

The  
**DevOps**  
**Handbook**

HOW TO CREATE WORLD-CLASS  
AGILITY, RELIABILITY, & SECURITY  
IN TECHNOLOGY ORGANIZATIONS



The  
**DevOps**  
**Handbook**

**HOW TO CREATE WORLD-CLASS  
AGILITY, RELIABILITY, & SECURITY  
IN TECHNOLOGY ORGANIZATIONS**

**GENE KIM, JEZ HUMBLE, PATRICK DEBOIS,  
& JOHN WILLIS**

**FEATURING NEW FOREWORD AND UPDATED  
MATERIALS FROM NICOLE FORSGREN, PHD**

**ORIGINAL FOREWORD BY JOHN ALLSPAW**

IT Revolution  
Portland, Oregon































## FOREWORD TO THE SECOND EDITION

**I**t has been five years since the first edition of *The DevOps Handbook* was released. While so much has changed, so much has also remained the same. Some of our tools and technologies are no longer in vogue or might not exist, but that shouldn't detract any readers. Although the technology landscape has shifted, the underlying principles presented in this book remain as relevant as ever.

In fact, the need for DevOps is even greater today, as organizations need to deliver value quickly, safely, securely, and reliably to their customers and users. To do this, they need to transform their internal processes and leverage technology to deliver value—using DevOps practices. This is true for organizations around the world and across all industry verticals.

Over the past several years, I've led research in the annual *State of DevOps Reports* (first with Puppet and later with DORA and Google), with co-authors Jez Humble and Gene Kim. The research has confirmed that many of the practices described in this book lead to improved outcomes like speed and stability of software releases; reductions in deployment pain and burnout of our engineers; and contributions to organizational performance, including profitability, productivity, customer satisfaction, effectiveness, and efficiency.

For the second edition of *The DevOps Handbook*, we have refreshed the text with updated data based on the latest research and best practices, and included new case studies to share even more stories about what transformations look like “on the ground.” Thanks for joining us on this journey of continuous improvement.

—Nicole Forsgren, PhD  
Partner at Microsoft Research  
2021



## FOREWORD TO THE FIRST EDITION

**I**n the past, many fields of engineering have experienced a sort of notable evolution, continually “leveling up” its understanding of its own work. While there are university curriculums and professional support organizations situated within specific disciplines of engineering (civil, mechanical, electrical, nuclear, etc.), the fact is, modern society needs all forms of engineering to recognize the benefits of and work in a multidisciplinary way.

Think about the design of a high-performance vehicle. Where does the work of a mechanical engineer end and the work of an electrical engineer begin? Where (and how, and when) should someone with domain knowledge of aerodynamics (who certainly would have well-formed opinions on the shape, size, and placement of windows) collaborate with an expert in passenger ergonomics? What about the chemical influences of fuel mixture and oil on the materials of the engine and transmission over the lifetime of the vehicle? There are other questions we can ask about the design of an automobile, but the end result is the same: success in modern technical endeavors absolutely requires multiple perspectives and expertise to collaborate.

In order for a field or discipline to progress and mature, it needs to reach a point where it can thoughtfully reflect on its origins, seek out a diverse set of perspectives on those reflections, and place that synthesis into a context that is useful for how the community pictures the future.

This book represents such a synthesis and should be seen as a seminal collection of perspectives on the (I will argue, still emerging and quickly evolving) field of software engineering and operations.

No matter what industry you are in, or what product or service your organization provides, this way of thinking is paramount and necessary for survival for every business and technology leader.

—John Allspaw, CTO, Etsy  
Brooklyn, NY, August 2016





























































## PART I: INTRODUCTION

**I**n Part I of *The DevOps Handbook*, we will explore how the convergence of several important movements in management and technology set the stage for the DevOps movement. We describe value streams, how DevOps is the result of applying Lean principles to the technology value stream, and the Three Ways: Flow, Feedback, and Continual Learning and Experimentation.

Primary focuses within these chapters include:

- The principles of Flow, which accelerate the delivery of work from Development to Operations to our customers.
- The principles of Feedback, which enable us to create ever-safer systems of work.
- The principles of Continual Learning and Experimentation, which foster a high-trust culture and a scientific approach to organizational improvement and risk-taking as part of our daily work.

### A Brief History

DevOps and its resulting technical, architectural, and cultural practices represent a convergence of many philosophical and management movements. While many organizations have developed these principles independently, understanding that DevOps resulted from a broad stroke of movements, a phenomenon described by John Willis (one of the co-authors of this book) as the “convergence of Dev and Ops,” shows an amazing progression of thinking and improbable connections. There are decades of lessons learned from manufacturing, high-reliability organizations, high-trust management models, and others that have brought us to the DevOps practices we know today.

DevOps is the outcome of applying the most trusted principles from the domain of physical manufacturing and leadership to the IT value stream. DevOps





emphasizing the desire for small batch sizes—incremental releases instead of large, big-bang releases. Other principles emphasized the need for small, self-motivated teams working in a high-trust management model.

Agile is credited for dramatically increasing the productivity and responsiveness of many development organizations. And interestingly, many of the key moments in DevOps history also occurred within the Agile community or at Agile conferences, as described below.

### Agile Infrastructure and Velocity Movement

At the 2008 Agile conference in Toronto, Canada, Patrick Debois and Andrew Shafer held a “birds of a feather” session on applying Agile principles to infrastructure as opposed to application code. (In its early days, this was referred to as “Agile system administration.”) Although they were the only people who showed up, they rapidly gained a following of like-minded thinkers, including co-author John Willis.

#### CONTINUOUS LEARNING

Around the same time, a few academics started studying system administrators, how they applied engineering principles to their work, and how it impacted performance. The leading experts included a group from IBM Research, with ethnographies led by Dr. Eben Haber, Dr. Eser Kandogan, and Dr. Paul Maglio. This was extended to include behavioral quantitative studies led by co-author Dr. Nicole Forsgren in 2007–2009. Nicole went on to lead the research in the 2014–2019 *State of DevOps Reports*, the industry-standard research into practices and capabilities that drive software delivery and performance; these were published by Puppet and DORA.

Later, at the 2009 Velocity conference, John Allspaw and Paul Hammond gave the seminal “10 Deploys per Day: Dev and Ops Cooperation at Flickr” presentation, where they described how they created shared goals between Dev and Ops and used continuous integration practices to make deployment part of everyone’s daily work. According to firsthand accounts, everyone attending the presentation immediately knew they were in the presence of something profound and of historic significance.



## AGILE, CONTINUOUS DELIVERY, AND THE THREE WAYS

**I**n this chapter, we present an introduction to the underpinning theory of Lean Manufacturing, as well as the Three Ways—the principles from which the observed DevOps behaviors can be derived.

Our focus here is primarily on theory and principles, describing many decades of lessons learned from manufacturing, high-reliability organizations, high-trust management models, and others, from which DevOps practices have been derived. The resulting concrete principles and patterns, and their practical application to the technology value stream, are presented in the remaining chapters of the book.

### The Manufacturing Value Stream

One of the fundamental concepts in Lean is the value stream. We will define it first in the context of manufacturing and then extrapolate how it applies to DevOps and the technology value stream.

Karen Martin and Mike Osterling define a value stream in their book *Value Stream Mapping: How to Visualize Work and Align Leadership for Organizational Transformation* as “the sequence of activities an organization undertakes to deliver upon a customer request,” or “the sequence of activities required to design, produce, and deliver a good or service to a customer, including the dual flows of information and material.”<sup>1</sup>

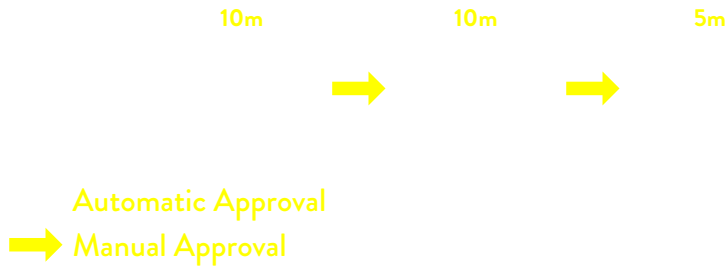
In manufacturing operations, the value stream is often easy to see and observe: it starts when a customer order is received and the raw materials are released onto the plant floor. To enable fast and predictable lead times in any value stream, there is usually a relentless focus on creating a smooth and even flow of work, using techniques such as small batch sizes, reducing work in process (WIP), preventing rework to ensure defects are not passed to downstream work centers, and constantly optimizing systems toward global goals.







This is most easily achieved when we have architecture that is modular, well encapsulated, and loosely coupled so that small teams are able to work with high degrees of autonomy, with failures being small and contained, and without causing global disruptions.



In this scenario, our deployment lead time is measured in minutes, or, in the worst case, hours. Our resulting value stream map should look something like Figure 1.3.

### *Observing “%C/A” as a Measure of Rework*

In addition to lead times and process times, the third key metric in the technology value stream is percent complete and accurate (%C/A). This metric reflects the quality of the output of each step in our value stream.

Karen Martin and Mike Osterling state that “the %C/A can be obtained by asking downstream customers what percentage of the time they receive work that is ‘usable as is,’ meaning that they can do their work without having to correct the information that was provided, add missing information that should have been supplied, or clarify information that should have and could have been clearer.”<sup>3</sup>

## CONTINUOUS LEARNING

### **Flow Metrics to Measure Delivery of Business Value**

When measuring the end-to-end value of any value stream it is important to stay away from proxy metrics (counting the number of lines of code committed or solely the frequency of deployments). While these metrics can reveal local optimiza-





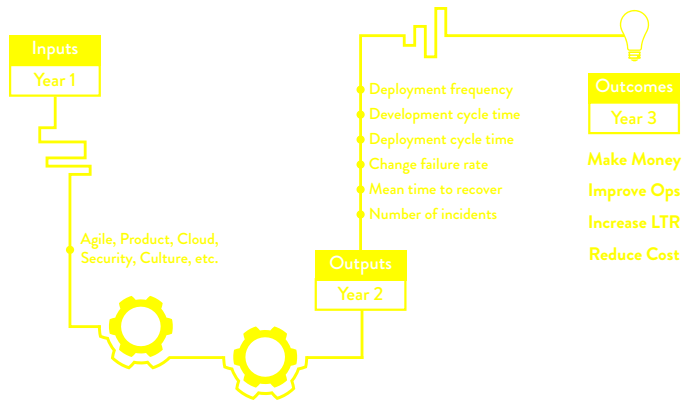








This visibility built the trust needed for experimentation. Finance took four product teams and gave them a set budget for the year. The teams defined the OKRs and used the budget for the top priorities they felt met those OKRs. This allowed the team to test before rollout and focus on accountability and outcomes, and Finance was able to gain even more visibility.



Source: With permission of Ross Clanton

This success allowed them to scale the new model against all of their products and define a new funding process. “This was a huge accelerator in our journey,” said Leibman.<sup>12</sup>

With Finance on board and new processes in place, American Airlines discovered the third question in their DevOps journey: How do we know what the score is? With each small success, the team wanted to better understand how they were doing overall. In other words, they wanted to know what the score was.

For the American Airlines team, year one of their DevOps journey was really focused on inputs: learning about Agile/DevOps, focusing on products, cloud, and security, etc. Year two of their journey focused more on outputs, including the metrics they began measuring, like deployment frequency and mean time to recover. Finally in year three they started to focus not just on inputs and outputs but on outcomes. “At the end of the day, what do we really want to do?” said Leibman.

They came up with the following outcomes: make money, improve Ops, increase LTR, and reduce cost.<sup>13</sup>















## Large Batches



## Single-Piece Flow



(Fold, insert, seal, and stamp the envelope.)

Source: Stefan Luyten, “Single Piece Flow,” Medium.com, August 8, 2014, <https://medium.com/@stefanluyten/single-piece-flow-5d2c2bec845b>.

Like in manufacturing, this large batch release creates sudden, high levels of WIP and massive disruptions to all downstream work centers, resulting in poor flow and poor quality outcomes. This validates our common experience that the larger the change going into production, the more difficult the production errors are to diagnose and fix, and the longer they take to remediate.

In a post on *Startup Lessons Learned*, Eric Ries states,

The batch size is the unit at which work-products move between stages in a development [or DevOps] process. For software, the easiest batch to see is code. Every time an engineer checks in code, they are batching up a certain amount of work. There are many techniques for controlling these batches, ranging from the tiny batches needed for continuous deployment to more traditional branch-based development, where all of the code from multiple developers working for weeks or months is batched up and integrated together.<sup>6</sup>

The equivalent to single piece flow in the technology value stream is realized with continuous deployment, where each change committed to version control is integrated, tested, and deployed into production. The practices that enable this are described in Part IV of this book.

## Reduce the Number of Handoffs

In the technology value stream, whenever we have long deployment lead times measured in months, it is often because there are hundreds (or even thou-

















