esper

DevOps for Devices

The innovator's guide to enterprise fleet transformation

Yadhu Gopalan

First edition May 2022.

Cover art by Dinesha Kumar and Veeresh Antapur

Bellevue
3600 136th Pl SE, #210,
Bellevue WA US, 98006

Bangalore
Salarpuria Adonis #3/1 Swami
Vivekananda Road, Old Madras
Road, 560038

Preface

The digital transformation moment for enterprise device fleets is now. With the rise of technology, we've seen a massive market pull to incorporate intelligent devices into the customer experience across verticals as far apart as healthcare, retail, education, and travel. As enterprises have acknowledged the value and ROI of implementing these devices, their proliferation has increased and the underlying complexities of managing them effectively, particularly at scale, can no longer be ignored.

I want to give you a DevOps blueprint to deliver customer experience innovation and build agility into the ways you deploy, manage, update, and secure your fleet of devices and the software it runs. The insights are for anyone with an interest in DevOps and devices, but senior decision makers who need to leverage devices to deliver exceptional customer experiences will benefit most.

I'm obsessed with DevOps. It is the best framework we have today for building agile engineering and operations teams. In over twenty years of leading teams at the intersection of hardware and software, from Windows CE to Windows Phone, Amazon Fire OS, Amazon Go, and the AWS Hardware Fleet team, I saw the impact of DevOps firsthand. It is the north star that has allowed me to confidently push my code to hundreds of millions of devices over the years.

For those of you who are already familiar with DevOps, devices and device fleets present challenges cloud servers do not. A large cloud deployment might peak on the order of a thousand servers, but device fleets can easily reach into the hundreds of thousands or even millions. The largest restaurants or retailers have tens of thousands of stores globally, each with potentially hundreds of devices. Consumer products like connected cars and fitness equipment are produced in the millions. And to make things even more challenging, compared to carefully maintained cloud servers in fixed and secure environments under ideal conditions, fleets of devices are highly distributed and experience vastly different environmental conditions. Customers and employees touch them, they may be exposed to the elements, and they may not always be reachable.

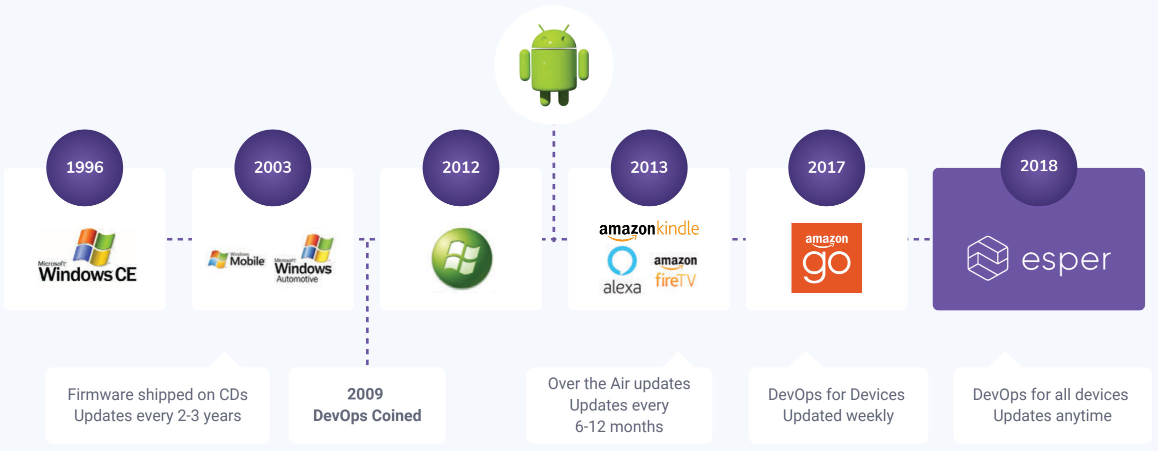
The infrastructure and tooling needed to deploy, manage, update, and secure device fleets mandates a completely different framework.

I built device infrastructure in bespoke environments several times, and I've become passionate about democratizing that power. Microsoft and Amazon invested significant resources building and maintaining that infrastructure — but as the cloud taught us, building infrastructure is incredibly challenging. In fact, I've spoken with many CTOs and engineering leaders who have visions of building their own proprietary device infrastructure, only to realize the immense cost of developing and maintaining their own tooling. Not only was it more resource-intensive than they thought, it distracted them from developing their product.

I founded Esper so that everyone can benefit from better infrastructure using our DevOps for devices platform — and spend more time and energy focused on making customers happy.

For 20 years I have been enabling devices one-off. The time is right to democratize this

100's of millions of devices run code I have personally built.



I hope you'll come to share my conviction that **DevOps is the single best solution to the complexities of device fleets at enterprise scale—and that it can be your next big competitive advantage.**

Table of Contents

1 The Rise of DevOps

In This Chapter:

• A quick refresher on DevOps fundamentals
• The proven benefits of DevOps for business apps
• Adding device management to your DevOps processes

What allows companies like Microsoft or Amazon to remain so consistently innovative? As a veteran of both, I can share that innovation isn't accidental or the result of a unique company culture; it's a result of deliberate focus on building the right infrastructure to enable innovation.

A quick refresher on DevOps fundamentals

DevOps isn't new, but many executives outside of engineering and ops might still be getting up to speed. That's okay! These are what I see as the core principles of DevOps — as a foundation for understanding how these now proven practices are beginning to transform dedicated device fleets at enterprise scale.

DevOps Principle #1: Continuous improvement

DevOps focuses on constant (daily) outputs and accomplishments. This cultural shift can feel strange at first. Gone are six-month-long projects with massive budgets and deliverables and, with it, the thinking that considers this the best approach.

A culture of continuous testing, experimentation, and development supports agile business practices at every level of the organization. DevOps can fuel smarter investment decisions into product improvements and help technologists and executives better understand real-time changes to customer preferences.

DevOps Principle #2: Automate everything (that can be automated)

A practical benefit of DevOps culture is that teams are always on the lookout for ways to automate tasks and processes. Complementary AI and cloud services have given companies and teams a huge opportunity to adopt software that automates previously manual processes.

Managing systems at a manual level is surely tedious, but worse yet, it's error-prone. You want your engineers to be innovating, not babysitting deployments. Additionally, human mistakes can be very demoralizing to the individual responsible, as well as the team and wider organization. This is yet another area where mechanisms can (and frequently, should) supplant intentions.

DevOps Principle #3: Customer needs are the North Star

Customer experience (CX) matters more than ever before. It's one of the most important factors in determining a company's annual revenue growth and customer loyalty. According to IBM and Adobe, in 2021, organizations that elevated CX digital transformation to the status of a formal business priority reported three times higher revenue growth for the prior two fiscal years.[1]

DevOps practices ensure every team member understands how their work impacts the customer. It's a powerful tool for maximizing customer retention and satisfaction. Increased employee retention and overall satisfaction are often welcome, and inevitable, side effects.

DevOps Principle #4: Scale up and scale down

Scripting, batching, and automating previously manual processes lets you scale rapidly— up or down. In cloud computing, for example, we see this happening when a company needs to add server capacity on the fly to address an unexpected spike in traffic. Companies themselves often need to scale up (and, as the pandemic demonstrated, scale down) quickly, though they tend to find doing so quite difficult.

DevOps builds into the DNA of a company a requirement that operations of any size (from small teams to large departments) should be structured to grow or shrink quickly. As dedicated devices increasingly become core to business strategy, you need to be able to scale your device deployment workflows as easily and flexibly as your cloud workflows.

1. Ablett, J., & Wellwood, S. (15 C.E.). Optimizing Your DXP Capabilities [Review of Optimizing Your DXP Capabilities]. IBM and Adobe. https://www.ibm.com/thought-leadership/institute-business-value/report/optimize-dxp

DevOps is an established practice, but device fleets have yet to fully benefit from this revolution. Usage of dedicated devices to deliver customer experiences is exploding. Esper partnered with 451 Research to survey developers and IT leaders who manage dedicated devices at companies from 500 to well over **10,000 employees** to look deeper into this trend. The results show 86% of respondents agreed that dedicated device fleets, such as IoT devices that fulfill a single use case, are increasingly viewed as a core element of overall business strategy.[2] Furthermore, 89% of respondents agreed that dedicated devices are a critical tool for differentiating services and customer experiences.[3]

The world we know today would not be possible without cloud computing infrastructure — DevOps for devices will be similarly critical in enabling the next generation of customer experiences.

DevOps for devices is the next wave of digital transformation, fueled by two converging trends: the proven benefits of DevOps (Chapter 1) and the rising importance of devices that bring digital agility to the physical world (Chapter 2).

DevOps Principle #5: Plan to fail … and learn from those failures

Continuously delivering work product in smaller bites also means there are more (smaller) missteps along the way. Companies should embrace failure by de-stigmatizing mistakes and turning them into learning opportunities that immediately provide blueprints for how to do things better or more efficiently going forward.

While deploying quickly can be great, it's critical to have excellent monitoring and partitioning of your fleet in place as well. This enables the ability to stop deployments and roll them back, which is crucial to deployment agility. When failures occur you need to have an operational excellence process in place to learn about each failure (e.g. through root cause analysis) and address them immediately.

(2022). Enterprise-Class Dedicated Device Fleets Set to Explode, but Operational Challenges Loom [Review of Enterprise-Class Dedicated Device Fleets Set to Explode, but Operational Challenges Loom]. In https://blog.esper.io/digital-transformation-strategy-around-dedicated-devices/. - Esper and 451 Research.

DevOps Principle #6: Measure everything measurable

Organizations that measure everything can better understand the potential impact of their choices. DevOps is an important step for integrating real-world data into business decision-making. Data can help teams make better decisions about changes to services, processes, and other variables across the organization.

The only real way to know you're making the right choices is by measuring variables and tracking their progression over time. And any variable that can be measured, should be measured. You never know what data will become relevant in the future (good storage is cheap, good data is not). If you're already measuring it, you've got a great baseline to start from.

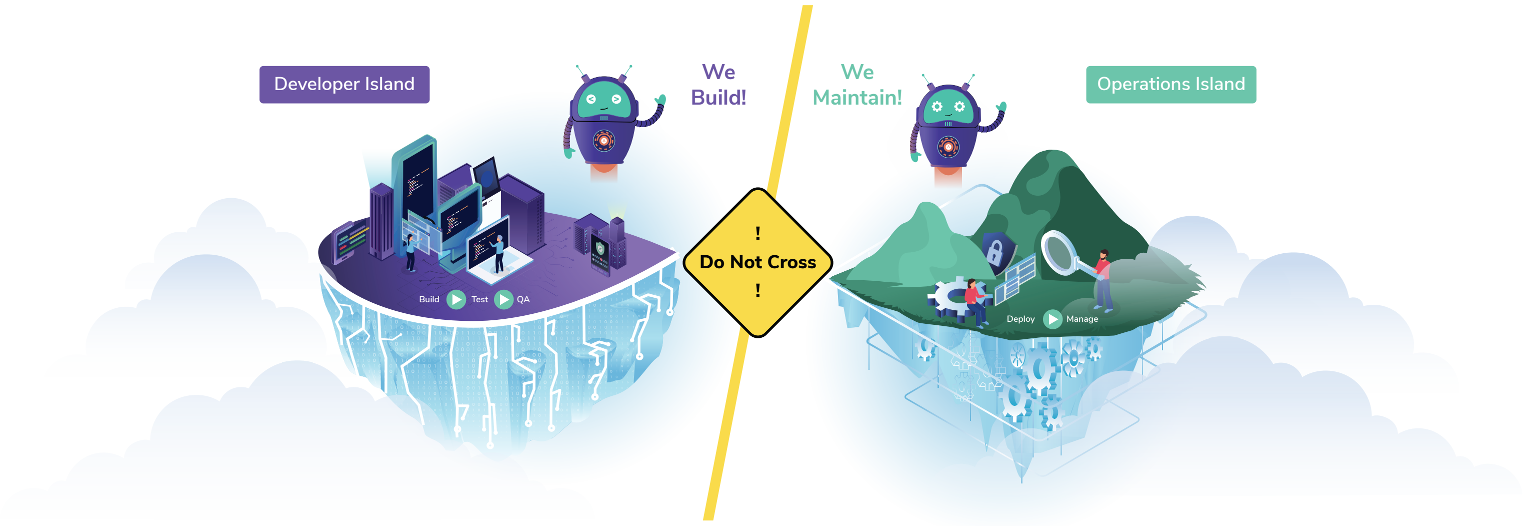
As CEO at Esper, I try to lead our company with these six DevOps principles and ingrain them in every team and every department. If you want to grow with confidence and continuously improve, I believe this is a great foundation to start with. As we'll see in Chapter 2, these principles unlock the value of device fleets.

The proven benefits of DevOps for business apps

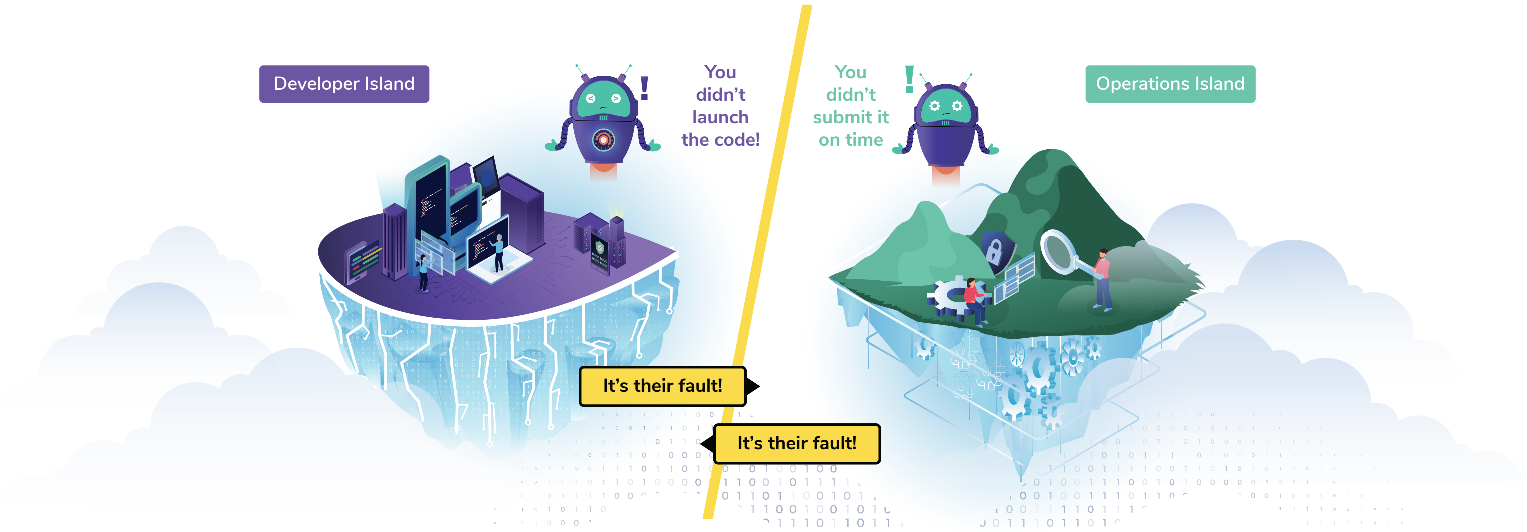
Amazon was among the first companies to show how impactful DevOps can be: constantly delivering the most up-to-date, feature-packed, and secure version of your software to customers is a game-changer. Combining sophisticated tooling, infrastructure, and development practices, you can continuously iterate software and automatically push changes to production (and in turn, to your customers). But what kind of outcomes can this approach enable?

DevOps streamlines development and operations by addressing shortcomings in the traditional software development life cycle. Today, every company is a software company. By extension, any company can be a DevOps-first company.

The legacy approach to software deployment treats developers and operations as two distinct "islands" inside an organization. Developers build and test, while operations deploy and monitor.



But this structure creates silos. Devs see themselves as separate from ops, and vice versa.



The shortcomings of silos become apparent when issues arise. Developers want the operations team to be more agile so code and fixes can be deployed faster and more frequently. Operations, meanwhile, wants development to more rigorously test and optimize that software before deployment and get feedback after deployment. Both sides have valid concerns.

DevOps is a framework to resolve this bilateral frustration through better orchestration between teams. Silos give way to collaborative alignment enabled by tools, processes, and productive partnerships.

DevOps is not a technology

It's a mindset focused on high-velocity code changes and deployments based on custome feedback and telemetry — in a continuous loop.

DevOps breaks down the development and operations wall by organizing tasks into smaller, more manageable processes. This reduces the impact of any potential issue, shortening your integration, deployment, and feedback loops. This speeds up your development and deployment cycles while simultaneously decreasing risk and increasing control over those processes. ***"When you put together a lot of small updates and optimizations, you can deliver better, more reliable customer experiences faster."*** It's a win-win-win.

DevOps success story

Imagine an enterprise IT department has just finished updating 3,000 mobile devices operated by field technicians, only to discover a serious new bug in the core customer-facing app. In the legacy pre-DevOps world, the company would instantly go into panic mode because now they need to recall every device to update the software. Tens or even hundreds of thousands of hours in lost utilization might accrue as you scramble to deploy a fix.

A proper DevOps team would be able to drive wildly different outcomes in this situation (using Esper, of course!). Continuous integration and continuous delivery (CI/CD) pipelines automatically push updates to every device in the field. Instead of an all-hands fire drill, they've likely remedied the issue before most of the field techs clock in (or notice the issue), and updates are applied to each device in stages defined by the customer's specific deployment needs.

In case it wasn't clear: Long iteration cycles hurt your customers. As Gene Kim, the former CTO of Tripwire, put it, ***"When you have long iterations, your ability to out-experiment your competition is tremendously compromised. In an age when almost all the major initiatives are nearly 100 percent reliant on the technology value stream, we're talking about the biggest business problems of any organization."***[4]

A lot has been written about the many advantages of DevOps in cloud software development. But much less has been said about how DevOps can revolutionize the dedicated device ecosystem.

Waters, J. (n.d.). The evolution of DevOps: Gene Kim on getting to continuous delivery [Review of The evolution of DevOps: Gene Kim on getting to continuous delivery]. TechBeacon. https://techbeacon.com/app-dev-testing/evolution-devops-gene-kim-getting-continuous-delivery

2 DevOps for Devices Was Inevitable

In This Chapter:

• **DevOps transforms how enterprises manage dedicated device fleets**
• **Laying the groundwork for success with DevOps for devices**
• **You're competing in the age of dedicated devices … are your tools?**

During my time working on the Amazon Go store concept, DevOps was the beating heart of everything we did, from new tools for managing code to novel practices in how software is architected, engineered, and written. A lot of the hard work that went into making Amazon Go go was bridging DevOps from the familiar realm of software to include the new aspects of hardware and software interaction.

The experience taught me that you did not need to reinvent the wheel to tame hardware complexities at scale. But you did need new thinking, new habits, and most importantly, new infrastructure. That's how Esper came to be.

In truth, this is inevitable! DevOps is transforming every corner of the software development universe and enterprise-scale device ops is no exception.

Why now?

Why wasn't DevOps for devices evangelized five or ten years ago alongside DevOps for cloud development? Inevitability doesn't explain why this is the moment for DevOps for devices.

To answer, we need a quick history of the evolution of developing and deploying software for dedicated devices. Early in my career, we'd write the code, put it on a CD, and then ship those CDs to the hardware manufacturer. If we wanted to update the code, we'd need to wait for the next hardware cycle, which could be a year or more away.

Next came side-loading with USB sticks. We could put the updated OS or application code on the USB stick, plug it into the device, and run the update.

With the rise of cloud computing and cloud connectivity in the mid-to-late 2000s, it became possible to make updates over the air. Typically you had to pay for it, so updates were still done infrequently. Around the same time, Apple, Microsoft, and Google launched their respective app stores, which rapidly shifted the paradigm for app updates for mobile phones. However, for dedicated devices, those app stores did not provide the granular control that enterprises require. Through the 2010s, Apple's App Store and Google's Google Play Store (and Managed Google Play for enterprise) grew in maturity and have become pervasive across consumer, commercial, and enterprise applications.

Back to today. We've seen the rise of cloud DevOps practices and cloud connectivity for devices reach near-ubiquity. At the same time, customer expectations have never been higher. From mobile apps and SaaS products that have frequent updates to cars that get updates with new features even after they're driven off the lot, the bar for innovation is higher than ever. And it will continue to rise as digital-native generations become the lion's share of the population. To continue to thrive, the time is now for enterprises to embrace DevOps for devices.

DevOps transforms how enterprises manage devices

The typical Esper customer manages a fleet of Android devices running custom software for use cases like in-store interactive promotion or field team project management. Esper provides the critical infrastructure that lets these companies focus on creating innovative customer experiences.

Esper is the first company to bring the power of DevOps to device fleets, and I gladly spend a lot of my time evangelizing this philosophy to enterprise executives. Getting their devices "working" in the field is just a step on the path to true DevOps maturity — a means to an end. Because we all know what business leaders truly want to focus on is delivering delightful customer experiences. That's why Esper focuses specifically on the devices so many companies increasingly depend on to interact with their customers: The impact is real, measurable, and (in the case of software deployment) immediate.

DevOps for devices delivers three key business benefits that share a common root theme: Confidence.

DevOps gets you there with:

Reliability. Manual processes done at scale are error prone and slow. DevOps lets you automate batches of update rollouts and can alert and respond to exceptions in real time. You do less and more gets done (and done safely).

Repeatability. There is no "done" in device management. Deployments and updates should always be advancing your near and long-term goals and delivering maximum business value.

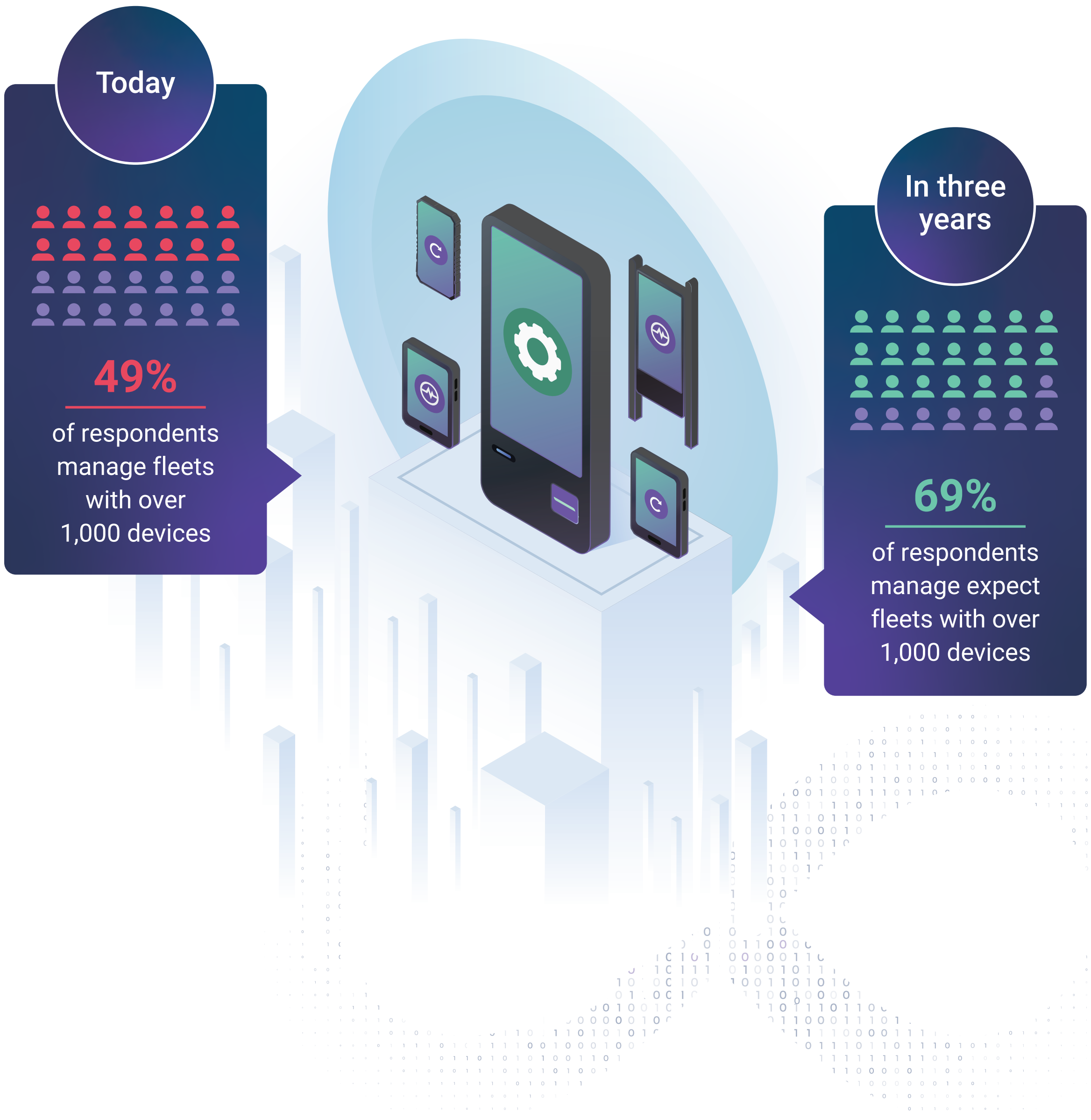
Security. Connected devices are subjected to dynamic environments and require ongoing monitoring and management to ensure devices fleets are functioning properly. DevOps can help your device fleet withstand complex and adverse conditions.

You don't even have to look to Esper to see customers who are already benefiting from the DevOps for devices phenomenon. Tesla can update its cars over the air, including changes for driver safety or optimizing the available range. Amazon Alexa constantly receives new skills that make it easier and more convenient for customers to intelligently manage and interact with their homes, cars, and offices. Companies that are taking full advantage of this convergence are being rewarded with highly visible marketplace dominance.

Device performance is crucial to customer experience

Dedicated device fleets are growing rapidly and have been for some time. Devices are already delivering customer interactions, transactions, and experiences — and are a new source of recurring revenue. By 2023, Gartner estimates that there will be 43 billion connected devices in the world; three times as many as in 2018.[5]

Organizations are increasing the size & diversity of their dedicated device fleets



Behind this explosive growth are dynamics like Moore's Law, hardware commoditization, and innovative device form factors that can reach customers in contexts previously found only in science fiction. These continued advances in technology (and drastically lower costs) mean virtually anything can become a connected device: bicycles, watches, kiosks, televisions, industrial machines, point-of-sale systems, cars, inventory scanners — the list goes on as far as your imagination can take it.

Dedicated devices open doors to new use cases available for any company big or small to deliver innovative customer experiences and generate new revenue streams. In fact, I believe planned obsolescence is a thing of the past. Instead, in line with demands for environmental sustainability and other consumer preferences, companies are moving to subscription models where the product constantly improves even though the hardware remains the same.

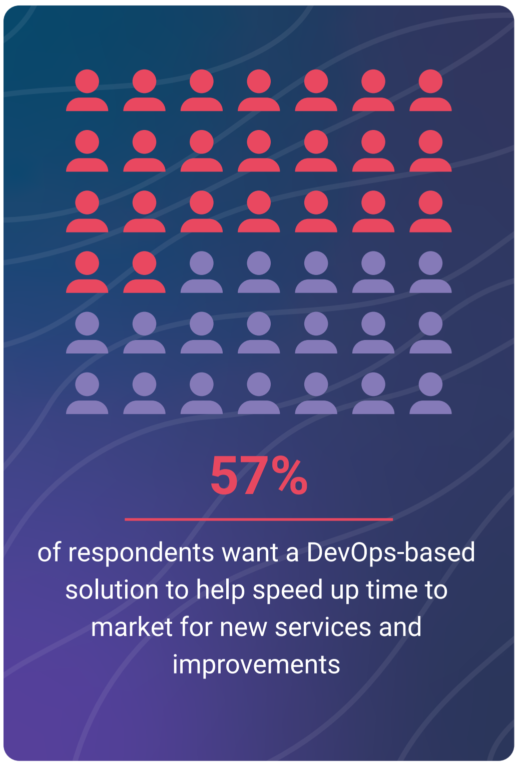
And subscription-based services are exactly what modern consumers are asking for. A 2021 Zuora study found nearly two-thirds of customers choose subscriptions to feel "connected" to brands in the age of self-service.[6] These subscriptions provide continuous personalization that improves over time, building an ongoing connection with a customer that extends far beyond an individual transaction.

In the age of subscription models and recurring revenue, devices are mission critical to delivering those experiences. Esper partnered with 451 Research on research to validate this assumption. 89% of companies we surveyed consider their dedicated device fleets a critical tool for differentiating their services and customer experiences.[7] These devices deliver a continuous stream of value across use cases and industries. For example, innovative hotels and entertainment venues are introducing personalized experiences, exclusive access with subscriptions, and member rewards. Smart fitness brands stand out through curated content and gamifying progress over time. Retailers are offering immersive experiences and personalized subscription programs that connect the brand from home to their brick-and-mortar stores.

To stay competitive, enterprises need to innovate on their customer experience quickly and often. The 451 Research survey revealed an awareness that delivering exceptional customer experiences at speed and scale requires new thinking:

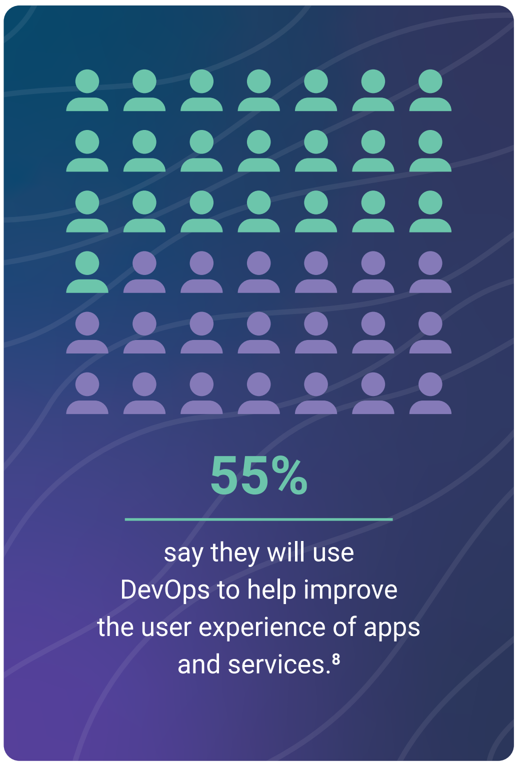
(2021, March 21). SUBSCRIPTION ECONOMY INDEX LEVEL REACHES 437 OVER NEARLY A DECADE AS CONSUMER BUYING PREFERENCES SHIFT FROM OWNERSHIP TO USERSHIP [Review of SUBSCRIPTION ECONOMY INDEX LEVEL REACHES 437 OVER NEARLY A DECADE AS CONSUMER BUYING PREFERENCES SHIFT FROM OWNERSHIP TO USERSHIP]. Zuora Financial. - https://www.zuora.com/press-release/subscription-economy- index-level-reaches-437-over-nearly-a-decade-as-consumer-buying-preferences-shift-from-ownership-to-usership/

(2022). Enterprise-Class Dedicated Device Fleets Set to Explode, but Operational Challenges Loom [Review of Enterprise-Class Dedicated Device Fleets Set to Explode, but Operational Challenges Loom]. In - https://blog.esper.io/digital-transformation-strategy-around-dedicated-devices/. - Esper and 451 Research.



57%

of respondents want a DevOps-based solution to help speed up time to market for new services and improvements



55%

say they will use DevOps to help improve the user experience of apps and services.[8]

Brands are rethinking their approaches to deploying, updating, and monitoring device fleets at the edge — placing DevOps at the center of their action plans.

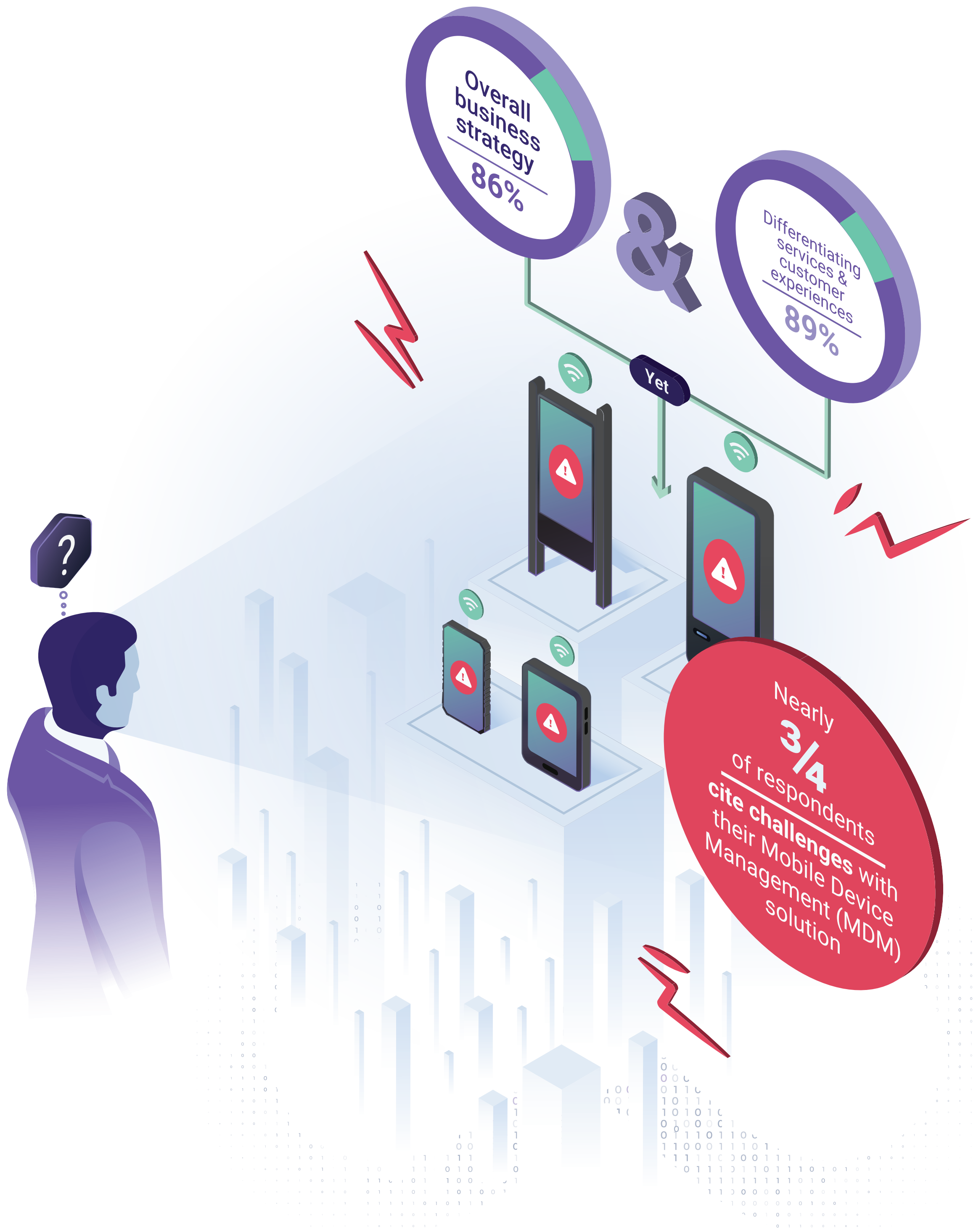
Managing devices isn't easy. Hardware is, well, hard. Later chapters focus on how DevOps for devices reverses that dynamic in your favor.

You're competing in the age of dedicated devices … but your tools aren't

Esper was born from the realization that the "best" option for device fleet management, Mobile Device Management (MDM) software, just doesn't do the job that organizations need it to do. MDM platforms were built to address the Bring Your Own Device (BYOD) reality of smartphones and other mobile devices inside many large companies. To maintain any semblance of security in an environment where corporate data lived on employees' personal devices, IT teams needed help. As a result, MDM was born.

(2022). Enterprise-Class Dedicated Device Fleets Set to Explode, but Operational Challenges Loom [Review of Enterprise-Class Dedicated Device Fleets Set to Explode, but Operational Challenges Loom]. In - https://blog.esper.io/digital-transformation-strategy-around-dedicated-devices/. Esper and 451 Research.

Among enterprise organizations dedicated devices are increasingly viewed as critical to


Overall business strategy
86%
&
Differentiating services & customer experiences
89%
Yet
Nearly
3/4
of respondents
cite challenges with their Mobile Device Management (MDM) solution

But MDM tools were never intended as a solution for dedicated device fleets. The MDM DNA is to protect the company from employee devices. ***"In the MDM paradigm, the company is first and the device is second. When it comes to dedicated devices, the devices are mission critical to the business strategy."*** The device, in essence, is the company. Whether it is a self-service kiosk, digital signage, a forklift, or an exercise bike, these are the primary methods of interaction between customers and companies.

When customers (internal or external) rely on these dedicated devices to function properly and securely, that's what DevOps for devices is for. As a result, any tool comparison between MDM and DevOps for devices is apples to oranges:

- Always-connected, mission critical systems like point of sale, kiosks, and digital signage can't be effectively managed using BYOD tools. These tools cannot effectively deploy, control, or update dedicated devices individually, much less at enterprise scale.
- Just because a dedicated device looks like a smartphone or tablet doesn't mean it can be managed like one. DevOps-based solutions with APIs, SDKs, and OS savvy are needed to deploy, monitor, and update real fleets at real scale.

These comparatively simplistic solutions aren't up to the task of managing mission-critical devices. If building exceptionally reliable and consistent experiences to your customers matter, frequent updates matter. Continuous improvement matters. Robust remote connectivity matters. Modern device strategies hinge on truly rapid iteration to ensure you're always delivering the best possible customer experience.

Many dedicated device fleets struggle to meet even quarterly software release cycles — DevOps can enable weekly, daily, or even hourly deployment cycles. The competitive advantages of such a shift speak for themselves.

The rise of DevOps alongside the rise of dedicated devices sets up what I believe is the next natural evolution of both: DevOps for dedicated devices. Why this should matter to most companies is the focus of the next chapter.

3

Bringing the Cloud Experience to Device Fleets

In This Chapter:

- Devices are mission critical in the age of subscription revenue and elevated customer expectations
- The technologies enabling DevOps for devices
- What success looks like

Devices are the crucial last mile in your continuous value delivery chain — if your software can't reach the devices your customers depend on, you may as well not be delivering that software at all. That last mile can be grueling — or even impassable — without the infrastructure and practices a DevOps for devices platform can enable.

DevOps (mostly) works for your devices like it works for your cloud servers

Successful development and operations teams build on proven DevOps principles to solve for resilience and speed in fleet deployments. Device fleets present a unique set of challenges, even compared to the cloud services ecosystem where DevOps was born.

DEVOPS FOR CLOUD TECH STACK	FOR...	DEVOPS FOR DEVICES TECH STACK
JFrog	Pipelines	esper
circleci	CI/CD	esper
docker	Containers	esper
ANSIBLE	Configuration Management (CM)	esper
aws	Cloud enablement / APIs	aws

Cloud deployments, even large ones, are at the magnitude of maybe a thousand servers. On the other hand, device fleets often reach magnitudes of hundreds of thousands or even millions of devices. New challenges come up at this level of scale, which I will cover in Chapter 7. To make things even more challenging, device fleets are much more distributed than cloud servers, with vastly different environmental conditions.

The DevOps lifecycle for devices must also adjust to the fact that hardware, OS, firmware, configurations, application services, and auxiliary content can all impact total device health and customer experience.

Customers (whether internal company employees or the public) who use dedicated devices are virtually always remote from product teams, and end-users are rarely product experts. There are no built-in redundancies or failover hardware for consumer devices, and any downtime can result in costly customer churn.

Enterprises need a single pane of glass view into their device fleet to manage health at scale, as well as sufficient alerting and monitoring to detect and correct early warning signs before customer devices fail. DevOps for devices is the approach needed to accomplish this.

DevOps for Dedicated Devices Compared to Cloud

	Cloud	Device	Esper Value
Provisioning	Procure and configure cloud resources (compute, network, storage)	Procure devices, install management solution, configure	Touchless when possible (EFA, Knox) Flexible onboarding (preload, automated full configuration)
Resource Management	Grouping concepts to match business needs, staging environments	Grouping concepts to match business needs, staging environments	Flexible device groups Set group blueprint to control config for entire group (many as one)
Manage Configuration	"Codified", automatic configuration Drift management (config drifts)	"Codified", automatic configuration Drift management (config drifts)	Device Blueprints Drift Detection / Management
Deploy Updates	Automated, staged rollouts of updates (code, content, config)	Automated, staged rollouts of updates (code, content, config) including error handling (devices offline, etc)	Device Pipelines (default and custom) for all updates to device fleet including Esper SW (agent and OS)
Monitoring	Health monitoring, alerting Resource monitoring App monitoring	Health monitoring, alerting Resource monitoring App monitoring Device HW monitoring (battery, wifi)	Real-time health monitoring Rich telemetry over time for trends Default and custom dashboards

Update with caution

Enterprises have historically approached firmware and software updates with an abundance of caution. Operations teams spend months of sleepless nights testing updates before going live, hoping the update does not wreak havoc on their fleet. Updating production devices with a traditional MDM solution can yield unpredictable and potentially irreversible harm, as most lack features to safely and systematically roll out updates, let alone monitor and respond to issues in real time during a deployment.

Early in my career, I was working with a fleet of devices that was densely concentrated in a location where it was physically difficult to remove and retrofit. Since this was prior to having a DevOps for devices infrastructure in place, the system had no rollback or safety mechanism. One time we rolled out an update that produced issues in the real world and had functionality that clashed with future updates. Because we couldn't easily roll back, the only option was to manually replace and rework the devices which took many weeks for the development team to resolve.

Having these experiences, I know the fear that deploying updates can create. On the other hand, not updating or delaying updates frequently isn't an option, either, as doing so creates security risks as well as risks to customer experience and loyalty.

I also worked on a device fleet that faced hardware issues that could have easily been worked around with a software fix. But since there wasn't a built-in mechanism to deploy updates to the hardware, the only way it could be resolved was a full hardware recall. In this instance, having the capability to update would have reduced device downtime and saved the team weeks of work.

DevOps for devices strikes the balance, recognizing the potential consequences of updates, both positive and negative. Ultimately, it is necessary to give organizations a reliable, repeatable, and secure way to roll out and roll back updates to customers with predictable results. Yet research shows only 35% of companies release applications in a DevOps manner.[9] Teams need automated ways to safely deploy updates and partition their fleets to tightly target and perform updates often without disrupting the customer experience.

Automation is king

Chances are, you have more than a handful of applications that are updated frequently to deliver new features and fixes to your customers. And chances are even better you don't want to perform this never-ending stream of updates manually, across your entire fleet, every time. With a DevOps approach, you can easily configure automated device fleet workflows that scale with you.

If you're already in the DevOps world, the tools for managing and releasing content (applications) on devices aren't far off from the cloud-based tools you're already using — think Jenkins, JFrog, or CircleCI. Acting as an automation server, these tools compile your content and push it through to your repository, where it then becomes available to test or deploy. DevOps for devices is about bringing this same functionality to that critical fleet last mile. Simply put, DevOps for devices seamlessly deploys software from your repository directly onto your devices.

The automated, integrated version of device management greatly increases the efficiency of releasing, configuring, and deploying your content. The fewer hands involved, the less chance for things to go wrong. The automation here is also highly repeatable and, therefore, scaleable. Grow and diversify your fleet as much as you like, and your device management will stay simple while still giving end-users the innovative experiences they expect.

(2022). Enterprise-Class Dedicated Device Fleets Set to Explode, but Operational Challenges Loom [Review of Enterprise-Class Dedicated Device Fleets Set to Explode, but Operational Challenges Loom]. In - https://blog.esper.io/digital-transformation-strategy-around-dedicated-devices/. - Esper and 451 Research.

Automation in action: Moving from manual to scripted

Let’s say you don’t have those kinds of powerful automations in place. The most basic method to update those devices is the manual way. Take Android, for example: First, you upload the APK installation file for your content to a cloud web console, where it will be hosted. With your app hosted in the console, you can then push that content to your devices with a variety of methods, like Android’s handy ADB CLI.

Sounds simple enough. But what happens when you try to scale this process? More likely than not, you have a diverse device fleet made up of thousands, or even hundreds of thousands, of devices that require different content and updates at different times. Imagine updating each device manually in a one-off cadence for each update. It’s a painful picture.

Conversely, DevOps is built on the principle of automation. And many organizations consider infrastructure automation the lynchpin of their DevOps implementations. If you integrate this workflow with the cloud tools you’re already using to build software, the aforementioned manual drudgery is fully replaced by an automated workflow — from build to deployment — that puts your software on your devices as quickly as your business demands.

Here's how. (Note that this workflow uses pipelines, which I'll expand on in our next chapter. For now, think of them simply as a set of automated processes used to complete a specified task.) It all starts with your existing CI/CD app development pipelines. Esper Pipelines APIs integrate with your existing CI/CD infrastructure, delivering software update packages directly to your Esper endpoint. Then, a second API call is placed to build a pipeline in Esper. Once this pipeline is built, it's used to push your content to the devices you wish, such as a set of test devices used to ensure proper behavior before pushing to your production devices. From this point on, the integration is set, and it's simply a matter of how you want to test and deploy your content. With Esper, you can deploy it to your devices in stages or deploy it to any segment of devices — a single device, a defined group, a percentage of devices, or your entire fleet.

Successful implementation comes down to ensuring the integrations between pipelines are in harmony. Since this workflow relies on standard APIs, there are no configuration options to sort through or edge cases to consider. As long as you're using standard REST APIs to communicate with Esper’s pipelines, you can use your cloud-based tool of choice. If you’re unfamiliar with REST APIs, REST is an architectural style for API that uses HTTP requests to access and use data. They’re format-agnostic and will standardize the formatting of data requests and the data received, making them fast and flexible to use.

Protect your updated fleet

Now that you've updated your devices, monitoring their health, security, and performance is the next most important thing. Robust monitoring is essential for fleet devices. Replacements can take weeks or even months to ship from brand headquarters or the manufacturer (OEM) overseas. Troubleshooting over the phone is costly and frequently ineffective. Too often, minor failures can lead to a major product recall.

Dedicated device fleets are uniquely heterogeneous and are only becoming more fragmented and specialized in response to consumer demand for personalized experiences. A single fleet can contain countless subsets of hardware configurations, content, and cloud services. These variations may reflect a combination of subscriber preference, demographics, behavior, and countless other variables.

Traditional approaches to fleet segmentation aren't pragmatic for monitoring at scale, especially not in a fleet that contains thousands, let alone millions of edge devices. Device operations teams need a single pane of glass to monitor the entire fleet and intelligent alerts to avoid excessive noise and uncover meaningful patterns.

Monitoring an enterprise fleet of edge devices requires dynamic partitioning so DevOps teams learn from common failures and isolate unique ones. And, perhaps most importantly, it requires the telemetry and operational agility to restore total device health before an isolated warning signal impacts the experience of one or many customers.

Tip of the iceberg

This is a small window into the power of using a DevOps approach when managing a modern device fleet. The automation capabilities we've discussed in this chapter are built on pipelines, which help you build and define workflows like the one we just outlined. Now, let's dive into the ins and outs of pipelines and how they help build the foundation of DevOps success.

4

CI/CD Pipelines: Paths to Continuous Innovation

In This Chapter:

- **Overview of Continuous integration / Continuous delivery (CI/CD) and pipelines concepts in DevOps**
- **CI/CD in the business context**
- **CI/CD in the DevOps for devices (hardware) context**
- **Typical CI/CD use cases**

Software release cycles are highly dependent on the specifics of your business. Whether it's rapidly changing customer expectations in your industry, logistical challenges, or even security and compliance requirements, there are many factors you need to consider when identifying how frequently you want to release updates.

DevOps isn't a silver bullet — it's not going to eliminate these challenges, but it can heavily degrade their ability to negatively impact your business. By breaking the software update process up into smaller chunks through CI/CD, you reduce risk. Then, by automating how those updates are staged and deployed at scale via pipelines, you can deploy software faster to more devices and make your customers happier.

Continuous integration / Continuous delivery (CI/CD)

Since virtually all modern software is cloud-first, software development has moved to smaller, more frequent releases via the process of continuous integration and continuous delivery, or CI/CD. In the devices context, CI/CD is a key element of the infrastructure that allows you to manage the business apps running on dedicated devices.

CI/CD covers the steps in the modern software development and release process:

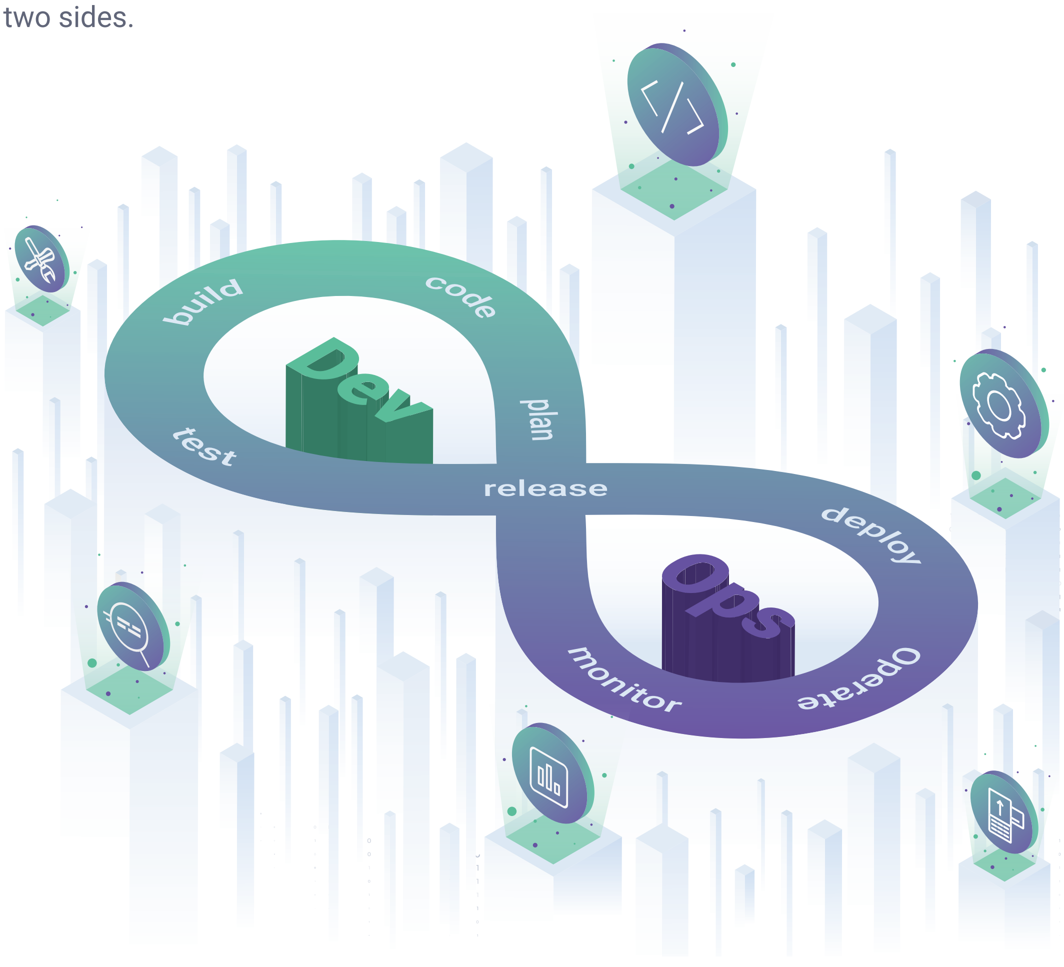
• Continuous integration (CI): Comprises four phases: Plan, Code, Build, Test. The next phase, Release, is the transition from CI to CD.

Continuous delivery (CD): CD starts with Release and then comprises the next three phases: Deploy, Operate, and Monitor.

DevOps pipelines

	INTEGRATION	DELIVERY
Component	CI: Plan/Code/Build/Test	CD: Release/ Deploy/Operate/Monitor
What is it?	DevOps phases of software operations	DevOps phases of software operations
Who owns it?	Dev team	IT/Ops team

In terms of team ownership, developers own the integration phases, and IT operations own the delivery phases. The transition phase, Release, is supposed to be collaborative. But in reality, it often becomes a wall. Developers toss their code over the wall to the IT operations team, which is then expected to release that code into production. DevOps is the paradigm shift that smashes this wall and aligns the two sides.



As the complexity of your software and device fleet grows, so too will your need for complex tools to enable efficient and effective deployment of that software in the field through each of the phases. When applied properly, CI/CD gives you the confidence and control you need to continuously innovate — another CI!

The business case for CI/CD

How often you release updates is a result of multiple competing business, technical, and logistical factors. The product team (including developers and engineers) wants to release as fast as they have created meaningfully better experiences for the customer. The operations team balances that with logistical concerns (e.g., is the device currently in use?) and risk factors (e.g., will the update affect 10 customers or 10 million customers? If issues arise, will this just slow the customer experience or break it completely?). Higher risk leads to less frequent updates, and fewer updates mean customers wait longer to receive new experiences.

As your software deployment footprint grows, the complexity of those deployments will grow in equal or greater measure. Without proper CI/CD, you could find yourself with a lot of angry customers, stale products, and a fleet that's constantly trying to catch up to your missteps by rushing out the product before it's ready.

Bringing CI/CD to device fleet management

Introducing the hardware component into your DevOps practice adds new wrinkles. Since DevOps is a framework for delivering quality software experiences quickly, DevOps practices — breaking things into smaller components, reducing risk, adding confidence to deploy faster, monitoring, etc. — can certainly be applied to devices.

In the CI/CD framework, this involves another CD: continuous deployment. You write the code, which is automatically integrated into the codebase, and then automatically delivered to the testing or production environment. Continuous deployment takes the process further by automatically releasing a tested-and-proven update to your customers and devices in the field — whether all of them or a chosen group.

For apps, DevOps gives you ways to manage versions and constantly roll out new functionality. For devices, DevOps gives you ways to manage configurations at scale, ensuring your devices are ready to run your apps, including all the benefits of continuous building, testing, and deploying.

DevOps pipelines

In DevOps, a pipeline is a catch-all term for the processes a particular company links together to enable CI/CD. In practice, it's made up of the automations and tools your teams use to tackle the individual steps in software development and hardware management.

Pipeline automation goes up over time as DevOps processes mature within a company. As we've seen, tooling and teams together increase the number of subprocesses (build, test, etc.) that can be automated. Early in the DevOps maturity phase, pipelines will likely have points where developers or operations need to step in and take manual action (e.g., promoting a deployment stage).

Pipelines give you flexibility that isn't possible with traditional siloed development and operations — you can adjust for variability across devices and simultaneously enforce consistency when and where you need it. A couple examples could be customer experience segmentation and regional version control. Pipelines built to account for such complexity let you release faster through simultaneous testing and deployment.

Device pipelines in action

We've worked with customers from across industries, all of which benefit from DevOps pipelines. Whether a global restaurant chain, retail company, or medical device, there's been a defining commonality in how pipelines are applied — one that gets the standard, general availability configuration, and one that is treated as a test or beta group. This could be the devices in the innovation lab, a test market, or a group of devices that need to be treated uniquely, perhaps because of local regulations or other regionalization features. Regardless of the reason, whenever they want to test new experiences, they're able to get them out to the right group of devices every time.

Without DevOps for devices and pipeline tooling, rolling out an update that manages multiple versions is cumbersome and typically requires field operators to get hands-on by manually updating devices. As a result, far fewer updates end up deployed, and the pace of innovation is reduced.

With CI/CD pipelines and over the air (OTA) updates, you can point the pipeline with the new version to the test group, deploy the update, and test it. After QA, testing, and receiving positive customer feedback, you can then point the pipeline to run on the production fleet, automating and staging a bulk rollout to your specifications.

Pipelines break down silos

Pipelines also help you manage multiple teams pushing updates to the same devices. When your teams are innovating and pushing updates frequently (which is good!), you can sometimes end up with unintended issues. Fortunately, DevOps practices can solve those, too.

I once worked on a project where we built, thoroughly tested, and rolled out a custom update. Soon after the update went out, another team pushed their update, which was also thoroughly tested on its own. But together, there were issues and the resolution required working with individual devices one by one. A mature DevOps culture and toolset doesn't just break down silos between developers and operations; it also breaks down silos between engineering pods or teams. With pipelines that can be shared between teams, the consequences from subsequent interactions can be identified and solved earlier in the development cycle.

Once your pipelines are running smoothly, the natural next step is improving your ability to capture and respond to feedback. Known formally as observability, it's the subject of our next chapter.

5

Observability: Learn from Real-World Performance

In This Chapter:

- How Observability enables continuous feedback
- Observability in the business context
- Differences with observability for devices (hardware)
- Use cases highlighting effective Observability

Observability builds on traditional performance monitoring to offer deeper context into software issues. As rapid software release cycles become more commonplace, observability gives fleet operators better ways to understand what's going on with the fleet.

Great companies that practice DevOps with observability through sophisticated and automated measurement can detect and rectify any adverse changes in real time. As a result, you can create systems where deployments can happen at any time.

Many years ago, I worked in a product org where code from thousands of developers flowed regularly through the system with no formal release cycles. The final production systems were continuously measured for any abnormal changes in the metrics. This includes not just technical metrics but business value such as drops in usage and orders. When abnormal conditions were detected, all code updates that were deemed to be in the range possibly affecting the abnormality were pulled back. It was then up to developers to understand which one of the changes impacted the system.

This system trades off flow of changes and efficiency versus having to make the rare analysis in retrospect when things go wrong. Of course, this only works when you have great processes in place throughout the development cycle.

If you want to improve how you use feedback, proactively rectify errors, and continuously improve software, you need observability.

Observability concepts

Before we dive into the benefits of observability in the dedicated device lifecycle, let's quickly recap some key concepts.

Observability is a buzzy catch-all term that comes down to using the right data to better understand how a system is performing. I think of it as "performance monitoring" for the cloud age. Telemetry data powers these insights, making use of different flavors of output data:

Logs show you what happened "behind the scenes" and should include a timestamp to tell you when it happened (e.g., X loaded at Y time). Of course, events happen all the time and most are not meaningful — capture, stream, and retention strategy are important. Logs are the basic building blocks used in both metrics and traces.

Metrics, as the name suggests, measure performance over time. Tracking metrics is critical for knowing whether you're making the right decisions and analyzing improvement. Common metrics are usage, lead times, release frequency, and time to recovery.

Traces represent data movement through an application. Traces are used to investigate issues such as performance degradation and errors and become immensely valuable as workflows increase in complexity.

Why Is observability important?

In order to effectively implement DevOps, you need to create a feedback loop. Going back to the familiar DevOps infinity loop, you see that the "monitor" phase feeds back into the "plan" phase. Observability lives at this transition, providing insight into how software is performing in the real world.

Here's a practical example: An IT operations team identifies signs of performance degradation. Customers are abandoning transactions late in the typical flow. The team's observability tools analyze traces and logs to identify the root causes. They move quickly from realizing there's a problem to working on fixing it.

Properly implemented DevOps observability really moves the needle, making it easier to:

- Identify errors and catch drift early, limiting the potential for downtime or lost revenue.
- Improve your release cadence because you can quickly identify and resolve issues that might otherwise stop a deployment in its tracks.
- Achieve better visibility into production environments lets you deliver software improvements with more confidence more quickly.

An emphasis on observability pays off — a Forrester study found one observability tool generated an ROI of 296% and a net present value of $4.43 million over three years.[10]

(2021). Honeycomb's Forrester Total Economic Impact (TEI) Study [Review of Honeycomb's Forrester Total Economic Impact (TEI) Study]. In Honeycomb.io. Honeycomb. - https://www.honeycomb.io/blog/forrester-tei-benefits-observability-roi-2021/

Done right, observability can ultimately be a foundation for predicting issues and improving reliability. For example, device telemetry data can be hooked up to predictive analysis platforms to give you powerful insights about fleet performance or even individual devices.

Observability for device fleets: Managing by exception

Telemetry is always important for cloud products, but it is especially important when it comes to dedicated device fleets. With more devices in the field than any one human being could hope to personally monitor, it's both critical and challenging to understand the state of your fleet while maintaining powerful visibility into individual devices. You need tooling and automation to achieve this at scale. And conceptually, you need to take a management by exception approach.

Fleet hardware and software composition introduce more wrinkles. If your fleet is small and uniformly configured, you may try observing manually — knowing the state of every device at all times. However, it only takes one outage or failed deployment to learn that the manual approach is more than a little onerous (and expensive). And as the permutations of the configurations multiply, it can quickly become impossible to effectively monitor with manual processes. Enter observability tooling designed for dedicated device fleets.

Rather than trying to keep track of the state of every device (which you do want to have access to), managing by exception means you only want to be alerted to take action when a device deviates from its expected state. This is the concept of drift. Every device has a stated ideal configuration. This could be the application version, admin settings, online or offline state, geolocation, etc. If the device's actual state, for whatever reason, does not match the ideal configuration, it has drifted and you want to be notified.

Telemetry data and device fleet observability enable you to more efficiently and effectively manage your device fleet. It allows you to quickly become aware of drift, investigate, identify, and remedy — ideally before the end user even notices.

For example, if an update results in an error in one device but not the other 1,000 devices in the fleet, how do you go about making a diagnosis? Was connectivity disrupted at the time of the update? Was the device powered off? Is the error only occurring on devices running a specific version of the OS? Does the presence of a peripheral change introduce a previously undocumented incompatibility? All of these questions can be answered using robust telemetry data in concert with observability tooling.

Observability in action: Remote healthcare

Esper worked with a company building a connected product for remote healthcare. It goes without saying that reliability was priority one for this customer. Whenever the user needs the device, it simply must work. This is a very valid reason to be concerned by the prospect of pushing frequent updates. If an update takes the device temporarily offline (or worse), the consequences may be severe — even life-threatening. Without proper observability practices, your end users may notice the error before you do, resulting in support tickets and many hours of headaches. This can create a culture of organizational fear, in which updates take weeks or months in testing to play out every potential scenario.

With DevOps observability in place, as this company achieved using Esper, the provider can proactively identify and prevent issues before they become real problems. Releasing updates stops being about avoiding disaster and starts being about delighting the end customer.

Observability in action: Restaurant PoS system

National fast-food restaurant chains operate some of the largest and most distributed point of sale (PoS) fleets in the world, meaning something as simple as deploying an update to their PoS to add seasonal menu items could theoretically bring down the entire business's ability to take orders or process customer payments. The highly distributed nature of those fleets also creates immense challenges: How can you be sure every restaurant is always online, let alone that its software is behaving?. For example, restaurant franchisees in one market might turn off certain stations for weeks at a time to save on electricity during the off-season. Now that they're back online, they fail. This inhibits employees from jumping on more stations to respond to swells in foot traffic.

With observability, fleet managers can respond to this situation quickly or proactively prevent it from occurring.To restore the devices, using observability fleet managers could spot the non-functional systems and see these devices never received the OS update the new patch relies on. They can then quickly deploy the required OS updates to these select systems and then deploy the software patch.

To prevent this situation from occurring again, or at all, fleet managers can use observability to see that these systems have been off for an out-of-policy amount of time and set up an automatic alert to franchise or district managers to power them on.

Observability in action: Warehousing and logistics

Observability can have a huge impact in many different use cases. Let’s say a manufacturing company oversees a fleet of mounted forklift tablets that assist workers on the floor. The on-device fulfillment software includes a spreadsheet that forklift operators rely on to find and store their inventory. However, on a select number of tablets, the data is taking far too long to load — the fields are completely blank for 10 seconds before finally populating. This is hurting productivity and causing workers to avoid using these forklifts. With the right observability in place, operators can see that the devices with high latency are set to use a default cloud database storage location instead of a geographically optimized one. They then deploy a simple patch that instantly resolves the latency issues before employees start sending in support tickets.

The presence of hardware adds a complicating layer (layers, even) when compared to just observability in the cloud. As devices increasingly become mission critical to business strategy as well as the wellbeing of everyday consumers, it’s important for leaders to understand the value of observability and for their organizations to get it right.

6 DevSecOps: DevOps for Security and Compliance

In This Chapter:

- DevOps + Security = DevSecOps
- Common questions about DevSecOps
- Rapid security reviews even in regulated industries

While great for delivering rich customer experiences, device fleets are also a prime target for bad actors. There are more hostile actors than ever, more media scrutiny of breaches, and more people inside your company working hard to keep threats at bay. The most successful security teams approach security in ways that are appropriately flexible and adaptive: continuous improvements, rapid deployments, and intelligent automation strategies.

After transforming how software is developed, DevOps is now transforming how enterprises manage cybersecurity. This emerging area is called DevSecOps, and it's the best approach out there for securing cloud applications and the devices that host them.

DevOps expands to integrate security

As DevOps organizations mature, it's a natural step to integrate security and compliance practices, knocking down the walls between security compliance and development teams. This dynamic is most visible in regulated industries like finance or healthcare, where applications must be approved by a security compliance team that applies rules from a regulatory document. That team passes the app back to the developers to implement needed changes, who then send the latest app version back to security compliance. This back-and-forth exchange continues until the security compliance team is satisfied.

In DevSecOps, security review processes are built into application planning and development processes. The security team communicates with the devs in real time, improving knowledge sharing and integrating once separate goals and processes.

If you're getting started with DevSecOps, start by looking at how you will bring security into application development. It is fundamentally a process engineering, communication, and culture shift project.

DevSecOps flows seamlessly from DevOps processes

A common fear is that DevSecOps will undo all the work that went into the process improvements that make up your well-oiled DevOps machine. Good news: You can get started with DevSecOps at any time, and with ***no disruption*** to your DevOps processes that are already working.

That doesn't mean DevSecOps comes for free. It does require updated processes, training, and new automation tooling. Those costs should always be compared to the value created by improved risk management. When security is embedded into app development (and not approached as a late-stage add on), you get better security and lower incidence rates (and the expense) of security breaches.

DevSecOps is even more impactful in regulated industries

Regulated industries are a great test case for DevSecOps. That's because the status quo (overwhelming amounts of compliance requirements) is dreadfully slow and expensive. One of the largest global banks has 200 pages of rules, written out in plain English, that are painstakingly applied to every app version before release. That compliance "step" can take months of back and forth to complete.

The DevSecOps approach to this use case starts with translating rules into code and ensuring developers know when to use that compliant code and security logic in their microservices. Compliance becomes foundational — not a slow, after-the-fact review process. And when your developers are thinking about security during development, that reduces your compliance burden down the line. Your overworked security and compliance teams will thank you.

DevSecOps in action

I once spoke to a product leader at a company in the financial tech industry who had to go through six months of compliance testing after code complete for a product launch. As you can imagine, waiting six months after code complete before you launch your product is both bad for team morale as well as for customer innovation. All because compliance and security were not integrated with the other stages of the development process.

If you integrate compliance testing into every step — by practicing DevSecOps — the result is zero months of compliance testing at the end of the development process

Device fleets add a new layer of vulnerability. To combat that, today's organizations need to ingrain security into the entire development process rather than tack it on at the end like an afterthought.

The benefits of DevSecOps are, in many ways, nearly identical to those of DevOps: Teams move faster, communicate more, and code reaches deployment sooner (and more frequently). Rather than looking at compliance and security as roadblocks, DevSecOps views them as pipelines to be widened — with proven processes and modern tooling. Implementing practices of continuous improvement, rapid deployment, and automation put you in a strong position to adapt those very same practices for security.

7

Managing Devices at Scale

In This Chapter:

- The challenges you'll face scaling DevOps for devices
- Tools for scaling DevOps for devices
- A DevOps for devices maturity scale

In this chapter, I will explain four key principles to help you overcome challenges as your fleet grows in size and complexity. I learned various aspects of each of these the hard way as we brought DevOps principles to our device teams at Microsoft and Amazon. My team at Esper and I have helped many customers avoid these pitfalls, and I hope to help you as well.

1: Manual processes fail at scale

So much of DevOps revolves around speeding up processes with systems and tools to automate away rotework, allowing developers and operations teams alike to focus on their core business functions. When it comes to manual processes, the question of failure is not if, but when. According to Smartsheet's Automation in the Workplace report, where they surveyed 1,000 information workers, the top two problems that automation can solve are reducing wasted time on repetitive work and eliminating human error.[11]

When it comes to device configuration, deployment, monitoring, and updating, many of the tasks are rotework: applying the same changes to all or many devices, following the same set of steps each time. One company that I worked with had a near-100 page instruction manual that needed to be followed for every new device deployed. That's a lot of potential for human error, and as a result, it needs to be followed incredibly carefully, which takes time. I'm sure you agree that the time and resources following those 100 pages could be better spent on higher value tasks. With a DevOps approach to device provisioning, we were able to automate many of the steps and reduce them by as much as 80%.

(2017). Automation in the Workplace 2017 [Review of Automation in the Workplace 2017]. Smartsheet.
https://www.smartsheet.com/sites/default/files/smartsheet-automation-workplace.pdf

Manual-to-scripted in action:

Andi's Coffee Shops have self-serve kiosks in the drive-thru. To make sure the kiosks get all the necessary updates and perform optimally, they need to be wiped, rebooted, and re-provisioned regularly. In the manual world, before they open every morning, an employee manually reboots the kiosk. If Andi's only ran a few coffee shops with a few kiosks, it would be easy to ensure this manual task is done daily. But Andi's has hundreds of coffee shops with hundreds of kiosks and relies on many more people to complete this manual task. Instead, with DevOps, Andi's can write a script to perform the wipe, reboot, and re-provision operation and set it to run across the entire fleet at 5:00 am every morning in the local time of each location. They can even take one more step and automate the reboot operation to only perform the reboot when a new version is available or an event is triggered.

This is a simple example, but you can see how scripting reduces the chances of human error, reduces the chance that the kiosk is running an old version or is offline for the day, and ultimately improves the customer experience.

2: Managing by exception

In Chapter 5, Observability, I discussed one major challenge of scaling your device fleet: as your fleet grows, it becomes impractical and often impossible to monitor the state of every device. For fleets of hundreds or thousands of devices or more, you need observability tooling in place to help you monitor and manage by exception. The point of this is that rather than looking at every individual device, you only look at the ones creating problems.

When it comes to devices, you need to know when a device has drifted away from its desired configuration. Unlike cloud servers, which are largely fungible, there are many unique reasons why devices drift. To name a few, people (e.g., employees, customers) are constantly touching them, and they can be in different environments with sometimes harsh conditions such as industrial kitchens or outside in freezing temperatures.

Breaking down the concept of drift even more, there are two types of drift: accidental drift and managed drift. It is critical to know which bucket your drift falls under.

Accidental drift can happen for multiple reasons, but most often is due to manual error. Perhaps someone accidentally unplugged a device, and it was offline when it was supposed to receive an update. Or it could be as simple as an employee accidentally taking the device home with them instead of leaving it at the store. At scale, with inevitable accidental drift, you need anomaly detection and alerting to let your team know in a timely manner. And with remote monitoring or by reviewing logs, you can identify the issue and resolve it.

Managed drift is when you intentionally set a device to deviate from the norm. This is often the case in test labs, demo devices, or in early stages of staged rollouts. These intentionally different devices are sometimes called snowflake devices because they have unique configurations. It's important to be able to distinguish between devices that have managed drift and those that have accidental drift because you don't want to set off unnecessary alarms. You also want to be able to track and monitor those devices specifically because eventually you'll want an automated way to return them back to the standard configuration after a specific event or elapsed time.

Alerts in action:

Andi's Coffee Shops likes to stay at the cutting edge of in-store customer experiences. So, in addition to their hundreds of stores, they also have a private test lab, where they can test their dev team's latest and greatest app updates before they are rolled out widely. This morning, all of the point of sale (POS) devices are supposed to be updated to and running version 8.2 of their POS app. With robust monitoring and anomaly detection in place Andi's fleet manager gets a text message alert at 5:10 am that one device is still running version 8.1. Looking at the telemetry data, she can see that the device is offline and also uses ethernet for power. Easy to solve.

Now monitoring all the devices, she also sees that two devices are running version 8.3. However, she can see that they are in the test lab device group. The drift was planned, and therefore no alert was sent.

3: Organize your devices rigorously

Since managing device fleets gets more difficult as the device count grows, it's helpful if you can think of multiple devices as one. Enter the concept of groups.

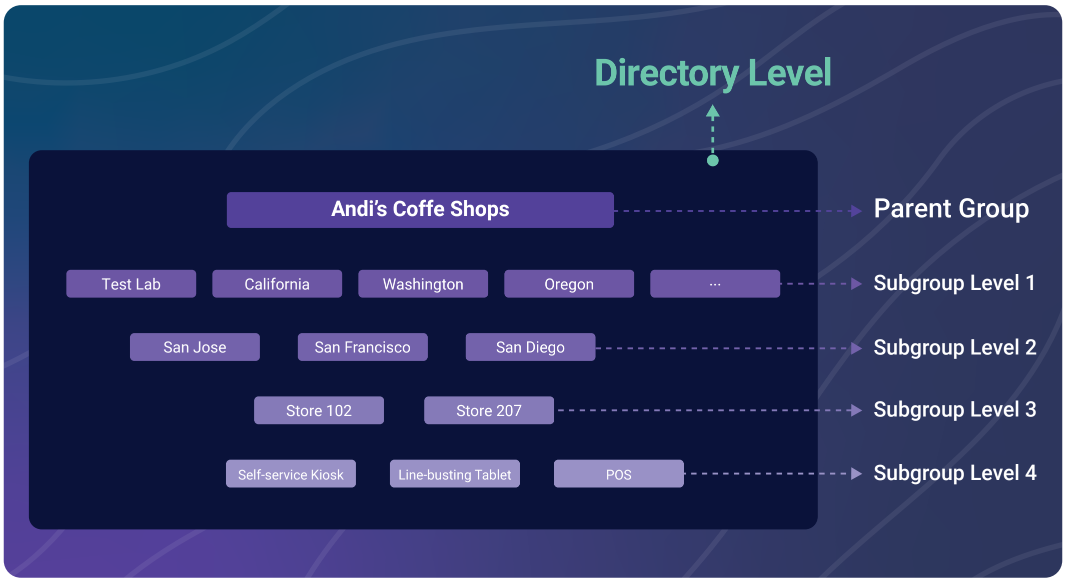
Rather than hosting thousands of devices in a single inventory, creating groups simplifies fleet management through a smaller collection of device configurations and content.

How you group devices is dependent on your unique business, logistical, and technical needs. Common ways to group devices are by location, device type or functionality, or common configuration. You will also likely want to create sub-group hierarchies (e.g., a device type group within a location group) for more granular control of devices within a group.

Not only do grouping and staged rollouts make it easier to manage large fleets of devices, but it also makes it possible to reduce the risk of any update. Groups are a way to finely control the consequences of a single action, both positively and negatively. For example, a restaurant chain may want to group every other POS terminal in a store together so that when they roll out an update they can roll it out to half of the terminals per pipeline stage. If the update fails and temporarily takes those devices offline, the other half will still work. Fortunately, because you can also automate the successive rollout to stages with larger groups based on desired outcomes, you can scale the positive consequences without scaling the negative.

Groups in action: Andi's Coffee Shops operates 1,000 devices across all of its locations. However, there are not 1,000 unique configurations with unique functionality. In the diagram below, we can see how a device hierarchy consisting of five levels can enable Andi's Coffee Shops to more easily manage their fleet.

Andi's is testing a new menu item and wants to first roll it out to their test store, Store 102 in San Francisco. To do so, they simply direct a pipeline run to update the devices in that subgroup in Subgroup Level 3. When the new menu item proves popular, they can direct that same pipeline to update all the stores in California in Subgroup Level 1 and so on. No matter the grouping level, this makes deploying an update to a group of devices as easy as deploying an update to a single device.



4: Plan to scale before you think you have to

When you have 10 devices, you can pretty much do everything manually. Even if it takes 10, 15, or 20 minutes per device, you can feasibly do it. You can touch every device, plug a USB stick in, and make an update. It would even be possible to send someone out in the field if that's where the 10 devices are.

But as you get to 100, 200, 1,000 devices, and more, all of that becomes prohibitively expensive. To do some quick math, 10 minutes per device for 10 devices is 100 minutes. But 10 minutes per device for a fleet of 1,000 devices is 10,000 minutes. That's a month of work! If you don't put a plan in place early, you'll eat up time and resources before you know it. Invest in the infrastructure early, and you'll thank yourself later.

Planning to scale in action:

Provisioning (or enrolling, staging and kitting, etc.) a device is the process of installing device management software onto the device with the desired device configuration. On Android, for example, there are many provision methods such as 6-Tap and Android for Work that may take 10-15 minutes of hands-on device time per device. Depending on the complexity of the configuration recipe, it can take magnitudes longer.

For the first five locations, Andi's Coffee Shops outsourced device provisioning to a staging and kitting firm, which used the Android for Work method to provision new devices to the required configuration. As Andi's Coffee Shops expanded from five stores to 100 stores, where they'd eventually have thousands of devices, the cost of outsourcing the staging and kitting was going to quickly add up.

Thinking about scale early, Andi's implemented a solution to mitigate this future expense. They chose to go the Android Open Source Project (AOSP) route and, with Esper, built their own flavor of the Android OS for their devices with Esper Foundation for Android. This enables Seamless Provisioning, the ability to work with the device manufacturer to flash the Esper Agent onto the device. Andi's can then ship that device straight from the manufacturer to the new store locations, and once the device boots, the Esper Agent loads and automatically provisions the device. No manual tasks, no staging and kitting.

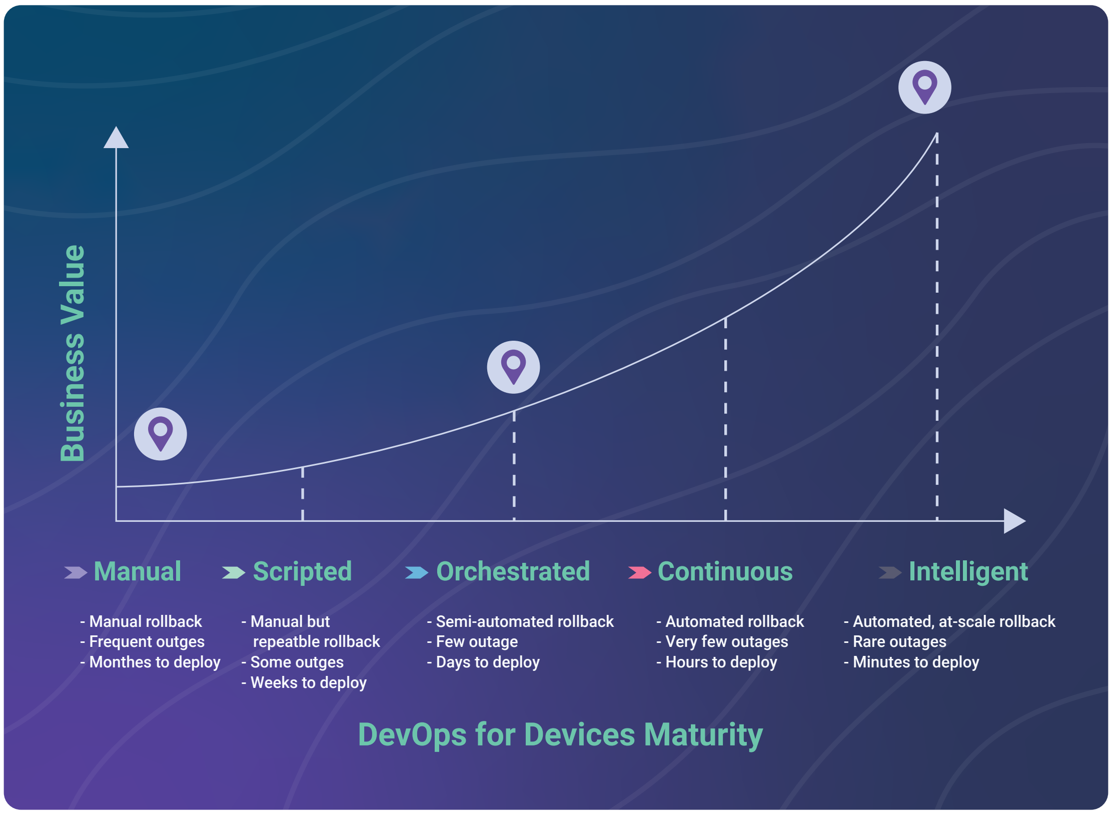
When it comes to devices, thinking about scale becomes really important much earlier than you think. Not only do you need to change your approach to monitoring, but you want to automate as much as possible because manual tasks inevitably fail at scale. You also want to organize and group devices so that you can reduce (conceptually and process-wise) the number of devices you need to manage. And finally, you want to think strategically about how you will manage at scale and put a plan and infrastructure in place before it's too late.

Growing in DevOps for devices maturity

Just as I advise all customers, you do not need to jump from 0 to 100 all at once. In fact, significant shifts such as this need to be an evolution, bringing along people, processes, and tools together.

Fortunately, as your device fleet grows in volume and complexity, you will actually be forced to evolve and mature. When you start to grow beyond what you can handle manually, the challenges I shared earlier are very painful. I hope you take my advice and learn from my experience so you don't feel that pain. Conversely, because the pain is significant, the business value you realize by automating tasks, organizing your fleet, and putting in the infrastructure will also be significant.

To that end, I suggest using the following maturity curve to help guide your evolution. No matter where you self-identify on the curve, you can use it as a tool to see how to get to the next phase until you reach your North Star, which is, in my opinion, an Intelligent DevOps organization.



Additionally, you may find the following traits helpful in identifying and driving self-awareness in your maturity stage. Manual is often correlated with Early Stage DevOps, Orchestrated is often correlated with Team Scale DevOps, and Intelligent is often correlated with Enterprise Scale DevOps.

Early Stage:

- Primarily UI or console-driven
- Siloed delivery teams
- Manual build, testing, and deployment
- Small number of off-the-shelf devices
- Basic management and basic monitoring

Team Scale:

- UI/Console and script-driven
- DevOps teams
- Automated and scaled operations through CI/CD
- Larger number of off-the-shelf devices
- Advanced monitoring

Enterprise Scale:

- APIs, scripts, and integration-driven
- Scaled DevOps teams
- Intelligent automation and canary (staged) deployments
- Very large number of custom devices (or mixed off-the-shelf and custom fleet)
- Advanced telemetry and anomaly detection

The North Star

The Intelligent stage is the last stage of the DevOps for Devices maturity model. When you are here, DevOps principles are embraced across the enterprise. You reduce incidents to very rare occurrences and can very quickly (in minutes) push fixes and updates to your systems. You can manage hundreds of thousands of devices seamlessly, you implement automated canary (staged) deployment methods, and you have advanced monitoring, forecasting, and anomaly detection processes. From app to hardware in the field, your infrastructure systems allow for frictionless updates and upgrades.

Most importantly, the time your teams save not doing manual coding, patching, or configuration is spent on customer thinking and innovation. And from your customers' perspective, little do they know how much work it took to scale up the DevOps program that made it possible to streamline everything from app development to management of your fleet of smart devices in the field. They just know that every interaction with your company is consistently great.

Getting here is a journey, not a destination. Every organization is at different stages along the DevOps for devices maturity curve, and as you grow along the maturity model stages, your business value and your ability to deliver exceptional customer experiences increase.

The DevOps for Devices Action Plan

DevOps for Devices First Steps

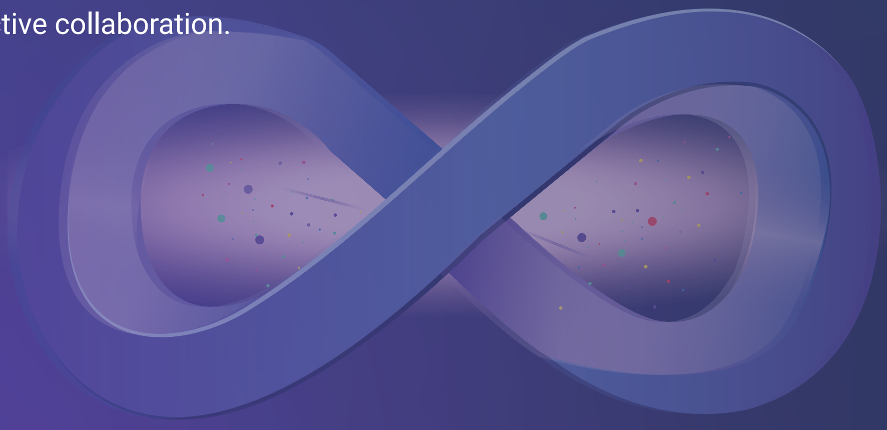
Successfully managing your devices with DevOps requires an enterprise-wide investment in automation, autonomy, trust, and collaboration. Early on, this can mean large-scale changes for organizations new to DevOps. Here are the key things to know about getting started.

Eliminate points of friction

You can't fix something if you don't know what the problem is. DevOps asks us to continually work to identify points of friction throughout our teams, process, and tools. When you expand DevOps to your devices, encompassing new tools and processes will inevitably reveal new points of friction. Plan for that by deliberately adding new elements to your systems and jettison approaches that don't work well.

Break down team silos

Most organizations say that communication between teams is the biggest challenge in integrating DevOps tools and methodologies. Competing priorities and resources, another artifact of siloed teams, is another big one.
When you're practicing DevOps for devices, you'll need open collaboration between not just your developer and operations teams but your hardware teams as well. Getting them to work seamlessly together can be challenging. Start with improving communication between teams, creating shared goals, and celebrating each other's accomplishments. These lay the essential groundwork for them to work smoothly together— sharing creates trust, and with trust comes better, more effective collaboration.



Prioritize efficiency

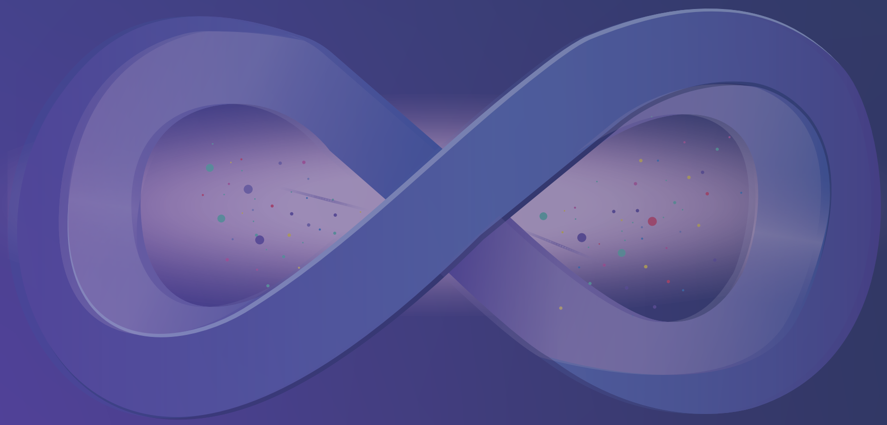
Speed matters in DevOps. It matters even more when you're applying DevOps to your mission-critical, revenue-generating devices. Whether it's trying to reboot a failed device or push out an update to better the user's experience, you need to move quickly and effectively.

Moving efficiently requires automation. It may seem obvious, but we can often overlook inefficiencies if we're used to them. Find anywhere in your processes you can automate tasks, consolidate tools, streamline approvals, or give employees more autonomy. There is no automation too small. Daily progress has a way of adding up and, perhaps more importantly, can help you reduce the impact of mistakes.

There Are No Minimum Requirements for Device DevOps

You don't need to wait to reach a particular point of maturity before expanding DevOps to encompass your device fleet management. Instead, let your current state inform your expectations about the impact DevOps for devices will have today, tomorrow, and over the long term.

Continuous improvement is a core principle in DevOps—let that guide your progress and expectations. ***You don't need to achieve Tesla-level autonomous updates overnight — but nor should you think that's out of your reach.***



The Benefits at the End of the DevOps Rainbow

DevOps for Devices First Steps

Running your dedicated device operations the DevOps way is immediately impactful, creating sweeping change across your organization and for your customers. The benefits are many — here are a few standouts:

Faster time to market

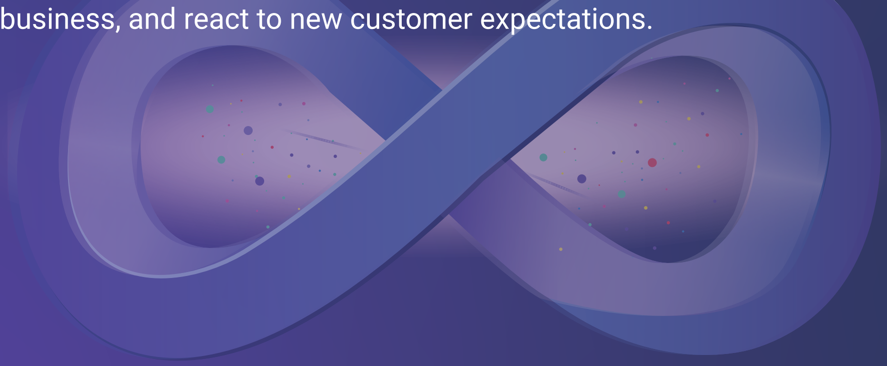
Once you've applied DevOps principles to your team, processes, and devices, you'll create a more agile company. One that has the processes, visibility, and control to deliver faster and react at high speed. You can move at the speed of your customers' expectations, turning ideas into market ready products in record time.

Better customer and fleet manager experiences

DevOps on devices infinitely improves how you can best serve your customers and employees. With these practices, you'll be able to continuously deliver personalized, immersive experiences to customers on the devices they rely on to work, play, and connect. Fleet managers, on the other hand, can use automated processes to manage menial device management tasks, minimize headache inducing fire drills, and feel confident in the performance of their fleet.

Long term foundation for success

While DevOps for devices may be a big change for some, it's one that will bring you benefits for the long term. These practices create a foundation for you to continuously manage your mission-critical devices effectively. One that's agile, not set in stone. With them, you can continue to manage your devices with efficiency even as you scale and diversify your fleet, make changes based on the needs of your business, and react to new customer expectations.



Acknowledgements

So many people came together in order to help me create this book. I am constantly filled with gratitude to work with a team full of people who take ownership, work hard, and are kind.

First and foremost, for the support in writing this book, thank you to Jordan Con, Alexandra Deane, David Ruddock, and Brian Walker. I am grateful to you all for transforming my musings into a concise and thoughtful manuscript. You made this book a reality.

At the various stages of writing this book, I benefited from the guidance of many of the experts on my team. Thank you to Cole Jaillet, Sudhir Reddy, and Chris Stirrat, who have each influenced this book in a meaningful way. Their insights are woven into this book, and it would not be the same without them.

I'd also like to thank Emily Carrion, who read early versions of this book. This book has benefited greatly from your feedback and advice. I am indebted to you for turning the first drafts of this book into an insightful read.

To Dinesha Kumar and Veeresh Antapur, thank you for the beautiful cover of this book. Your designs never cease to amaze me.

Thank you to Kaleen Skersies for defying logic and making time in my schedule to dedicate to this book. I am, as always, deeply grateful for your help.

I am extremely grateful to my wife Dharini, who has supported me throughout the process of starting and growing Esper to where it is. Her encouragement and patience have allowed me to focus on my passion and bring our vision to reality.

And finally, to anyone who has taken the time to read this book, thank you. Your time and attention are valuable, and I appreciate you choosing to spend some of it with me.

About the Author

Yadhu is Co-Founder & CEO at Esper, a leader in DevOps for Devices. He is the visionary behind Esper's mission to bring DevOps to smart Edge devices in the enterprise and is responsible for Esper's product innovation, marketing, customer success, finance, and people teams. Under Yadhu's leadership, Esper was named on three of Built In Seattle's "best of" lists: ranking 18 out of 100 on Seattle Best Places To Work, three out of 100 on Seattle Best Midsize Companies To Work, and 17 out of 50 on Seattle Best Paying Companies. Esper was also honored as a G2 2022 Best Software Award Winner in the Best IT Management Products category, ranking #13 out of 50 and the top-ranked in the fleet and device management space.

Yadhu has over 25 years of experience and 35 patents in embedded systems and security. He has held engineering leadership roles at Microsoft as Chief Architect of the Windows CE and Windows Phone. At Amazon, he designed back-end solutions for FireOS and AWS before owning Systems Engineering for Amazon Go. Yadhu has a BS in Electrical and Electronics Engineering from the University of Maine and an MS in Electrical and Electronics Engineering from Auburn University. He lives in the greater Seattle area.

Summary

No matter your industry or focus, if you have a dedicated device fleet, DevOps for Devices delivers a proven way to architect and deliver the next generation of continuously improving customer experiences.

If you're struggling with how to use your device fleet as a competitive advantage, you're not alone. Revenue-generating devices are a new arena, becoming mission critical in today's age of subscription revenue, and traditional management tools lack the functionality to truly turn them into a competitive edge. In DevOps for Devices, you'll learn exactly how to bring your tools into the dedicated device age and use your devices to capture previously unattainable opportunities.

Written by Yadhu Gopalan, the visionary behind Esper, the industry's first and leading DevOps platform for Android devices. Yadhu is known for his innovation with device infrastructure, having spent over two decades developing software and leading engineering teams at Amazon and Microsoft. In DevOps for Devices, he draws on his experiences to provide an easy-to-understand guide for how to apply DevOps to your dedicated device fleet.

DevOps for Devices will reshape your entire approach to managing devices. From deploying updates confidently to implementing the right security practices to easily scaling your fleet, you'll walk away with the answers to your toughest fleet management challenges.