WHITE PAPER – **APRIL 2017**

DEVOPS AND AGILE DEVELOPMENT

A VMware Field Perspective

vmware®

ABOUT THE AUTHORS

VMware's DevOps experts collaborated to build an overview on DevOps, reflecting a perspecive built from their experience enabling agile software development and infrastructure delivery for customers across the globe.

KEVIN LEES

Kevin Lees is the field Chief Technologist for IT Operations Transformation at VMware.

JOHN GARDNER

John Gardner is a practice manager with the DevOps and Cloud Native Applications group, part of VMware Professional Services.

PEG EATON

Peg Eaton is a practice director with the DevOps and Cloud Native Applications group, part of VMware Professional Services

Table of Contents

INTRODUCTION

DevOps is both a hot topic and an overused term today. There are almost as many definitions of DevOps as there are people writing about it. Is it a technology? An organizational construct? A combination of the two? Is it the way Netflix and other internet-based companies develop software? It depends on who you read. The purpose of this paper is to provide guidance in the form of a set of concepts, a taxonomy, and an overview of VMware's approach to DevOps.

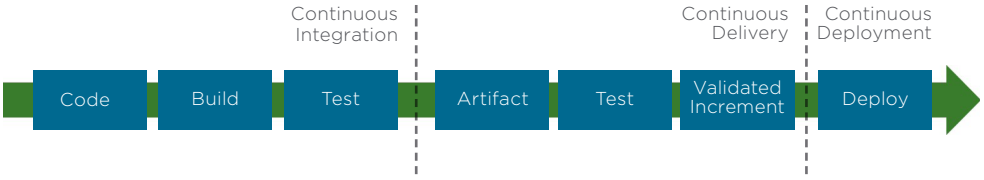
1. DevOps Key Concepts

DevOps represents a new model for application development that requires deep collaboration between software developers and IT operations. This deep collaboration is indicative of a culture focused on defining software with operations in mind and automating the process of software delivery so that building, testing, and releasing software happens rapidly, frequently, and reliably.

1.1 Definitions

Following are a set of foundational definitions for commonly used (but variously defined in the industry) terms associated with DevOps:

Term	**Definition**
DevOps	Conceptually, DevOps is that which is necessary to take an idea (feature, code, documentation, whatever) from inception through delivery to a customer in the most expedient and sustainable way possible. DevOps is most often applied in the context of software delivery and refers to a model for application development focused on deep collaboration between software developers and IT operations. That said, the DevOps model can be applied to the development lifecycle of anything with distinct development and operations components – for example, infrastructure or IT services.
Continuous Integration	Continuous Integration is the process of combining source code (most likely from different developers or teams) into a single application, and then typically running some automated suite of tests on the resulting application. This integration process runs "continuously", either polling source control on a regular interval or triggered by code check in.
Continuous Delivery	Continuous Delivery is the process of packaging, testing, and storing an application unit in a continuous fashion to be always ready to deliver into production. This extends the continuous integration process to arrive at an application unit that has enough testing, compliance and validation that it is production-ready.
Continuous Deployment	Some organizations or applications extend Continuous Delivery into Continuous Deployment, where the application is deployed into the production environment automatically. This typically requires robust production validation mechanisms and comprehensive rollback capability.



Term	Definition
Delivery Pipeline / DevOps Pipeline	The Delivery Pipeline refers to the stages of the process by which an application is delivered to production. This will include as a minimum build, integration, testing and deployment, but will likely also require code quality scans, security and compliance checks, performance testing, as well as progressing through different environments.
Delivery Pipeline Tool Chain	The delivery pipeline tool chain is a collection of best of breed tools based on common industry patterns and environment's requirements to support application development using C#, Java, Python, Go, or other programming languages. See Section 3: Delivery Pipeline for a description of the delivery pipeline tool chain.
Cloud Native Applications	Cloud Native Applications refers to applications architected and developed according to principles for optimally running and managing distributed applications on a cloud platform. A good example of these principles are those documented as the Twelve-Factor App. The DevOps model is a key enabler of the Twelve-Factor App governing principle, which shares ideal practices for app development, paying particular attention to the dynamics of the organic growth of an app over time, the dynamics of collaboration between developers working on the app's codebase, and avoiding the cost of software erosion. The DevOps model can be seen as the required approach for developing Cloud Native Applications.
PaaS	Platform as a service (PaaS) is a category of cloud computing services that provides a platform allowing organizations to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an application, for example Pivotal Cloud Foundry.

1.2 Guiding Principles

Following is a set of DevOps guiding principles from a VMware perspective:

DevOps Guiding Principle	Description
Dev & Ops as one team	Traditionally Dev and Ops has been synonymous with "us and them." Developers developed software and handed it off, hopefully through some defined release process, to Infrastructure & Operations (I&O) for production deployment and on-going operations. This approach was ripe with miscommunication and misaligned expectations. More often than not, the first time I&O became aware of a new application going into production was during release review – if there was one. DevOps requires developers and I&O to collaborate extensively and continuously, essentially becoming one team. While development and I&O must collaborate extensively, collaboration extends to actively including other teams, such as security for example, at different points during the delivery pipeline. This also represents a significant cultural shift for traditional Dev and Ops, a shift to a no-blame culture based on trust and deep collaboration. To carry this through the execution of the delivery pipeline, employ an Agile methodology including both development and I&O activities in a single sprint plan.
DevOps team is accountable and responsible	The DevOps team retains sole responsibility and accountability for the application it delivers and virtual environment (i.e., virtual machines and/or containers) in which it runs.

DevOps Guiding Principle	Description
Shift to the left	Instead of only focusing I&O activities on deploying to and operating software in production, I&O activities move into the delivery pipeline itself. Integrate capabilities into delivery pipeline such as early security and authentication testing as well as not only opening up monitoring to the developers but I&O actively monitoring application environment impacts as it moves through the delivery pipeline and provide feedback to the developers.
Automate the delivery pipeline	Automate all aspects of the delivery pipeline process to the greatest extent possible so that building, testing, and releasing software can happen rapidly, frequently, and reliably. Automate everything that can be automated and maintain manual steps only where regulatory or company policies require it.
Immutable infrastructure	Infrastructure should be created once, from original source code (see Infrastructure as code) and not manipulated after the fact, particularly not with error prone manual processes. A closely linked concept, any combined processes addressing the same infrastructure (such as configuration management) should be idempotent, so that repeated application does not result in errors or expected changes.
Infrastructure as code	Infrastructure should be treated as code, keeping infrastructure definitions with application code in version control while providing the capability to deploy consistent, production-like environments throughout the delivery pipeline stages. This ensures consistency and provides an audit control point.
Smaller more frequent releases	Each release introduces a small number of changes so production releases can be made more frequently and with less risk. This also provides the opportunity to receive end-user feedback and make adjustments more quickly.

1.3 Infrastructure and Operations Implications

The DevOps guiding principles outlined above certainly impact I&O. The table below describes the key I&O impacts.

Note: *While the I&O implications in the following table are presented in the context of supporting application development, they equally apply in the context of IT development such as, for example, when developing IT services, scripts, blueprints, automation and orchestration workflows, and integration code. IT can realize the same benefits when applying DevOps to internal IT development as realized when applied to developing applications.*

DevOps Guiding Principle	Infrastructure & Operations Implication
Dev & Ops as one team	• Understand the delivery pipeline and customize it to meet the needs of the application and its developers. • Actively and continuously collaborate with development from planning through testing and release into production in addition to on-going operations. • Collaboratively develop integrated sprint plans. • Understand the application architecture and its implications for the virtualized environment in which it runs.
DevOps team is accountable and responsible	• Understand security, audit and compliance requirements for the delivery pipeline as well as the virtual environment in which the application will run in production. • Understand the impact of deploying the application and its virtual environment into production from a change management perspective.

DevOps Guiding Principle	Infrastructure & Operations Implication
Shift to the left	• Automate security, compliance, and operational readiness tests for early inclusion in the delivery pipeline. • Actively work with developers to address operational considerations directly in application code. • Provide open access by developers to monitoring and logging throughout the execution of the delivery pipeline and as allowed by compliance policies in production. • Work with developers to provide customized application-level monitoring throughout the execution of the delivery pipeline and in production. • Actively monitor application virtualized environment throughout delivery pipeline and provide feedback to developers.
Automate the delivery pipeline	• Participate in delivery pipeline automation definition, design, and implementation. • Build compliance controls and audit capabilities into delivery pipeline automation. • Provide self-service, on-demand and/or programmatic access to software-defined infrastructure, blueprints, and policies. • Use same tools and code for operations activities in production as used in the delivery pipeline.
Immutable infrastructure	• Always modify the versioned definition of the virtual environment associated with the application and use the delivery pipeline to re-deploy into production; never update, patch, or modify in-place in production.
Infrastructure as code	• Ensure ability to declare infrastructure as code and put it under version control. • Always define the virtualized environment as code and maintain the virtualized environment definition under version control.
Smaller more frequent releases	• Work with IT change management and compliance auditors to focus governance on the version controlled definitions (application code, infrastructure as code, blueprints, policies, etc.) of what is to be deployed as well as the automated delivery pipeline itself rather than each application release unit.

1.4 Developer Implications

Just as for I&O, the DevOps guiding principles will impact developers as well. The table below describes the key developer impacts.

DevOps Guiding Principle	Developer Implication
Dev & Ops as one team	• Actively and continuously collaborate with I&O team members from planning through testing and release into production as well as for on-going monitoring of the application in production. • Collaboratively develop integrated sprint plans. • Ensure I&O team members understand the application architecture.
DevOps team is accountable and responsible	• Accept responsibility and accountability for the application in production as well as throughout the execution of the delivery pipeline. • Be prepared to accept accountability and responsibility for releasing the application into production. • Be prepared to actively monitor and be on call for the application in production.

DevOps Guiding Principle	Developer Implication
Shift to the left	• Actively work with I&O team members to address operational considerations directly in application code. • Work with I&O team members to develop customized application-level monitoring throughout the execution of the delivery pipeline and in production. • Actively monitor application throughout the execution of delivery pipeline and work with I&O team members to tune the virtual environment in which it runs.
Automate the delivery pipeline	• Work with I&O team members to define how delivery pipeline automation should be customized for the application. • Decide if they are ready for programmatic configuration and provisioning of the virtual environment for their application.
Immutable infrastructure	• No direct implication for the developer, but the same best practice should be used: always modify the versioned definition of the application and use the delivery pipeline to re-deploy into production; never update, patch, or modify the application in-place in production.
Infrastructure as code	• Always keep application code with its virtualized environment definition under version control.
Smaller more frequent releases	• Develop applications based on iterations focused on a series of small application changes testable and released as a unit, such as adding a single or, at most, a very limited set of new features.

2. Delivery Pipeline

While DevOps refers to a model for application development focused on deep collaboration between software developers and IT operations, it is applied to the process of software delivery. As such, it is applied to the execution of the delivery pipeline so that building, testing, integrating, and releasing software can happen rapidly, frequently, and reliably. To achieve this, processes such as continuous integration and continuous delivery, or continuous deployment, are used. These processes are enabled by interrelated software tool stacks that support the delivery pipeline stages and activities.

The delivery pipeline tool chain is a collection of best of breed tools based on common industry patterns and environment's requirements to support apps development using C#, Java, Python, Go, or other programming languages. Figure 1 on the following page depicts the delivery pipeline tool chain.

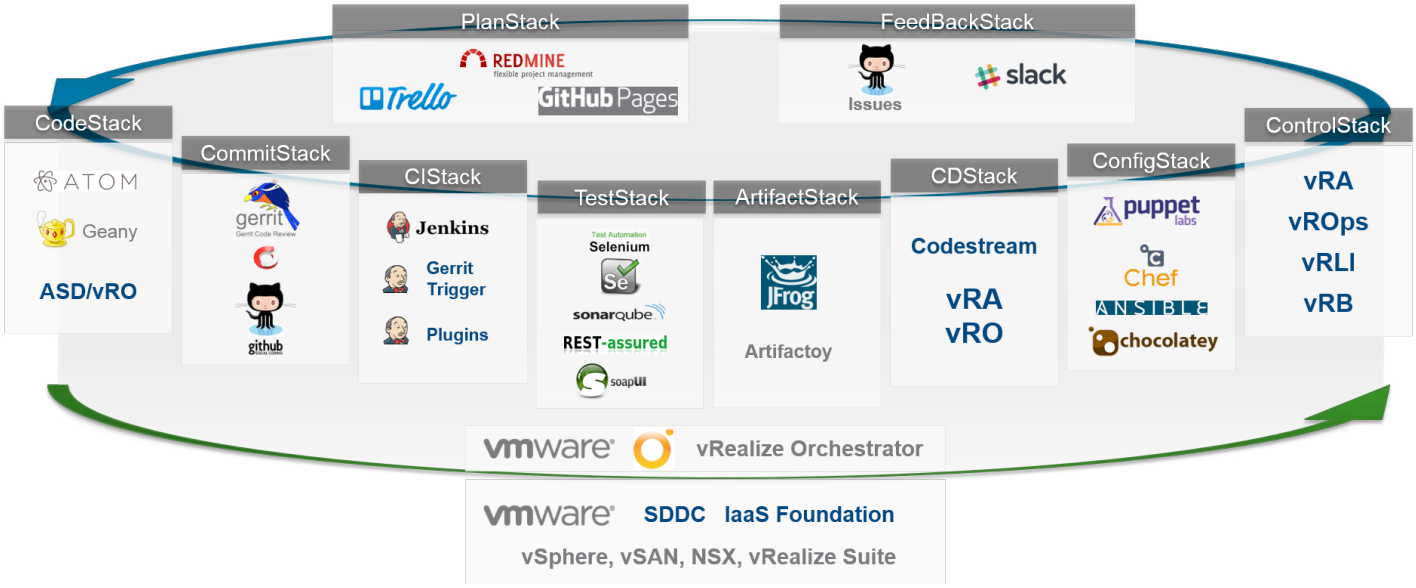


Figure 1: Delivery Pipeline Tool Chain

This section describes the software stacks that comprise the delivery pipeline tool chain by stage and associated best practices.

Planning

The Planning Stack supports agile development throughout the delivery pipeline — planning and tracking software releases; creation of user stories; sprint planning, backlog management, and issue tracking. Typical tools used in planning are: JIRA™, Redmine, Trello, GitHub Issues, and Microsoft Team Foundation Server (MS TFS).

Coding & Unit Testing

Next in the delivery pipeline tool chain is the Coding Stack. The software in this stack is used for the coding effort against the user stories. It includes integrated development environments, editors, debugging tools, and unit-test tools. Commonly used tools include: SpringSource Tool Suite, MS TFS, Geany, Atom, Eclipse, and vRealize Orchestrator. Two other items in this stack are used to support the application development: pre-configured developer-workstations and the environments for application unit-tests (essentially a set of VM blueprints) for the developer.

Commit

The Commit Stack supports version control and many best practices including: daily check-ins, committing assets early and often, automated style-checking, and code reviews. Typical tools include: Git, GitHub, MS TFS, and Gerrit.

Continuous Integration

In the next stack, Continuous Integration, the tools support software builds and automated smoke tests which provide developers with fast-useful feedback. Continuous integration tools include: like Jenkins, Gerrit Triggers, and vRealize Automation.

Testing

The Testing Stack is used to manage testing throughout the software development lifecycle. Developer-driven testing in production-like environments for each phase is the key to high productivity and quality code as it is more effective to expose and remediate issues earlier in the lifecycle. This stack includes tools such as: Selenium, REST-assured, soapUI, SonarQube, and vRealize Automation.

Artifact Management

The Artifact Management Stack supports the management of application artifacts including binaries; and provides package version control and dependency management of the artifacts. Tools for artifact management include: JFrog Artifactory, and vRealize CodeStream.

Continuous Delivery

The next stack focuses on Continuous Delivery and provides support for consistent deployments to every environment – for example, UAT, Staging, and Production. Continuous delivery tools include: Jenkins, Ansible, vRealize Automation, vRealize Orchestrator, and vRealize CodeStream.

Configuration Management

Next in the tool chain is the Configuration Management stack, which supports application and environment configuration and should be tightly integrated with the deployment stack. Typical tools in this stack are: Ansible, Chef, Puppet, and Chocolatey.

Control Management

The Control Stack comes next, and is used for application and infrastructure behavior monitoring — for example, alerting, dashboards, logging, and capacity management throughout the release process and into production. Components of the vRealize Suite can be used at this stage, including vRealize Operations and vRealize Log InsightTM, as well as tools such as Nagios.

Feedback

Last but not least is the Feedback Stack, which provides automated feedback to the right people at the right-time during all phases — providing for example, alerts, auditing, test results, build results, deployment status — touching all areas of the pipeline. Github Issues and Slack can be effective tools in this stage.

Best of Breed Approach

VMware supports a best of bread approach to tools supporting each stage of the delivery pipeline. Tool decisions are based on criteria such as the language being used for development, best fit for the organization's environment, and the type of applications being developed.

3. DevOps Team Models

As previously stated, DevOps represents a new model for application development that requires deep collaboration between software developers and IT operations. This deep collaboration is indicative of a culture focused on defining software with operations in mind and automating the process of software delivery so that building, testing, and releasing software happens rapidly, frequently, and reliably. Achieving this requires a set of software development and IT operations skills to form a team with this singular focus. The team model used depends on the degree to which the IT organization commits itself strategically to embracing the DevOps way of working.

3.1 DevOps Roles

The roles along with associated skillsets and responsibilities are shown in Table 5 below. This table represents the on-going roles required on a DevOps team. Depending on the nature of the software or application being developed, there may be additional subject matter experts, such as for security, frequently involved but not necessarily (thought they may be) a full-time member of the DevOps team.

Note: *Some role titles are shown in parenthesis. This is meant to represent the fact that these are roles and do not necessarily correspond to distinct and separate headcount; they could be shared roles so the real focus is on the required skillsets.*

Role & Skillsets	Responsibilities
Customer	• Increment (Scrum Artifact –usable version of product ready for release) • Sprint review
Product Owner	• Cares about business value • Product Backlog • Grooming of Backlog • Sprint planning
Scrum Master	• Time-box • Sprint
Development Team (self-organizing)	• Definition of "Done" • Sprint backlog • Sprint Goal • Daily Scrum • Sprint Retrospective • Application / service development
(Engineer) • Delivery pipeline design & implementation • Delivery pipeline integration design & implementation • Blueprint & policy architecture & design • Application "virtualized environment" design & build • Expertise on consuming virtualized components (network, security, storage, compute, etc.) • I&O SME	• Design / build / integrate / customize delivery pipeline & components • Design & build on-demand infrastructure and platforms/full PaaS • Ensure ability to declare infrastructure as code • Provide guidance to application developers for integrating operational considerations into application code (for example, logging, resilience, security, compliance, etc.)
(Automation Developer) • Delivery pipeline tool-specific development • Delivery pipeline integration development • Deliver pipeline workflow development • Blueprint & policy development & management • Monitoring tool dashboard & automation development (both application- and I&O-related)	• Automate team-specific pipeline actions • Automate "production-specific" actions using same tools and code used for pre-prod deployment • Develop blueprints and automate actions • Automate tests (security and certification; integration, etc.) • Collaborate with Analyst to automate monitoring and remediation • Implement rules for moving between stages

Role & Skillsets	Responsibilities
(Test Engineer) • I&O-related test case design • Test case development in automated test tool – both application- and I&O-related	• Develop security and compliance, integration, resilience, etc., tests • Develop promotion rules for moving between stages
(Analyst) • Monitoring tool SME • Monitoring tool dashboard, analytics, & automation design & development • I&O monitoring tool data analysis	• Develop and maintain application and virtualized container monitoring activities (e.g., dashboards, dynamic thresholds, smart alerts, automated actions). • Actively monitor virtualized containers through pipeline and in production
(Release Engineer) • Release process expertise (if not automated) • Automated release process techniques, design & implementation expertise	• Owns team-specific release deployment process if not automated
Administrator	• Monitor, administer and maintain delivery pipeline components

3.2 Ideal End State Model

The ideal end state team model VMware recommends for DevOps is shown in Figure 2 below. This represents DevOps "nirvana": deep collaboration between software development and I&O teams. In this model software development and I&O applies their unique specialization where needed but also work closely together where needed; all involved have shared goals – which are explicitly reflected in their job descriptions and annual review criteria.

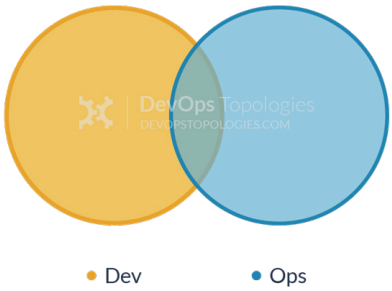


Figure 2: Dev and Ops Collaboration, http://web.devopstopologies.com/ - DevOps Team Topologies, Type 1

In this model there are most likely many software development teams, most of which are dedicated to distinct applications, services, or products. Ideally, each dedicated software development team would have corresponding, dedicated I&O team members. This model also applies to software development teams working on multiple, smaller applications, services, or products. In this case there would be a set of IT operations resources dedicated to DevOps but working across multiple software development teams. Regardless of whether the teams are dedicated or not, each team is fully accountable and responsible for all aspects of automating and maintaining their software delivery pipeline as well as executing all associated processes.

3.3 Pragmatic Concession Model - DevOps as a Service

Those enterprises that, for whatever reason, aren't ready to fully embrace the DevOps ideal end state model of full Dev and Ops collaboration, can begin by implementing "DevOps as a Service" as depicted in Figure 3 below. In this model IT offers the integrated delivery pipeline tools and automation as well operational capabilities needed for DevOps to the enterprise software development teams as a service. I&O skillsets, responsibilities, and shared goals are identical to those present on the dedicated teams described in Section 3.2 above. The difference is they are provided by a set of I&O resources dedicated to delivering DevOps as a Service when and as needed across the development teams.

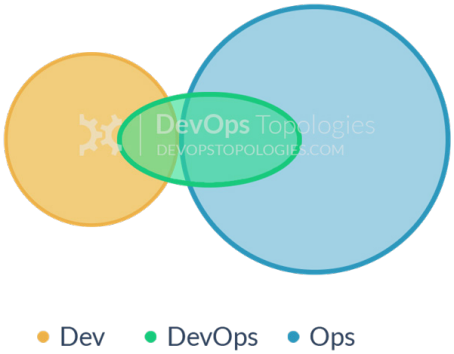


Figure 3: DevOps as a Service, http://web.devopstopologies.com/ - DevOps Team Topologies, Type 4 (modified)

The DevOps as a Service approach also provides an opportunity for centralized DevOps enablement. In this way, DevOps as a Service can provide leadership and delivery team training, support continuous DevOps process improvement and a repository for shared best practices, blueprints, and code. Since I&O resources aren't dedicated full-time to a software delivery team, this model is most efficient if a standardized set of tools are used across all automated delivery pipelines. The DevOps as a Service team is also responsible for actively marketing the benefits of DevOps to the broader IT and development community.

The DevOps as a Service team would include a Service Owner as well as the following roles from Section 3.1:

- Engineer
- Automation Developer
- Test Engineer
- Analyst
- Release Engineer
- Administrator

3.4 Cultural Shift

As with almost every transformation, the biggest challenge is the cultural shift required to move from "how we've always worked" to the new way of working. This is no different when implementing DevOps. Some of the key, cultural attributes required for DevOps are:

- Culture of trust
- Collaborative culture
- Collective ownership
- Never done, always improving
- Empowering

- Constantly learning; promotes skills enablement
- Everybody markets and communicates, socializing DevOps concepts to the larger organization
- Customer value oriented
- Low to no egos
- Everyone's responsible; no blaming

Some people inherently exhibit these attributes, some have the attributes but the attributes have been suppressed due to existing culture, and some individuals just don't have these attributes and aren't willing to change. The people who inherently exhibit these attributes are typically known. Holding DevOps workshops is a good way to identify individuals who may have the attributes but are suppressed as well as those who just aren't willing to change. Hiring fresh talent into the enterprise is always an option, but hopefully it is to fill gaps and expand capabilities rather than being required to fundamentally implement DevOps.

As with any transformation requiring cultural change, communication and enablement are critical. A proactive communication and marketing plan as well as an enablement plan should be developed and executed to raise DevOps awareness and prepare for DevOps adoption.

4. Agile Processes

4.1 Key Process Implications

DevOps implements Agile software development principles while embodying several Lean principles. As with Lean for example, DevOps focuses on value to the customer supported by optimizing the application development and delivery value stream. Agile software development is based on a set of principles in which "requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change." (https://en.wikipedia.org/wiki/Agile_software_development) This results in DevOps focusing on establishing a culture and environment in which reliable applications are released more frequently, utilizing tools and techniques supporting a continuous delivery pipeline as described in Section 3 above.

In addition to impacting the software delivery process itself by applying Agile principles and through the implementation of Continuous Integration and Continuous Delivery/Deployment processes, DevOps directly impacts the following, IT processes:

Request Fulfillment

Modifying the request fulfillment process either through self-service, on-demand automated provisioning or by providing programmatic access is fundamental to supporting rapidly and frequently delivering software which is a goal of the DevOps model. Increasingly, the emphasis is being placed on providing developers a programmatic interface as this approach is much more conducive to the infrastructure-as-code principle.

Change Management

While a certain amount of governance is always required, to support the Agile and Lean principles associated with DevOps the challenge is to apply the appropriate level of governance without slowing down application releases. Governance now focuses on the blueprints defining what is to be deployed as well as the automated delivery pipeline itself rather each application release unit. Additionally, instead of using the traditional IT change management process based on cross-functional Change Advisory Boards that meet at most weekly, change management is localized to the DevOps team supporting a given application. As such, the DevOps team is accountable for production deployment decisions.

Release Management

In support of an Agile approach, DevOps follows a "fail fast, fail forward, fail small" philosophy for release management. This applies to iterative application releases as well as application and associated virtual environment patching and updating. The philosophy is to never rollback or modify in-place in production, but rather deploy new.

Compliance Management

In order to support rapid and frequent delivery, compliance is also both "shifted left" and automated. It's shifted left by moving compliance controls and audit points to earlier stages of the delivery pipeline. For example, compliance control and auditing is applied to the infrastructure as code, automation scripts, blueprints, and policies under version control as well as the automated delivery pipeline itself rather than at the time of deployment into production. Compliance is automated, for example, by building compliance policies into the delivery pipeline automation and logging auditable information as the application and its virtualized environment move through the delivery pipeline. And, for example, by implementing compliance checks audit information logging into the rules determining whether the application and its virtualized environment are promoted to the next stage in the delivery pipeline.

Incident Management

The DevOps team now represents level 2 support where previously development and operations were level 3 support.

Event Management

The DevOps team is now responsible for monitoring the application and its immediate environment (virtual machine, containers). This could include direct monitoring through tools such as vRealize Operations Manager but at a minimum requires they be alerted to related events. This also implies they have direct responsibility for defining and/or customizing monitoring capabilities as well as defining and providing application-specific events.

Capacity Management

Increased emphasis is placed on capacity monitoring and reclamation due to the increased pace of virtual machine/container creation and deletion in a virtual environment supporting DevOps.

5. Evolving DevOps In the Enterprise

5.1 Evolving from Pragmatic Concession to Ideal End State

DevOps as a Service is a good starting point and stepping-stone to the ideal end state of Dev and Ops collaboration. It provides an opportunity to start small and grow in a controlled manner without requiring immediate and extensive organizational and leadership cultural changes or large re-organization efforts. A Service Owner can be named and the service defined and developed like any other service provided by IT. Candidates with the right cultural and mind-set attributes can be recruited from an existing and well-known talent pool.

When the decision is made to begin moving towards a Dev and Ops collaboration model, take the following high-level steps:

1. Start small by identifying an early adopter application development team most conducive to DevOps.
2. Identify individuals to provide the IT operations skillsets as part of the initial DevOps team. These individuals would ideally come from the DevOps as a Service team. Their positions on the DevOps as a Service team would be backfilled with sufficient time to train replacements.
3. Accelerate DevOps enablement activities described in Section 3.3 above for the early adopter application development team.
4. Form the initial, dedicated DevOps team and plan to have it execute for 3 to 6 months including frequent interaction with DevOps enablement to capture lessons learned.
5. Begin rolling out to additional application development teams.
6. Maintain DevOps as a Service to support development teams on applications not large enough to justify a dedicated DevOps team.

5.2 Continuity from the SDDC to DevOps

There is a continuum between the SDDC and DevOps. Organizations who have invested in a SDDC solution find themselves well under-way towards enabling DevOps. For example:

- Extensive automation is a cornerstone of the SDDC. The shift from routine tasks and fire-fighting to implementing automation workflows is a common trait to the SDDC and DevOps.
- The service-oriented operating model used to optimize SDDC operations focuses on breaking down IT's traditional plan, build, and run organizational silos to create integrated teams (whether virtual or physical) providing those functional capabilities. DevOps involves breaking down these same silos from a development and operations perspective.
- A major benefit of the SDDC is the enablement of closer alignment with the business, positioning IT as an innovative service provider. DevOps furthers this alignment by enabling business agility through providing a mechanism for quickly responding to changing business needs and opportunities.

6. Resources

The following resources are useful for further understanding and exploring DevOps and associated processes:

Description	Resource
Introduction to DevOps	"The Phoenix Project", by Gene Kim, Kevin Behr, and George Spafford "The DevOps Handbook" by Gene Kim, Jez Humble, Patrick Debois & John Willis
Continuous Delivery	"Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation", by Jez Humble and David Farley
High performance organizations	"Lean Enterprises", by Jez Humble, Joanne Molesky, and Barry O'Reilly
Transformation at Scale	"Leading the Transformation: Applying Agile and DevOps Principles at Scale", by Gary Gruver and Tommy Mouser

7. About VMware Professional Services

VMware Professional Services transform IT possibilities into business outcomes. Our comprehensive portfolio of consulting and education services help you uncover and exploit the unique opportunities made possible by VMware technology and solutions.

Drawing on our unparalleled product expertise and customer experience, we collaborate with your team to address the technical, people, process, and financial considerations for your VMware solution to deliver results that are positive, tangible and material to IT and your business.

For more information on VMware Professional Services and how we can help you, contact your local VMware representative or visit http://www.vmware.com/professional-services.html.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com