Systems Engineering)

Artifact Management Repository – These tools provide a software repository for storing and versioning binary files and their associated metadata (Artifactory, Archiva and JFrog).

Build – Build automation is used to script or automate the process of compiling computer source code into binary code (ANT, Maven, Bamboo, Jenkins).

Testing – Automated testing tools include unit and acceptance testing, performance testing, load testing, and many more (JUnit, NUnit, Maven, Cucumber, FitNesse).

Continuous Integration (CI) – CI tools automate the process of compiling code into a build after developers have checked their code into a central repository. After the CI server builds the system, it runs unit and integration tests, reports results, and typically releases a labeled version of deployable artifacts (Cruisecontrol, Jenkins, Continuum).

Continuous Deployment – Deployment tools automate application deployments through to the various environments. They facilitate rapid feedback and Continuous Delivery while providing the required audit trails, versioning, and approval tracking (Capistrano, UrbanCode, Ansible, Puppet).

Additional tools – There are numerous other important DevOps support tools: configuration, logging, management and monitoring, provisioning, source code control, security, code review, and collaboration.

Lean Flow

SAFe teams strive to achieve a state of continuous flow, enabling new features to move quickly from concept to cash. The three primary keys to implementing flow make up Principle #6 Visualize and limit WIP, reduce batch sizes, and manage queue lengths. All three are integral to systems thinking (Principle #2), and long-term optimization. Each is described below in the DevOps context.

1. Visualize and limit Work in Process (WIP). Figure 3 illustrates an example of a Program Kanban board, which makes WIP visible to all stakeholders. This helps teams identify bottlenecks and balance the amount of WIP against the available development and operations capacity, as work is completed when the new feature or functionality is running successfully in production.

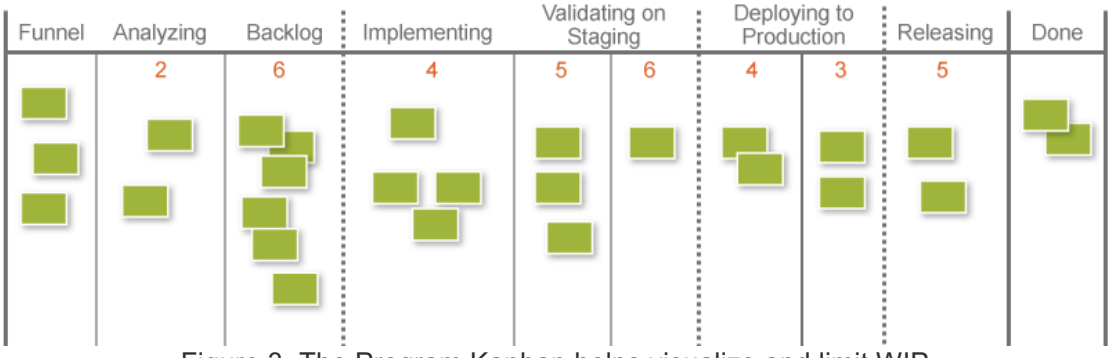

Figure 3. The Program Kanban helps visualize and limit WIP

2. Reduce the batch sizes of work items. The second way to improve flow is to decrease the batch sizes of the work. Small batches go through the system faster, and with less variability, which fosters faster learning and deployment. This typically involves focusing more attention on, and increasing investment in, infrastructure and automation. This also reduces the transaction cost of each batch.

3. Manage queue lengths. The third way to achieve faster flow is by managing, and generally reducing, queue lengths. For solution development, this means that the longer the queue of work awaiting implementation or deployment, the longer the wait time, no matter how efficiently the team is processing the work. The shorter the queue, the faster the deployment.

Measure the Flow of Value

In a DevOps environment, problem resolution is less complex because changes are made more frequently, and in smaller batches. Telemetry, or automated collection of real-time data regarding the performance of solutions, helps to quickly assess the impact of frequent application changes. Resolution happens faster because teams don't need to wait for a different group to troubleshoot and fix the problem.

It's important to implement application telemetry to automatically collect data on the business and technical performance of the solution. Indeed, basing decisions on data, where "the facts are always friendly" rather than intuition, leads to an objective, blameless path toward improvement. Data should be transparent. It should be accessible to everyone, be meaningful, and easily visualized to spot problems and trends.

The goal is to build applications that:

- Collect data on business, application, infrastructure and client layers.
- Store logs in ways that enable analysis.
- Use different telemetry for different stakeholders.
- Broadcast measurements and be hyper transparent.
- Overlay measurements with events (deploys, releases).
- Continuously improve telemetry during and after problem solving.

It's also important to measure the flow of value through the continuous delivery pipeline.

Please see the metrics article for specific recommendations on DevOps measures.

Recover – Enable Low-Risk Releases

To support the continuous delivery pipeline and the concept of Release on Demand, the system must be designed for low-risk component or service-based deploy-ability, release-ability, and fast recovery from operational failure.

Techniques to achieve a more flexible release process are described in the Release on Demand article. In addition, the following techniques support fast recovery:

Stop-the-line mentality – With a stop-the-production mentality, everyone swarms to fix any problem until it's resolved. When there's a problem with the continuous delivery pipeline, or a deployed system, the same thinking must apply. Findings are integrated immediately into the process or product as they're discovered.

Plan for and rehearse failures – When it comes to large-scale IT applications, failure is not only an option, it's guaranteed at some point. A proactive approach to experiencing failures will increase the team's response practices, and also foster built-in resilience into the systems. (See the 'Chaos Monkey' in [2]).

Build the environment and capability to fix forward or roll back – Since mistakes will be made, and servers will fail, teams need to develop the capability to quickly 'fix forward' and, where necessary, roll back to a prior known good state. In the latter case, planning and investment must be made to revert any data changes back to the prior state, and not lose any user transactions that occurred during the process.

To achieve these recovery capabilities, the organization will typically need to undertake certain enterprise-level initiatives to enhance architecture, infrastructure, and other nonfunctional considerations to support deployment readiness, release, and production.

Learn More

[1] Gene Kim. Jez Humble, Patrick Debois, John Willis. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press.

[2] 2015 State of DevOps Report https://puppet.com/resources/whitepaper/2015-state-devops-report?link=blog