Solution Path for Continuous Delivery With DevOps

Published 16 June 2021 - ID G00741321 - 24 min read

By Analyst(s): Gregg Siegfried, Bill Holz, Paul Delory

Initiatives: Application Development and Platforms for Technical Professionals

Achieving continuous delivery with DevOps requires agile culture, modern architecture, infrastructure automation and pervasive security, but those alone are not enough. DevOps technical professionals must embrace full life cycle automation to scale their efforts while delivering business value.

Overview

Key Findings

- Automation is integral to DevOps and to achieving continuous delivery — specifically automation across a variety of disciplines and targets. This research catalogs the most important of them.
- The security of the software delivery supply chain has been in the news due to several high profile compromises within both commercial and open source ecosystems. Securing your artifacts and toolchains is an integral part of that.
- The DevOps toolchain is a production workload, but is not always managed to the same performance, availability and governance standards.

Recommendations

Applications and I&O technical professionals adopting DevOps practices to achieve CD should:

- Build skills maintenance into your DevOps process to position your initiative for success. Myriad options are available to level each other up, including formal training, online learning platforms and communities of practice.
- Protect your artifacts and the toolchain used to produce them by closely managing access to the tools themselves, and carefully handling critical assets like certificates and signing keys. Although not a new concern, recent software supply chain vulnerabilities have illustrated the potential impact of not doing so.

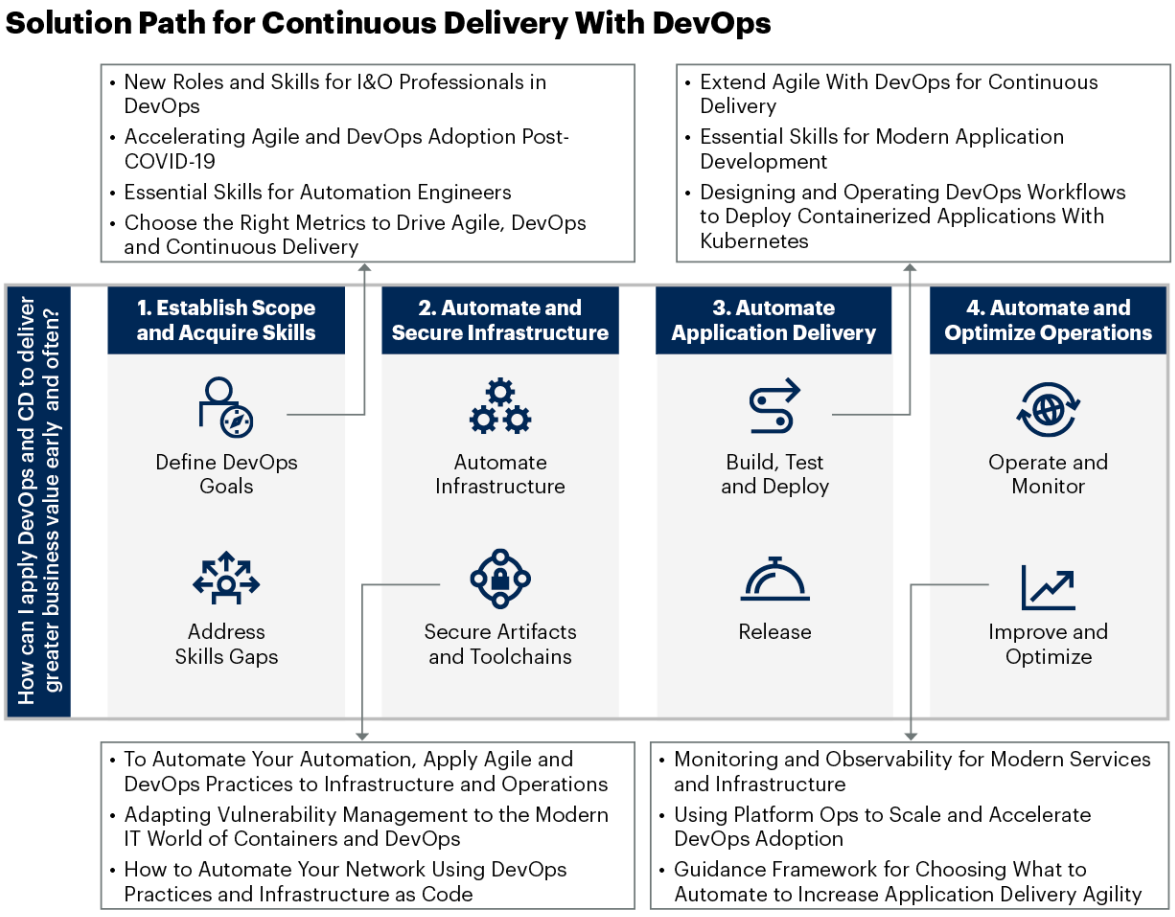
- Build support for ongoing operations by ensuring that health, performance and integrity-related telemetry is available to operators. Instrument processes, as well as products, in a way that supports continuously improving people, processes and technology.

Problem Statement

How can I apply DevOps and CD to deliver greater business value early and often?

Solution Path Diagram

Figure 1: Solution Path for Continuous Delivery with DevOps



Source: Gartner
741321_C

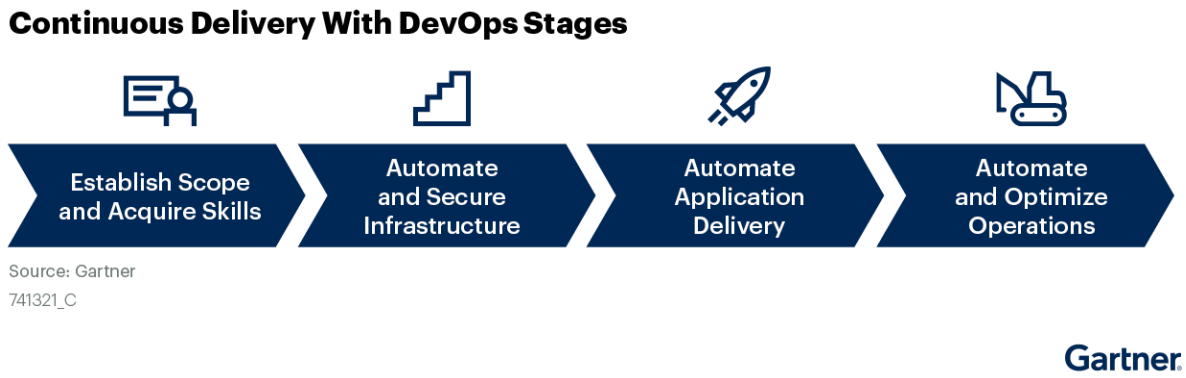
Gartner

Solution Path

This Solution Path, in combination with Solution Path for Agile Transformation, provides guidance for technical professionals seeking to achieve CD. Teams that have not yet embraced agile should start with that research. Those that are familiar with agile should follow both solution paths together by ensuring that their skills and practices emphasize agile fluency. This is particularly important to ensure that I&O technical professionals, who may not be as steeped in agile culture and terminology, are equally prepared for the transformation (see Figure 1).

As Figure 2 illustrates, the stages in this Solution Path overwhelmingly emphasize automation. There are many reasons for this, but recall that CD includes the word "continuous." Any process intended to be continuous relies heavily on automation, and this is no exception. Additional benefits include reducing human error, amplifying output independent of labor and, improving quality and robustness through increased validation.

Figure 2: Continuous Delivery With DevOps Stages



Step 1: Establish Scope and Acquire Skills

For any process improvement or business transformation initiative to succeed, it is best to begin with the end in mind. DevOps means different things to different organizations and teams, so an important part of the planning process is to establish what that is for you.

Step 1.1: Define DevOps Goals

There is not a universal consensus on what DevOps entails and how it is implemented in every organization. One of the most common definitions for DevOps used within Gartner is included below:

A business-driven approach for delivering business value using agile methods, collaboration and automation.

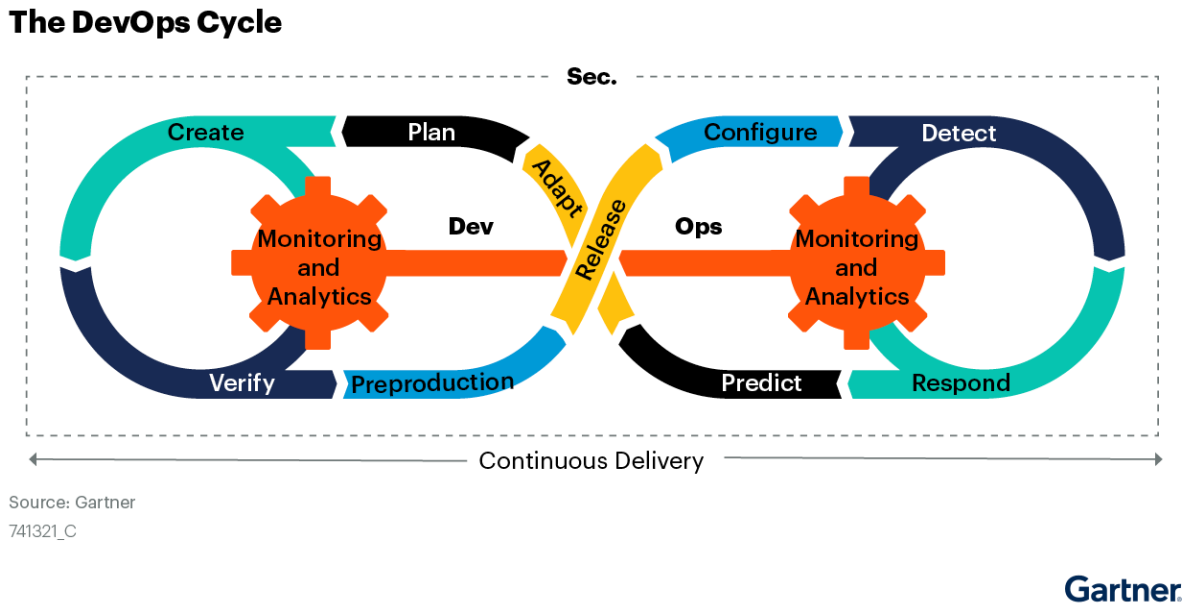
We have also seen variations on the term DevOps that are intended to reflect additional functions and roles that need to be included as well. This is where terms like DevSecOps and NetDevOps appear. All of these expanded terms reduce simply to DevOps. As used in this research, DevOps should be considered as inclusive as possible.

Gartner webinars are available for further grounding in DevOps practices and principles:

- The Keys to DevOps Success
- The 4 Stages of DevOps Adoption to Maximize Business Success

One of the most common illustrations of DevOps has been an infinity symbol, often referred to as the "lazy-8." Figure 3 is a variation of the lazy-8, which is intended to include explicit treatment of security as well as continuous delivery.

Figure 3: The DevOps Cycle



The objective of achieving continuous delivery is to regularly update software to incorporate user feedback, shifts in the market and changes in business strategy. This represents the practical manifestation of your DevOps goals.

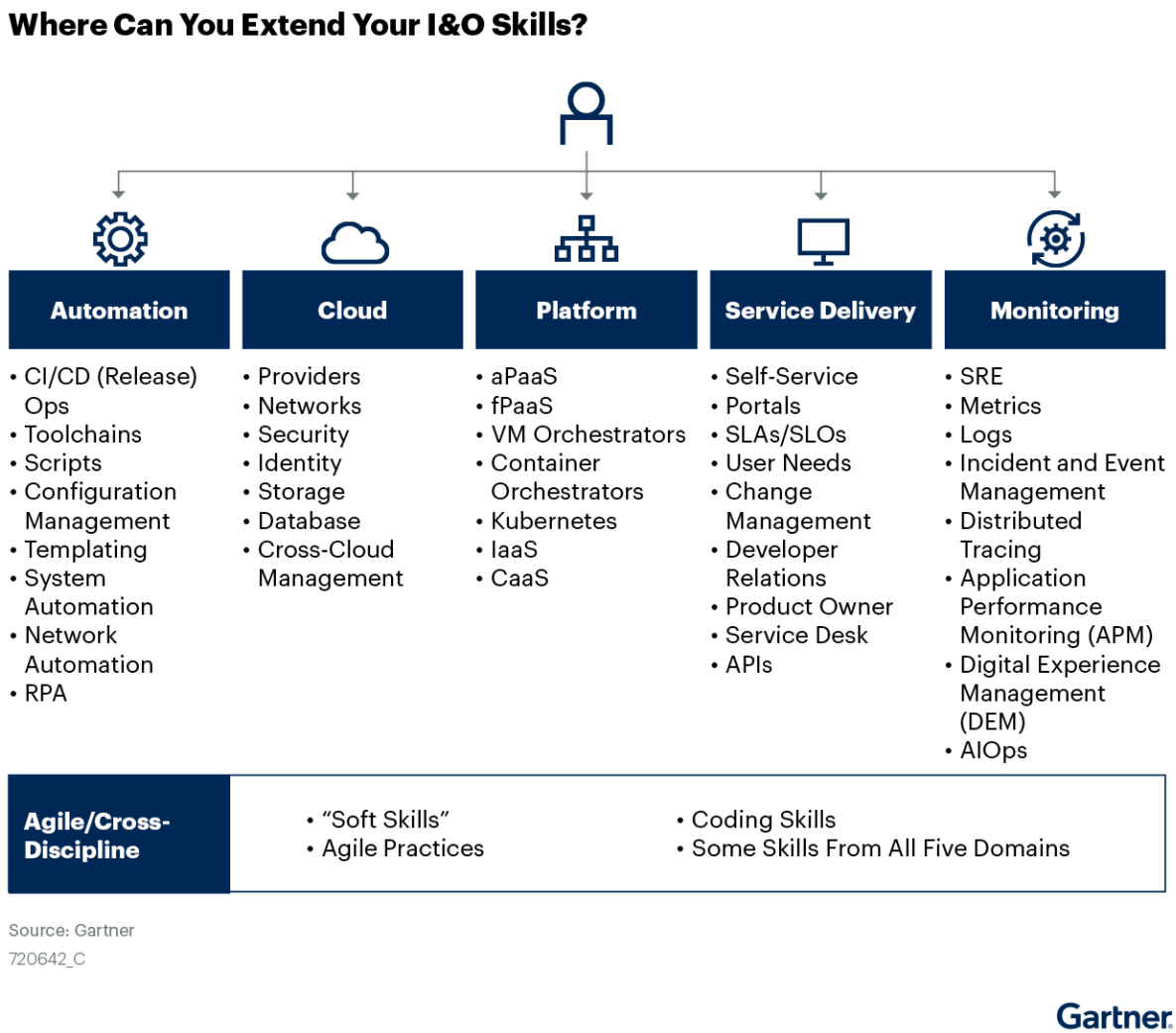
To successfully achieve CD with DevOps, technical professionals must first build a solid DevOps foundation. The following Gartner research will inform your ability to do so:

- The global pandemic of 2020 and 2021 has had a profound impact on how and where many of us work. Organizations whose DevOps practices presumed geographic proximity were left at a disadvantage and had to rapidly iterate. Avoid those types of assumptions in your transformation by reading Accelerating Agile and DevOps Adoption Post-COVID-19 as you define your objectives.
- As mentioned earlier, successful DevOps requires agile, and a companion to this research can ensure that your agile journey gets off on the right foot. See the Solution Path for Agile Transformation for the necessary prework.
- News of security breaches and system compromises has become increasingly high profile, and no organization wants to end up on the front page for that reason. Ensure that you account for the strategies and tooling used to integrate application security throughout the product life cycle by reviewing Structuring Application Security Tools and Practices for DevOps and DevSecOps, and incorporate that guidance into your plan.
- Teams whose primary purpose is not building software products can benefit from DevOps and CD as well. AI, machine learning and data technical professionals can increasingly benefit from the same agile principles that have made DevOps as valuable for software development. See Demystifying XOps: DataOps, MLOps, ModelOps, AIOps and Platform Ops for AI for examples of how to build agility into your data pipelines as well.
- Continuous improvement is the backbone of lean, agile and DevOps. Understanding how to measure your performance and assess its improvement over time is critical to attaining the promised business value. See Choose the Right Metrics to Drive Agile, DevOps and Continuous Delivery for assistance with baselining and measurement techniques you can use to demonstrate your teams' improvement over time.

Step 1.2: Address Skills Gaps

Organizations planning DevOps initiatives will identify skills gaps. Formal training is desirable in areas requiring immediate remediation, such as when your organization adopts Scaled Agile (SAFe). Other deficiencies can be left for development using cross training and cohort development. For I&O technical professionals, the adoption of DevOps has a profound impact on the way they work — particularly if the prior operating model was more traditional and silo-based. The need for new roles and skills are emerging as the role of operations evolves. More information about this is available in New Roles and Skills for I&O Professionals in DevOps. Figure 4, from that research, illustrates some of the areas into which technical professionals may choose to expand.

Figure 4: New Skills for I&O Professionals



Source: Gartner
720642_C

Gartner

Automation and cloud are two of the most important areas for skills growth. Technical professionals still have options, such as whether to go broad, cross domain or deep within a single domain — but consider these domains along with some of the more traditional I&O domains, such as servers, storage and networks. Review the research below to guide your skills development as you prepare for DevOps adoption.

- It has never been more important for technical professionals interested in career longevity to take command of their professional development and skills relevance. The guidance included in Upskilling in Crisis — Elevate Your Skills with Continuous Learning: A Gartner Trend Insight Report, Proven Practices to Enhance Skills in Application Architecture, Development and Software Engineering and Assessing Online Learning Platforms for Technical Skills Development takes the uncertainty and upheaval associated with the world in 2020-2021 and offers the continuous learning approach as a way to mitigate obsolescence and maximize value.

- Application development comprises a set of core skills around architecting, writing, testing and integrating software as part of a larger initiative. As languages and frameworks and platforms fall in and out of favor, a certain amount of regular reskilling should be considered part of the job. See Essential Skills for Modern Application Development and Essential Skills for Application Architecture for Gartner's guidance on the most important practices and technologies.

- As identified previously, automation is the fabric that ties DevOps and CD together and makes them go. Understanding how to automate tasks — and deploy and manage automation artifacts — is something that everyone on the team should be aware of. Research that will help you acquire and maintain these skills is available in Essential Skills for Automation Engineers and Essential Skills for Automation Architects.

- Many organizations find that cloud skills are the most in-demand — particularly if they are just establishing a presence there. Hiring the necessary people might be an option, but that is not available to everyone. It is also not uncommon for organizations to bring in a managed service provider (MSP) to help get them started with the cloud, but the MSP may need an internal team to turn things over to as well. Cloud architects and engineers are key to orchestrating cloud adoption and operating cloud platforms in steady state. See The Cloud Architect: Skills Guidance for Modern Technical Professionals and The Cloud Engineer: Skills Guidance for Modern Technical Professionals to help you and your teammates grow these skills.

- In the same way that the best way to learn a new programming language is to write programs in it, cloud skills are similarly perishable in that using the cloud platforms to solve realistic problems is ideal for reinforcing learning. Reading Jump-Start your Cloud Skills with These Objective-Driven Assignments and working through the provided exercises will help reinforce the newly acquired cloud skills.
- Fostering a culture of continuous learning is often supported through communities of practice. See How to Build Successful Communities of Practice for Knowledge Management to see how.

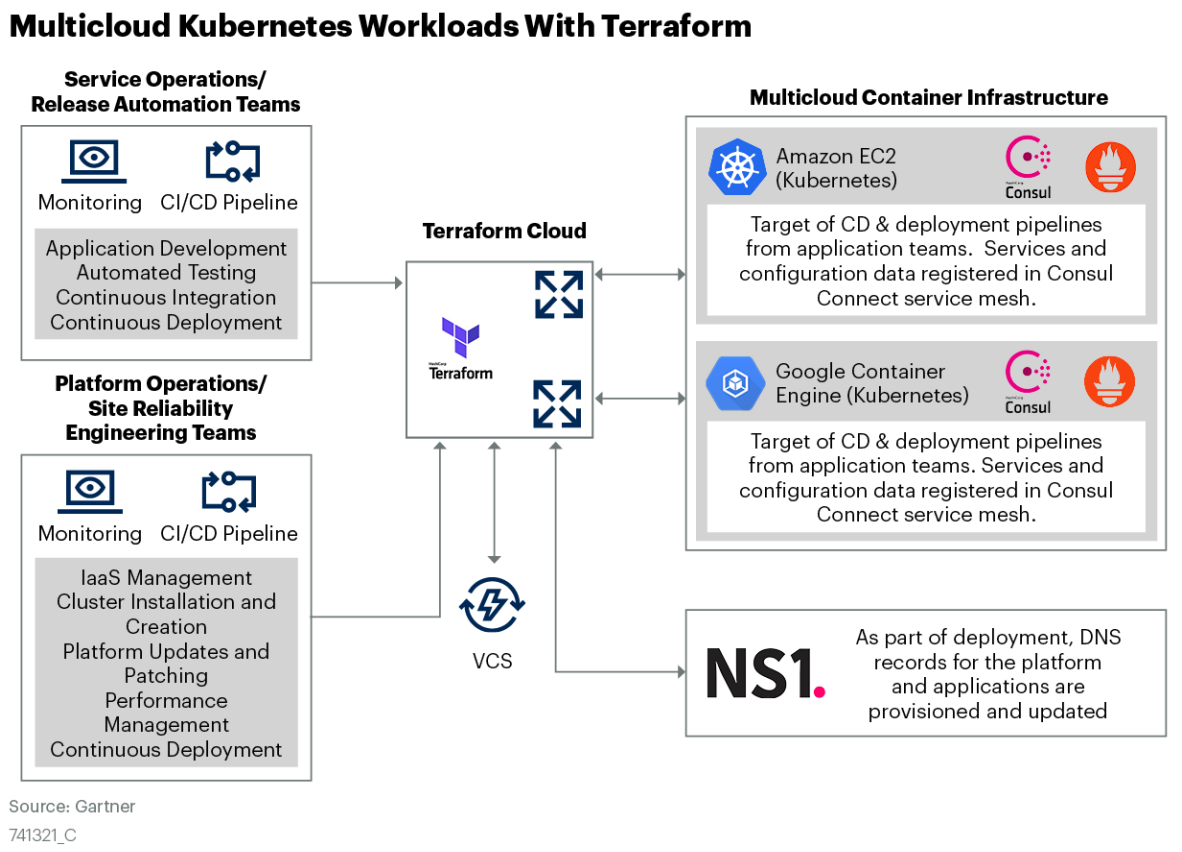
Step 2: Automate and Secure Infrastructure

In this step, infrastructure may refer to a few different things. First, and most commonly, this is the actual compute, storage, network, cloud resources on which application workloads are deployed. Automating and securing these to facilitate build, test, deploy and release (see Step 3) is in many ways synonymous with DevOps in pursuit of CD. Next, there is the DevOps toolchain — that is, infrastructure that falls into this category as well. The impact of an attacker compromising your build and packaging process to incorporate malicious, but authentic, looking capabilities into your software product can be severe. This is evidenced by SolarWinds breach that was disclosed in December 2020.

Step 2.1: Automate Infrastructure

Infrastructure automation is not new, but the mechanisms used to automate infrastructure have evolved substantially in the cloud era. It was once common for each platform or equipment manufacturer to ship their own automation toolkits, often at additional charge. Now, however, the prevalence of APIs and software-defined infrastructure (SDI) has ushered in a relative uniformity — at least on the provisioning side — that has given rise to products such as HashiCorp Terraform. Terraform's role in DevOps and CD is discussed in Assessing HashiCorp Terraform for Provisioning Cloud Infrastructure (see Figure 5, which illustrates a Terraform-based infrastructure automation with Kubernetes workflow).

Figure 5: Multicloud Kubernetes Workloads With Terraform



Today's cloud infrastructure and SDI platforms are easily automated, but there are often concerns about policy, governance and orchestration. Is an event-based continuous integration style enough, or is a self-service request portal necessary?

- Many organizations struggle to get started with infrastructure automation. The vendor landscape is increasingly fragmented and complex, it is easy to find yourself facing tool sprawl and redundancy, and vendors often oversell their capabilities. See Visualizing the Infrastructure Automation Pipeline for a graphical introduction to the process and tools and a way to build perspective quickly.
- A set of guidance that you can use to adopt infrastructure automation at scale is included in Solution Path for Infrastructure Automation. This is a good starting point for organizations that do not have established capabilities.

- Achieving CD with DevOps is often a strategy to improve agility. Due to the complexity of today's environments and toolchains, determining the most valuable aspects to automate poses a challenge. Automating the wrong thing not only fails to address the original problem, but it adds needless complexity and reduces efficiency. See Guidance Framework for Choosing What to Automate to Increase Application Delivery Agility to avoid that pitfall and ensure that the right things are being automated to achieve your objectives.

- Containerizing software was originally a packaging and dependency management technique, but as containers have become more common, the need to automate the life cycle of containerized software has increased. Kubernetes is in wide use as a container scheduling and orchestration platform. See Solution Path for Implementing Containers and Kubernetes to learn how to incorporate containerized software into your DevOps practices.

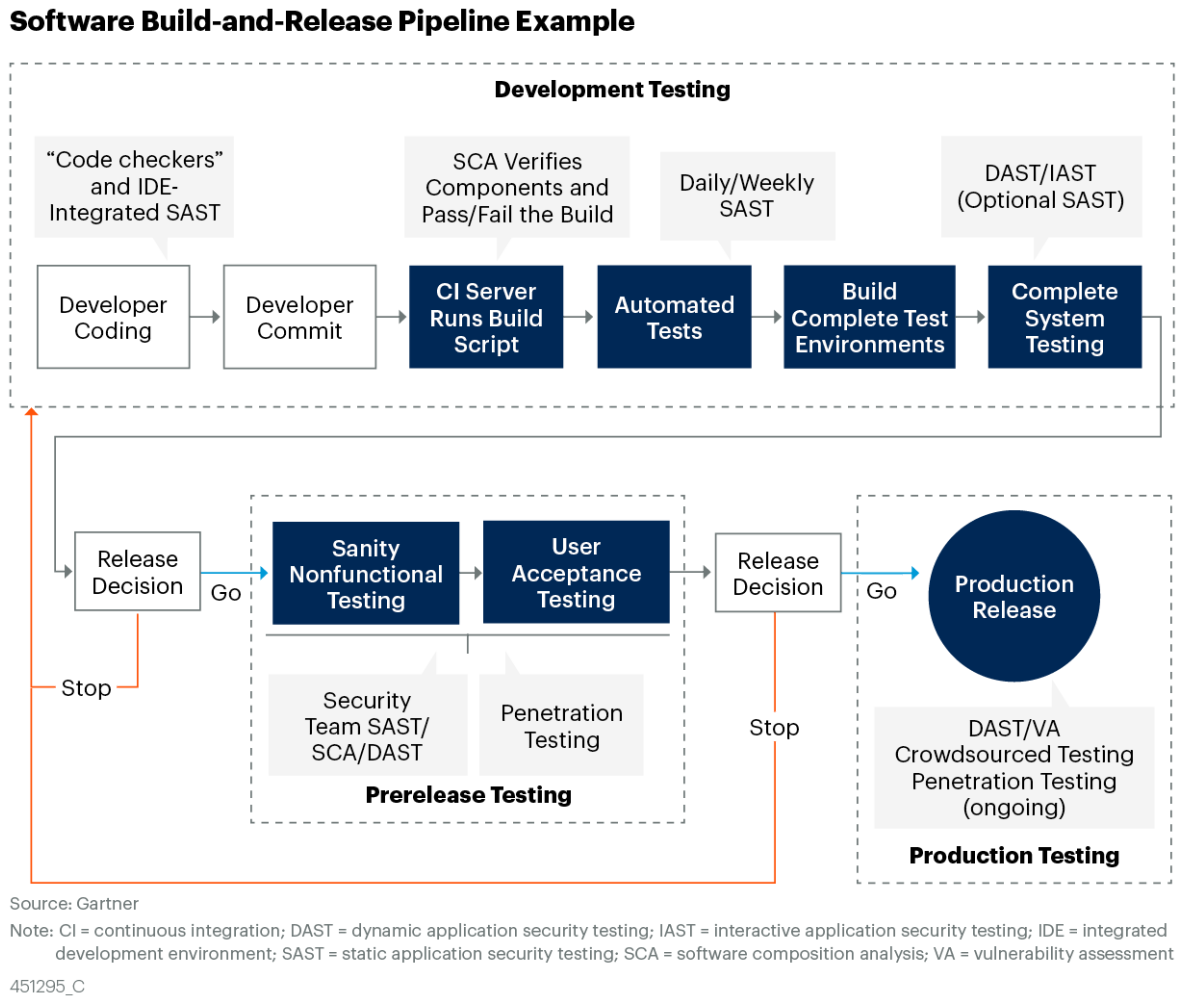
- Kubernetes is complex to operate, and requires expertise and training to manage securely. The Guidance Framework for Securing Kubernetes provides detailed recommendations to ensure that you adopt Kubernetes without subjecting your organization to unknown and unnecessary risk.

- When organizations unify their DevOps and infrastructure automation toolchain around a CI/CD model, rather than establishing separate toolchains for both, it may improve stability, encourage reuse and cross training. See To Automate your Automation, Apply Agile and DevOps Practices to Infrastructure and Operations to learn more about this. This is also helpful when an I&O organization wishes to adopt DevOps practices, but does not develop many custom applications.

Step 2.2: Secure Artifacts and Toolchains

To achieve CD with DevOps in any realistic environment, security must be accommodated at multiple levels. Figure 6 illustrates a build pipeline that includes the most common aspects of security testing.

Figure 6: Software Build-and-Release Pipeline



Three types of security testing are performed in the above build pipeline.

- Software composition analysis (SCA): Assesses the open source components of a software application for known vulnerabilities (CVEs), as well as license compliance and compatibility. Commercial SCA tools may also include remediation capabilities.
- Static application security testing (SAST): Performs a static analysis on the application source code looking for exploitable vulnerabilities.
- Dynamic application security testing (DAST): Attempts to detect vulnerabilities in running applications.

A fourth technique called runtime application self-protection (RASP), is when an application or application runtime can detect active or attempted exploitation. RASP tools have historically been stand-alone solutions, but application performance monitoring (APM) solutions, such as Cisco AppDynamics and Dynatrace, are incorporating RASP features. Sysdig, although not an APM, supports RASP-like capabilities for some container-based workloads.

Review the research below for further discussion on security in DevOps and CD.

- Security and risk management is a topic of considerable depth and complexity. This research centers on DevOps and continuous delivery. To appreciate where these elements fit within broader security and risk management practices, see Building the Foundations for Basic Security Hygiene.
- Organizations just starting with DevOps security, or technical professionals not functioning in a security capacity that would like to understand the concepts behind cloud and application security, should read Guide to Cloud Security Concepts and Guide to Application Security Concepts. These provide concise yet thorough examinations of these often thorny topics in a way most any technical professional can understand.
- Protecting your applications is not a one-size-fits-all exercise, and there are a variety of factors that determine the appropriate controls and countermeasures to deploy to protect each one. To ensure that you are leveraging the right ones, see Decision Point for Securing Application Architecture.
- Infrastructure-as-code (IaC) is an increasingly common approach to building and managing the artifacts associated with application development or testing, as well as infrastructure automation. For organizations that work primarily with COTS software, IaC may be their primary source of managed, text-based artifacts. When your application is deployed in containers — COTS or not — the vulnerability management practices and tools must evolve. See Adapting Vulnerability Management to the Modern IT World of Containers and DevOps to see how to do so.
- The success of IaC has given rise to policy-as-code, as a way for security teams to build, test, manage and reuse policy and governance artifacts within cloud and SDI environments. Widespread use of policy-as-code is still nascent. Each hyperscale provider has long had their own policy language and mechanism, and industry neutral initiatives such as the open policy agent (OPA) are still emerging — but quickly. Security teams should plan to start adopting policy-as-code. See Using Cloud-Native 'Policy as Code' to Secure Deployments at Scale to learn how.

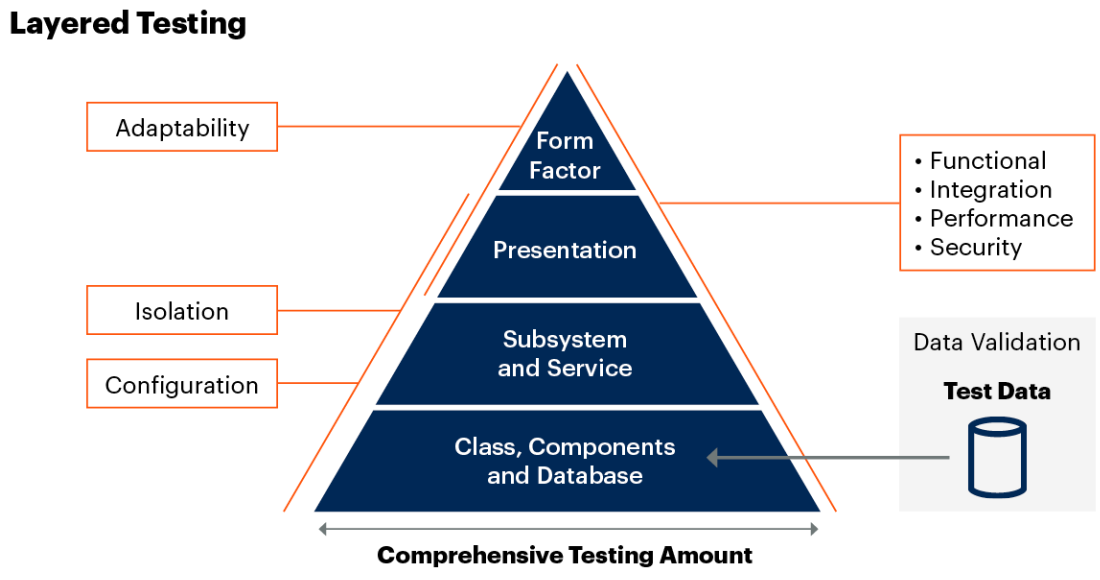
Step 3: Automate Application Delivery

Automating the delivery of application software is core to what we consider the modern, pipeline-based CI/CD toolchain. As the toolchain expands in scope due to product expansion or additions to the validations required for products to progress, this orchestrated, event-driven process is the focal point of continuous delivery. Technical professionals must ensure that this delivery pipeline continues to meet organizational requirements.

Step 3.1: Build, Test and Deploy

The automated cycle of build, test and deploy activities defined by the modern continuous delivery pipeline should be considered the heartbeat of healthy product development. Provided that the individual components are healthy and the validation coverage sufficiently thorough, the software products produced will be functionally robust, performant and secure. Thorough, automated test coverage that facilitates the identification and correction of defects as early in the process as possible, so-called "shifting left" reduces rework and improves correctness. What, however, is thorough testing? Figure 7 illustrates a layered approach to testing that balances efficiency and coverage. See Solution Path for Testing Software Applications for much more detail on testing.

Figure 7: Layered Testing



Source: Adapted From Concepts Created by Mike Cohn of Mountain Goat Software
741321_C

Gartner

Tool selection is a key component in achieving continuous delivery, but it is inefficient to be in a state of continuous tool selection. A layered approach — similar to testing — is often most effective. The tools you choose must support your processes, not the other way around. Consider this research as you seek to balance these competing forces.

- Software development presumes version control software is used to track and manage changes to text-based artifacts. Ensuring that your branching strategy is compatible with your agility and delivery requirements is paramount. See Use Trunk-Based Development for Agility and Continuous Delivery to select and validate your approach.
- Testing should not mean after-the-fact. Many organizations leverage test-first methodologies, such as test-driven development (TDD) and behavior-driven development (BDD) to reinforce acceptance criteria and build collaboration into their development processes. See Use Test-First Development Processes to Jump-Start your SDLC to learn how to do so.
- Continuous integration is a prerequisite for CD and is a distinct process, even though many have become accustomed to treating CI and CD as a unit. CI is in actuality more tightly coupled with version control than with CD. See A Guidance Framework for Continuous Integration: The Continuous Delivery 'Heartbeat' for guidance on CI.
- Databases can be a source of friction as DevOps practices are adopted and may even prevent CD from being achieved. The impact may range from dependency sprawl that prevents software components from being independently released to a source of catastrophic outage when testing is incomplete. See Implement Agile Database Development to Achieve Continuous Delivery to identify strategies to avoid these antipatterns.

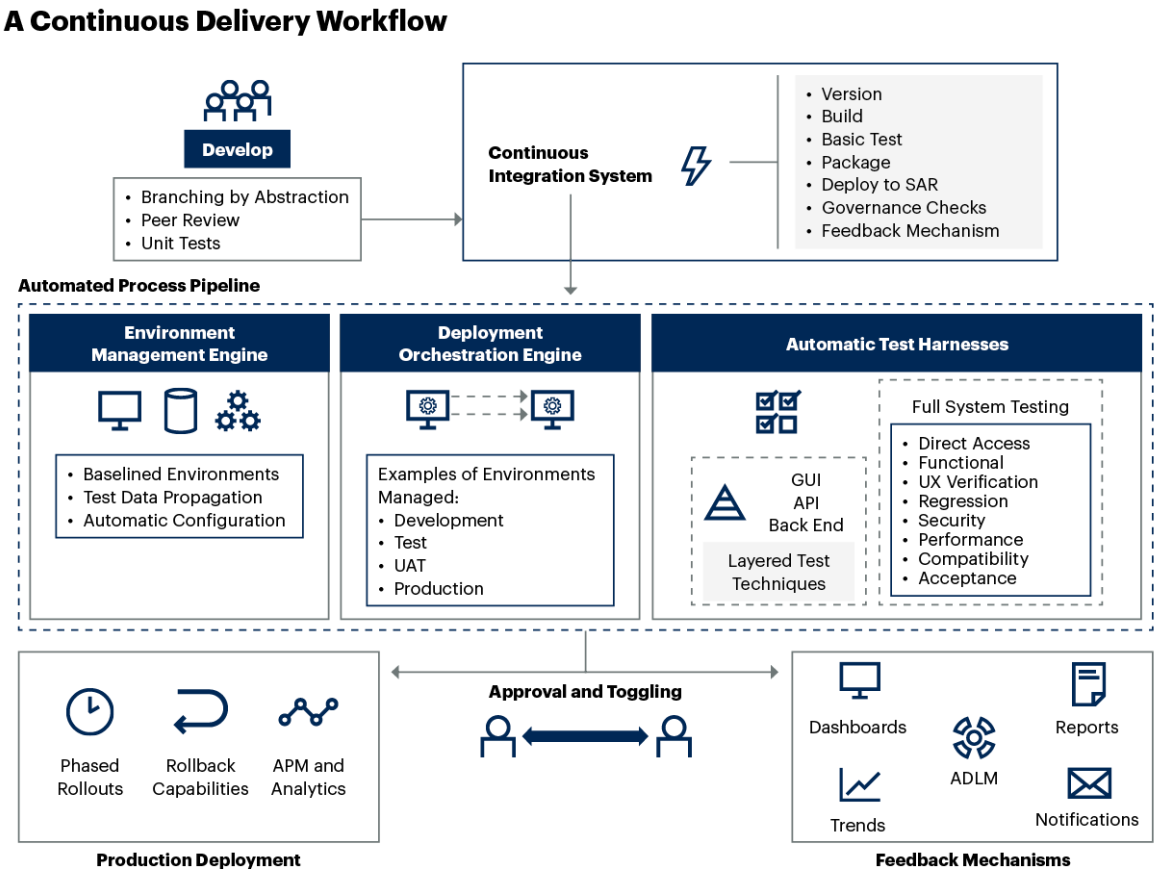
Step 3.2: Release

When continuous delivery has been achieved, the release process should become anticlimactic. The release process has been executed under non-production circumstances so many times that there is little to be concerned about. Even under continuous deployment conditions, where software travels from developer commit to production release without human intervention, each stage of intermediate validation — as the software moves through testing environments and scenarios — should exercise the same update process.

Amazon has become an oft-quoted case study in continuous deployment, given the rate at which software is released into production. An excellent description of how this is done is available in the Amazon Builders' Library. [1] Those interested in continuous deployment should study it.

For the rest of us on the way to achieving continuous delivery, release is intentional. Figure 8, from Extend Agile With DevOps for Continuous Delivery, illustrates the canonical CD workflow.

Figure 8: The Continuous Delivery Workflow



Source: Gartner
385600_C

Gartner

Endeavor to make your release process as lightweight and transparent as possible. Reversibility is important if you need to back out changes. Consider this research and guidance as you plan your releases.

- Two common software production release styles are blue-green and canary. The former acts as a toggle, moving traffic from one instance of the application (blue) to the next (green) version. There are some drawbacks to this, depending on the scope of the deployment apparatus that must be duplicated, and the risk of the idle environment becoming obsolete if change activity is not frozen during the release interval. The canary release style subjects a percentage of traffic to new builds of software, increasing the percentage as success allows. Of course, the system must support the use of multiple versions in production simultaneously, and is generally more suitable for services based applications. These styles can be used in combination as well. See How to Automate Server Provisioning and Configuration Management for more detail on these release styles.

- Within the Kubernetes ecosystem, Gartner is seeing an increasing interest in the GitOps release and configuration management pattern. GitOps uses a git-based version control repository as the source of truth about infrastructure configuration and application release. It is a continuous deployment mechanism, with a commit to the underlying git repository acting as the event that initiates a deployment event or release process. Flux and Argo CD are GitOps-friendly CD orchestration tools. These orchestrators run in your Kubernetes clusters, watch the underlying repositories and ensure that the changes committed to the repository are reflected in the cluster. More information on GitOps is available on its site, [2] as well as in Gartner's To Automate Your Automation, Apply Agile and DevOps Practices to Infrastructure and Operations.

- The use of feature flags or feature toggles is one mechanism to construct a very lightweight release process. Feature flags allow you to deploy new software to production behind what amounts to an on/off switch that defaults to 'off'. To release a feature, its flag is turned on. This may be done in stages using a canary release approach. Once the feature is in production and stable, the flag is customarily removed in a subsequent release to reduce the complexity of deployed code. See Learn From the Spotify Model for Better Enterprise Agile Scaling and Use Trunk-Based Development for Agility and Continuous Delivery for more insight on feature toggles.

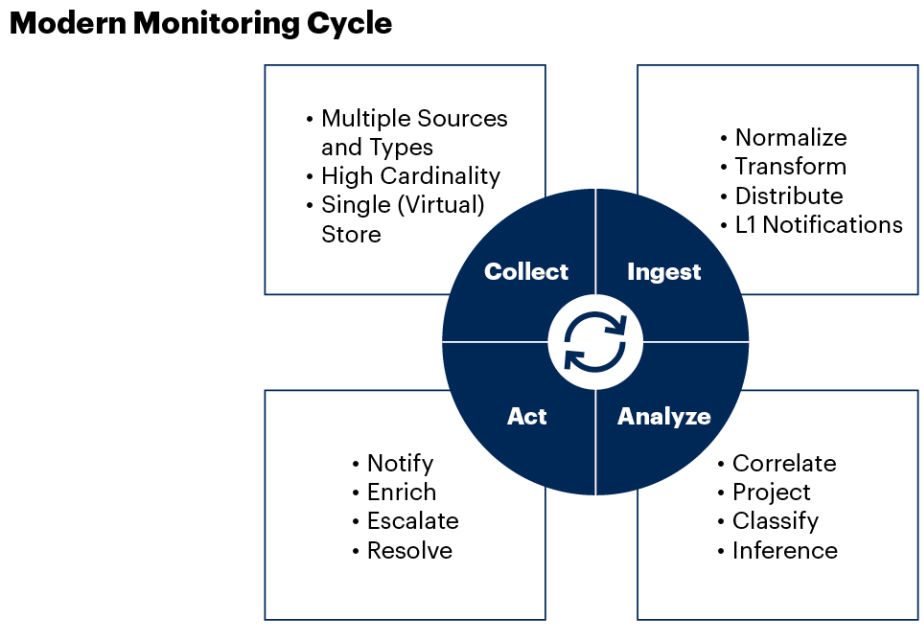
Step 4: Automate and Optimize Operations

The DevOps practices of continuous integration and continuous delivery (and this solution path, so far) emphasize product development, and not the day-to-day (or "day 2") operations that are required for the products to actually deliver business value to your customers, including internal users. The delivery in continuous delivery does not end at deployment — ensuring that your products are available, performant and secure on a daily basis is included as well.

Step 4.1: Operate and Monitor

Monitoring is required to ensure that your systems remain operational within their expected health and performance objectives. This includes not only those systems touched by customers, but the entirety of the toolchain as well. The CI/CD-based build pipeline, the IaC mechanisms, the monitoring systems: all of it must be monitored. When many think about monitoring, a mental picture of a room full of people staring at displays (also called "monitors") comes to mind. These traditional operations centers make for an impressive backdrop, but are not the primary way most monitoring is carried out. Gartner's Solution Path for Modern Infrastructure and Application Monitoring discusses monitoring in detail and includes Figure 9.

Figure 9: The Modern Monitoring Cycle



Source: Gartner
720254_C

Effective monitoring requires attention to each stage in this cycle, and might appear to be more of a data acquisition and management process than anything else. This is the right way to look at it. Gartner has published research that covers monitoring and service management that will help you to focus your efforts:

- Monitoring relies on telemetry, which takes many forms. Managing the collection, ingestion and analysis of this telemetry on a time critical basis is fundamental to operations. See Monitoring and Observability for Modern Services and Infrastructure for guidance on your telemetry strategy.
- Log and event data can be among the most difficult to manage due to its size and the forms that it can take. This is illustrated by the myriad products that have emerged recently to support log monitoring. Gartner recommends a holistic approach to log monitoring, as described in Guidance Framework for Deploying Centralized Log Monitoring.
- Containerized software and the Kubernetes platform have specific operational requirements to manage service stability and collect health and performance data. See Monitoring Containers and Kubernetes Workloads for the details.
- As with many daily operations and incident response activities that you automate (see Step 4.2), there will be incidents that require human operators and engineers to intervene. There are products and tools available to optimize the interactive and collaborative aspects associated with incident response. See Automate Incident Response to Enhance Incident Management to learn more.
- Many organizations have adopted IT service management (ITSM) practices and principles to support monitoring and service management. The way ITSM and DevOps interact and interoperate can become a substantial point of friction or conflict in organizations. For two different perspectives on this, see Extend Agile With DevOps for Continuous Delivery and ITSM Best Practices: How to Optimize IT Incident Management with Automation.

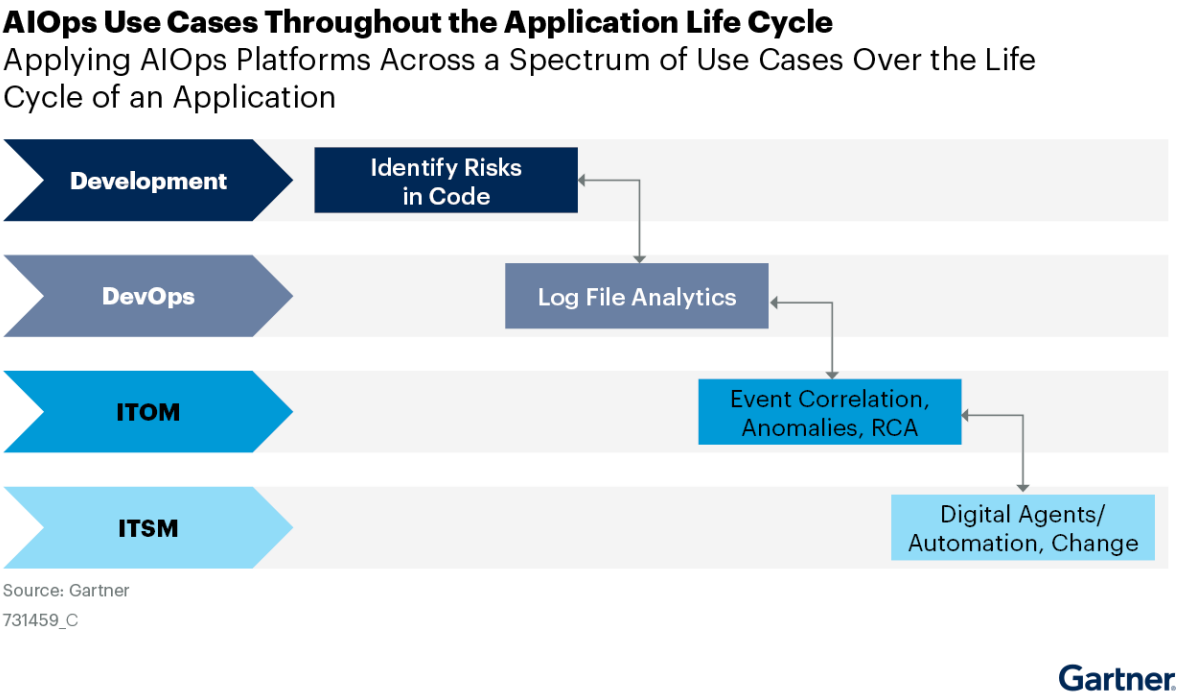
Step 4.2: Improve and Optimize

Much has been written about self-driving IT operations, AIOps and NoOps, which intend to automate or eliminate tasks that are typically performed by human operators. This accelerates performance and availability metrics, such as mean time to detect (MTTD) and mean time to restore (MTTR). Aside from a few specific, curated examples, there is little evidence that human operators will be out of the picture any time soon. However, given the telemetry and event management challenges referenced in Step 4.1, opportunities to deploy technology to assist operators by reducing the volume of data they are required to process — or decisions they are required to make — are much more achievable.

Consider the opportunities for release-aware monitoring. As new builds of a software product are deployed, a monitoring solution — likely a digital experience monitoring (DEM) or application performance monitoring (APM) tool — detects the presence of the new build. It applies additional scrutiny to attenuation in things like latency and error rate as production traffic is introduced by a canary process. If the behavior of this new build is worse than the prior build, the monitoring tool can generate an event that results in the new build being backed out and the prior build restored. Provided the regressive behavior can be characterized, this process is quite straightforward, but eliminates what can be a stressful and manual process of manually deploying, watching and un-deploying new builds. A mechanism like this is required for organizations considering achieving continuous deployment.

Interest in deploying AIOps platforms is on the rise among Gartner clients. Figure 10 from Solution Path for Adopting AIOps illustrates some of the use cases these products have.

Figure 10: AIOps Use Cases



Improving and optimizing operations continuously is the perpetual challenge for organizations using DevOps practices to achieve CD. Review Gartner's operations research when planning your approach:

- For more details on how AIOps can be deployed within IT operations to support use cases, such as anomaly detection and event correlation, see Understanding the Application of AIOps Disciplines within IT Operations.
- Organizations that are employing Kubernetes may find that productive and efficient life cycle management of containerized software requires additional steps. See Designing and Operating DevOps Workflows to Deploy Containerized Applications With Kubernetes to accelerate your journey.
- As networks have become increasingly virtual and software defined, modern software has become more reliant on robust networking, so monitoring them requires a different approach. See Transform Network Monitoring and Analytics for the Modern Era to learn what must be done.
- Once you have some successes under your belt, one of the next challenges will be scaling DevOps and ensuring that the operations teams share these practices. Adopting the Platform Ops approach as described in Using Platform Ops to Scale and Accelerate DevOps Adoption is one way to do so.

- Site reliability engineering (SRE), which originated at Google, is a developer-centric and data-driven operating model that treats failure at scale as normal, because it is. The objectives of SRE and DevOps overlap. See Assessing Site Reliability Engineering (SRE) Principles for Building a Reliability-Focused Culture to learn how and why.

Evidence

[1] Automating safe, hands-off deployments, Amazon Web Services.

[2] GitOps

Recommended by the Authors

Some documents may not be available as part of your current Gartner subscription.

Proven Practices to Enhance Skills in Application Architecture, Development and Software Engineering

Essential Skills for Modern Application Development

Solution Path for Agile Transformation

Solution Path for Infrastructure Automation

How to Succeed With Microservices Architecture Using DevOps Practices

Solution Path for Implementing Access Management

Best Practices to Enable Continuous Delivery With Containers and DevOps

Why DevOps Success Requires Platform Teams

Use 8 Simple Steps to Get DevOps Right

3 Steps to Integrate Security Into DevOps