Koen Vastmans

Debunking DevOps Delusions

Save yourself from a

experience

Debunking DevOps Delusions

Save yourself from a DevOOPS! experience

Koen Vastmans

simulearn.koen@gmail.com
www.simu-learn.net

To my wife Hilde,
who always supports and motivates me
to keep learning new things
and to be creative.

Preface

My name is Koen Vastmans. Some of you may know me as creator of several agile and DevOps related serious games. Others may know me as co-creator of the Agile Bullshit cards, together with Noel Warnell and Michele D'Urzo. These are 56 cards with thought-provoking statements and matching images. After a while, Noel and I started sharing our opinions on LinkedIn about each of these statements. We did this for over a full year. This way we ended up collecting a full year of agile bullshit.

After that year Noel started focusing on processing all the shared agile knowledge generated, not only of our own opinions, but also a lot of valuable comments. He took care of all the the editing work of what would become the "Agile Bullshit" book. Meanwhile I was inspired to continue with a similar idea, but this time focusing on DevOps. And the approach was also different. Noel and I didn't start with the intention of sharing our opinion about each card; they were meant to trigger discussions in the first place. My intention with the DevOps Bullshit initiative was definitely to share my opinion.

It was more of an iterative approach:

1. Defining some statements
2. Drawing the image
3. Writing my opinion
4. Planning the post on LinkedIn
5. Back to one of the previous steps

As it turned out, it is harder to come up with DevOps-related thought-provoking statements than with agile-related statements. After 28 weeks, I had 28 statements, 28 images and 28 opinions about DevOps. I thought I had enough to make the card deck. For me this was a good opportunity to collect all the LinkedIn posts, structure them according to common themes and bundle all the information into a book.

The book you are currently reading, is the result of the bundling, editing, and structuring of these 28 LinkedIn posts. And two extra cards were added: while finishing the book, two extra cards were presented on a silver plate... I could not leave them out.

I hope you will enjoy this book!
Koen Vastmans

Foreword by Nathen Harvey

DevOps began in 2009 and has never sought a canonical definition. This lack of a formal definition has been freeing; it allows all of us to find the DevOps that works best for us in our own context, based on our lived experiences. However, the lack of a definition also opens up a lot of room for misinterpretation of the ideas and misapplication of the principles behind DevOps.

From the very early days, DevOps has focused on aligning the work and goals of teams, improving collaboration, and taking a human-centric approach to leveraging technology.

I discovered Koen's work via a pithy, DevOps BS post on LinkedIn. The humorous approach to illuminating some misconceptions of DevOps really resonated with me. In reading this book and looking through the DevOps BS cards that Koen shared, it strikes me that there is some amount of truth in each of the BS ideas.

As you read through Debunking DevOps Delusions, I encourage you to keep an open mind while you embrace and help extinguish many of the apparent contradictions. Teams can have speed with stability, DevOps is more than just "dev" and "ops," and pushing back against the status quo prevents stagnation.

Use the ideas presented in this book as conversation starters for the teams across your organization. Remember that we're all on a journey of continuous learning and improvement, there is no destination but we should all seek to enjoy the ride!

Nathen Harvey

DORA lead, community builder, and occasional DevOps farmer[1]

[1]Nathen made a video titled “DevOps: no horse sh**”. See link to the Youtube video in the References.

Introduction

Why this book?

There are quite a few misconceptions about agile and DevOps. These misconceptions about agile were the initial trigger to start making the Agile Bullshit cards and later write a book about it. Agile misconceptions are often related to (bad implementations of) frameworks or techniques. Bad implementations of frameworks or techniques often lead to inefficiencies. Bad implementations of DevOps-related principles and practices will, on the other hand, often lead to poor quality products. Whereas inefficiencies will lead to more time (and budget) consumption, poor product quality can lead directly to revenue loss and a damaged reputation. That's immediately hurting someone's wallet …

I don't pretend to have all the wisdom about DevOps; on the contrary! That said, I at least hope to save you from a **DevOOPS!** experience, by using 28 thought-provoking statements that help you understand how **not** to become a DevOps organization.

You can read this book cover to cover, but you certainly don't have to do so. You can also selectively read chapters based on themes that interest you most. Or you can use certain chapters or cards to trigger a discussion within your team or your organization.

Who should read this book?

This book is meant for anyone who is involved in the development, delivery and maintenance of software products. Technical profiles, like architects or middle management, people who shape the IT strategy and bring the vision of management into practice, can gain insights about DevOps and the pitfalls that can occur during this journey. Also coaches, who guide these aforementioned people, can benefit from reading this book. And lastly, trainers can find inspiration in this book for their training activities.

Structure of the book

This book is full of bullshit about DevOps. Yes, seriously! The chapters and sections in this book are inspired by things I have heard or experienced about DevOps; things that at least made me frown. These are often misconceptions, e.g. that DevOps is about delivering faster, but are also about practices and ways of working that have room for improvement, like not being able to separate release from deployment. This inspiration got translated into the 28 cards with thought-provoking statements and a matching image. I also shared my opinion about each of these cards on LinkedIn.

The cards and LinkedIn posts originally came into existence in a more organic way: I thought of a statement, created the matching card and wrote a post about the theme on LinkedIn. This meant that there was no structure or categorization defined up-front for these cards and statements. They were just created in the order that inspiration struck. The book however does follow a structure. The chapters of this book group cards according to the following themes:

- What's in a name?
- People and responsibilities
- Technologies
- Security-first
- Deploy and release
- Measuring is knowing

Each chapter begins with an overview of its theme, followed by individual sections dedicated to each related card. All sections have the same structure.

Each section contains:

The statement as title of the section, with an image that illustrates the statement.
This example here is the image for "DevOps is about delivering faster"

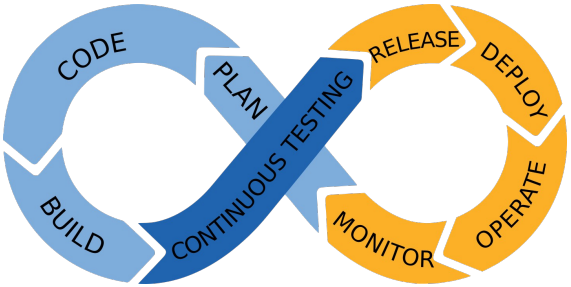
My opinion. This is the main part of a section, where I share some background about why I think this is BS and what to do instead.

The **Takeaway** for that statement. Consider this as a summary of Do's and Don'ts.

The chapter ends with general takeaways or things to remember.

A word about the DevOps cycle...

Throughout this book, I will frequently refer to the DevOps cycle. You will also find this DevOps cycle as part of the images for each statement. And some pages will also contain an image of this DevOps cycle, including the names of the different stages. This is the image of the DevOps cycle you will find in the book:



This is probably the most common visual representation of the DevOps cycle. While there are other variations, the majority of images you will find on the internet are based on this image.

However, a while ago I read Lisa Crispin's critique[2] of this visualization. She argues – and she definitely makes a valid point – that testing is not some stage in the middle of the cycle. Testing should come first (as in test-driven development). And testing is done everywhere in the cycle. Personally, I completely agree! The visual is flawed in this sense and in other ways as well. For instance, on the right hand side of the cycle Release appears before Deploy. This sequence is counterintuitive since deployment typically occurs multiple times before a change is ultimately released.

[2]The opinion of Lisa Crispin about the DevOps cycle and the role of testing was shared on LinkedIn (see References).

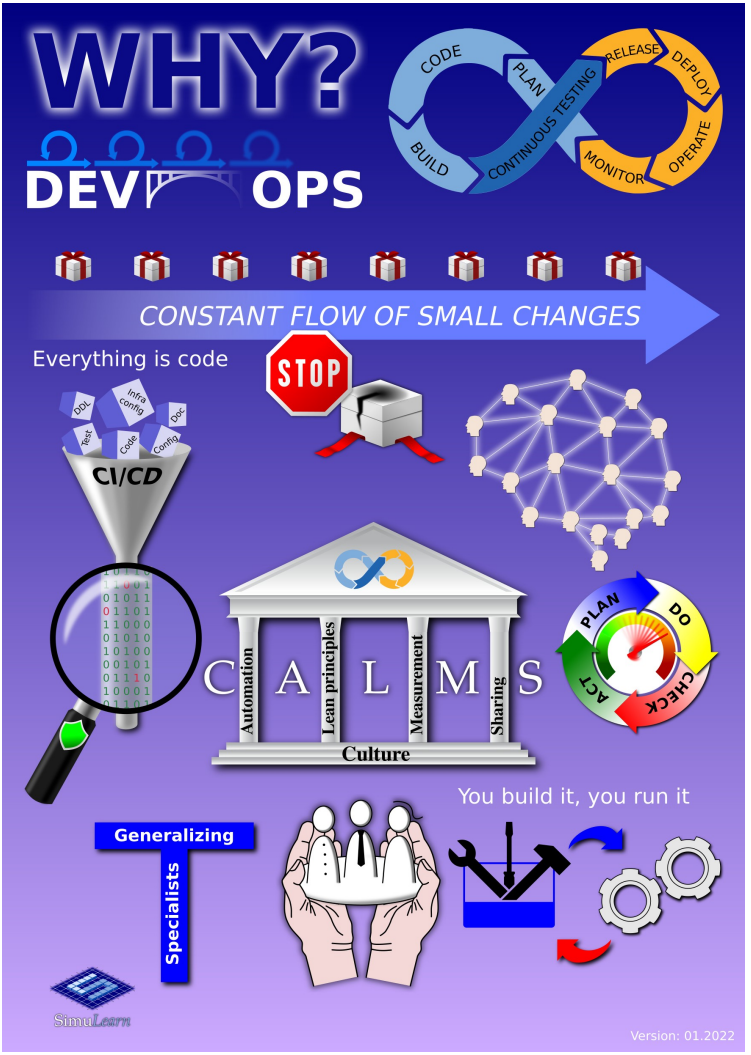
Even though the DevOps cycle visual is not perfect, it is widespread and very recognizable. I use it to make certain points clear regarding other misconceptions about DevOps. That's why I chose to repeatedly use this visual, both on the cards and in this book.

Table of Contents

Chapter 1:
What's in a name?

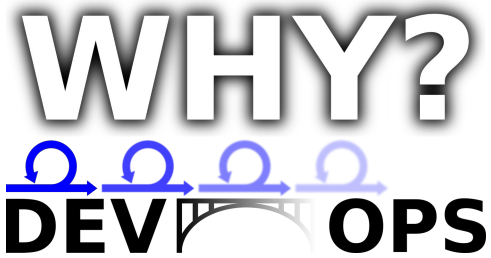
What I think DevOps is about...

Several years ago I made a visual about what I think DevOps stands for. The latest version[3] looks like this:



[3]The PDF of this visual – size A3 poster with explanation on the backside – can be found on my website (see References).

It all starts with the "Why?"

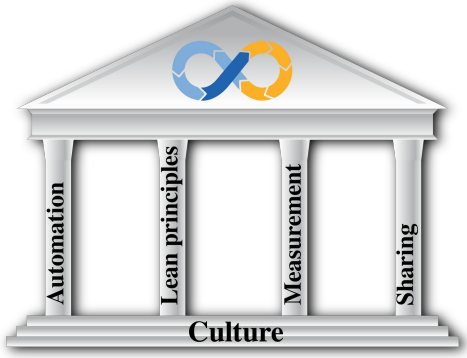


When Patrick Debois organized the first DevOpsDays conference in Ghent in 2009, he was looking for a way to apply agile principles to operations. You can do iterative development and deliver new features sprint after sprint, but what if you still have to deal with traditional release and deployment processes? Where is your **real** fast feedback? In this context, DevOps serves as a bridge between agile development practices and operations.

The CALMS acronym explains it all

CALMS stands for:

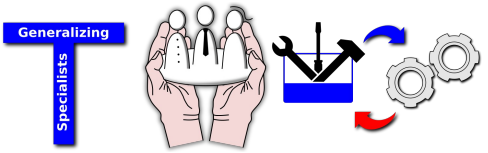
- Culture
- Automation
- Lean principles
- Measuring
- Sharing



The CALMS acronym is often used to explain the principles of DevOps.

C stands for Culture

Contrary to what people might think, culture, not automation, is the foundation of DevOps. The next section will talk in more detail about this misconception. For now, let's focus on what culture means in a DevOps context.



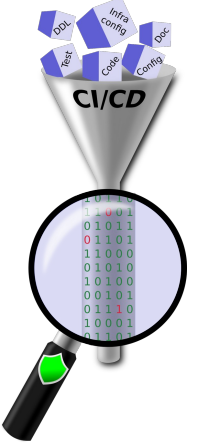
Culture focuses on making sure that teams can work in the most optimal way. That means that team members are so-called T-shaped profiles, generalizing specialists, with both a broad general knowledge foundation and a deep area of specialty. This way the team has all the knowledge needed to build and maintain the solution.

Culture is also about creating a safe environment for the team. In this environment it is safe to fail, make mistakes and learn from them in the form of blameless post mortems.

Culture is also about taking full responsibility for the application, not only developing it, but also making sure it runs properly. As Amazon CTO Werner Vogels said:

> *"You built it, you run it"*

A stands for Automation



Automation is key to reducing errors. Doing the same type of activity manually over and over again can be very error-prone. Doing this in an automated way reduces this risk: the outcome will always be the same if the input and circumstances don't change. Additionally, the fact that an activity is done in an automated way, means that it will probably be completed a lot faster than when done manually.

This is achieved by treating everything as code:

- The source code of your application (obviously)
- Database changes
- Application configuration
- Tests
- Documentation
- Infrastructure configuration

The last one might not sound very obvious, but it is the only way to ensure that you can roll out the same infrastructure over and over again in the same way. Let's say you need to install the same server configuration, with the same technical components multiple times. The only way you can do this without errors, is by applying infrastructure-as-code (IaC).

The vehicle to process all these sources is a CI/CD pipeline, a software component that takes care of continuous integration and continuous deployment/delivery. And yes, making sure that the application you deliver is secure, is part of the responsibilities of your CI/CD pipeline.

L stands for Lean principles

If you know anything about the Lean methodology, then you will remember that eliminating waste is one of its key principles. Work in progress is considered a waste. One way to reduce work in progress is to keep work items small. If you do this, you will be able to deliver a constant flow of small changes.

A defective system is also a form of waste. The automotive industry uses the Andon cord: when something goes wrong at one of the workstations, the worker at that station will pull the Andon cord to stop the line. The management gets a signal that there was an issue at that particular workstation. The entire production chain is stopped and they solve the issue before the construction flow can continue. Similarly, if something goes wrong with your build process – no matter in what environment, whether in development or in production – the team should stop the line and fix the build process immediately.

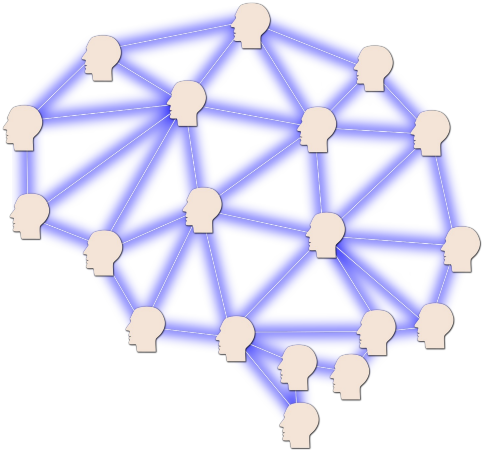
M stands for Measuring

There is only one way to improve objectively: that is based on measurement. Figures, numbers, data, whatever you want to call them, are the result of measuring. Observing these measurements gives you insights about where you and your team are now and where there is room for improvement.

These improvements are implemented in small steps. Again, that is the only way to measure the impact of the change and to see if you can continue with these improvements or need to try a different route. One way to do this, is by applying the Plan – Do – Check – Act cycle (PDCA):

- **Plan** an improvement, based on measurements
- **Do** the improvement
- **Check** the outcome
- **Act** according to the outcome of the improvement effort (continue or change)

S stands for Sharing



Sharing is about knowledge sharing. A team is composed of generalizing specialists who all should have the necessary knowledge to do their job: to build and maintain their application. But a team does not live on an island. Often they will need to acquire new knowledge, e.g. about a new technology. If another team in their organization already went through the same learning process, it is often better to learn from them. Re-inventing the wheel may lead to making the same mistakes that someone else already made and which therefore could have been avoided.

This stresses the importance of communities: the collective wisdom from different people that form the collective brains the team and of their broader organization.

So, what about the bullshit in this chapter?

Now that you have read what I think DevOps is about, let's look at some misconceptions about the term and the concept of DevOps. The remainder of this chapter is full of bullshit about:

- DevOps only focusing on automation
- DevOps being meant to deliver faster
- The nonsense of combinations of DevOps, like DevSecOps, DevTestOps or whatever

The cards that cover the last bullet may contain some statements you find blunt… These are all based on my own observations. You don't have to believe me. Check the information I refer to and then form your own opinion.

DevOps? That's all about automation

Let me tease out the BS level of this statement: automation does in fact play a key role in enabling frequent delivery of value to the customers. However, having "a key role" is not the same as being "the key role".

As explained in the introduction of this chapter, in the DevOps world the acronym CALMS is used for identifying the key principles of DevOps:

- **C** stands for **Culture**
- **A** stands for **Automation**
- **L** stands for **Lean Principles**
- **M** stands for **Measuring**
- **S** stands for **Sharing**

Some scaling framework talks about CALMR. In CALMR, the C stands for "a culture of shared responsibilities" and the R stands for "Recovery". But apart from some slightly different accents, I don't see a difference between CALMS and CALMR[4]. That could have been another BS story…

[4]Several years ago I wrote blog about similarities and differences between CALMS and CALMR. See the link in the References.

If automation were the most important aspect of DevOps, the acronym should probably start with the letter A; this is not the case. Instead, culture is the foundation of DevOps. As the foundation, culture profoundly impacts how DevOps teams function. Read the book *Accelerate*[5] from Nicole Forsgren, Jez Humble and Gene Kim about the research behind the DORA[6] assessments and you will fully understand how important culture is in growing towards a DevOps organization, and how it impacts other DevOps capabilities.

DORA distinguishes the following cultural capabilities:

- Generative organizational culture
- Job satisfaction
- Learning culture
- Transformational leadership
- Well-being

[5]The Book "Accelerate – Building and Scaling High Performing Technology Organizations" helps you understand the findings of the DORA DevOps survey. More info about the book in the References.

[6]DORA stands for DevOps Research and Assessment. They are known for their survey about the State of DevOps (aka "the DORA report"). More on DORA and their assessments in the chapter "Measuring is knowing".

Takeaway

So, DevOps, to what extent is it even about automation? It is true that you want to replace manual actions with repeatable automated actions in order to achieve faster and more frequent delivery of small changes with guaranteed quality. Or as someone once told me, “If you can shoot yourself in the foot over and over again in a controlled way, you're doing the right thing.” He used this expression to stress the importance of automation, so that repetitive tasks always have the same expected outcome. But without the right culture, your automation efforts will most likely not realize the expected added value…

DevOps is about delivering faster

A video[7] documents how Patrick Debois organized the first DevOpsDays conference and while doing so, kicked off the worldwide DevOps movement. In this video you will hear that Patrick was having a hard time with a migration project: the development team wanted frequent delivery to enable testing in small batches, while the operations team prioritized system stability. Every change represented a potential stability risk… Patrick was inspired after hearing a conference talk from people of Flickr, who were able to do 10 deployments per day (we are still talking about 2009, mind you!) and how Dev and Ops people collaborated to achieve this. In other words: how Dev and Ops collaborated to deliver frequently and still maintain stability.

If you read the above, it is fair to say that the main goal of DevOps is not to deliver faster (that's merely a by-product) but to bridge the gap between development and operations teams in the first place. Imagine an agile development team delivering something valuable every iteration (say, every 2 weeks), but you can only release (or even deploy) a big package every 3 months, then you miss a big benefit of working iteratively and incrementally: the fast feedback. You would only be able to test incrementally, and not get fast feedback from the real customers.

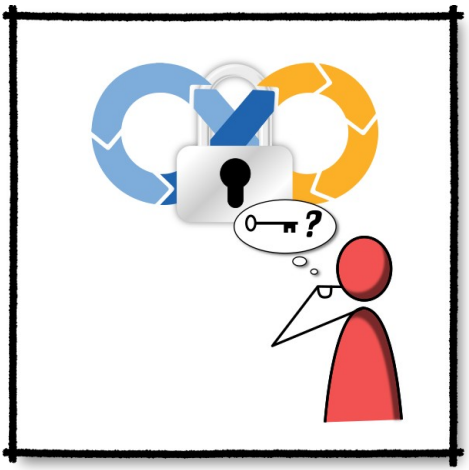
[7]The video "The (Short) History of DevOps" can be found on YouTube (see References).

An important prerequisite of DevOps is to be sure you are putting a quality product into production. Remember what someone once told me about shooting yourself in the foot? If your only focus were on delivering faster, neglecting quality (bugs, usability, security, performance, …), you would harm your system and your organization. That's also shooting yourself in the foot, but not in a controlled way...

Takeaway

So, is DevOps about delivering faster? You must prioritize quality in everything you deliver. And that the Dev and Ops people work together to achieve this. Only then you can start delivering faster…

Should we implement DevSecOps or SecDevOps?



When talking about DevOps, I have always been allergic to combinations with DevOps, like BizDevOps and especially DevSecOps. Readers of Gene Kim's book "The Phoenix Project"[8] may remember the final chapter of the book. That book is considered to be the reference book on DevOps. If you don't remember, read that chapter again. Let me quote a paragraph from that chapter:

> *There's a term that we're hearing more lately: something called "DevOps." Maybe everyone attending this party is a form of DevOps, but I suspect it's something much more than that. It's Product Management, Development, IT Operations, and even Information Security all working together and supporting one another.*

[8]The Phoenix Project is considered as the reference book to learn the basics of DevOps. It is a novel written by Gene Kim about a fictional company, Parts Unlimited, that struggles to get a major project released. See References for more information about this book.

This passage clearly indicates that DevOps encompasses all aspects of delivery, including security. Then why would there still be a need for a separate term like DevSecOps? Let me make a bold statement: it is because security tool vendors felt left out. They feared they were missing the boat of DevOps. So every combination of DevOps with security is pure marketing. And they narrow down the scope to security only, they don't address DevOps as a whole.

In an article of a security tool vendor[9] I literally read:

> *"Traditional DevOps processes do not include security. This means that many application development businesses that practice the DevOps culture initially have no security teams at all!"*

That same article makes a distinction between:

- **DevOpsSec**: doing security testing after the facts
- **DevSecOps**: security somewhere integrated
- **SecDevOps**: putting security first

To me this is utter BS. While this may sound reasonable, if you read the details, this is nothing but marketing fluff. Companies want to distinguish themselves from others with the products and services they offer. They stress the importance of secure coding guidelines and the integration of security testing in the CI/CD pipeline, and then they mention their products "as an example" of how to do this. But having secure coding guidelines is a good (agile) development practice that has nothing to do with DevOps exclusively. And a CI/CD pipeline alone won't make you a DevOps organization (more about that in chapter 3 – Technologies).

[9]Blog posts of both Acunetix and Whitesource – both security tool vendors – explain the "difference" between DevSecOps and SecDevOps. Also DevOps.com did a full webinar about this difference. Find the links in the References.

Takeaway

So should we implement DevSecOps or SecDevOps? Neither. Just transform to a DevOps organization and make sure you pay enough attention to security:

- During planning
- When coding
- When building
- While testing
- Upon release
- Upon deployment
- During operations activities
- When monitoring

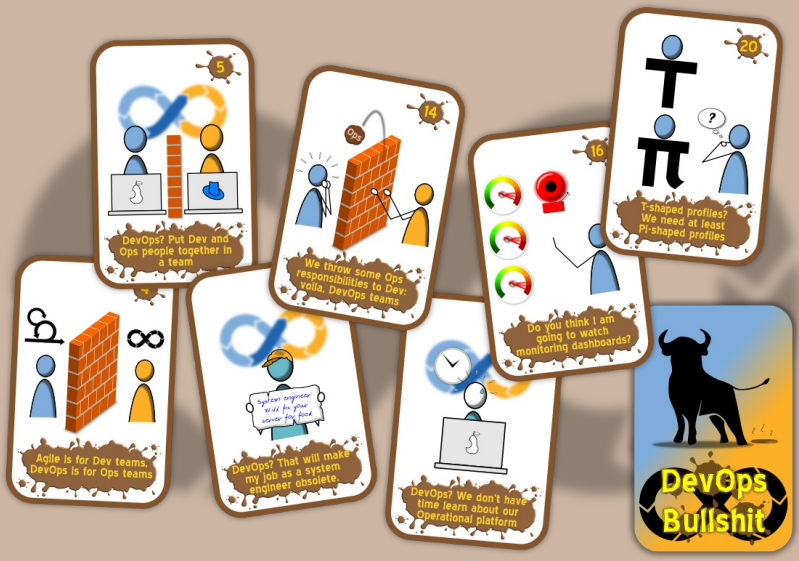
Heck, that's the entire DevOps cycle!

Takeaways of this chapter

What should you remember from this chapter?

- DevOps is about bridging the gap between agile (iterative & incremental) development and operations.
- The DevOps principles can best be explained with the acronym CALMS: Culture, Automation, Lean principles, Measuring and Sharing.
- DevOps is not only about automation. Mindset and culture are crucial.
- DevOps is not about delivering faster. That's a result, not a goal.
- We don't need another name combination of DevOps:
  - not for security
  - not for testing
  - for nothing (except maybe for chaos engineering ☺)
- Instead of inventing a new name, fix the trust issue.

Chapter 2:
People and responsibilities



Culture – that's about people

As you read in the previous chapter, the foundation of DevOps, according to the CALMS acronym and also the DORA findings, is culture. Culture is about how an organization treats their people and how people behave, how they treat each other and how they do their job.

Also in the previous chapter you read how culture could be applied in a DevOps team and an entire DevOps organization:

- Team members being generalizing specialists
- Creating an environment where it is safe to fail and learn from your failures through blameless post mortems
- The team both builds an application and maintains it

Each of these ideas only work well if people take their responsibility within the team, as well as within the organization.

So where's the bullshit in this chapter?

As of the first DevOpsDays conference, DevOps was considered a movement, not a methodology let alone a set of automation technologies you need to apply to deliver faster, more frequently. A movement is about people. People need to collaborate in order to deliver an end-to-end solution.

It doesn't help if Dev and Ops people are put together hierarchically in the same team but still do their own jobs separately. This will create bottlenecks because people will stick to their responsibilities like they are used to doing. It is as if there is still a wall between them.

Automation is an important part of a successful DevOps implementation. For Ops people treating infrastructure configuration-as-code can be confronting, especially when it is used for self-service provisioning of components and their configuration. They might think their job is in danger. But someone with knowledge needs to develop these automations so that they can be provisioned via self-servicing…

Once these self-servicing capabilities are in place, it is not enough to throw them over the wall to the developers, saying "you're a DevOps team now, this is your responsibility". People need time to acquire a new skill, before they can be responsible for it. One doesn't become a generalizing specialist overnight.

And then there is the "You built it, you run it" saying. Yes, running it also means that a team needs to be signaled when something goes wrong with their application, so that they can take corrective actions.

As you can see, there is a lot to say about people and responsibilities. Let's dive into the details…

Acknowledgements

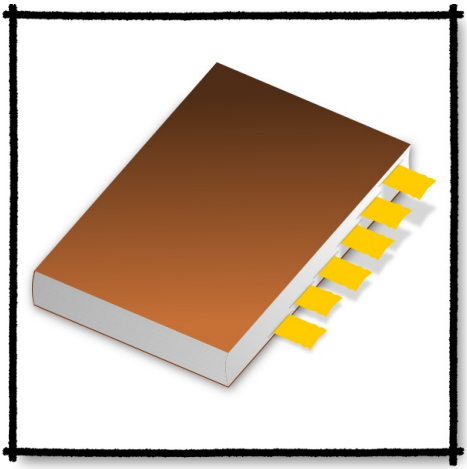
This book would never exist without the support of several people. I am very grateful for their contribution – in whatever way – to this book:

- Noel Warnell, with whom I started the initiative of the Agile Bullshit cards, which resulted in the Agile Bullshit book. Our collaboration inspired me to start this DevOps Bullshit initiative that resulted in this book.
- Nathen Harvey of DORA, who introduced me to the DORA community.
- All the people who participated in the proof reading and shared their feedback to make this the best possible book:
  - Doug Lane
  - Georg Link
  - James Marshall
  - Jens Weber
  - Lanre Okuyelu
  - Nadia Eldeib
  - Patrick Debois
  - Pooriya Shokri
  - Rénald Casagraude
  - Stijn Dejongh

Thank you very much!

References

Throughout this book, I refer to different articles, books or videos that inspired me during my writing and helped me form my opinion. This part gives an overview of all the sources I refer to, divided per chapter.



Reference of Nathen Harvey's foreword

- Nathen Harvey's video "DevOps: No Horse Sh**": **https://www.youtube.com/watch?v=0P0HD5pE-zU**

Reference of the introduction

- Lisa Crispin's opinion about the DevOps cycle and the role of testing **https://www.linkedin.com/posts/lisacrispin_holistictesting-agiletesting-activity-7247731772879171584-pQt-**

References of chapter 1

- The PDF of the DevOps visual: **https://www.simu-learn.net/download-devops-infographic**
- Blog post "DevOps in 5 letters – Should we say CALMS or CALMR?": **https://www.knowledgehut.com/blog/devops/devops-in-5-letters-should-we-say-calms-or-calmr**

- Nicole Forsgren, Jez Humble and Gene Kim. *Accelerate – The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution, 2018. ISBN number 978-1942788331 (paperback), 978-1942788355 (eBook) **https://itrevolution.com/product/accelerate/**
- The video "The (short) History of DevOps": **https://youtu.be/o7-IuYS0iSE**
- Gene Kim, Kevin Behr, and George Spafford. *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win*. IT Revolution, 2013. ISBN number 978-1950508945 (paperback), 978-1950508969 (eBook) **https://itrevolution.com/product/the-phoenix-project/**
- Blog post of Acunetix.com about the difference between DevSecOps and SecDevOps: **https://www.acunetix.com/blog/web-security-zone/devsecops-vs-secdevops**
- My blog post titled "DevSecOps, SecDevOps or DevOpsSec - really?" **https://www.linkedin.com/pulse/devsecops-secdevops-devopssec-really-koen-vastmans/**
- Blog post of Mend.io – formerly known as Whitesource – about the difference between DevSevOps and SecDevOps: **https://www.mend.io/blog/devsecops-vs-secdevops**
- DevOps.com webinar recording about the difference between DevSecOps and SecDevOps: **https://youtu.be/orlrck3_mfc**

- Blog post of SauceLabs about DevTestOps: **https://saucelabs.com/resources/blog/what-is-devtestops-and-how-does-it-impact-software-quality**
- The video of Patrick Debois' conference talk "DevOps Is More Complex and Harder Than You Think – Personal Lessons" at QCon in 2020: **https://youtu.be/sB_mmpCHvRU**

References of chapter 2

- Team 3 game: **https://brain-games.com/products/team3-green-party-board-game**
- Enterprise DevOps: Why you should run what you build: **https://aws.amazon.com/blogs/enterprise-strategy/enterprise-devops-why-you-should-run-what-you-build/**

References of chapter 3

- The Agile Bullshit card about TWWADI – The Way We Always Did It – explained by Noel Warnell: **https://lnkd.in/evtYUQ9d**
- The game Continuous Interruptions, Continuous Delays, Quite Annoying: **https://www.simu-learn.net/ci-cd-qa**
- ZOWE, the Open Mainframe project: **https://www.zowe.org**
- Open Service Broker: **https://www.openservicebrokerapi.org**

References of chapter 4

- The full story about the Struts vulnerability exploit at Equifax: **https://www.breachsense.com/blog/equifax-data-breach/**
- Common Vulnerabilities and Exposures: **https://www.cve.org/About/Overview**
- European Union Vulnerabilities Database: **https://euvd.enisa.europa.eu/**
- Article on TechTarget explaining the difference between SAST, DAST and IAST: **https://www.techtarget.com/searchsoftwarequality/tip/SAST-vs-DAST-vs-IAST-Security-testing-tool-comparison**

References of chapter 5

- The LinkedIn post of Noel Warnell about "We can't release every iteration": **https://www.linkedin.com/posts/noelwarnell_agilebullshit-devops-platformengineering-activity-7094988504484491264-K7Jq**
- My LinkedIn post "We don't test before everything is complete": **https://www.linkedin.com/posts/koenvastmans_when-i-was-still-working-as-a-developer-activity-7077167003848859648-ASPI/**
- More about the Agile Bullshit Cards: **https://www.simu-learn.net/agile-bullshit-cards**

References of chapter 6

- "DORA's Journey: An Exploration" – Jez Humble explains the history of DORA: **https://medium.com/@jezhumble/doras-journey-an-exploration-4c6bfc41e667**
- Westrum Organizational Culture (as referred to in the book "*Accelerate – The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*"): **https://itrevolution.com/articles/westrums-organizational-model-in-tech-orgs/**

References of A word about the Cards

- The DIB Guide: Detecting Agile BS: **https://media.defense.gov/2018/Oct/09/2002049591/-1/-1/0/DIB_DETECTING_AGILE_BS_2018.10.05.PDF**
- Agile Bullshit Cards: **https://www.simu-learn.net/agile-bullshit-cards**
- DevOps Bullshit Cards: **https://www.simu-learn.net/devops-bullshit**

Debunking DevOps Delusions

When it comes to DevOps, you often hear or read some misconceptions, delusions if you like. For example:
DevOps is about delivering faster.
Sounds familiar? This book covers 30 of such delusions.

Divided over 6 chapters, you will get food for thought about different aspects of DevOps, including people, technologies, security, release and deploy. With a sense of humor, this book also helps you discover things that will **not** bring you closer to a DevOps organization.

If you are looking for ready to use solutions, like in a cook book, to transform your organization to a DevOps organization, you may be disappointed. This book doesn't pretend to have all the answers; above all, it provides a lot of questions.

Then why should you read this book?

To save yourself from a **DevOOPS!** experience.