Hewlett Packard Enterprise

Business white paper

Getting grounded with DevOps

An introduction to DevOps

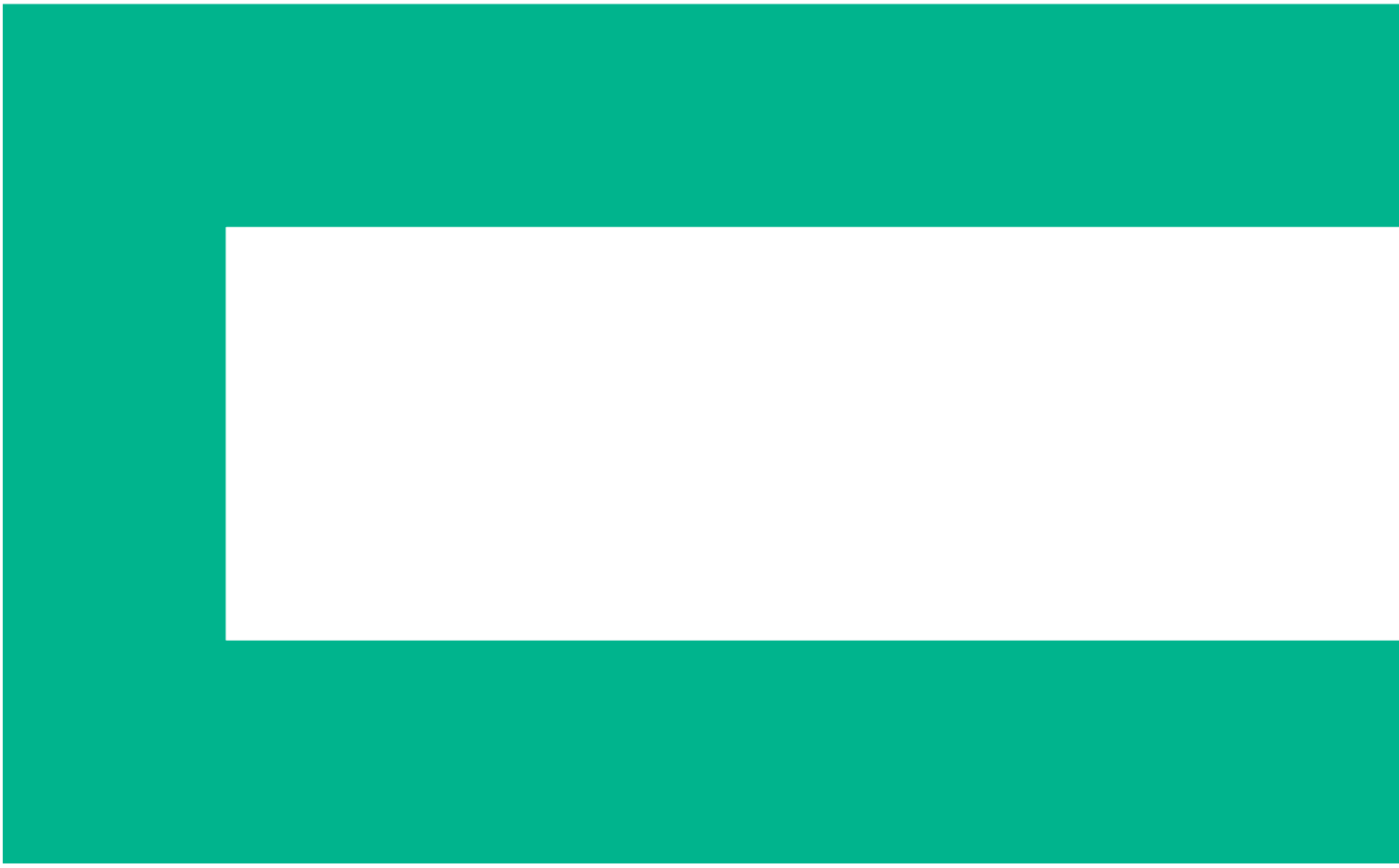


Table of contents

Why reading this guide is time well spent

DevOps is not a new phenomenon, so why is it attracting so much attention now? The answer is in how the market is changing. Forrester reports they are seeing organizations go from four application releases per year in 2010 to 120 releases per year by 2020.[1] This is a 30X increase. As organizations adapt to this pace of business, many struggle with the learning curve of truly understanding what DevOps is all about.

As you read this guide, you'll learn the truth about DevOps and discover the benefits DevOps offers your organization. This introductory guide also explains how the DevOps movement supports the demands for today's IT—characterized by new types of applications, new approaches to IT operations, new demands and uses for insight, and new threats and risks. This quickly evolving business environment also brings with it new customers, buying behaviors, and IT consumption models.

Getting grounded—what is DevOps?

Dev plus Ops is equal to the streamlined flow of work to deliver faster, higher-quality software, with less risk.

DevOps is best defined as a philosophy or ideology. It is not a job title, nor a specific toolset, nor a marketplace. Many of the underlying principles and language of the DevOps philosophy are grounded in a combination of agile software development plus Kaizen, Lean Manufacturing, and Six Sigma methodologies. Even if you are more technically inclined, the relevance of these methodologies will be apparent as you read on.

Embracing a DevOps philosophy means adopting an ideology that fosters a highly productive culture of collaboration between your development (Dev) and operations (Ops) teams. The common goal of DevOps is to remove friction, risk, and other constraints to enable faster, more successful application production rollouts, as often and as rapidly as the business requires.

Most companies that implement DevOps methods today still have a development team and an operations team in place. You can think of DevOps as the processes and individuals that build the bridges between these teams to improve the business and enhance the end-customer experience. Various tools and platforms facilitate the work of DevOps, but they do not define it. Organizations that embrace DevOps might have all IT resources within a traditional data center, all resources in an offsite cloud, or distribute their resources in a hybrid environment.

The DevOps movement is not defined nor led by traditional IT software, hardware, or management vendors. In addition, there are currently no codified rules or manuals for DevOps, only generally accepted guidelines. With that said, adoption and implementation of DevOps vary greatly from one organization to the next.

The learnings of DevOps are primarily evangelized by a passionate grassroots community of IT practitioners, spread across a wide variety of IT disciplines. Most members of the DevOps community have active jobs within various organizations, and they share their learnings in numerous online and in-person forums and gatherings. Depending on the number and maturity of the practitioners in an organization, the benefits of a DevOps implementation are significant, as shown in the next page.

[1] "Better outcomes, faster results. Continuous delivery and the race for better business performance," Forrester Thought Leader Paper commissioned by HP (now Hewlett Packard Enterprise), December 2013.

The story of DevOps. How it began.

In the mid-2000s, numerous conversations began between IT operators and developers. One can only imagine the dynamics of these conversations. Historically, the relationship between these two groups has often been challenging—with each group having specific goals to achieve, and frequently feeling the other was a roadblock. However, discussions soon centered on a core issue that was becoming more prevalent across a growing number and variety of IT organizations: How do we bridge the gap between development and operations for the betterment of the business?

One of the early conversations was between Patrick Dubois and Andrew Shaffer, IT professionals, at the Velocity Conference 2008. The result of that interaction was one of the first communities discussing this issue—the Agile Systems Administration Group. This community and subsequent conversations led Dubois to create the first Velocity Conference (held in Belgium in October 2008), which he named "DevOps Days." Thus, the DevOps name and movement were born.[2]

One of the best synopses of the origins of DevOps is narrated by Damon Edwards, founder of DTO Solutions in "The History of DevOps."[3] Click here to watch an informative video.

The DevOps community was largely underground until 2011, when analysts at Gartner and RedMonk became interested in the topic. Shortly thereafter, enterprise companies began to show growing interest in DevOps. As the DevOps community expanded, it spawned the development of myriad tools—including Vagrant, Rundeck, Puppet, Chef, Juju, Logstash, and many more—designed to help DevOps accomplish its goals. (For more information, refer to the "Tools in your toolbox" section of this document.) And this range of tools the community uses continues to expand and grow. These tools have not yet been consolidated or codified into traditional best practices, but this will most likely occur over time.

Due to the early age of the movement, DevOps is not fully established and still prides itself in being loosely defined. Documenting DevOps practices has begun, but no clear documentation of the community has been consolidated into standard best practices. This means that each operational team applies the DevOps practices according to organizational needs. Doing so allows people to serve many different roles, but can lead to confusion when trying to implement a DevOps culture in a traditional operational environment.

[2] The Agile Admin, "What is DevOps?"

[3] Damon Edwards, "The History of DevOps"

[4] Source: Gene Kim

The term "DevOps" typically refers to the emerging professional movement that advocates a collaborative working relationship between development and IT operations, resulting in the fast flow of planned work (i.e., high deploy rates) while simultaneously increasing the reliability, stability, resilience, and security of the production environment.[4]

Who's taking advantage of DevOps?

DevOps was originally considered a methodology found only in large Web-based companies (e.g., Netflix, Google™, etc.) and startups. Within the DevOps community, these types of organizations are often referred to as "cloud-natives," "born in the cloud," or "unicorns." Increasingly, large traditional enterprises are embracing DevOps principles to become more agile so they can better address market needs and compete. When this report was written, 38 percent of large enterprises have implemented DevOps practices in some way.[5]

Today, the implementation of DevOps can be found in three significantly different groups of adopters, all challenged by the rapidly increasing pace of application releases:

1. **Daily to monthly releases**—Early practitioners of agile software development methodologies who seek to accelerate delivery and deployment to match the rapid pace of software creation. The typical release pace in these organizations ranges from daily to monthly. A typical goal of early practitioners is to eliminate development work-in-progress delays caused by testing and operational release wait times.

2. **Multiple daily releases**—Web-scale practitioners, who adopt DevOps to achieve many micro releases in a day for an application or run-time service. In these environments, the rapid update and release of code is tied to new features that are critical to meeting ever-evolving customer expectations and competitive threats, as well as to the imperative to continuously experiment and improve.

3. **Structured releases**—Traditional IT organizations operating in well-established (and often regulated) industries, such as finance have structured release practices. They are now turning to DevOps methodologies not only for greater speed, but also for improved quality and risk mitigation. In this group of adopters, the current release pace might be considerably slower than in the other two, but demands for more frequent releases or more efficient releases at the appropriate level of quality and security still exist.

All three groups share a common goal: eliminating the time and resource constraints of application delivery, while ensuring the quality, stability, and continuous improvement of their releases and the experience for their end users. The primary differences between the groups are driven by the frequency and scale of application releases, usually based on the industry and nature of the service delivered.

Common misconceptions about the DevOps community

While every organization will implement DevOps in different ways, there are some consistent characteristics and drivers that make this unique group of individuals tick.

1. Is DevOps a job title?
The short answer is, "No." As DevOps involves collaborating between various groups, DevOps employees can work in a number of different areas. In fact, DevOps is embraced and driven by practitioners and evangelists who hold a variety of job titles. Even so, many companies assign "DevOps" titles to employees, such as DevOps Engineer, DevOps Security Engineer, and others.

2. How do DevOps practitioners think?
Evangelists and enablers of DevOps methods are typically senior operations analysts, developers, architects, and most often, senior generalists with a broad understanding across multiple disciplines. Based on the increasing system complexity these individuals have witnessed, they have embraced a continuous improvement mentality. These people have adopted new tools that enable them to optimize processes and systems. At their core, employees facilitating DevOps methodologies are problem solvers who want to find the fastest path to high-quality output.

[5] "Enterprise understanding of DevOps grows, but key benefits prove elusive," Cloud Services World, January 2015

3. How do DevOps practitioners like to work?
These professionals are tool savvy; they use tools and the information the tools provide to enable continuous delivery of system improvements. These people are data-and-information-driven, which allows for data-based improvements in operations and development procedures. In addition, these professionals bring together groups to help achieve standard optimization of processes and systems. IT organizations with high levels of employee satisfaction often include large numbers of DevOps practitioners. Inversely, DevOps practitioners often flee from highly political or bureaucratic command and control environments.

4. What motivates DevOps practitioners?
DevOps practitioners are action takers and problem solvers who want to work with like-minded people to create meaningful value for their organization, as well as for customers, citizens, employees, and the work-life environment. These individuals contribute to successful outcomes by:

- Removing bottlenecks and friction in processes and systems
- Driving toward faster resolution of business problems
- Driving ever-increasing customer satisfaction, quality, and outcomes
- Using data to drive continuous improvement
- Facilitating continuous experimentation and learning
- Enabling collaboration between teams to achieve common goals

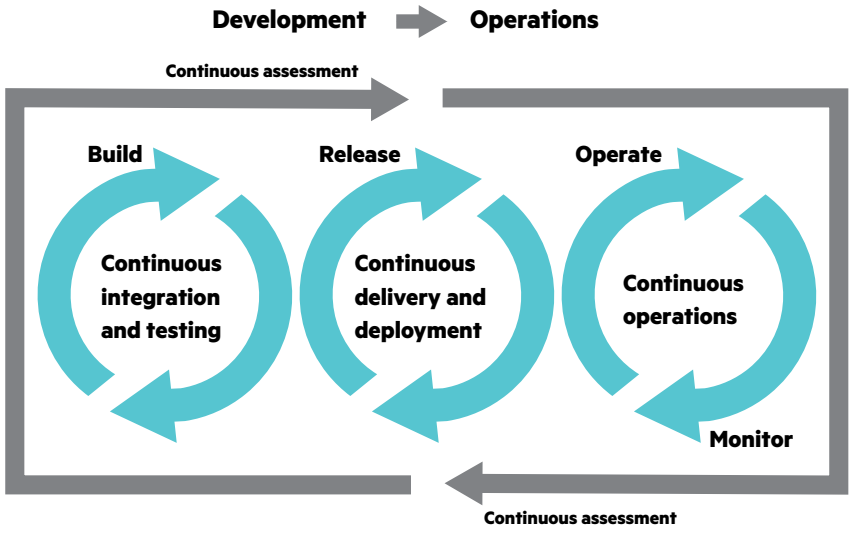
5. What worries DevOps practitioners?
DevOps concerns vary, based on the maturity of the IT environment. For example, in an immature environment, implementing a DevOps culture commonly includes the following concerns:

- Removing the barriers between developers and operations
- Deploying code multiple times a day (such as for Facebook or Amazon) without impacting the user experience
- Working with operations to homogenize the data center hardware
- Standardizing management tools throughout enterprise
- Working with developers to simplify the software architecture
- Minimizing deployment failures and the associated blame that follows

In a more modern IT environment where a DevOps culture is already in place, common concerns include:

- Keeping communications flowing freely between developers and operations
- Increasing the number of developers on a project, while ensuring accurate version control
- Automating tasks
- Identifying software infrastructure load issues to improve applications
- Increasing customer satisfaction

The DevOps cycle



Within a typical IT organization, the scope of DevOps spans four functional areas.

1. Continuous integration and testing
2. Continuous delivery and deployment
3. Continuous operations
4. Continuous assessment

Figure 1: Model of a mature DevOps cycle

Continuous integration and testing

Continuous integration is a development practice that requires developers to integrate code into a shared repository several times a day. Each check in is then verified by an automated build, allowing teams to detect problems early in the cycle. Continuous testing is the practice of automating and integrating unit, functional, and non-functional (performance, security, etc.) tests into the software delivery chain, and automatically executing those tests against each build of the code base. Specific tests and test conditions can be prioritized to ensure the most relevant test cases are given priority, or the entire test suite can be executed with each build. Continuous testing can be utilized as a natural extension of test-driven development practices.

Continuous delivery and deployment

Continuous delivery is a discipline where your goal is to build software so it can be released to production at any time, and it is always production-ready. You achieve continuous delivery by automating the delivery flow from development to production. Key elements of continuous delivery include standardizing infrastructure configuration, and managing configuration details by following the same discipline applied to managing source code. Other elements include automating the delivery of the software completed by the development team to specific environments, and configuring the infrastructure so that automated tests can detect problems. The changes (executables and configurations) can be pushed into increasingly production-like environments to ensure the system will work in production. Continuous deployment is the rare practice of automatically releasing and installing every good code build to production and end users. Continuous deployment is a unique and extreme form of continuous delivery, where every change that passes the automated tests is automatically deployed to production.

Continuous operations

Managing software and hardware changes in a way that is non-disruptive to end users. Processes such as patching and compliance fall under this function. Even though software and servers can be taken offline during planned maintenance, continuous operations enable customers to be serviced by the previous version of the application until the new version has been successfully tested and deployed.

Continuous assessment

Continuously evaluating an application, based on three types of feedback:

- **Feedback loops**—Continuously monitoring the availability, health, and performance of the application, as well as capturing the user experience throughout the lifecycle (i.e., development, quality assurance, staging, and production) and feeding it back to the appropriate teams. Doing so enables continuous optimization and fine-tuning of the application and end-user experience.
- **Planning prioritization**—As the planning team receives feedback, they can continuously assess and prioritize new features, functions, and defect fixing based on business needs and end-user demands.
- **Portfolio investment**—As the planning team receives feedback, they can continuously assess and prioritize investments based on business drivers.

Each functional area applies specific tools, skills, and metrics for success. Gaps between these functional areas often exist where most of the friction, latency, and bottlenecks occur. To close these gaps, much of the DevOps effort focuses on collaboration, data capture, data sharing, and automation, coupled with applying a holistic view and a shared set of metrics across the entire system. Organizations that adopt DevOps principles work with developers and operations teams to determine the needs of the business as a whole, and then use tools that simplify, synchronize, and automate the current infrastructure and services housed on that infrastructure. Let's take a closer look at these four crucial areas.

Expected outcomes from DevOps

Typically, DevOps will enable three key changes in an organization:

 **Velocity and speed of delivery will increase**	The overarching objective is to be responsive to the business demand for new features and functionality. A DevOps team will organize, align, and automate to streamline delivery.
 **Quality and stability will improve**	Standardization and automation reduce the opportunity for manual errors to be injected into application delivery, and the gauntlet of automated testing combine to increase the quality of the final product.
 **Efficiency and capacity will increase**	The combination of changes that DevOps introduces eliminates waste and rework, enabling the same size team to complete more work.

When you think about the desired outcomes of DevOps, consider the goals of a traditional manufacturing enterprise. In such an environment, you can see the entire supply chain—from raw materials and assembly to final end-customer delivery. In fact, as you walk around a manufacturing plant, you can see every step in the process, with each step handing off material and task from one station to the next. During this first-hand experience, you can discover opportunities for driving out waste, friction, and mistakes that lead to rising costs, unnecessary delays, and poor quality. Enter Lean manufacturing, Kaizen, and Six Sigma—targeted methodologies for creating highly efficient production environments.

In a digital enterprise, where the product is based in software or data, the development process is not immediately as transparent as it is in manufacturing. The goal of DevOps is to clear away barriers to success, and to do for the digital enterprise what Lean manufacturing and Kaizen did for traditional manufacturing: eliminate waste, friction, and mistakes to improve the agility, speed, efficiency, and quality of the output, and ultimately enhance the satisfaction, productivity, and experience of the end user.

Today, most IT departments focus primarily on availability, costs, project status, utilization, power, or space consumption, and application performance. In the traditional IT department, the Ops team focuses on the efficient operation of the business systems, where high availability and minimal disruptions are rewarded. However, the Apps team is rewarded for delivering more projects on time and on budget. In simple terms, the Ops team and the Apps team often have opposing goals and objectives.

In a DevOps world, one combined team shares goals and accountability to deliver whatever the business needs. Rapid and frequent changes, stability, and availability are all common goals of the DevOps team. They share responsibility for support as much as they share responsibility for speedy delivery.

Setting up for success

To get started with DevOps, many organizations apply the following approach and framework:

- Continuously import new concepts including tests, experiments, and implementation of new tools or processes
- Get everyone on the same page by:
  - Seeing the entire system through value-stream mapping, document the end-to-end system, and understand the steps, players, and goals
  - Seeing the flow in terms of timeline analysis (identify the bottlenecks) and waste analysis (identify the errors, non-standards, and waste)
- Identify organizational behaviors to improve feedback loops and the data capture and sharing processes that are in place (or need to be)
- Compare projects/experiments against the baseline, look for continuous improvement opportunities

Dispelling common myths

Even though many myths are rooted in truth, the common myths about DevOps are most often rooted in misunderstanding or misinformation. Let's take a few minutes to set the record straight about DevOps.

The roles of developer, operations, and DevOps are blending into one. Actually, these are three distinct areas of work. Developers write code. Operations teams manage the infrastructure that houses the code. DevOps helps optimize the processes and infrastructure for application effectiveness. The number of roles might change over time and be more enabled by software, but all these key skills and expertise will continue to be required.
Operations teams are unnecessary with DevOps because everything will move to the cloud. The goal of DevOps is not to move primarily to the cloud, but to a simpler, standardized infrastructure that can be more easily monitored for problems, deliver application updates more often, and identify system optimization opportunities.
When transforming to a DevOps culture, the only consideration is what new tools are needed. Although introducing a DevOps culture will require new tools, research conducted by 451 Research indicates there are other important factors as well, such as the ones shown in figure 2.[6]
DevOps cannot be used in a large enterprise. The automation lifecycle has different needs at different stages. As such, automation can be incorporated in an enterprise, regardless of the maturity of the IT system in use.
We need to hire DevOps roles. DevOps is not a role; it is a way of doing things. A formal DevOps department is not required to implement a DevOps culture, but you need to adopt the culture to become more agile.
DevOps is a new name for something previously done in IT. In the past, there was not a large agile presence, and there were no tools to help simplify the development, implementation, and automation of applications. DevOps was created to address the increasing need for speed to meet customer demands and reduce the growing complexity of IT systems.
DevOps gives developers the opportunity to do unlimited development. The key to DevOps is increasing the speed of development and the number of developers working on a project at the same time. Development must be based on customer needs and improvement suggestions gathered from infrastructure data monitoring.
Developers will now understand infrastructure, and operations will now understand coding. This is not the goal of DevOps. DevOps is meant to increase communication and collaboration between the teams and make the overall process more agile, while enabling each function to excel within their area of expertise.
DevOps requires you to use certain tools. There are many DevOps tools, and not all tools are used all the time. Tools should be chosen based on business needs, which will vary for each situation.

[6] Source: 451 Research (Analyst) white paper—IT Ops Can Thrive in a DevOps World, August 2015

Source: 451 Research (Analyst) white paper—IT Ops Can Thrive in a DevOps World, August 2015

Figure 2: Factors that influence DevOps implementation

Tools in your toolbox

The DevOps movement was a catalyst for creating new tools that could enable automation across all aspects of development, production, and operation, while also solving other core goals unique to DevOps. Currently more that 80 percent of software development organizations rely on automated tools for IT management and deployment.[7]

Five primary types of tools support DevOps, and each tool in each category has its own specialty. You should choose tools based on your specific needs.

1. Version control	Tool that versions all application code, infrastructure configuration changes, and Big Data database changes, enabling a single version of the truth for the entire IT system. Different version control apps work better than others for different developer platforms.
2. Build and test	Tools that automate common developer tasks including creating executables, running tests, and compiling source code.
3. Configuration management	Tools that track and control changes to the software code base. These tools enable multiple developers to work on the same code, while avoiding version control issues and continuously integrating code.
4. Application deployment	Tools that enable the automation or continuous delivery and deployment of software releases, enabling continuous service delivery.
5. Monitoring	Tools that enable systems engineers to proactively fix problems. These tools either continuously assess and monitor application performance and remediate any performance issues, or they provide visibility into infrastructure capacity, memory, and CPU consumption.

[7] "Predicts 2010: Agile and Cloud Impact Application Development Directions," Gartner

Hewlett Packard Enterprise and DevOps

With more than 75 years of experience in the technology industry, Hewlett Packard Enterprise grounds every customer conversation in innovative products and services. Customers around the world trust Hewlett Packard Enterprise to help transform their infrastructure from one that simply keeps the lights on, to one that creates competitive advantage today and prepares for tomorrow.

Wherever you are in your DevOps journey, you can find a Hewlett Packard Enterprise solution that fits your needs. If you are just beginning your journey, you can attend an HPE Transformation Workshop to develop your roadmap. If you already have a roadmap, you can choose HPE Software Solutions that enable the automation you need for DevOps, as well as the simplified infrastructure that supports scalability.

Hewlett Packard Enterprise IT has been going through its own DevOps cultural transformation. It was based upon several guiding principles that are shared below. Follow the evolution of our IT culture as well. Here is the first piece.

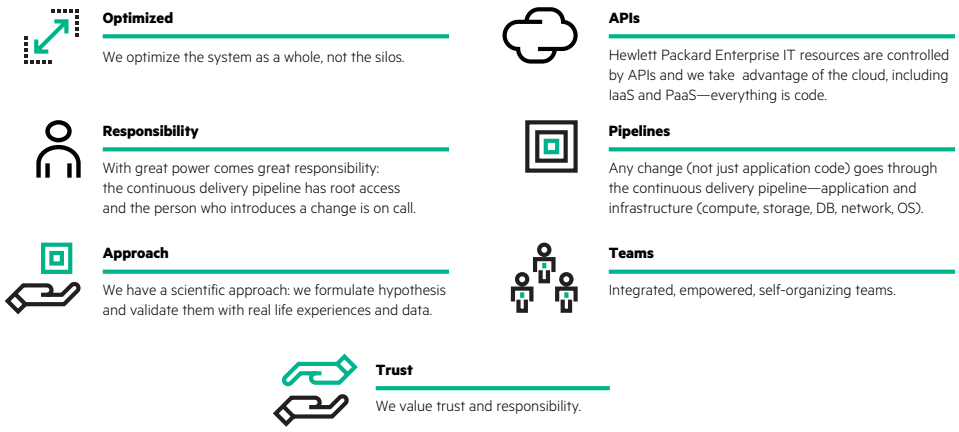


Figure 3: Hewlett Packard Enterprise IT DevOps Manifesto

Glossary

Agile development: Promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, as well as encourages rapid and flexible response to change. Agile development incorporates iteration and continuous feedback to successively refine and deliver a software system. It involves continuous planning, testing, integration, and other forms of continuous evolution of both the project and software. Forms of Agile development include Agile Scrum, Lean development, and Extreme Programming.

Agile Scrum: With Scrum methodology, the product owner works closely with the team to identify and prioritize system functionality in form of a product backlog. The product backlog consists of features, bug fixes, and non-functional requirements—whatever needs to be done to successfully deliver a working software system. With priorities driven by the Product Owner, cross-functional teams estimate and commit to delivering potentially shippable increments of software during successive sprints, typically lasting 30 days. Once a Sprint's Product Backlog is committed, no additional functionality can be added to the sprint except by the team. Once a sprint has been delivered, the product backlog is analyzed and reprioritized, if necessary, and the next set of functionality is selected for the next Sprint.

Canary release: A technique for reducing the risk of introducing a new software version in production by slowly rolling out the change to a small subset of users before rolling it out to the entire infrastructure and making it generally available. (Source: Martin Fowler)

Canary testing: In software testing, a canary or canary test is a push of programming code changes to a small number of end users who have not volunteered to test anything. The goal of a canary test is to make sure code changes are transparent and work in a real-world environment. (Source: WhatIs.com)

Continuous assessment: The function of continuously assessing the application through:

1. **Feedback loops:** Continuously monitoring the availability, health, and performance of the application, while also capturing the end-user experience throughout the lifecycle (development, quality assurance, staging, and production), and feeding the information to the appropriate teams to continuously optimize and fine-tune the application and end-user experience.
2. **Planning prioritization:** As feedback returns to the planning team, continuously assessing and prioritizing new features, functions, and defect fixing based on business needs; doing so ensures that application changes are prioritized and delivered to end users.
3. **Portfolio investment:** As feedback returns to the planning team, continuously assessing and prioritizing investments based on business drivers (includes application portfolio management).

Continuous delivery: A software development discipline where software is built so it can be released to production at any time. You achieve continuous delivery by continuously integrating the software created by the development team, building executables, and running automated tests on those executables to detect problems. You push the executables into increasingly production-like environments to ensure the software will work in production. Continuous delivery is sometimes confused with continuous deployment, but they are actually two distinct practices. (Source: Martin Fowler, Continuous Delivery)

Continuous deployment: The practice of automatically releasing and installing every good code build to production and end users. Continuous deployment is the next step in the continuous delivery process. (Source: Jez Humble, Continuous Delivery)

Continuous integration: A development practice that requires developers to integrate code into a shared repository several times a day. An automated build verifies each check-in and allows teams to detect problems early. (Sources: thoughtworks.com/continuous-integration, and wikiwand.com/en/Continuous_integration)

Continuous operations: Managing software and hardware changes in a way that is non-disruptive to end users. Processes such as patching and compliance fall under this function. Even though software and servers can be taken offline during planned maintenance, continuous operations enable customers to be serviced by the previous version of the application until the new version has been successfully tested and deployed. Gartner defines continuous operations as, "those characteristics of a data-processing system that reduce or eliminate the need for planned downtime, such as scheduled maintenance. One element of 24-hours-a-day, seven-days-a-week operation."

Continuous testing: The practice of automating and integrating unit, functional, and non-functional (performance, security, etc.) tests into the software delivery chain and automatically executing those tests against each build of the code base. Specific tests and test conditions can be prioritized to ensure the most relevant test cases are given priority, or the entire test suite can be executed with each build. Continuous testing can be utilized as a natural extension of test-driven development practices.

Deployment pipeline: Orchestrating a build through a series of quality gates, with automated or manual approval processes at each stage, culminating with deployment into production. Also referred to as CD pipeline, delivery pipeline, build pipeline, and deployment production line. (Source: Jenkins: The Definitive Guide, by John Smart)

DevOps: Typically refers to an emerging professional movement that advocates a collaborative working relationship between development and IT operations teams, resulting in the rapid flow of planned work (i.e., high deploy rates), while simultaneously increasing the reliability, stability, resilience, and security of the production environment. (Source: Gene Kim)

Gemba Walk: The action of going to see the actual process, understand the work, ask questions, and learn. Gemba Walk is one of the fundamental components of the Lean management philosophy. (Source: Gemba Walk)

Kanban: Scheduling system for lean and just-in-time (JIT) production. Kanban controls the logistical chain from a production point of view, but it is not an inventory control system. Kanban was developed by Taiichi Ohno (Toyota) to improve and maintain a high level of production. (Source: Toyota)

Kaizen: Japanese for "good change." Kaizen has been applied in health care, psychotherapy, life coaching, government, banking, and other industries. When used in the business sense and applied to the workplace, Kaizen refers to activities that continually improve all functions, and Kaizen involves all employees from the CEO to the assembly line workers. Kaizen also applies to processes, such as purchasing and logistics that cross organizational boundaries into the supply chain. By improving standardized activities and processes, Kaizen aims to eliminate waste (see Lean manufacturing). (Source: Kaizen.com)

Latency: A measure of the time delay experienced by a system.

Lead time: The latency (delay) between the initiation and execution of a process, such as the lead time measure of the time delay experienced by a system, or the time between the placement of an order and delivery of the item from the manufacturer. For example, the lead time on a new car can be anywhere from two weeks to six months. Lead time reduction is an important part of Lean Manufacturing. (Source: Investopedia)

Lean development: An iterative Agile methodology that focuses a team on delivering value to the customer and on the efficiency of the value stream (i.e., the mechanisms that deliver that value). Lean methodology eliminates waste through such practices as selecting only the truly valuable features for a system, prioritizing the features elected, and delivering the features in small batches. Lean development emphasizes the speed and efficiency of the development workflow, and relies on rapid and reliable feedback between programmers and customers.

Queue time: The time between sub-processes where an item is moved around or waits for someone to work on it. Also known as "Waiting & Transportation Time" or "Inventory/ Transportation Time." (Source: Velaction)

Value-stream mapping: A Lean management method for analyzing the current state of a product or server, and then designing a series of events that move the product or service from its beginning through to the customer. This methodology can be applied to practically any value chain. (Source: iSixSigma)

Wait time: Waiting is one of the seven wastes that most people readily recognize. Eliminating time spent waiting has been a focus of manufacturing improvement activities since the beginning of the industrial age. The motivation to eliminate wait time has been the driving force behind many of the other wastes—defects, overproduction, transportation, inventory, motion, and processing.

For example, to eliminate any chance of an employee waiting, large queues of work-in-progress (WIP) would be accumulated throughout the production process. When people think of wait time, most picture a worker in front of a machine waiting for material to arrive or for the machine to cycle. This is one of the common types of wait time, but there are more subtle instances that are every bit as costly. Wait times are a major challenge in supply chain operations, as companies often wait days or weeks to replenish raw materials. Wait times also occur in many administrative functions, including the delays in the flow of information or approvals from one department to another, or the delay of waiting for an open position to be filled. (Source: Lean Genie)

WIP: Work-in-progress (or work in process) is the amount of unfinished product in development and test. Typically, the goal is to reduce work in progress, as it is investment that is not yet providing a return and requires continued investment until it becomes productive.

Resources

To learn more about DevOps, we recommend reading the following publications:

- "The Phoenix Project," by Gene Kim
- "The Goal," by Eliyaho M Goldratt
- "Continuous Delivery," by Jez Humboldt
- "The Machine that Changed the World," by James Womack

If you prefer online reading, you can refer to the following blogs, communities, and podcasts:

- continuousdelivery.com
- bmc.com/blogs
- devops.com
- arresteddevops.com
- devopscafe.org
- community.dev.hp.com

Open source tools for automation, configuration management, and continuous integration

Jenkins: An open source continuous integration server. Jenkins features numerous plug-ins that support project building and testing. Jenkins monitors a version control system by maintaining a build system, monitoring it for changes, and providing appropriate notifications of those changes. Typically, Jenkins is paired with a build tool, such as Maven.

Chef: An open source configuration management tool by Opscode that manages server configuration and deploys applications. Chef helps to automate configuration, deployment, and scaling of servers and applications, regardless of whether the server or application is in the cloud, onsite, or in a hybrid environment. Chef runs on the Linux® operating system with agents to manage other platforms. Chef uses a Ruby-like scripting language called recipes. In addition to the open source versions of Chef, a commercial offering is also available. Chef is often used as a continuous delivery tool.

Puppet: An open source configuration management tool by Puppet Labs, the Puppet automation platform manages server configuration and deployment of applications. Following the client or server model, Puppet helps to automate, deploy, and scale up applications in the cloud or on site. Puppet runs on the Linux operating system with agents to manage other platforms. Puppet uses a proprietary scripting language based on declarative models known as manifests. In addition to the open source versions of Puppet, a commercial offering is also available. Puppet is often used as a continuous delivery tool.

Docker: An application or micro-service container developed by Docker, Inc., Docker automates application deployment inside software containers. This tool helps package an application and its dependencies as a virtual container. Docker is written in the Go programming language and integrates with CODAR. In much the same way that VMware® virtualizes hardware, Docker virtualizes the underlying services provided by the operating system. Where VMware can support multiple operating systems on a physical server, Docker can support multiple applications or services on an operating system, with no interference or conflicts.

Learn more at
hp.com/go/devops

Sign up for updates

★ Rate this document

Hewlett Packard Enterprise

4AA4-3696ENW, November 2015, Rev. 1