What is DevOps? The ultimate guide

February 2024

TechTarget

In this guide:

DevOps is a combination of the terms *development* and *operations*, meant to represent a collaborative or shared approach to the tasks performed by a company's application development and IT operations teams. In its broadest meaning, DevOps is a philosophy that promotes better communication and collaboration between these teams -- and others -- in an organization. Use this guide to learn more about how DevOps works, why it's important, its benefits and challenges, prominent DevOps tools and more.

In this guide:

What is DevOps? The ultimate guide

STEPHEN BIGELOW, SENIOR TECHNOLOGY EDITOR

The word *DevOps* is a combination of the terms *development* and *operations*, meant to represent a collaborative or shared approach to the tasks performed by a company's application development and IT operations teams.

In its broadest meaning, DevOps is a philosophy that promotes better communication and collaboration between these teams -- and others -- in an organization. In its most narrow interpretation, DevOps describes the adoption of iterative software development, automation and programmable infrastructure deployment and maintenance. The term also covers culture changes, such as building trust and cohesion between developers and systems administrators and aligning technological projects to business requirements. DevOps can change the software delivery chain, services, job roles, IT tools and best practices.

Although DevOps isn't a technology, DevOps environments apply common methodologies. These include the following:

- Continuous integration and continuous delivery (CI/CD) or continuous deployment tools, with an emphasis on task automation.

In this guide:

- Systems and tools that support DevOps adoption, including software development, real-time monitoring, incident management, resource provisioning, configuration management and collaboration platforms.
- Cloud computing, microservices and containers implemented concurrently with DevOps methodologies.

A DevOps approach is one of many techniques IT staff use to execute IT projects that meet business needs. DevOps can coexist with Agile and other continuous software development paradigms; IT service management frameworks, such as ITIL; project management directives, such as Lean and Six Sigma; and other strategies.

Some IT professionals believe that the simple combination of *Dev* and *Ops* isn't enough, and the term DevOps should include business (BizDevOps), security (DevSecOps) or other areas.

WHY IS DEVOPS IMPORTANT?

At its core, DevOps and DevOps practices are shown to improve software quality and development project outcomes for the enterprise. Such improvements take several forms, including the following:

- **Collaboration and communication.** DevOps eliminates many of the traditional organizational silos that can inhibit creativity and workflows. DevOps practices bring together developers, IT operations, business leaders and application

In this guide:

stakeholders to ensure that the software being built is designed, developed, tested, deployed and managed in a way that is best for the business and users.

- **Development outcomes.** DevOps adopts a cyclical process of ongoing, iterative development. Traditional development methodologies, such as Waterfall development, codify requirements and outcomes months or years in advance of the actual development process. DevOps projects typically start small with minimal features, then systematically refine and add functionality throughout the project's lifecycle. This enables the business be more responsive to changing markets, user demands and competitive pressures.
- **Product quality.** The cyclical, iterative nature of DevOps ensures that products are tested continuously as existing defects are remediated and new issues are identified. Much of this is handled before each release, resulting in frequent releases that enable DevOps to deliver software with fewer bugs and better availability compared to software created with traditional paradigms.
- **Deployment management.** DevOps integrates software development and IT operations tasks, often enabling developers to provision, deploy and manage each software release with little, if any, intervention from IT. This frees IT staff for more strategic tasks. Deployment can take place in local infrastructure or public cloud resources, depending on the project's unique goals.

A well-executed DevOps environment enables a business to deliver more competitive and better-quality software products to market faster, with lower support and maintenance demands, than traditional development paradigms.

In this guide:

HOW DOES DEVOPS WORK?

DevOps is a methodology meant to improve work throughout the software development lifecycle. You can visualize a DevOps process as an infinite loop, comprising these steps: plan, code, build, test, release, deploy, operate, monitor and -- through feedback -- plan, which resets the loop.

Ideally, the cyclical loop that comprises each DevOps iteration alleviates a great deal of stress from development outcomes. Traditional Waterfall development used an all-or-nothing approach where requirements were gathered up front, and then code was written, tested and released as events; any outcome, performance issues or reliability problems were dealt with as an afterthought. DevOps gives organizations far more flexibility in developing and releasing software that matures systematically over time, letting the team adjust, change, learn and try innovative new approaches and take risks that traditional development paradigms could never allow. Organizations use a combination of culture and technology to pursue this goal.

To align software to expectations, developers and stakeholders communicate about the project, and developers work on small updates that go live independently of each other.

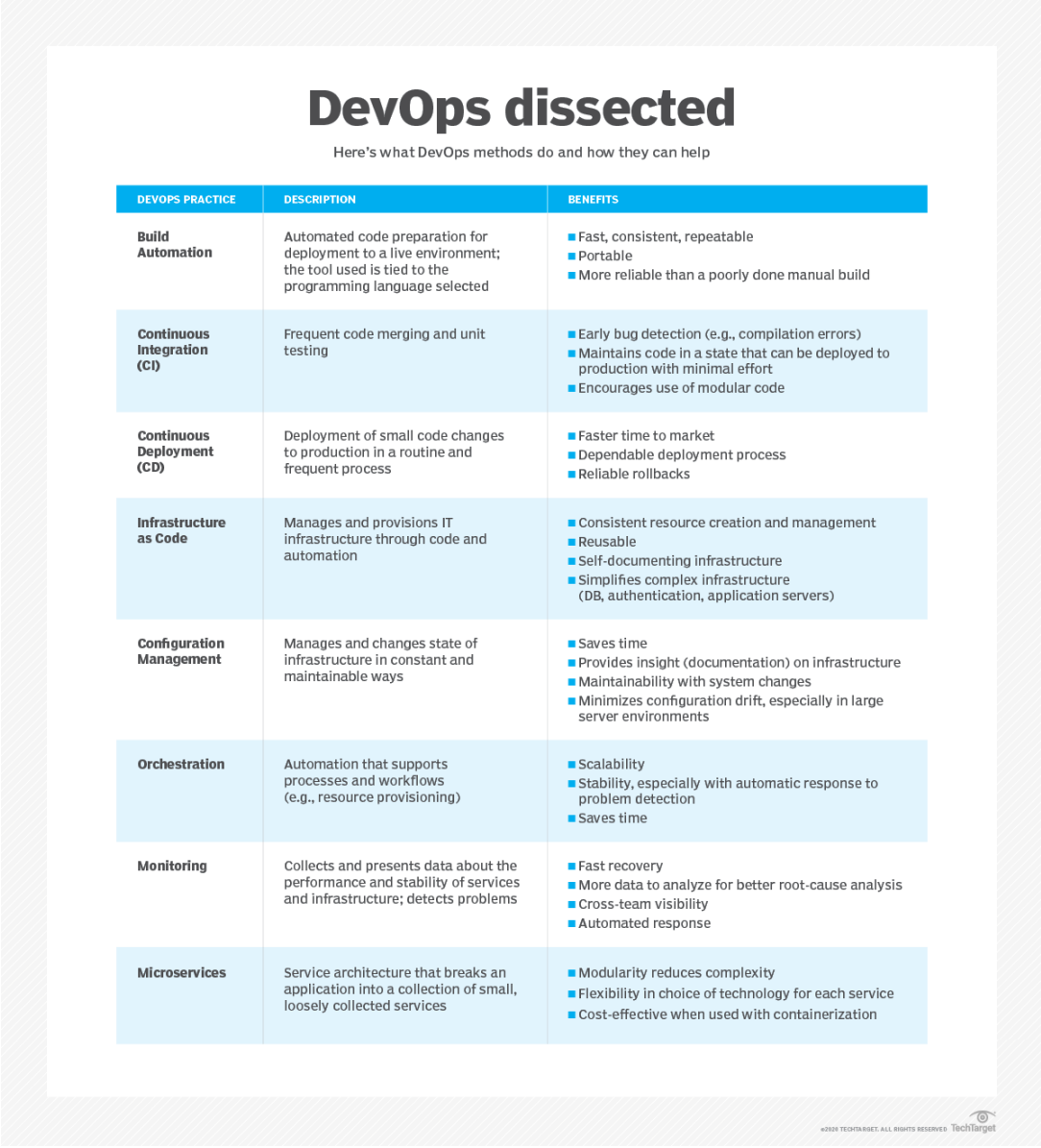
To avoid wait times, IT teams use CI/CD pipelines and other automation to move code from one step of development and deployment to another. Teams review

changes immediately and can rely on policies and tools to enforce policies and ensure releases meet standards.

It's easy to write software quickly; writing software that works is another story. To deploy good code to production, DevOps adherents use containers or other methods to make the software behave the same way from development through testing and into production. They deploy changes individually so that problems are traceable. Teams rely on automation and configuration management for consistent deployment and hosting environments. Problems they discover in live operations lead to code improvements, often through a blameless post-mortem investigation and continuous feedback channels.

Developers might support the live software, which puts the onus on them to address runtime considerations. IT operations administrators might be involved in the software design meetings, offering guidance on how to use resources efficiently and securely. Anyone can contribute to blameless post-mortems. The more these specialists collaborate and share skills, the more they can foster a DevOps culture.

In this guide:



DevOps dissected

Here's what DevOps methods do and how they can help

DEVOPS PRACTICE	DESCRIPTION	BENEFITS
Build Automation	Automated code preparation for deployment to a live environment; the tool used is tied to the programming language selected	■ Fast, consistent, repeatable ■ Portable ■ More reliable than a poorly done manual build
Continuous Integration (CI)	Frequent code merging and unit testing	■ Early bug detection (e.g., compilation errors) ■ Maintains code in a state that can be deployed to production with minimal effort ■ Encourages use of modular code
Continuous Deployment (CD)	Deployment of small code changes to production in a routine and frequent process	■ Faster time to market ■ Dependable deployment process ■ Reliable rollbacks
Infrastructure as Code	Manages and provisions IT infrastructure through code and automation	■ Consistent resource creation and management ■ Reusable ■ Self-documenting infrastructure ■ Simplifies complex infrastructure (DB, authentication, application servers)
Configuration Management	Manages and changes state of infrastructure in constant and maintainable ways	■ Saves time ■ Provides insight (documentation) on infrastructure ■ Maintainability with system changes ■ Minimizes configuration drift, especially in large server environments
Orchestration	Automation that supports processes and workflows (e.g., resource provisioning)	■ Scalability ■ Stability, especially with automatic response to problem detection ■ Saves time
Monitoring	Collects and presents data about the performance and stability of services and infrastructure; detects problems	■ Fast recovery ■ More data to analyze for better root-cause analysis ■ Cross-team visibility ■ Automated response
Microservices	Service architecture that breaks an application into a collection of small, loosely collected services	■ Modularity reduces complexity ■ Flexibility in choice of technology for each service ■ Cost-effective when used with containerization

WHAT PROBLEMS DOES DEVOPS SOLVE?

Each company faces its own challenges, but common problems include releases that take too long, software that doesn't meet expectations and IT that limits business growth.

Without wait times, manual processes and lengthy reviews, a DevOps project moves faster from requirements to live software. Shorter cycle times can keep requirements from shifting so that the product delivers what customers want. When requirements must shift to address changing user expectations and market demands, short cycle times can help when implementing those changes and getting updates to market faster.

DevOps solves communication and priority problems between IT specializations. To build viable software, development teams must understand the production environment and test their code in realistic conditions. A traditional structure puts development and operations teams in silos. This means developers are satisfied when their code delivers functionality -- and if the release doesn't perform well or fails in production, it's up to the operations team to fix the problems.

With a DevOps culture, developers don't resort to the "It worked on my machine" response when a problem arises. Changes rolled out to production are small and

reversible. Plus, the whole team understands the changes, which simplifies incident management.

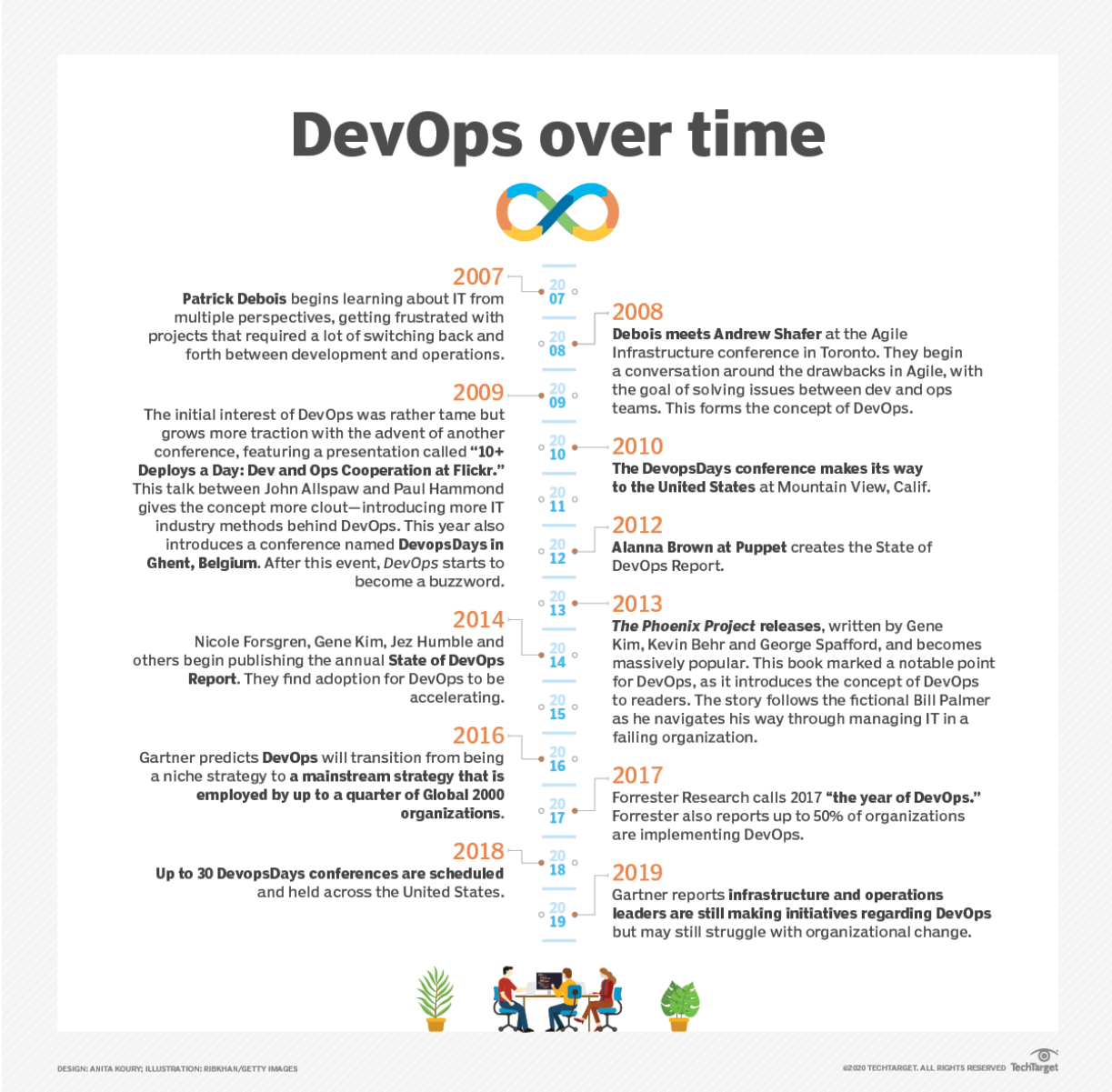
With a faster process from idea to live software, companies can capitalize on market opportunities. In this way, DevOps provides a competitive advantage for businesses.

THE EVOLUTION OF DEVOPS

Patrick Debois, a software development consultant, is credited with creating the term *DevOps* in 2009 by naming a conference DevOps Days. DevOps addressed a shortcoming of the Agile software development methodology: Iterative, rapid code development didn't necessarily lead to iterative, rapid code deployment.

Concurrent with the Agile push deeper into operations, IT administrators chafed against sometimes laborious and overly complex change management steps in the ITIL framework. ITIL champions stable, reliable and predictable IT, while Agile advocates for collaboration and change. DevOps struck a chord with people on both sides. In fact, organizations can do both ITIL and DevOps, especially if they embrace the cloud.

In this guide:



TechTarget

The concept of DevOps was then popularized with the book *The Phoenix Project* in 2013. *The Phoenix Project* uses a fictional narrative to illustrate endemic problems and help IT managers understand the concepts and benefits of collaboration and shared technologies.

Early proponents of DevOps included these key players:

- Debois and collaborator Andrew Clay Shafer.
- *The Phoenix Project* authors Gene Kim, Kevin Behr and George Spafford.
- Paul Hammond and John Allspaw, influential early adopters at Flickr.
- Continuous delivery pioneers Jez Humble and Dave Farley.
- Containerization advocate John Willis.
- State of DevOps Report organizers, including Alanna Brown and Nicole Forsgren.

As DevOps became popular, organizations formalized DevOps approaches. Retailer Target originated the DevOps Dojo training method, for example. Vendors touted DevOps-enabling capabilities in tools, from collaboration chatbots to CI/CD suites built into cloud services. Technologies such as virtual containers, public cloud services and microservices application architectures offered a natural fit to the fast and flexible nature of DevOps approaches. "DevOps engineer" soon emerged as a job title.

DevOps continues to evolve, as artificial intelligence surfaces to aid in everything from code creation to incident management. AI for DevOps (or *AIOps*) means

smarter automation and even shorter wait times, even seamless translations from business need to technological offering -- but plenty of barriers remain before AIOps becomes reality.

Although DevOps has achieved mainstream status, not all adopters are full DevOps converts. Many rely on a DevOps approach for revenue-generating IT projects, where they see a return on investment in the leading-edge tooling and skills. For many internal IT services that are stable and mature, such as traditional, well-established legacy applications, DevOps doesn't offer significant benefits.

DEVOPS METHODOLOGIES, PRINCIPLES AND STRATEGIES

DevOps is associated with Agile software development because Agile practitioners promoted DevOps as a way to extend the methodology into production. The approach has even been labeled a counterculture to the IT service management practices championed in ITIL. DevOps doesn't have an official framework.

To hone their strategies, organizations should understand the related contexts of DevOps, Agile and Waterfall development, site reliability engineering (SRE) and SysOps, and even the variations within DevOps.

DEVOPS VS. WATERFALL DEVELOPMENT

Waterfall development comprises a series of steps and gates in a linear progression to production. Its phases are requirements, analysis, design, coding and implementation, testing, operation, deployment, and maintenance. In Waterfall teams, development tests new code in an isolated environment for quality assurance (QA) and -- if requirements are met -- releases the code to operations for use in production. IT operations deploys multiple releases at once, with extensive controls. Support is operations' responsibility. Waterfall approaches engender long waits between software releases. Because development and operations teams work separately, developers aren't always aware of operational roadblocks that prevent code from working as anticipated.

The DevOps model aligns development, QA and IT operations efforts with fewer gates and more continuous workflow. For example, some of the operations' team responsibilities shift left in the app delivery pipeline to the development team. IT operations provides feedback for code improvements. Rather than gated steps, DevOps relies on CI/CD and continuous monitoring processes.

DEVOPS VS. AGILE DEVELOPMENT

Agile is a software development approach defined in the Agile Manifesto. Agile teams focus on incremental and rapid cycles of code creation and delivery, referred to as *sprints*. Each sprint iterates upon the last, which makes the software flexible and

adaptable to changing requirements. It's possible for the original vision of a project to be lost through this cycle.

DevOps arose from Agile's success at improving development speed, and the realization that disconnects between development and operations teams -- as well as between IT and the business side of the organization -- significantly hindered the Agile software's delivery to users.

In an Agile-only workflow, development and operations teams have separate objectives and leadership. When an organization uses DevOps and Agile together, both development and operations teams manage code throughout the software development lifecycle. Although Agile work is often formalized with a framework, such as Scrum, DevOps doesn't have a framework.

DEVOPS VS. CD

CD methodologies involve a frequent cyclical pattern of building and committing code changes to a central repository. The code can then be tested and validated using automated and manual testing technologies. If testing is successful, the build can be considered for release to production.

DevOps follows the same cyclical development and testing pattern. However, CD typically stops with the mechanical aspects of code creation. It doesn't offer the

principles and emphasis on communication and collaboration across the organization that DevOps seeks to improve.

DEVOPS VS. SRE

SRE arose concurrently with Agile and DevOps. Started in the early 2000s at Google, it's a programming- and automation-focused approach to the software development lifecycle. SRE emphasizes attention to software reliability and scalability intended to foresee and mitigate issues that might compromise an application's stability and health. Problems should be solved in a way that prevents them from occurring again. Rote tasks should be minimized.

The SRE toolbox closely matches that for DevOps. Both disciplines aim for continuous improvement. SRE and DevOps engineers seek to abolish silos between development and operations. Although DevOps also can extend to business stakeholders, SRE typically stays within the confines of IT processes.

DEVOPS VS. SYSOPS

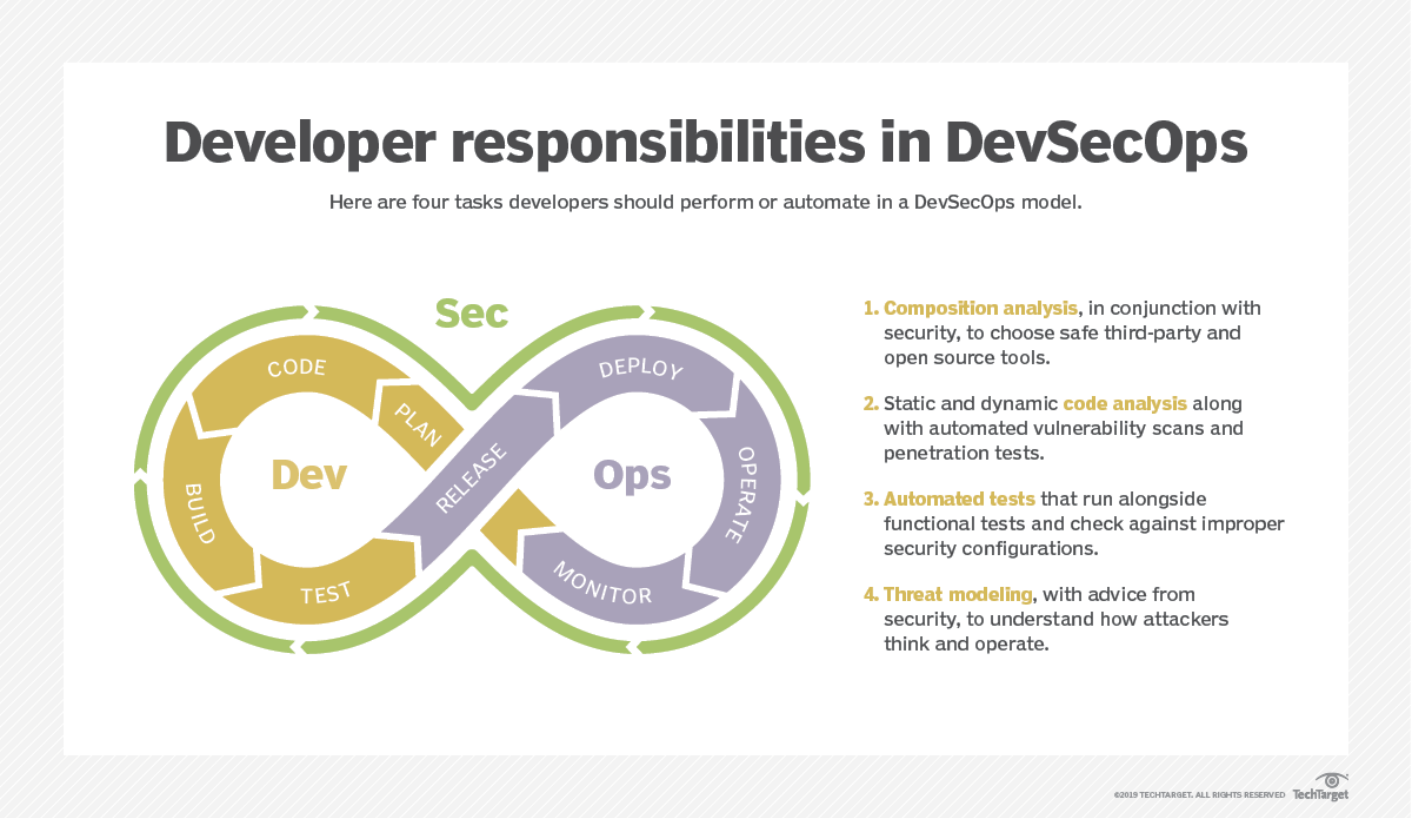
SysOps typically denotes that an IT administrator or IT team manages production deployment and support for a large distributed application, such as a SaaS product. As with DevOps adopters, SysOps teams should be versed in cloud computing and automation, as well as other technologies that enable applications to perform well at a large scale. SysOps teams troubleshoot IT outages and incidents, monitor for performance problems, enforce security rules and optimize operations.

They also focus on high availability, fault tolerance, security and performance just like other IT admins. Although SysOps professionals are likely to use some development tools and understand development processes, their work isn't as enmeshed with development as in a DevOps job. However, SysOps roles can exist within DevOps and SRE organizations.

DEVSECOPS VS. BIZDEVOPS VS. GITOPS

Some organizations broaden the scope of DevOps to include other roles, such as security teams, or departments. In DevSecOps, security and compliance planning, scans, testing and reviews occur continuously throughout the DevOps loop. BizDevOps focuses on connecting executives, application owners and other business stakeholders to the technical team, which develops, tests and supports the software. Although more collaboration is better than less, these collaborators must share effective, timely and precise input.

In this guide:



Another variation on DevOps, or a different faction of the same movement, is GitOps. Named for its focus on the eponymous repository and version control technology, GitOps espouses declarative source control over application and infrastructure code. Everything about the software -- from feature requirements to the deployment environment -- comes from a single source of truth. GitOps is often associated with change management and deployment techniques such as *immutable infrastructure*,

which dictates replacing or reinstancing -- rather than upgrading or changing -- deployed code each time a change is made.

WHAT ARE THE BENEFITS OF DEVOPS?

DevOps has been widely embraced by developers and organizations because of the many improvements that it brings over traditional development paradigms. DevOps benefits include the following:

- Fewer silos and increased communications between IT groups.
- Faster time to market for software, enhancing revenue and competitive opportunities for the business.
- Rapid improvement based on user and stakeholder feedback.
- More testing results in better software quality, better deployment practices and less downtime.
- Improvement to the entire software delivery pipeline through builds, repository use, validations and deployment.
- Less menial work across the DevOps pipeline, thanks to automation.
- Streamlined development processes through increased responsibility and code ownership in development.
- Broader roles and skills.

WHAT ARE THE CHALLENGES OF DEVOPS?

However, DevOps challenges abound. The DevOps paradigm poses its own complexities and changes that can be difficult to implement and manage across a busy organization. Common DevOps challenges include the following:

- Organizational and IT departmental changes, including new skills and job roles, which can be disruptive to development teams and the business.
- Expensive tools and platforms, including training and support to use them effectively.
- Development and IT tool proliferation.
- Unnecessary, fragile, poorly implemented and maintained, or unsafe automation.
- Logistics and workload difficulties scaling DevOps across multiple projects and teams.
- Riskier deployment due to a fail-fast mentality and job generalization vs. specialization where access to production systems is handled by less IT-savvy personnel.
- Regulatory compliance, especially when role separation is required.
- New bottlenecks such as automated testing or repository utilization.

In short, DevOps doesn't solve every business problem, or benefit every software development project in the same way.

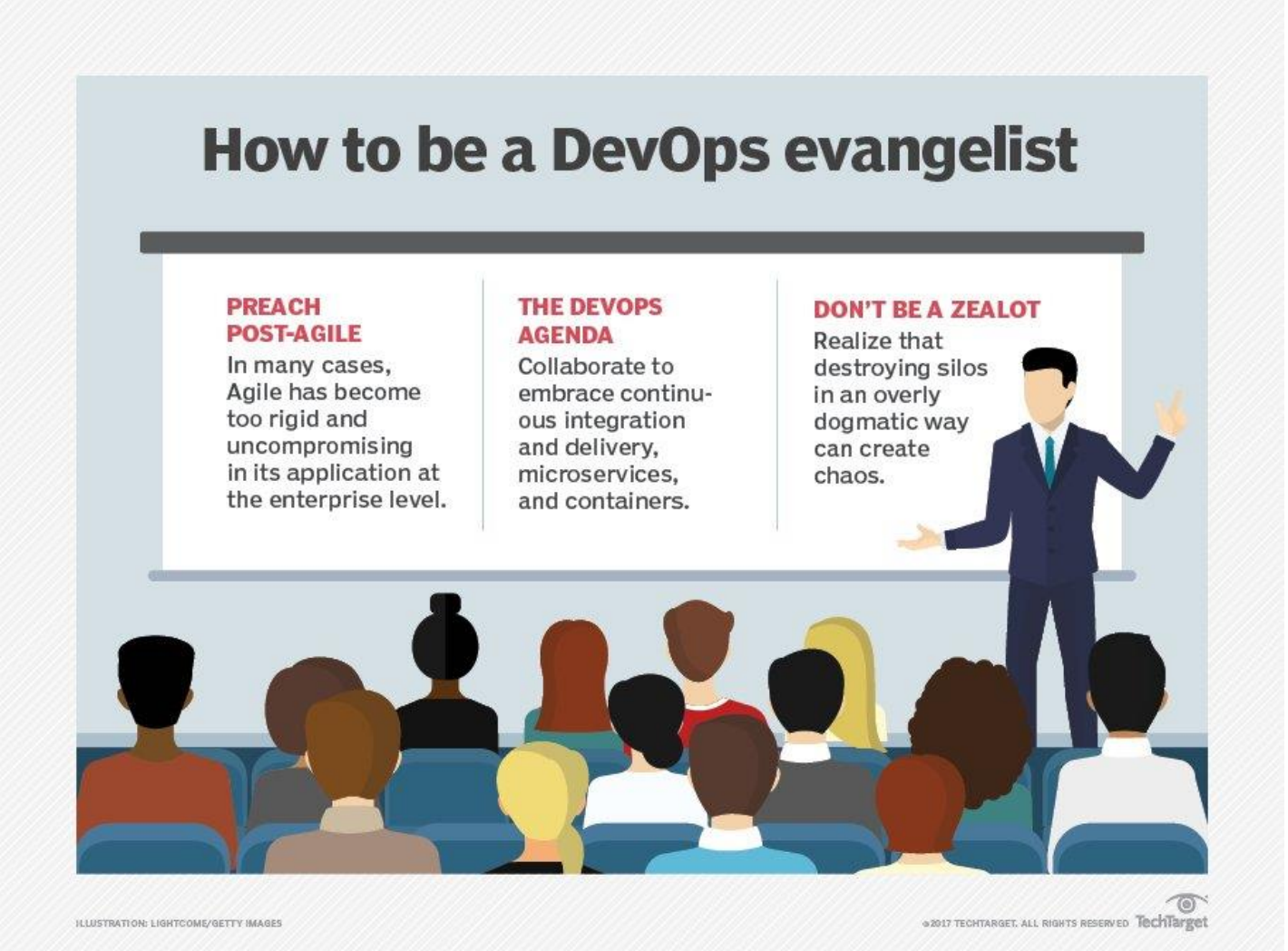
DEVOPS ADOPTION BEST PRACTICES

The adoption of DevOps or other CD paradigms can be disruptive to software and management teams. There are inevitable changes to workflows, processes, tool sets and even staffing that will drive the need for more training. It's easy for DevOps adoption to get off track and eventually fall by the wayside. Below are some tips that can help ease DevOps adoption and maximize the chances for DevOps success.

START SMALL

DevOps transformations don't happen overnight. Many companies start with a pilot project -- a simple application where they can get a feel for new practices and tools. DevOps teams look for some quick and easy wins to refine workflows, learn the tools and prove the value of DevOps principles. For large-scale DevOps adoption, try moving in stages.

In this guide:



CONSIDER THE WORKFLOWS

Initially, DevOps can mean a commitment from development and IT operations teams to understand the concerns and technological boundaries that exist at each stage of the software project. Agree upon KPIs to improve, such as shorter cycle times or fewer bugs in production. Understand how DevOps practices map against current development and deployment practices, and plan for suitable changes to the workflow while enhancing software quality, security and compliance. Lay the groundwork for continuous processes by communicating across job roles.

SELECT THE PROPER TOOLS

Evaluate the existing tools for software development and IT operations. More or different tools might be needed. Identify process shortcomings, such as a step that's always handled manually -- moving from a code commit to testing, for example -- or a tool without APIs to connect with other tools. Consider standardizing one DevOps pipeline for the whole company. With one pipeline, team members can move from one project to another without reskilling. Security specialists can harden the pipeline, and license management is eased. The tradeoff with this approach is that DevOps teams give up the freedom to use what works best for them.

EMPLOY MEANINGFUL METRICS

Simply adopting DevOps isn't enough to ensure success. Understand the need for DevOps in the first place -- what problems is it intended to solve, or what benefits is it intended to deliver. Select metrics and KPIs that will show those outcomes, and then

plan to measure and report on those metrics as an objective gauge of DevOps success. For example, a metric such as *defects found* or *code commit velocity* can help manage DevOps outcomes.

The organization now has a DevOps mindset in place, metrics to track for success and capable tools. Focus on best practices, knowledge sharing and skills development to continue improving. Optimize tooling and technologies, identifying roadblocks and gaps that affect your KPIs.

USE THE DEVOPS MATURITY MODEL

The DevOps maturity model illustrates five principal phases of adoption, ranging from novice to well established. Organizations can use the DevOps maturity model as a guide to adoption by identifying their place in the model:

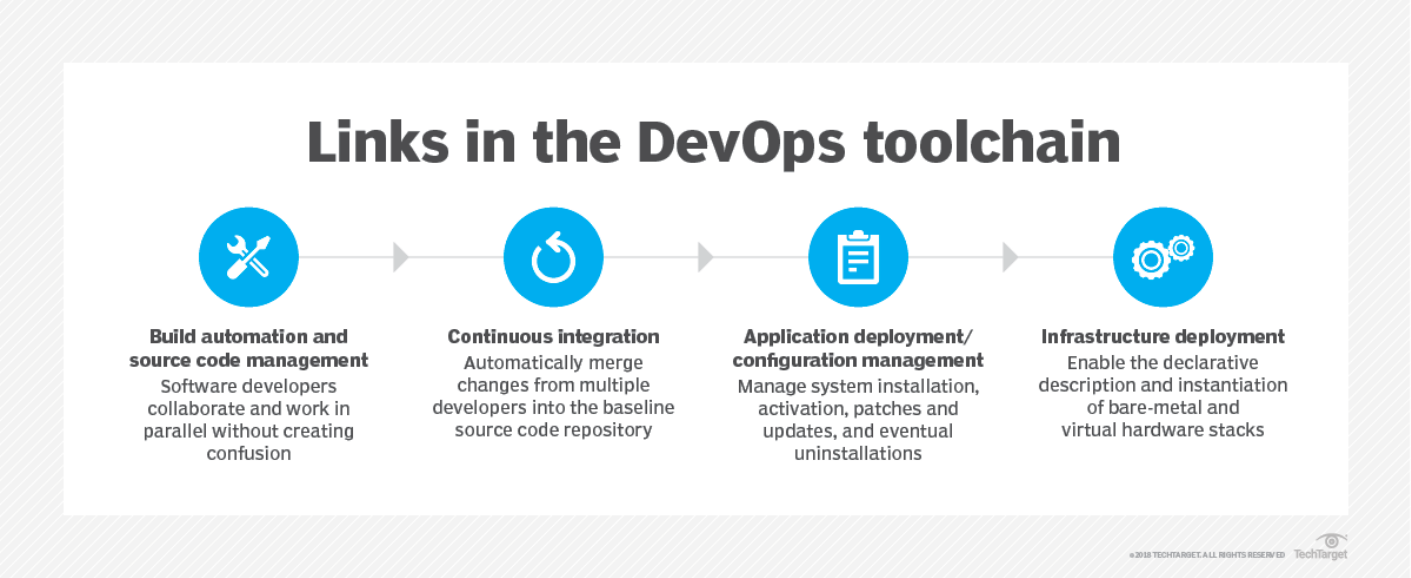
- **Initial.** Teams are siloed; work is reactive and done with ad hoc tool and process choices.
- **Defined.** A pilot project defines a DevOps approach, basic processes and tools. It's a proof of concept.
- **Managed.** The organization scales up DevOps adoption with lessons learned from the pilot. The pilot's results are repeatable with different staff members and project types.

- **Measured.** With processes and tools in place, the teams share knowledge and refine practices. Automation and tool connectivity increase, and standards are enforced through policies.
- **Optimized.** Continuous improvement occurs. DevOps might evolve into different tool sets or processes to fit use cases. For example, customer-facing apps have a higher release frequency, and financial management apps follow DevSecOps practices.

DEVOPS TOOLS

DevOps is a mindset, not a tool set. But it's hard to do anything in an IT team without suitable and well-integrated tools. In general, DevOps practitioners rely on a CI/CD pipeline, containers and cloud hosting. Tools can be open source, proprietary or supported distributions of open source technology.

In this guide:



Code repositories. Version-controlled source code repositories enable multiple developers to work on code. Developers check code out and in, and they can revert to a previous version of code if needed. These tools keep a record of modifications made to the source code -- including who made the changes and when. Without tracking, developers might struggle to follow which changes are recent and which versions of the code are available to end users.

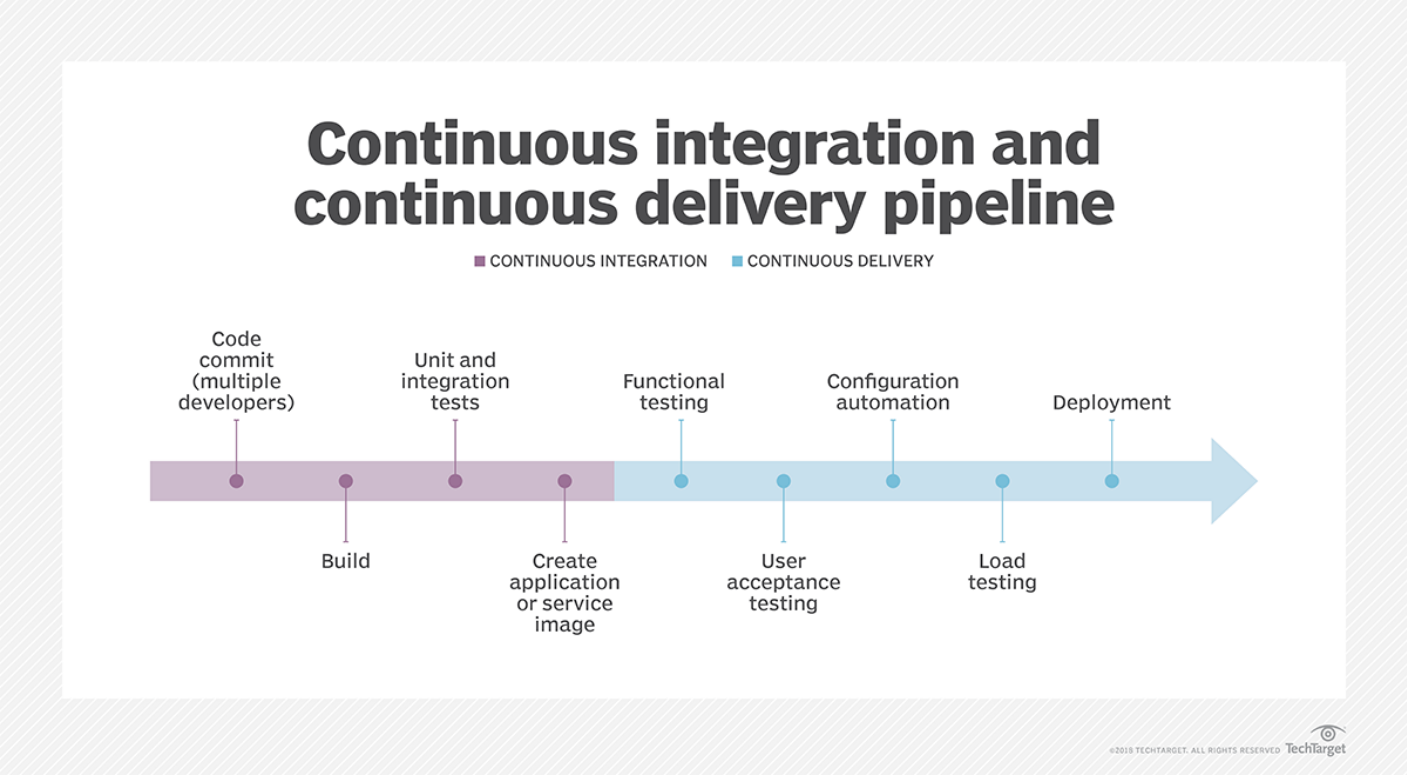
In a CI/CD pipeline, a code change committed in the version-control repository automatically triggers next steps, such as a static code analysis or build and unit

tests. Tools for source code management include Git and GitHub, along with Bitbucket, Mercurial, Subversion and others.

Artifact repositories. Source code is compiled into an artifact for testing. Artifact repositories enable version-controlled, object-based outputs. Artifact management is a good practice for the same reasons as version-controlled source code management. Examples of artifact repositories include JFrog Artifactory and Nexus Repository, Azure Artifacts, Amazon Elastic Container Registry, Cloudsmith, Dist, ProGet and Yarn.

CI/CD pipeline engines. CI/CD enables DevOps teams to frequently validate and deliver applications to the end user through automation during the development lifecycle. The continuous integration tool initializes processes so that developers can create, test and validate code in a shared repository as often as needed without manual work. Continuous delivery extends these automatic steps through production-level tests and configuration setups for release management. Continuous deployment goes a step further, invoking tests, configuration and provisioning, as well as monitoring and potential rollback capabilities. Common tools for CI, CD or both include Jenkins, GitLab, Travis CI, Atlassian Bamboo, Concourse and CircleCI.

In this guide:



Containers. Containers are isolated runtimes for software on a shared OS. Containers provide abstraction that enables code to work the same on different underlying infrastructure from development to testing and staging, and then to production. Docker and Apache Mesos are some of the most well-known containerization software, while Microsoft offers specific Windows container options. Container orchestrators -- such as Kubernetes and commercial Kubernetes

distributions Red Hat OpenShift and Amazon Elastic Kubernetes Service -- deploy, scale and maintain containers automatically.

Configuration management. Configuration management systems enable IT to provision and configure software, middleware and infrastructure based on a script or template. The DevOps team can set up deployment environments for software code releases and enforce policies on servers, containers and VMs through a configuration management tool. Changes to the deployment environment can be version controlled and tested, so DevOps teams can manage infrastructure as code (IaC). Configuration management tools include Ansible, Terraform, SaltStack, Puppet and Chef.

Cloud environments. DevOps organizations often concurrently adopt cloud infrastructure because they can automate its deployment, scaling and other management tasks. AWS, Google Cloud and Microsoft Azure are among the most used cloud providers. Many cloud vendors also offer CI/CD services.

Monitoring. Additionally, monitoring tools enable DevOps professionals to observe the performance and security of code releases on systems, networks and infrastructure. They can combine monitoring with analytics tools that provide operational intelligence. DevOps teams use these tools together to analyze how

changes to code affect the overall environment. Choices are wide-ranging, but include New Relic One, Dynatrace, Prometheus, Datadog and Splunk.

Cloud-based DevOps pipelines. Public cloud providers offer native DevOps tool sets to use with workloads on their platforms. An incomplete list includes AWS CodePipeline and CloudFormation, Azure DevOps and Pipelines and Google Cloud Deployment Manager. Cloud adopters have the option to use these pre-integrated services or run third-party tools. For example, an organization can use HashiCorp Terraform or CloudFormation to make IaC templates for its AWS workloads.

As-a-service models. Lastly, DevOps as a service is a delivery model for a set of tools that facilitates collaboration between an organization's software development team and the IT operations team. In this delivery model, the provider assembles a suite of tools and handles the integrations to seamlessly cover the overall process of code creation, delivery and maintenance.

In other cases, DevOps-as-a-service providers act as consultants that can assist businesses with DevOps projects, or supplement expertise that might be lacking anywhere in the DevOps process. Any service providers should be evaluated on factors including time in business, proven experience -- particularly in related market verticals -- and compatibility with existing DevOps tools and practices.

DEVOPS SKILLS AND JOB ROLES

DevOps is often said to be more of a philosophy or collaborative IT culture, rather than a strictly defined job description or skill set. Because the area is so broad, DevOps positions suit IT generalists better than specialists.

The role of DevOps engineer doesn't fall along one career track. Professionals can enter into the position from a variety of backgrounds. For example, a software developer can gain skills in operations, such as configuration of the hosting infrastructure, to become a DevOps engineer. Similarly, a systems administrator with coding, scripting and testing knowledge can become a DevOps engineer.

Many DevOps job listings call for container, cloud and CI/CD knowledge, as well as soft skills such as communication and team leadership. A DevOps engineer might also need to change processes and solve organizational problems to achieve business outcomes.

Other titles often found in DevOps organizations include the following:

- Infrastructure developer.
- Cloud consultant.
- Site reliability engineer.
- DevOps architect.
- Build and release engineer.

- Full-stack developer.
- Automation specialist.
- CI/CD platform engineer.

Don't judge the job by the title. Roles can vary dramatically, so take the time to look at the specific education, job requirements and desired experience indicated for each type of opportunity.

Most entry-level DevOps jobs require a degree in computer science or a related field that covers coding, QA testing and IT infrastructure components. Higher-level positions might require advanced degrees in systems architecture and software design. People on this career path should also expand their knowledge using DevOps books, and connect with other members of the community through blogs and conferences.

There are no industrywide recognized DevOps certifications as there are for formalized frameworks such as ITIL. IBM Red Hat offers DevOps certification, while DevOps Institute has vendor-neutral options for levels from foundational knowledge to expert.

Stephen J. Bigelow, senior technology editor at TechTarget, has more than 20 years of technical writing experience in the PC and technology industry.

Meredith Courtemanche has more than 10 years of experience at TechTarget Editorial, covering software development, DevOps, data centers and other technologies.

Alexander S. Gillis is a technical writer for the WhatIs team at TechTarget.

CONTINUED READING

- **Best free DevOps certifications and training courses**
- **DevOps KPIs you should track to gauge improvement**
- **Compare GitOps vs. DevOps for modern app deployments**