FOUR STEPS TO CREATING YOUR OWN DEVOPS SUCCESS

A guide to understanding DevOps implementation from an enterprise perspective

ConnectALL

TABLE OF CONTENTS

INTRODUCTION

In the digital era, many organizations are reimagining their business and operating models based on digital capabilities, to gain competitive advantage and differentiation. The speed of technological advancements is creating a division between winners and losers.

Enterprises cannot meet digital business demands with the traditional approach; it needs enterprises to be responsive. As your business moves to become digital, you need to be able to respond to the threats and opportunities of the digital economy.

You need to recognize the need to innovate more, manage uncertainties better, and most importantly establish more agility to meet the needs of the business. Agile and DevOps are the most common capabilities that organizations start to develop, and it is a great place to start because it teaches many of the principles that enable an organization to be responsive.

Figure 1: Transitions are About Culture

But transitioning from traditional to agile approach requires creating a culture that embraces agile principles to become successful.

The first issue faced by the application organization is to get a 'we' environment right within their organization. This involves establishing the team that removes the boundaries between 'development' and 'testing' teams. There is just one team, no hierarchy, where developers consider themselves more important than testers.

> **"Speed is useful only if you are running in the right direction."**
> – Joel A Barker, Business Futurist

The 'we' culture must expand beyond applications. Agile is not just about 'doing development differently,' but about doing business differently. This means blurring the line between the business and the development team by including one or more people from the business on the team.

The 'we' culture also expands to include operations team. This is known as DevOps and involves integrating the deployment and production support roles into the development team to form a cross-functional DevOps team. Production issues become development issues as well.

THE UNPLANNED BIRTH OF DEVOPS

While many development teams are applying agile software development practices to projects, the development and operations teams on a given project are not always aligned.

Generally, two organizations are responsible for delivering software to users: development and operations. Development teams are measured on the features they deliver to users, while operations teams are measured by the stability of the system after it has been delivered to users. These are often competing interests in organizations because the development team is not incentivized to ensure the stability of a system once it delivers it to the operations team.

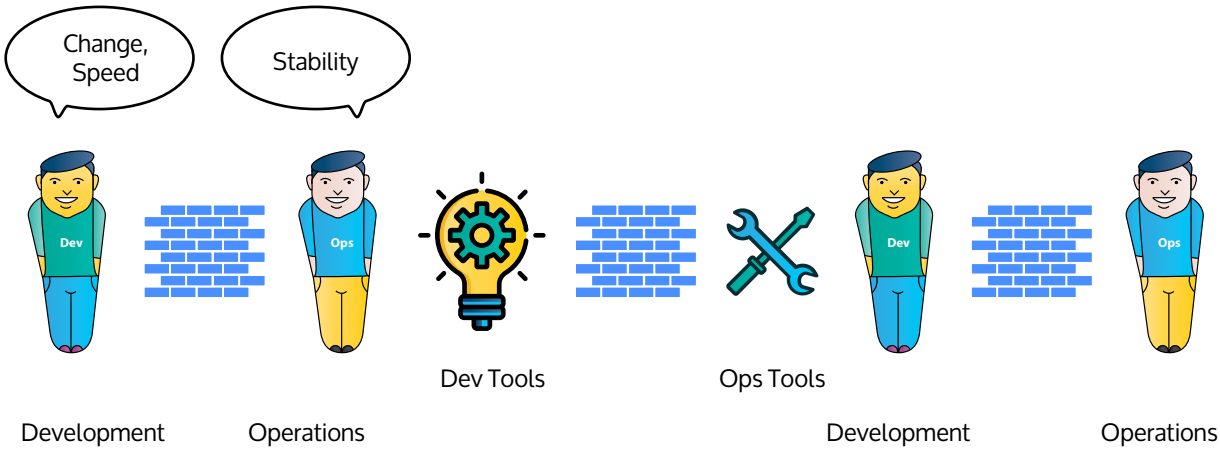

Figure 2

Likewise, the operations team is not given the appropriate incentives to care about the frequency of releasing new features to users, because its primary responsibility is ensuring the uptime of the system. It was this environment that led to the birth of DevOps. It was this environment that led to the birth of DevOps.

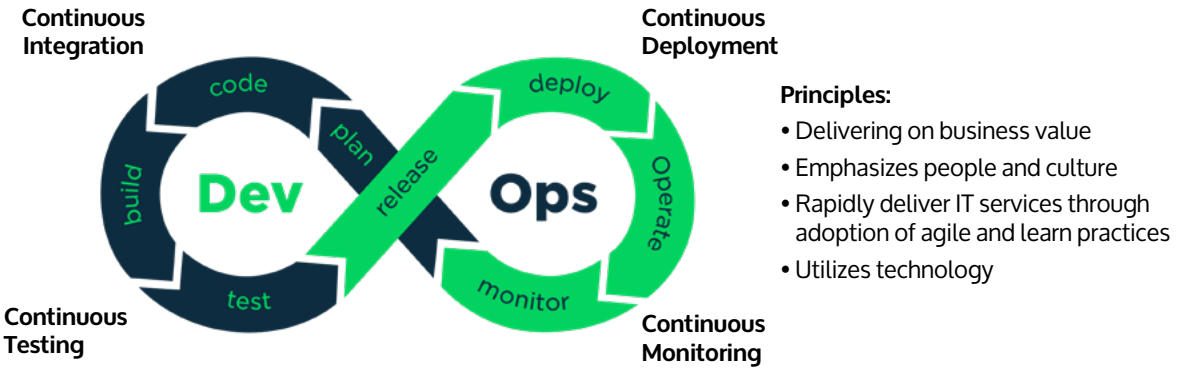

Figure 3: DevOps is a business-driven approach to rapidly deliver solution through collaboration

DevOps is a response to this continual frustration of lack of collaboration and communication between development and operations on software projects that increase the time and labor involved in delivering and maintaining software systems. It seeks to improve and increase this communication and collaboration by breaking down the barriers and silos that often exist in traditional organizations. DevOps does not have a concrete set of mandates or standards, or a known framework — like ITIL or CMMI — making it subject to a more liberal interpretation. For many, it is elusive enough to make it difficult to know where to begin and how to measure success. DevOps is a philosophy (no rules or manuals); it is a change in IT culture that is based on four principles (Figure 3). Continuous Integration (CI), Continuous Testing (CT), Continuous Delivery (CD), and Continuous Monitoring (CM) — are key components of a DevOps Process.

WHAT IS CONTINUOUS DELIVERY?

Continuous Delivery is an overarching philosophy which makes use of Continuous Integration and Continuous Deployment. Continuous Delivery is a methodology designed to increase agility and velocity through the deployment process. Its success depends on breaking down the silos and lack of communication between development, QA, and operations, so that the entire software lifecycle runs in a continuous and business-aligned fashion.

In a typical continuous delivery methodology, automation of processes and workflows is important in development, testing, and deployment of applications, but a well-devised continuous delivery plan is not something that can be invented overnight. It requires a strategic approach with specific goals. Continuous Integration is a practice of building, integrating, testing, and deploying software on a repeatable and frequent basis. The objective for Continuous Integration is to reduce development risk and cost while improving software quality. Continuous Deployment is the practice of automatically deploying release software into a production. Continuous Testing is about building test automation into your pipeline that is never finished, and that requires constant fine-tuning. Which mean there's always a need to balance available resource and the pressure for greater speed with the right standard of quality and an acceptable level of risk.

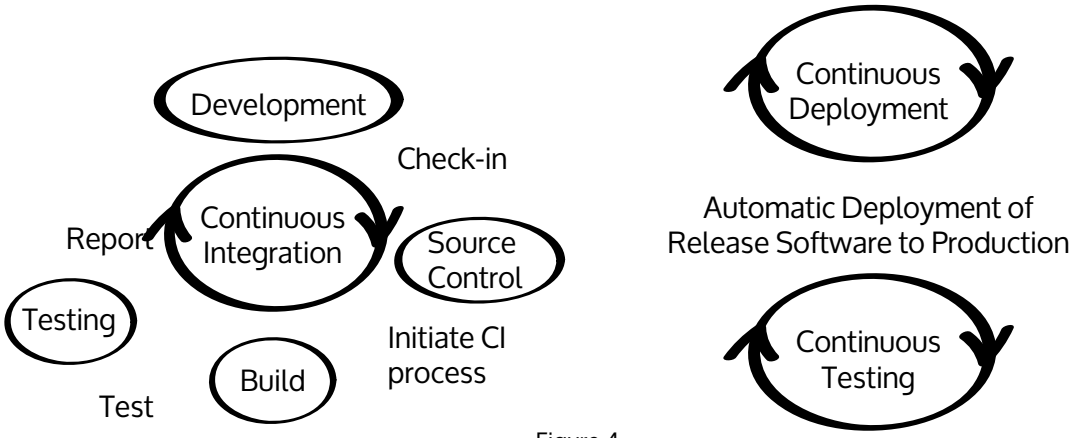

Figure 4

Adopting true DevOps and Continuous Delivery practices is a difficult and complex undertaking for the typical large enterprise. This book lays down some important yet basic points that enables you to learn what modern organizations need to do to make the adoption of DevOps work, to sustain its effectiveness, and to ensure that it is continually improving.

LET'S TAKE A LOOK AT THE FOUR IMPORTANT STEPS TO MAKE A GOOD START TO DEVOPS TRANSFORMATION.

STEP 1: DEFINE WHAT DEVOPS MEANS TO YOUR ORGANIZATION

The benefits of DevOps are clear — high-performance, faster deployment, and quicker response to crisis. Businesses today are either getting started with DevOps or have applied it in pockets but find it difficult to scale up to an enterprise-wide implementation. The most important step is to define what DevOps means to your organization.

Get everyone on the same page

DevOps is more than just pushing code to production. It involves cross-silos cooperation and is not easy. In a large organization, everyone gets excited about DevOps transformation, but each person from the grassroots has a different understanding and you end up with is something that doesn't enable you to release code more frequently.

Communication is key to getting everyone on the same page early on. As Gary Gruver, an experienced software executive, has stated in his book *Starting and Scaling DevOps in the Enterprise,* how starting the DevOps process and journey is like having five blind men build an elephant. Solution: He suggests that organizations use process maps to communicate with all their teams to give everyone a common understanding of what the issues are and introduce everybody at the same time to DevOps concepts from both front and back end.

Value stream mapping has proven to be a good methodology to not only get everyone together and understand problems, but also to identify areas of improvement, which is the main aim of DevOps. It allows you to assemble everyone involved with a workflow into the same room at the same time, to clarify their roles in this product delivery process and identify bottlenecks, friction points and handoff concerns. Value stream mapping reveals steps in development, test, release and operations support that waste time or are needlessly complicated.

It starts with the product person or team and moves through the development life cycle, QA testing, release and operations, ultimately looking at how the IT team manages and monitors a product/feature after release. And it doesn't end there — it goes all the way through deployment till the customer sees value.

Start small and focus on all things continuous

Identify a person such as director of operations, chief architect, or director of development, who can be the champion of your DevOps project. Start with a pilot project, set goals, and achieve the results, thus showing to the rest of the teams that you can scale up slowly.

A good practice is to pick a regular process in place and improve on it — developing an inspect-and-adapt cycle is key to DevOps success. DevOps gurus have always emphasized that the shift to DevOps should start small, but neither too small nor too big. You need to know what the business problem is alongside the anticipated benefits, therefore measuring every factor that contributes to DevOps success.

DevOps promises an array of automation. This can't be achieved in a single attempt or overnight. Enterprises have to focus efforts on continuous integration, following it with continuous delivery and eventually continuous deployment. Your continuous delivery pipeline is an opportunity for you to inject automated quality testing into your process. But don't stop at continuous delivery. Continuous improvement is also another step to doing all things continuous. It is imperative that teams spend some time every day deliberating about small things that can be improved and over time these small, daily changes can earn profound results.

Be consistent on the overall approach

As you move from one small project to the other, you will notice that consistency is not just a feature of DevOps for a software delivery process from development to product, but it requires a consistent approach. As stated in the early days of its inception, DevOps is all about consistency more than automation.

Agree on DevOps objectives and scope

With DevOps, you can't expect dozens of releases every single day. It takes time. The important step is to make sure everyone in the organization agrees on the overall objective and scope — accelerate delivery and manage risk. If you are a large organization,

your focus will be on discipline, checks, and balances, as you would need to consider the legacy technology in comparison to smaller companies that need not worry about all the nitty-gritty. Whether large or small, all organizations need to work toward producing a high-quality software product that can be delivered to the end-user.

STEP 2: ESTABLISH A UNIFIED TEAM — AN INTERSECTION OF FUNCTIONS

People are the main ingredient in a successful DevOps initiative. Having the right staff on the DevOps team will make or break the initiative. It is critical to designate people who are willing and able, and dedicated from the start to the DevOps initiative: willing, in

that they understand the initiative requires potentially a completely new way of working, and able, in that they are capable of doing work in a new way — one that is different from the way they have been doing their current work.

Build a required skills list before selecting or assigning people to the initiative. Note that in some situations: you will have to hire from outside and some existing employees may never be a fit for this type of initiative.

Build resilient systems

Building resilient systems allows teams to collaborate better and work together, knowing how to respond to failure. When an enterprise begins DevOps, it should scrap non-scalable and traditional change management approaches to tackling failure. DevOps is a response to increasing process complexities and the best way to handle complexities is to build resilience. Instead of trying to cut out failure, enterprises have to learn to become resilient to failure. This should begin with the top leadership and trickle down.

Figure 5

Emphasize the importance of collaboration

To make a DevOps transformation successful, enterprises need to get everyone on the same page — from individual contributors to the top management. This begins with making sure that there is motivation, incentive, and most importantly the right mindset and culture for cross-team collaboration. We live in a service economy and everything today is about delivering top-quality applications to customers, emphasizing the crucial need to work together and therefore continuous delivery. Organizations should also help teams visualize an end-to-end workflow that gives them a breakdown of what they need to work on next, how much work is in progress, where the bottlenecks are, and more. This acts as a catalyst for a relationship-building environment as everyone knows who is working on what part of a service at a given time and how much each person is contributing to the overall success.

Create an environment of experimentation and learning

While smaller organizations or start-ups may find it a lot more organic to create an environment of experimentation and learning, it can be adopted by a larger organization with some careful steps. Gene Kim, co-author of The DevOps Handbook, explains how one of the key principles of DevOps is fostering continual experimentation, taking risks and learning from them, and understanding that repetition and practice are prerequisite to mastery. This enables you to improve and also tread onto paths that are untraversed, giving team members opportunities to share their learning with stakeholders, both upstream and downstream, in a process. There is no checklist to what tools you should acquire or where you should begin — you make a plan as to where you want to go, experiment and learn on the go. In most DevOps implementations, things fall in place as you move ahead with resilience and adaptivity.

Break away from a culture of blame

One of the key steps to getting rid of silos is to make sure that information about failure or a problem is passed on in a non-blameful manner across a team. DevOps success hinges a lot on people working effectively together, and enterprises have to get rid of environments that breed on blame culture. The need is to make sure that the teams understand the problem and its solution and not focus on who caused it.

STEP 3: FIND YOUR BOTTLENECK

You can identify and remove bottlenecks only if you can see them. Identifying bottlenecks is a continual process within DevOps practice. A lot happens before development and after operations through building, maintaining, and delivering value. Tracing the whole workflow, automating it from ideation to production and collecting and consolidating data throughout the DevOps process is key.

It's true that CI/CD, and release automation smoothens out build and deployment of high-quality software products with faster delivery. However, during DevOps implementation, if your flow time is still long, unpredictable and unmeasurable, the bottleneck would have moved further upstream. If there are manual handoffs between development and operations teams, then there are bound to be bottlenecks even outside the DevOps process.

Here are some steps that you can follow and repeat:

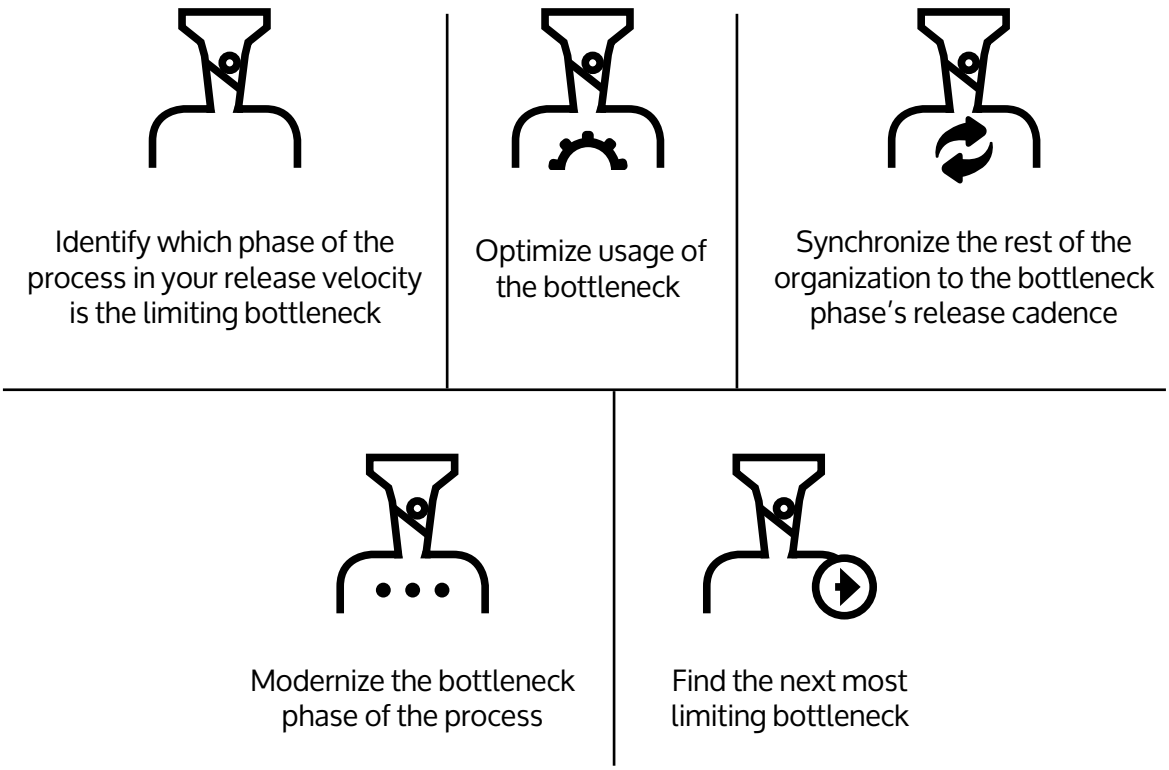

Figure 6

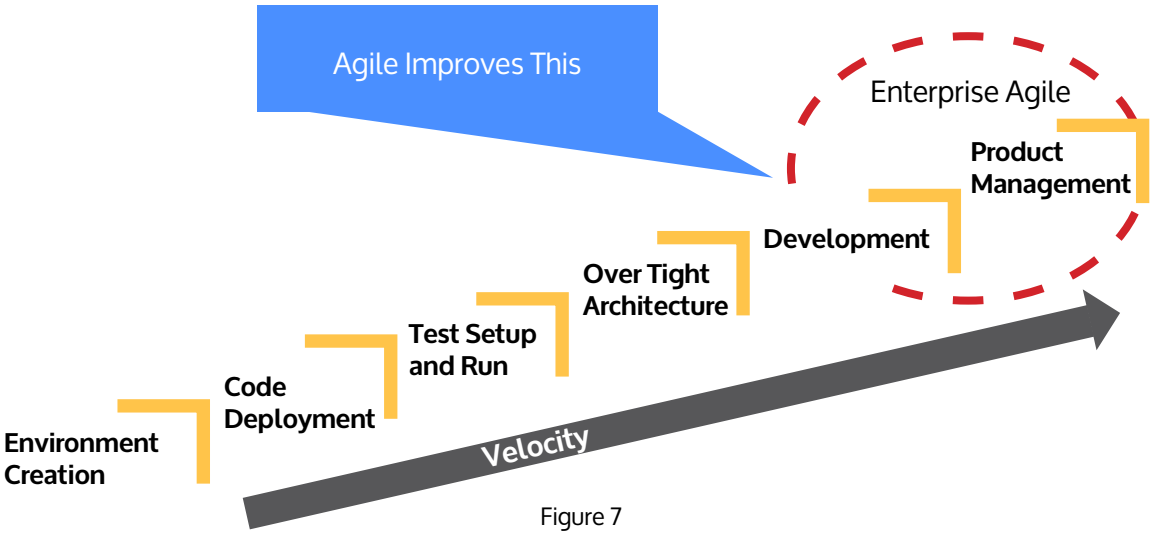

Figure 7

Source: Gene Kim @RealGeneKim

You can also avoid bottlenecks by answering these key questions:

- Are you monitoring and measuring the right metrics?
- Have you understood how value flows across the process?
- Do you have access to real-time metrics across the process?
- Can you monitor end-to-end traceability?
- Do you have access to performance reports?

If you encounter unsure answers, then you need to relook at your software delivery value stream and optimize it. The good news is that you can have visibility to the end-to-end value stream by unifying teams, processes, applications, and data, for improved collaboration, monitoring, tracking, and reporting, which brings us to the next important step: integration of tools and applications in DevOps.

STEP 4: CONNECT YOUR TOOLS AND ENABLE SYNCHRONIZATION

The need to sync tools, scripts, and data for rapid delivery, better collaboration for seamless flow of data across toolchains is imperative now than ever. With the rising demand of everything continuous, software teams are under the pressure to automate the entire software lifecycle. It is more than understood that manual setting up of text, build, and deployment is time-consuming and a burden for large enterprises.

Multi-tool integration is key to achieving continuous delivery, which in turn helps DevOps. When all the tools participating in DevOps sync with each other, it enables enterprises to implement DevOps through the life cycle of continuous planning, integration, testing, monitoring, delivery and feedback.

Once connected to an integrated tool system, developers, testers, and managers get complete visibility of the tool records, their relationships from within their own tools, and end-to-end traceability through the software delivery lifecycle, including defects. It closes the gap in communication between the Dev and Ops teams to a large extent. It also enables the teams to give real-time updates to customers, therefore improving product performance. Organizations are on the constant look out for integration methodologies that are scalable, reliable, and easy to maintain, without manual involvement or programming.

Such a high-quality enterprise application integration product is ConnectALL. It eliminates errors and delays in DevOps tools and applications by making workflow complexities invisible. It connects the systems already in use, automatically. Each DevOps team works with the best system for their needs and ConnectALL makes all the systems work together, without manual operations. It is infinitely configurable to work the way the company's staff works, with their systems, and their processes.

CONCLUSION

DevOps transformation and implementation of continuous delivery practices reduces time-to-market significantly, enhances quality of applications, value of services, and reduces development costs. However, implementation of either DevOps or Continuous Delivery methodologies is not a 'do once and forever' kind of task. It requires some amount of process changes that directly involve developers, testers, and operations teams.

While DevOps transformations begin with process reviews and automation of the deployment pipeline, the more viable approach is to get everyone working together, addressing the larger problem — development and operations teams have different approaches, concerns, and performance metrics and do not understand each other's processes.

Deploying a set of tools to users and expecting things to fall in place will not serve the purpose. The need is to create connectivity between people, processes, data, applications, and teams in the DevOps toolchain, and build a collaborative environment. Application integration enables continuous delivery and therefore sets DevOps initiatives in the right path, saving considerable amount of time and effort, reducing manual execution, monitoring, and reporting during DevOps implementation.

ABOUT CONNECTALL

ConnectALL® powers businesses in achieving higher agility and increased velocity. Teams from software development and delivery, IT and business units across large and small enterprises worldwide use ConnectALL's integration platform to unify people, processes, applications and tools from multiple ALM and DevOps providers, such as Atlassian, Micro Focus, Microsoft, IBM, Salesforce, BMC, ServiceNow, and more. Designed to break down barriers to continuous delivery, ConnectALL helps companies rapidly create business value by bringing software innovation to market faster and increasing productivity through cross-team collaboration.

For more information, visit

www.connectall.com

+1 800 913 7457

sales@connectall.com