



The Practical Guide to Preparing Edge Device Fleets for the Future

Where AI, DevOps, IoT, Security, and Operationalizing the Edge Converge

By Sudhir Reddy

Foreword by Yadhu Gopalan



Table of Contents

Chapter 1: Managing by Exception	04
Chapter 2: Drift Management and Remediation	07
Chapter 3: Continuous Delivery and Automation	12
Chapter 4: Compliance and Security	19
Chapter 5: Operationalizing AI at the Edge	23
Postface	28
About the Author	29

Foreword



By Yadhu Gopalan

In the realm of artificial intelligence (AI), the ability to execute models effectively at the edge has become paramount. As we witness the rapid proliferation of IoT and edge devices along with the increasing need for those devices to make decisions in real time, the concept of edge AI has emerged as a transformative paradigm. *The Practical Guide to Preparing Edge Device Fleets for the Future* delves into the intricacies of operationalizing AI models at the edge, providing a comprehensive guide for practitioners.

The convergence of AI and edge computing offers numerous benefits, including reduced latency, improved privacy, and enhanced security. By deploying AI models closer to where the data is generated, organizations can unlock the full potential of AI at the edge while overcoming the challenges associated with cloud-based solutions.

The Practical Guide to Preparing Edge Device Fleets for the Future is structured to provide a holistic understanding of the key considerations, best practices, and technical nuances involved in streamlining device management and preparing device fleets for a future where AI deployments are critical. Whether you are a seasoned AI practitioner, a researcher seeking to expand your knowledge, or a business leader exploring the possibilities of edge AI, this book offers a comprehensive and practical guide to operationalizing AI on your devices. By leveraging the insights and best practices presented, you can effectively harness the power of AI to drive innovation and achieve transformative business outcomes.

Chapter 1:

Managing by Exception

Modern companies leverage an array of business-critical devices — everything from point of sale systems to kiosks to digital signage. As the need to get more from these devices grows, future-ready organizations are looking to artificial intelligence (AI) to elevate their device experiences.

But the path to operationalized AI at the edge is anything but linear, regardless of whether you have a device fleet of 200 or 2,000,000. You must lay some groundwork to prepare your device fleet for this inevitable future. The first step is understanding what it means to manage your devices *by exception*.

As your device fleet grows from a few to dozens to thousands, how do you ensure that **all** your devices are in compliance and working as expected? Especially with complexities where each device (or group of devices) has different settings, apps, AI models, and content to manage. What if one device is compromised and no longer in compliance? Do you look at a report of all your devices every day and determine which needs attention? In the meantime, a whole day has gone by without addressing that issue! And, as your fleet continues to grow, the complexities only exponentially multiply!

This is where managing by exception comes into play.

Imagine this: you outline your “desired state” for a group of devices. That includes how they are set up after provisioning, what apps they have installed, user settings, privacy controls, and more — *every aspect of what your device should look like and how it should function*. Your 38-page setup guide is condensed to a single document that tells a device exactly what to do — we call this a “blueprint.”

As you bring new devices online into your fleet, you want to onboard the device and then immediately provision the device to its desired state. A blueprint is what accomplishes that for you. You define the desired state, tell the device that this is its desired state, and then apply your blueprint to see the magic happen! With granular control over every single aspect of your device, blueprints capture — and enforce — that desired state after provisioning. Now, it’s all about *keeping* your devices in that desired state.



Enforcing Your Desired State

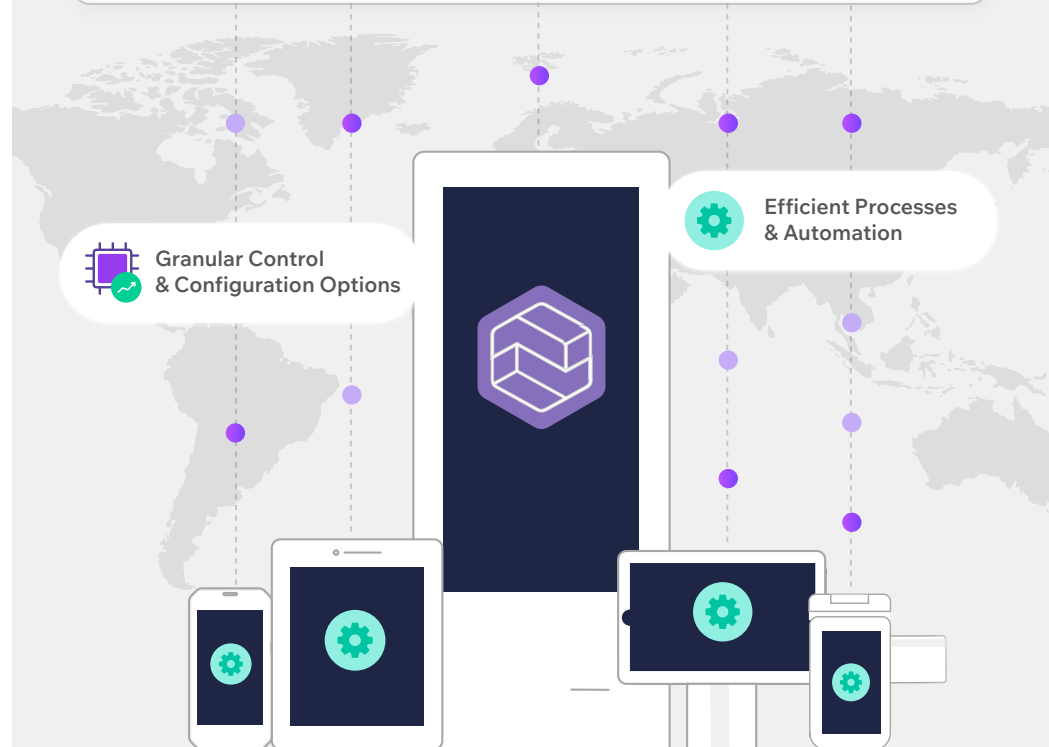
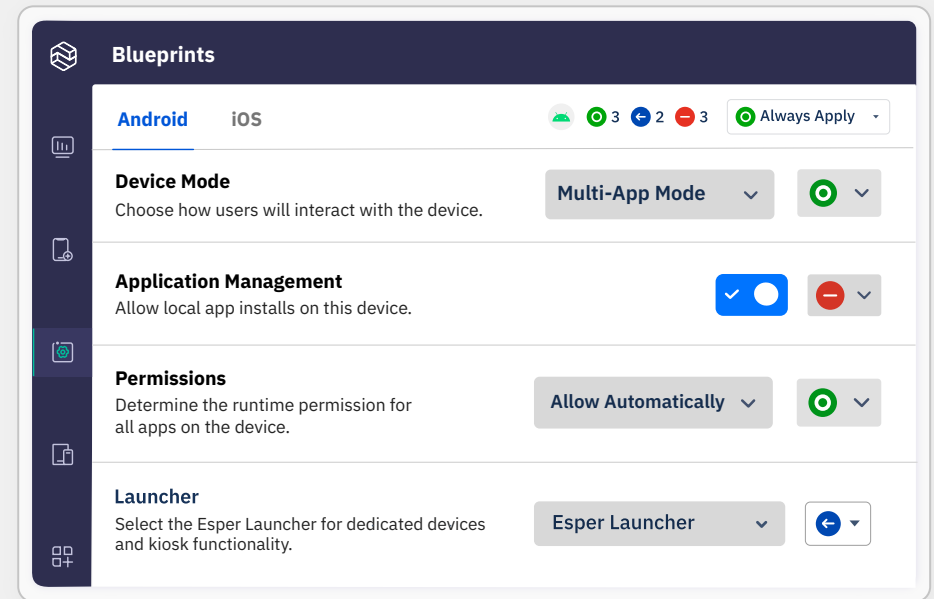
Once your device is initially in its desired state, you want to rest assured that the device will stay in that state. And, if you define a new desired state for that device (if you want a new version of your app on the device, for example), then you want to update it to that new state and make sure that configuration is enforced.

As great as that sounds, you haven't even seen the best part! What if your device management platform monitored these devices for *drift* (meaning they are no longer in that desired state you set up earlier) and then notified you via Slack, email, or a ticket in your ticketing system? You could immediately remediate this specific device and bring it back to its desired state.

That is *exactly* what “managing by exception” is! That is what [Esper Blueprints](#) enables for you (along with drift management, which we'll talk about in detail later in this book). Simple. Done. Now go enjoy that coffee break!

I internally refer to the capability of Blueprints and drift as “set it and forget it.” And yes, I do enjoy rotisserie chicken, for those of you who know the reference. 😊

Think of the alternative — every few hours or every day, you'd troll through a long report or, worse, log in and check each device to see if it's still in the state you told it to be in. And, of course, you could miss something. Disaster! Painful! Yuck!



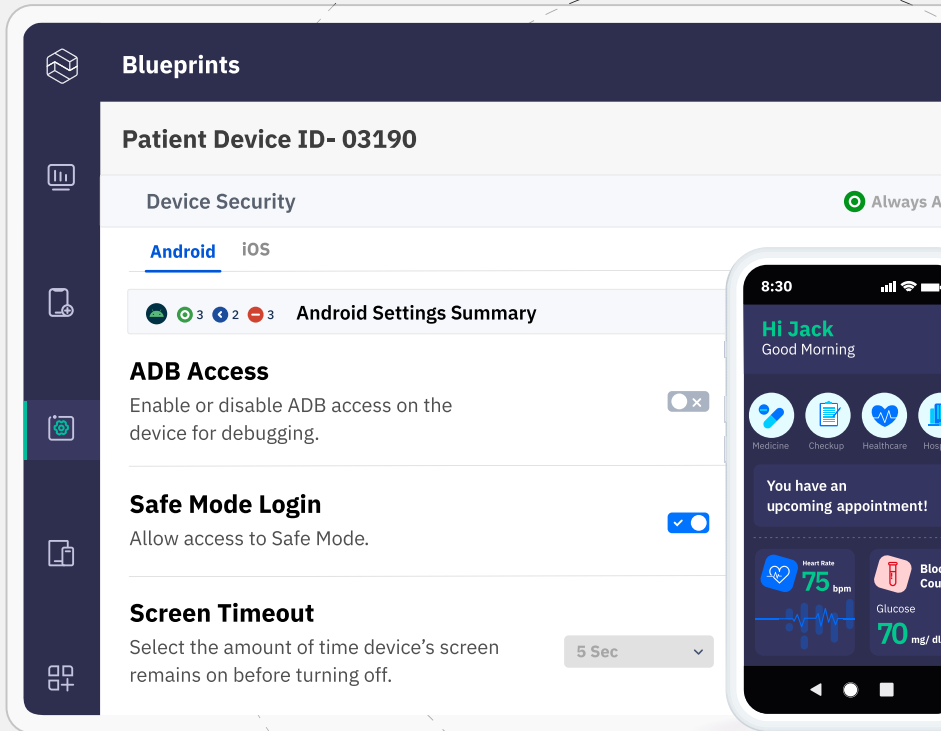
Managing by Exception is Just the Beginning

Maintaining your desired state is just the start of streamlined, scalable device management. But it's a crucial component to get right as you prepare your device strategy for the future. Failing to successfully enable a strategy that relies on managing by exception will ultimately stifle your ability to grow and scale, which is the opposite of what we're trying to achieve here.

Starting with this foundational philosophy, we believe the future we see will enable organizations to move and scale their device fleets like never before.

Esper's Blueprints feature is built on the “desired state” philosophy and is a core component of our next-gen device management platform. You set up your blueprint with the desired state for your devices, apply it at provisioning, then enforce it at scale. It's the best way to manage devices.

[Learn more about Esper Blueprints](#)



Chapter 2:

Drift Management and Remediation

Once you've learned to manage by exception, it's time to focus on the concept of drift management. A unified, systematic approach to optimizing your company hardware lies not just in managing by exception, but taking that a step further with remediation.

Drift management and remediation go together like peanut butter and jelly. Sure, they're both great on their own. But when you combine them, you get something truly extraordinary!



What is Drift?

As outlined in Chapter One, when you configure a device precisely how you want it to be, that's called the **desired state** — an all-encompassing term that covers applications and versions, screen configurations, security settings, AI models, and everything in between. With Esper, you encode your desired state into an artifact called a **blueprint**.

Now, when the device deviates from this desired state — whether it's a small change in volume or a severe change, such as improper application configuration or connection to an unauthorized Wi-Fi point — that is called **drift**.

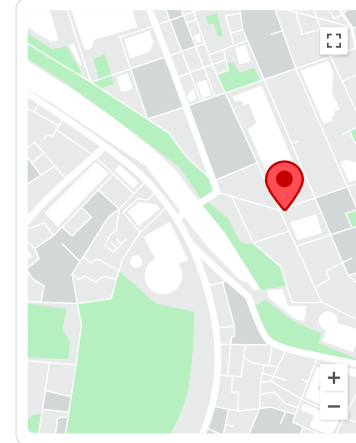
There are various reasons why a device can drift from its desired state. Software updates, a malicious actor tampering with a device, or an application causing an undesired change on the device are just a few examples. While some of these changes introduce mere inconveniences, others can dramatically affect security settings, the end user experience, or business operations (think a broken point of sale, health monitoring device, or an ATM)! The reactions to such events range anywhere from “Yuck” to “That's a serious incident — wake everyone up!!!”

Fortunately, there's a solution here, too: **Converging** is the act of restoring a device that is in drift back to its desired state. If your device is in drift, this is how you get it back into compliance.


Now, let's see why good drift detection and remediation are crucial to enforcing critical policies.

Drift Management - K10 A13 GSI

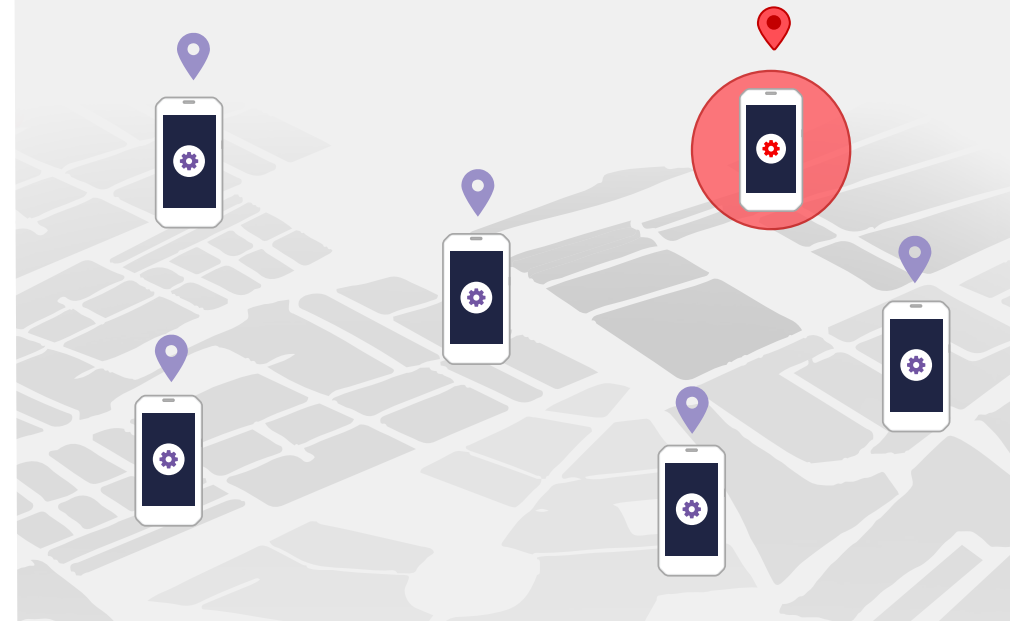
Overview



Device and Groups

 Device Fleet :	K10 A13 GSI
OS Platform & Version:	ANDROID-13
Model:	GMS on ARM64
Blueprints:	EventSprout_Test
Device State:	ACTIVE
In Drift:	YES Unauthorized app detected

Alert



A Look into the Future of Drift Detection with Automated Remediation

Imagine a tool that constantly monitors your fleet, detects every tiny change on each device, determines if the device has drifted from its desired state (while ignoring things you don't consider meaningful changes), and alerts you when the device is indeed in drift.

Now, *that* is what you need in your tool chest! Without such a tool, policy enforcement and managing device fleets becomes a laborious and manual effort. Detecting drift isn't something you should do passively, but rather have it be part of your overall fleet management strategy. The absence of this only makes managing by exception a manual hunt-and-seek effort.

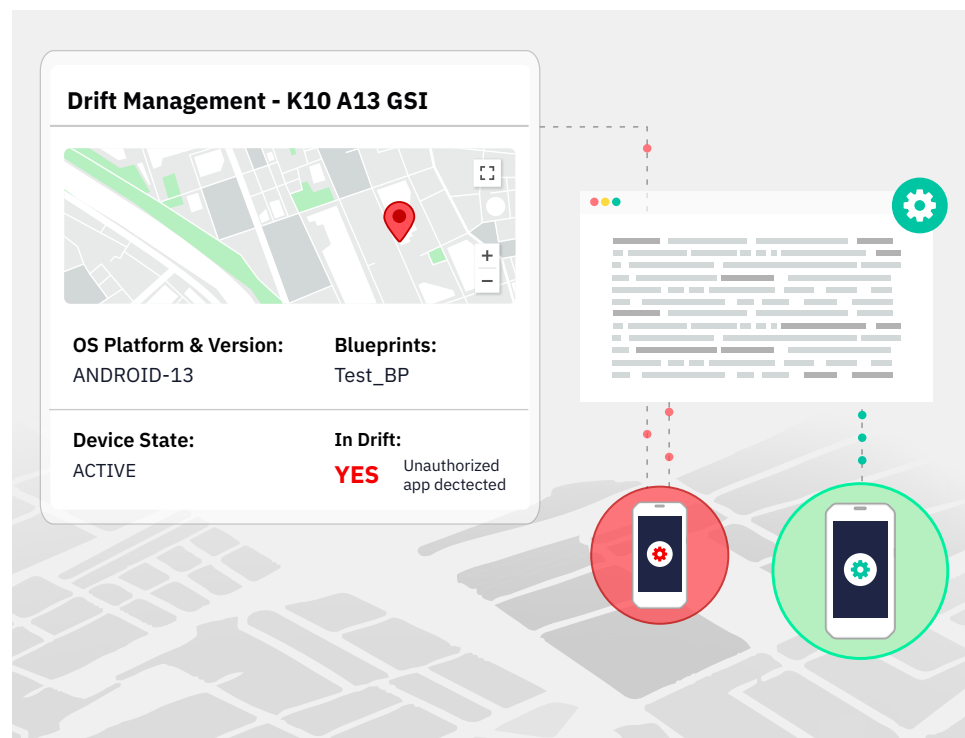
But wait! We can take this a step further — we can **remediate** this drift!

However, if good drift management relies on notifications and reactive solutions, **better drift management is automated and proactive**. That's exactly what automated remediation is about — it removes all the manual elements associated with drift management, virtually eliminating the need for manual oversight. That's next-level managing by exception! That's having your desired state cake and eating it too!

Sticking with our example above, now imagine if everything after the notification just *automatically happened*! The system detects a device went into drift, and then applies the correct desired state to bring it back up to snuff. You wake up from a good night's sleep and see a notification from your robot friend letting you know, "Howdy! I noticed things were broken, but I fixed them according to your specifications. Here are all the details.

And ... You're welcome." Bam!!! Day saved! Now, you can worry about that new project that's been sitting for a while.

Keep in mind that this isn't where the industry is as a whole right now, but it's the direction we're going. At the same time, these aren't theoretical examples, either — this is practical, tangible, and the features are in the works right now.



Let's Bring All This Together



Drift remediation is necessary — not a luxury — as you scale. The more devices you have, the harder it is to keep them all aligned with their desired state(s). Automated remediation fixes that. Period.

Consider the security implications here. **The longer a device is in drift, the more of a security risk it becomes.** And the more devices you have to manage, the higher the odds are that a device stays in drift longer than it should. In the future, automated remediation will fix problems almost in real-time, and these threats are suddenly nullified before they can even become threats. Until then, dependable drift detection is a great start!

Additionally, what happens if your security settings, application versions, or user experiences change? **The best thing about an integrated desired state, drift management, and remediation solution** is that all you have to do is update what your desired state looks like! The tool enforces that new state and automatically converges your entire fleet to the new desired state!

In the previous chapter, I said, “Set it and forget it.” Now, it’s also **“update it and forget it!”** — in a single operation!

Managing by Exception Across Device Fleets of All Sizes

As your device fleet scales and requirements change, reliable drift monitoring ensures security, compliance, and operational efficiency. It simplifies device management strategies, allowing teams to maintain control over assets in a streamlined manner, meaning they have more time to focus on strategic initiatives instead of being loaded with repetitive management duties (or worse, putting out fires).

This is yet another foundational example of managing by exception. I wouldn't go as far as saying your fleet can manage itself (we'll get there soon), but advanced drift management tools really make it feel that way.

But drift management and remediation are just the tip of the iceberg here! Let's talk about software delivery, automation, and how to operationalize AI at the edge.

Devices & Groups

Fleet T2/ 332 Devices

+Link Blueprint

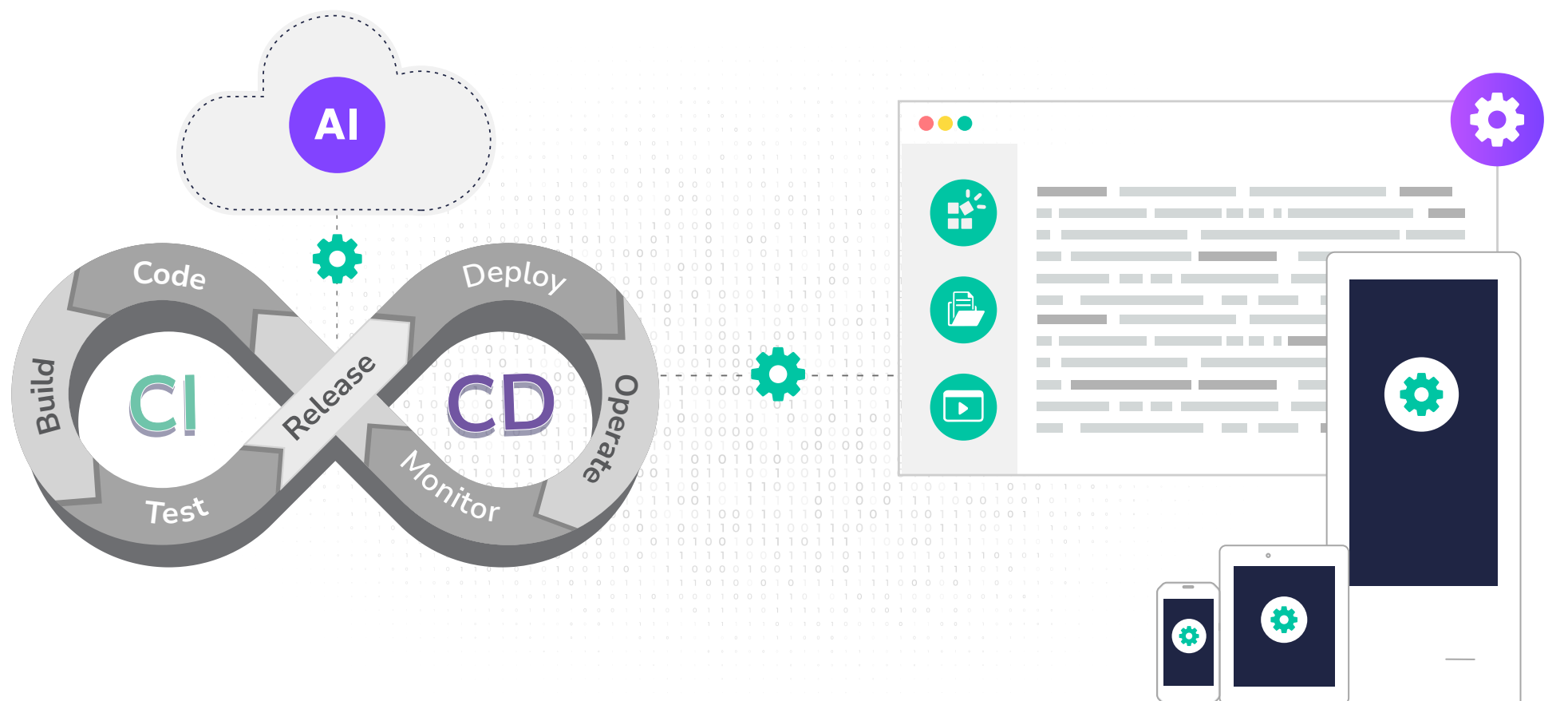
Device ID	Group Name	Blueprint	OS Platform	Actions
DEVE- LOP-AA6XL	Fleet T2	Blueprint 10	Android	...
DEVE- LOP-AA651	Fleet T2	Blueprint 10	iOS	...
DEVE- LOP-AA650	Fleet T2	Blueprint 10	iOS	...
DEVE- LOP-AAU10	Fleet T2	Blueprint 10	Android	...
DEVE- LOP-AAGLE	Fleet T2	Blueprint 10	Android	...
DEVE- LOP-AA6GO	Fleet T2	Blueprint 10	iOS	...

Chapter 3:

Continuous Delivery and Automation

Now that you're familiar with drift management, automated remediation, and managing by exception, it's time to get down to the nitty gritty with some new concepts that directly affect your edge AI strategy: **continuous delivery and automation**. (Starting to notice a trend here? Fleet automation is the future!)

The focus here is on AI models and deployment, but the beauty of this concept is that it's incredibly versatile! You can apply these philosophies to any content that you want to deploy at the edge, such as apps, files, media, and the like.



The Enterprise AI Deployment Model

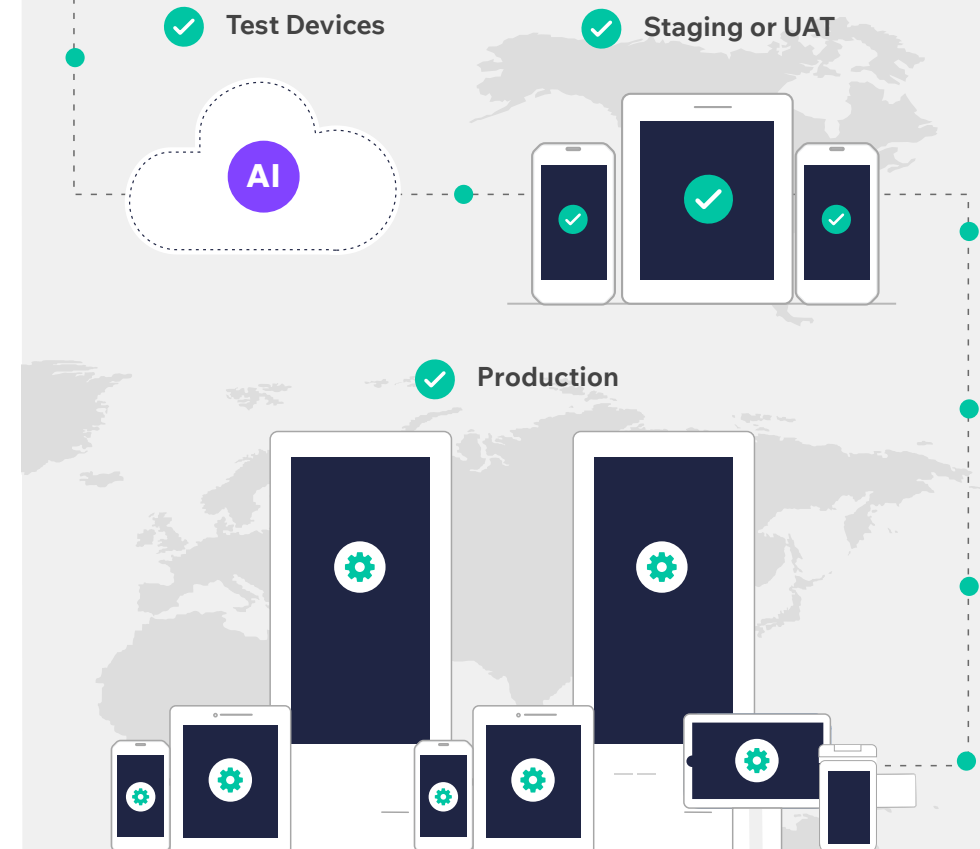
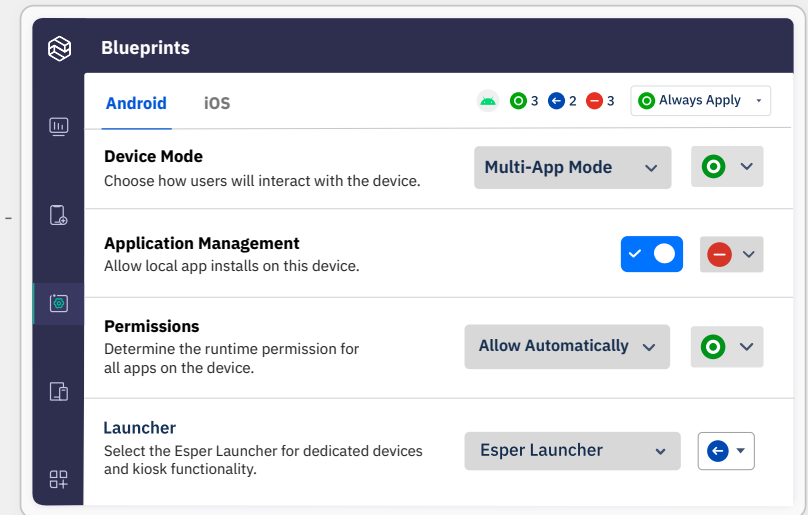
For the sake of completeness, let's take a quick walkthrough of what a typical enterprise edge AI model deployment cycle looks like:

- Once you get approval from your app development team, you'll roll out the AI model to a small set of test devices. Hopefully, you have some automated tests that can run on these devices or are manually testing to ensure the application is working as desired.
- If all tests on this small test fleet are successful, you might roll out the model to either a staging or UAT (User Acceptance Testing) environment and give some of your end users access to this environment to ensure that their expectations are met.
- Finally, you will roll the model out to your production environment. If you have a large fleet, you'll want to roll it out in stages to ensure compatibility.
- If the AI model is not up to spec at any point in this process, you will reject it and not proceed to the next stage of deployment.

Now, in a typical DevOps operation, this is called **Continuous Delivery (CD)** and is enabled via concepts such as pipelines, stages, and groups. With Esper, you have this ability at your disposal. Let's explore these in detail.

💡 **Note:**

If you're an expert in these concepts, please bear with me as I get everyone on the same page here.

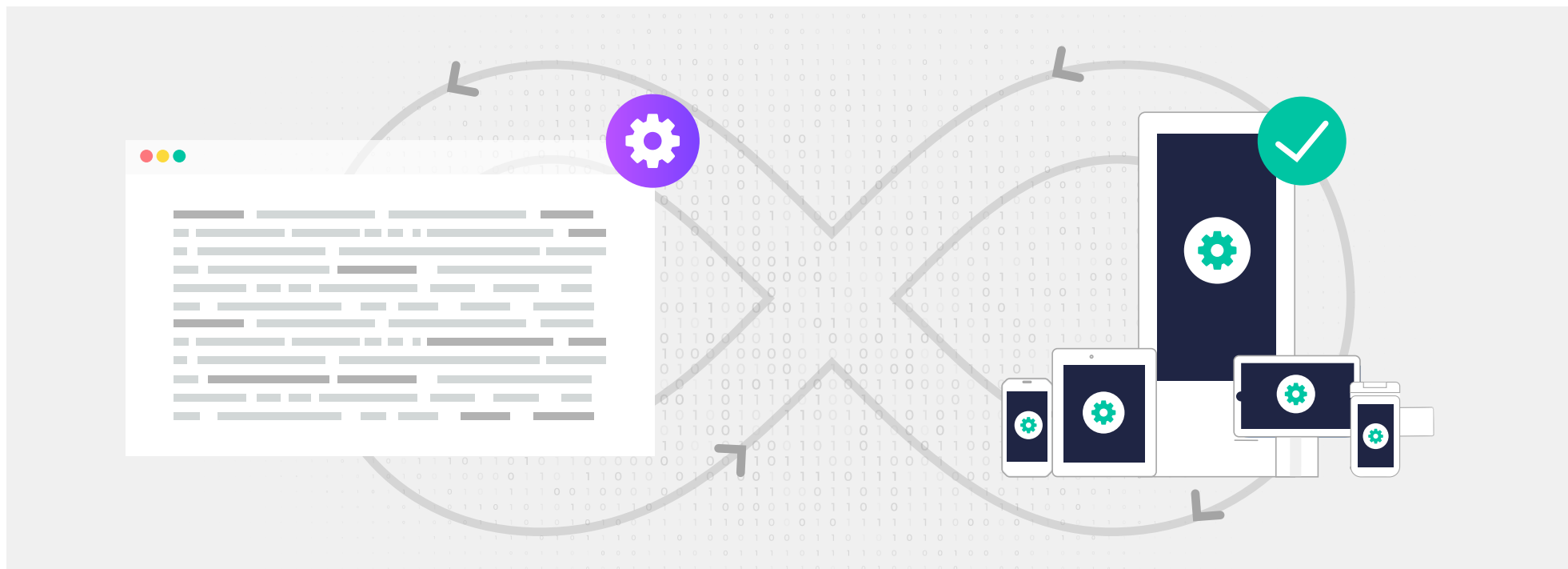


What is Continuous Delivery (CD)?

The DevOps software development philosophy hinges on the concepts of [Continuous Integration \(CI\)](#) and [Continuous Delivery \(CD\)](#). **Continuous integration (CI)** is all about how the applications are built and the development lifecycle. This is typically the realm of product engineering teams, and we'll leave that out of the scope of this chapter since it's not

relevant to the topic — awareness of CI's existence in the software development process (including AI models) is enough.

Once a new application version is developed, **Continuous Delivery (CD)** kicks in. This is typically where the team that manages production fleets gets engaged, and is the focus of this chapter.



Continuous Delivery is the Key to an Optimized Device Fleet

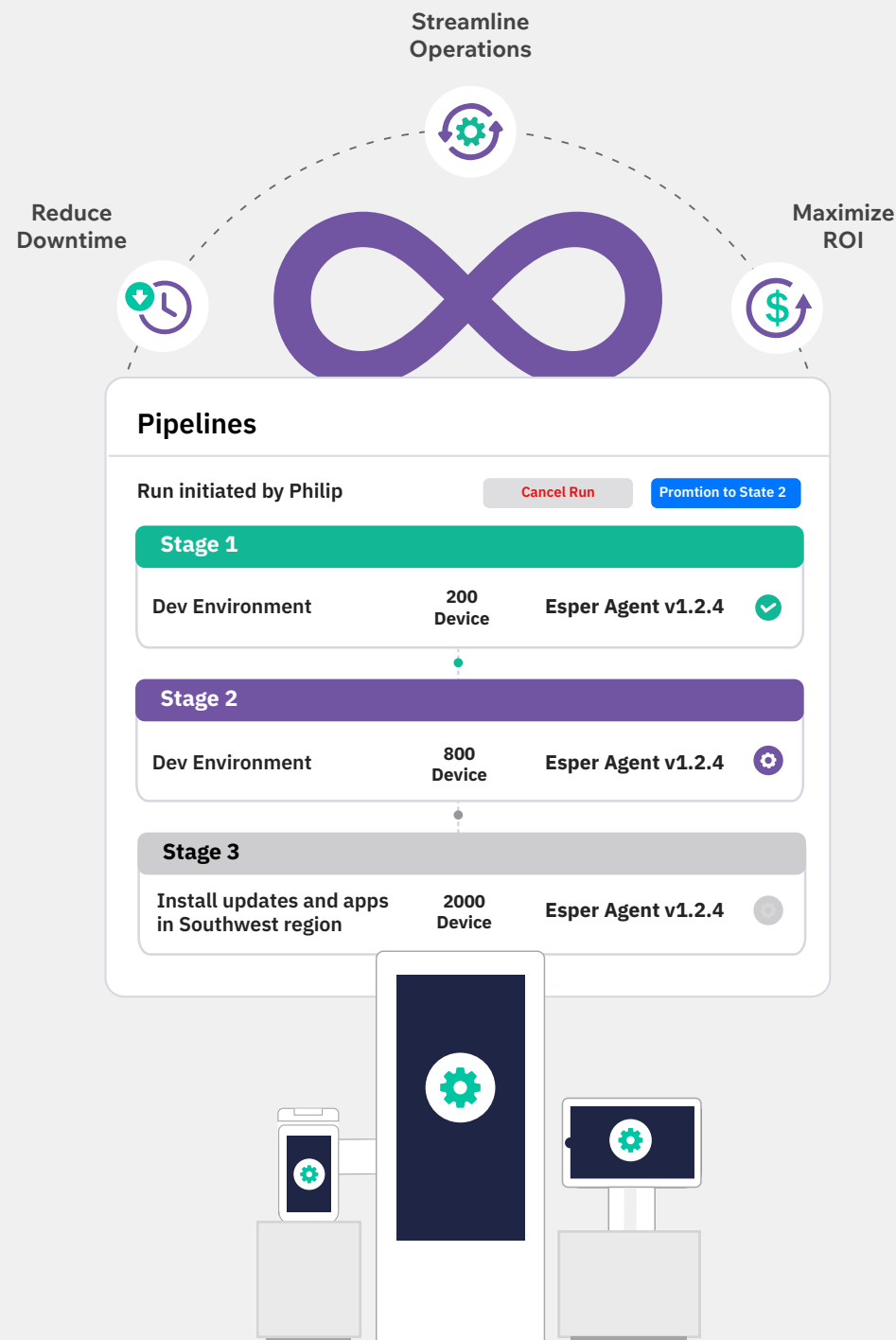
For most dedicated devices, such as kiosks, point-of-sale systems, self-checkout stations, etc., the software experience matters. For customer-facing devices, you want to put your best foot forward and provide an innovative, intuitive, user-friendly solution. For employee-use devices, you need to streamline operations, reduce downtime, and maximize ROI.

In all of these scenarios, that means constant optimizations to provide the most ideal experience. Continuous delivery is how you achieve that. But how do you get there?

Remember the pipelines thing I mentioned a bit earlier? Well, it's not just for software development!

The concept of pipelines is rooted in DevOps philosophies, but [Esper Pipelines](#)' advanced software delivery mechanisms take advantage of this idea to provide the seamless, predictable software delivery you need to optimize your device fleet without worrying about app updates — it's part of what makes our platform unique.

This is the foundation of reliable AI model deployment.



But You Can Take It a Step Further with Automation

Reliable AI model deployment is critical and predictable, repeatable software delivery is a boon for IT and development teams, but what if I told you that we can also automate the process? Using Esper Blueprints, you can automate everything from scheduling to delivery across your entire device fleet. Blueprints ties directly into Pipelines, so when you need to deploy a new version of your AI model (or app), you simply update your blueprint with the updated model, define the rollout using groups and stages in the pipeline, hit the button, and watch the platform do its thing!

The power of front-to-back automation with absolutely no unnecessary interaction is invaluable to the dev and IT teams responsible for building and deploying your AI models and apps. A side effect of full automation means teams have more time to innovate and optimize since they're not babysitting deployments.

[Learn more about Esper Pipelines](#)

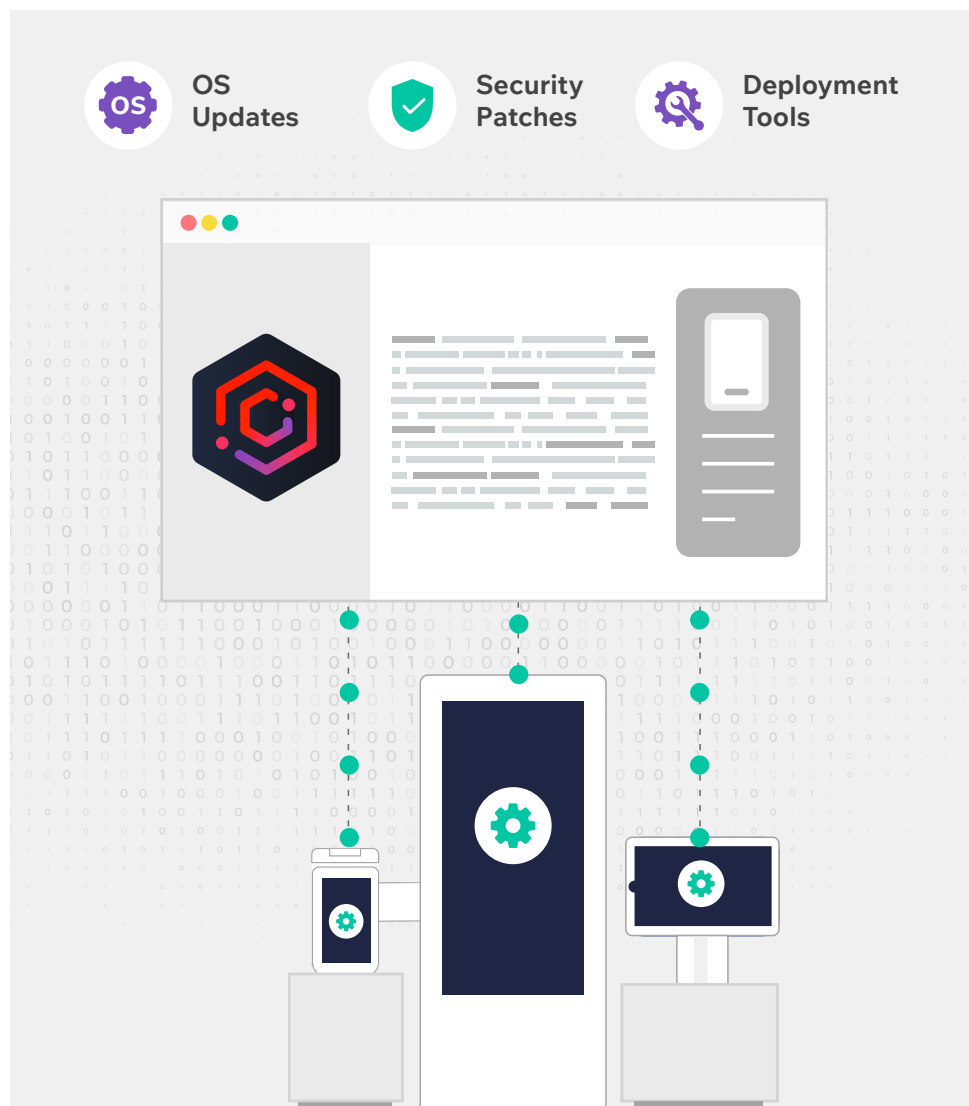
It's Not Just for AI Models or Apps, Either — You Can Streamline OS Updates, Too

Up to this point, we've focused on AI model deployments, but continuous deployment tactics and automation don't have to stop there — you can apply these philosophies to operating system updates, security patches, software, and other content, too! Users of our custom software solution [Esper Foundation for Android](#) receive regular updates for their builds. Using our OS deployment tools, they can leverage the same granular rollouts and automations as other software deployments.

That means operating system updates of any kind are fully controlled. You can apply them to your test devices to ensure full app and peripheral compatibility before deploying the update to your entire device fleet. Even then, you can stage the update to roll out using groups instead of a single bulk deployment.

Bringing the concept of drift management back around, consistent AI model, app, and software deployments are another piece of the puzzle. Or an answer to part of the problem, if you'd rather look at it like that. Compliance enforcement and drift management aren't just about software settings and ideal experiences — it's also about ensuring your devices are always running the latest version of your AI model, application, or operating system and that device security is always up to date. More about that in the next chapter.

The combination of drift management, automation, and intelligent software delivery is potent, primarily when used in concert. It's device management on another level, unlike anything you've experienced.



The Power of Predictable Software Delivery with Automation is a Game Changer

Yes, you *can* change a plane's wings as it is flying, and with Esper, you can automate that!

When we say that Esper is “next-gen device management,” *we mean it*. Predictable, repeatable, scalable software delivery at the edge has long been an issue for IT and development teams, so we built the solution. A future-forward approach to device management for modern device fleets.

Esper was founded on the principles of DevOps. Our founders saw the gaps in existing device management solutions, asked how they could improve it, and started Esper. We call this next-gen approach to device management “**DevOps for Devices**,” but if you’re not familiar with DevOps, it’s hard to piece it together. That’s why we have a resource dedicated to understanding this concept from the ground up — no software development experience necessary. We promise.

[Read *The Beginner's Guide to DevOps for Devices*](#)



Chapter 4:

Compliance and Security

Enforcing a strict compliance and security policy is crucial to a reliable device fleet, but the idea of automating such a critical part of your device fleet may sound risky. But stick with me on this one! Using many of the concepts we've covered up to this point, enforcing compliance is easy peasy.

But first, let's talk about what "compliance" actually means in the context of a device fleet.


If you already deal with compliance needs in your organization, then you know exactly what I'm talking about. For others, you are likely already "being compliant" in some way.


Depending on your industry and company, there are probably regulatory rules or compliance policies that you need to adhere to or enforce. Imagine a clinical trial setting where you are required to have devices adhere to a certain set of rules. Any deviation from these rules might render a particular study invalid — or, at the very least, you will need to report to an authority when a deviation happens.

On the other hand, there are many rules that your company (or even your team) might want to enforce in order to achieve a certain goal with your fleet. Think of rules such as:

- A specific app should always be running on your device
- Only a certain set of ports can be used
- The device should always be in kiosk mode

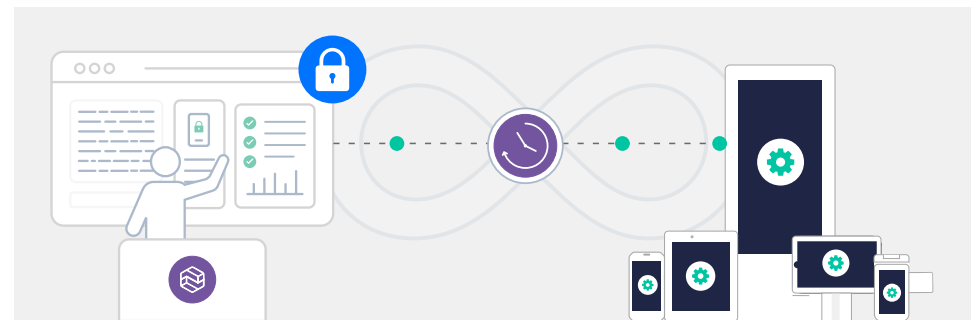
...And so on. The most common reasons why you'd have these rules are for:

 **Security:** From this perspective, compliance is a security tool. It's about ensuring your devices adhere to the best security practices and guidelines defined by your company (or authority).

 **User experience:** From this angle, compliance ensures your devices enforce the ideal customer / user experience.

This is an important distinction as you think about compliance because people often associate compliance solely with security. And while that is a critical aspect of compliance enforcement, it's not the only one! This is what I mean when I say you are most likely already "being compliant."

Further, in regulated environments, there is often a compliance officer involved who will demand a report and audit your adherence to policies. In this case, it is important to not only have the fleet compliant at any given time, but also to be able to obtain and deliver reports to the compliance officer.



Why Compliance Enforcement Is So Hard

Given the complex and nuanced nature of compliance as a concept, it's not difficult to understand why compliance enforcement is a challenge. While we consider compliance enforcement non-negotiable, we also recognize that this is a weighted scale of importance, where you must enforce critical security functions at all costs while other things might be a little less critical.

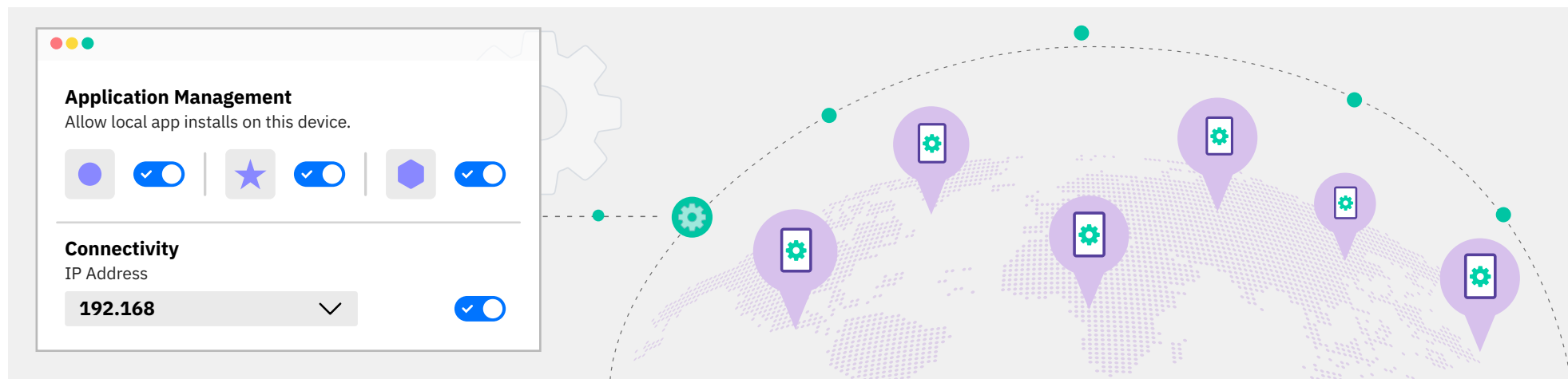
That's where the challenges of compliance enforcement start. For example, you might want to enable a company wallpaper during device provisioning to adhere to company standards. Is that something you want to enforce? Or simply enable once? That's a question only you can answer.

On the other hand, let's say all of your devices at a given location have access to three different Wi-Fi networks, and you want to ensure they don't

connect to any other network for any reason. This is almost certainly a setting you want to strictly enforce all the time — no exceptions!

And that's just two scenarios. Multiply these complexities across every setting on every device, and suddenly, you have dozens (or even hundreds) of settings that need to be enforced — or don't. That raises the question: Can you actually enforce these settings?

If you're using Esper, then yeah, of course you can! You can make policy enforcement a blanket setting or granularly control every setting and option. The importance of your company's policies is for you to decide, so we built a way to give you complete control. But we can take it a step further.



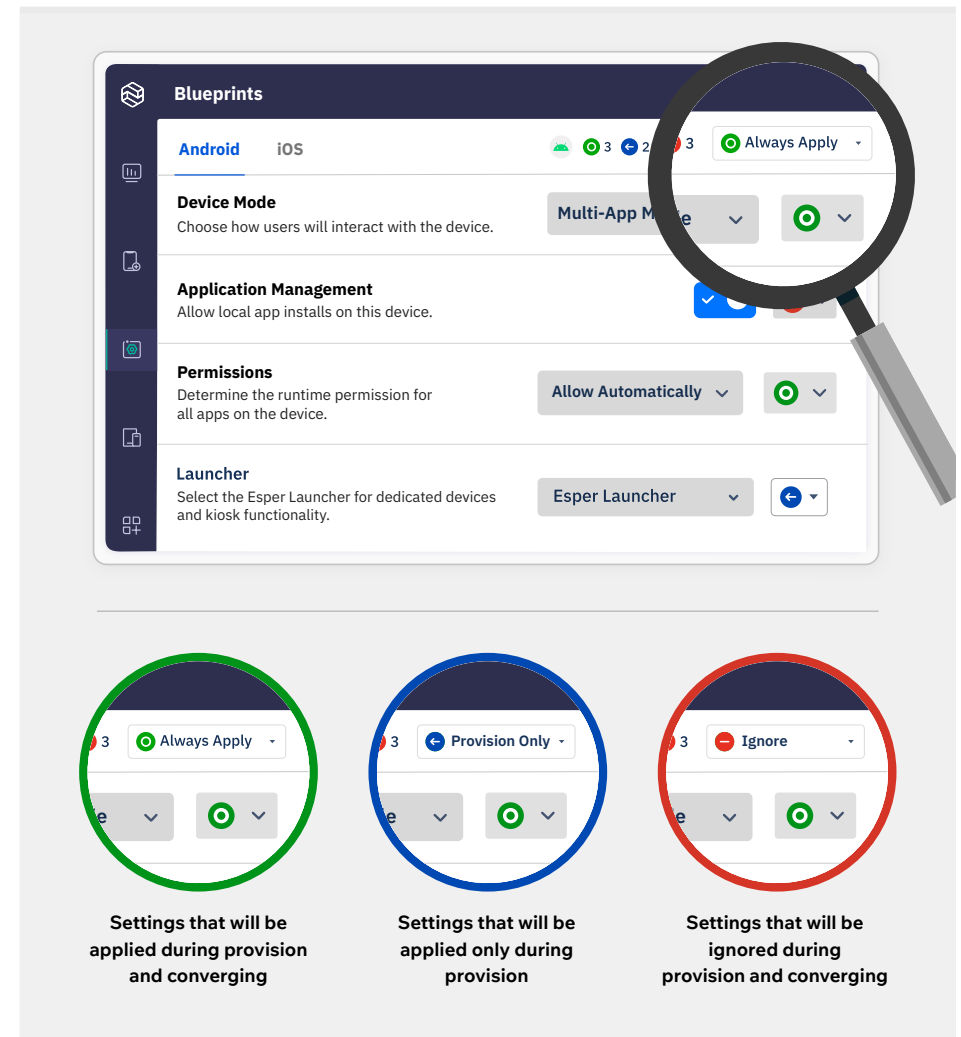
Automated Compliance Enforcement = Reliable Security and Predictable Experiences

On the Esper platform, we enable granular policy enforcement through our Blueprints feature with “when to apply” options for every setting.

In short, this allows you to define when the specific setting is enforced — either **during provisioning, always, or never**. It’s the most straightforward way to control policy enforcement at scale. This way, you can strictly enforce your Wi-Fi allow list, but only push the wallpaper at provisioning (if you so choose).

But now’s when compliance enforcement gets, dare we say, fun? Using these settings, you essentially automate the process! Sure, you *could* manually check each setting daily, weekly, or monthly, but why bother? You can simply use the “always apply” setting and have these settings automatically re-applied every time you converge the Blueprint. Talk about a time saver!

This gives you precise, granular control of every setting on every device in your fleet, which legitimately changes the game.





This is what device management at scale really looks like. It introduces a level of efficiency for Android and iOS devices like you've never experienced before.

[Learn more about Esper's software](#)

It All Comes Full Circle

This brings us back to the beginning: managing by exception and automated remediation — two core concepts we covered in chapters one and two of this book! When you can rely on drift detection to help you manage by exception, and automated remediation to strictly enforce the policies, you arrive at device management nirvana. Think about it: a self-healing device fleet that requires very little oversight and limited manual intervention. Bliss.

And, if you'll allow me to further future-cast a bit, we could take “self-healing” to another level — what if your device fleet could also self-diagnose and preemptively perform a series of actions based on predicted behavior? For example, let's say a device loses Wi-Fi connectivity. An automation might include toggling airplane mode after five minutes without a connection. If that doesn't work, the device could reboot.

To go one level deeper, what if the device could “understand” if it's the only one affected by having localized information about other devices on the same network? If all of the devices on the network lose Wi-Fi connectivity around the same time, the device could “understand” that this is likely a network-wide outage and not an issue with the device itself.

But, hey, we're just speculating here, right?

Yep.

For now. 🤔

Chapter 5:

Operationalizing AI at the Edge

In the final chapter, we combine all of the concepts we've covered so far to discuss operationalizing Artificial Intelligence (AI), Machine Learning (ML), and Computer Vision (CV) at the edge.

More specifically, we will discuss AI on edge devices, the complexities of delivering AI models to the edge, and how Esper is uniquely suited to help make AI dreams a reality for any business with a dedicated device fleet.

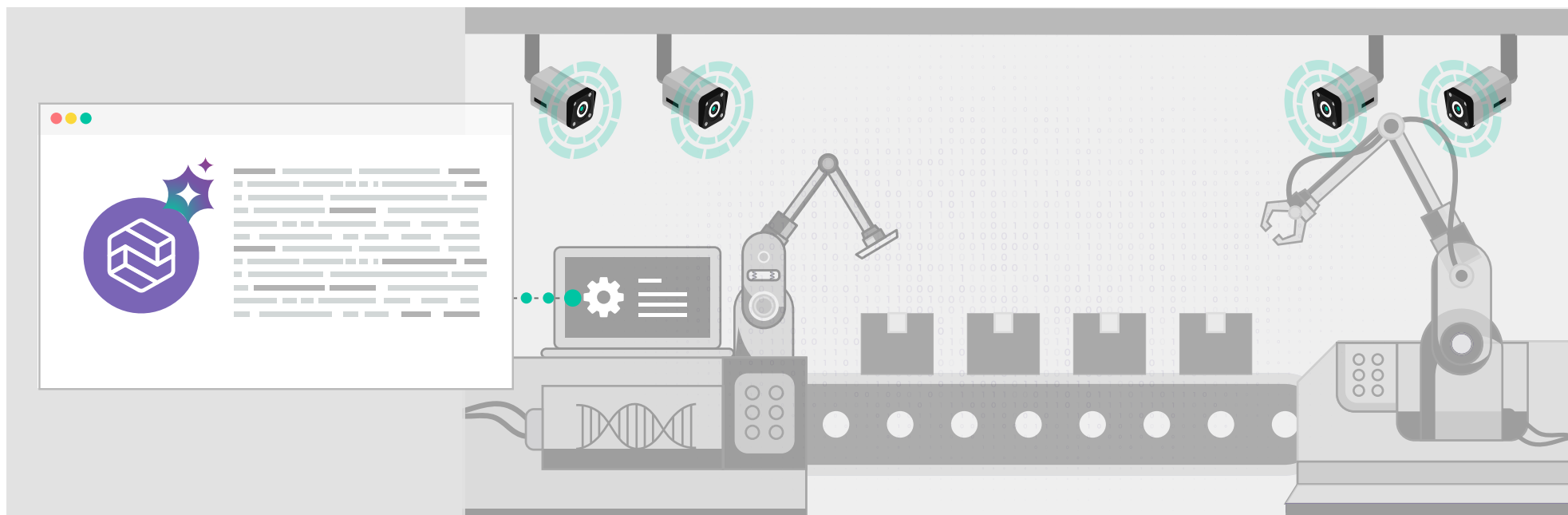


The AI Operationalization Problem: Model Delivery at the Edge

It seems like every tech-first company (and what company isn't these days?) is looking for a way to leverage AI. This isn't just a buzzword in the industry, either — it's the future. AI at the edge is no different. In modern businesses, there are numerous benefits to integrating AI into your edge devices: enhanced privacy, improved efficiency, scalability, enhanced user experiences, and so on. What makes it even more challenging, however, is delivering that AI model to the edge in a repeatable, predictable, and scalable way.

The issue here is twofold: Edge devices often need to run local AI models for latency issues and bandwidth constraints, and delivering content to them is challenging. It's quite the pickle.

For example, let's say you run a manufacturing facility and rely on a series of cameras to QA the product assembly process. These strategically-positioned cameras analyze the product from various angles and can detect anomalies or malfunctions in real time. But how can they do this with precision and accuracy without transmitting data back and forth to and from the cloud, which often takes several seconds? With localized AI running on the LAN!



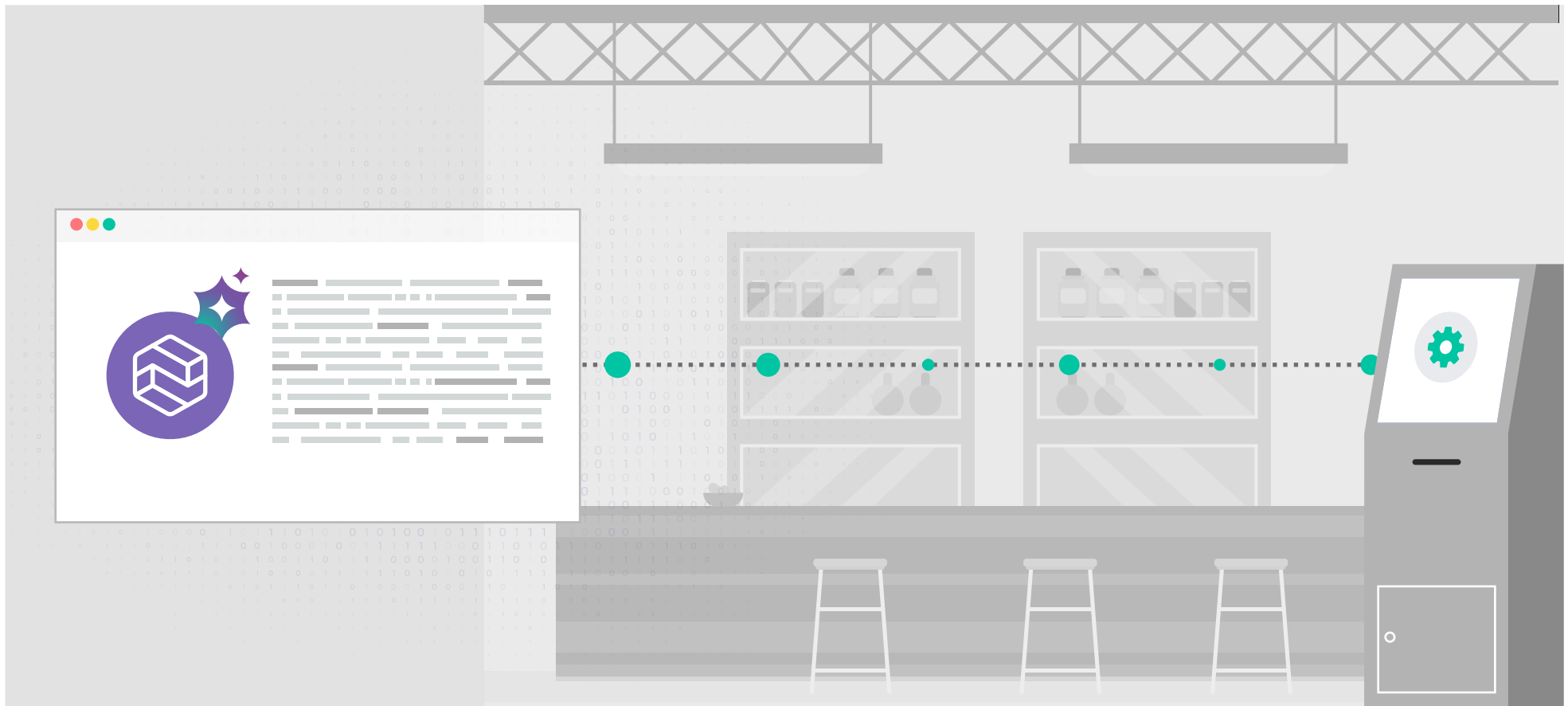
Another example here is if you run a restaurant loyalty program and want to deploy your loyalty model to the restaurant. Integrating AI into self-ordering kiosks (and tying it to the loyalty program) allows you to streamline ordering by offering smart suggestions based on the customer's ordering history or specific allergens, leveraging location-based promos, offering birthday rewards, etc. — all automatically.

The question in all these scenarios is pretty clear: how do you get there?

This scenario starts with a large AI model trained and running in the cloud.

It uses machine learning to improve its ability to identify manufacturing anomalies, continuously improving its job. When it reaches an accuracy level within the company's acceptable parameters, the core parts are put into a smaller model that can run on the LAN (or even directly on devices in some cases!) — no need for the cloud.

But therein lies the problem. How can you get that model to the edge network, especially with the foresight that these models will need continuous updates? AI is driven by continuous improvement, and you must continuously improve your model on the edge to operationalize AI efficiently.



The AI Model Delivery Solution: Streamlined Deployments

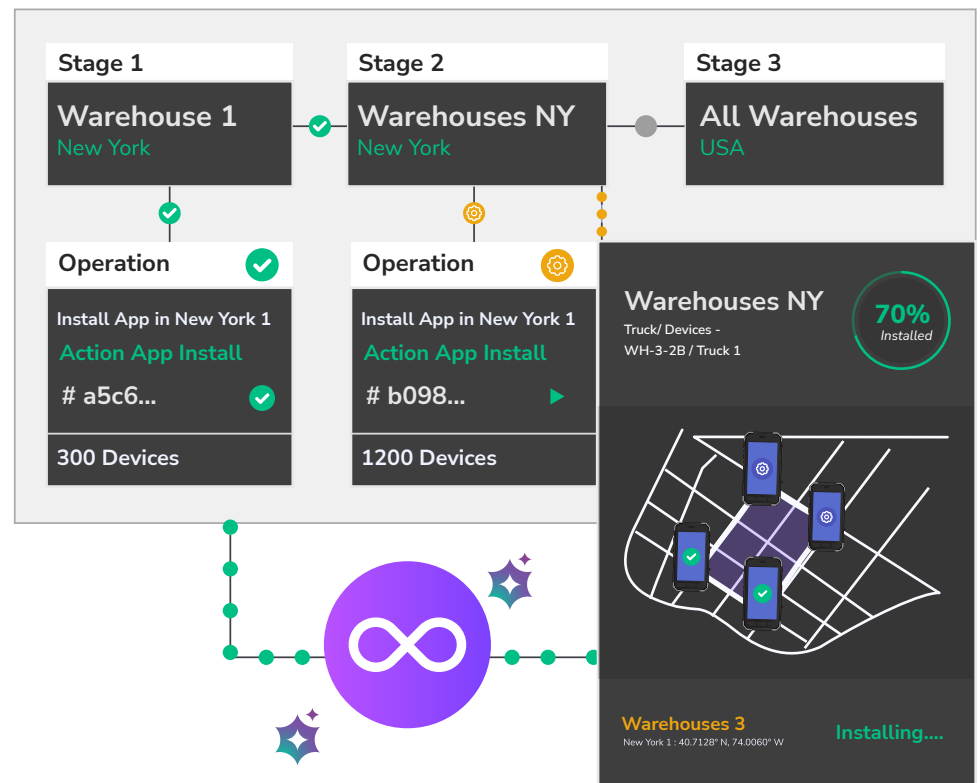
Your distribution model is just as critical as your AI model itself — after all, what good is AI if you can't reasonably deploy and update it? That's the first problem you must address when building an AI strategy.

Fortunately, AI models are conceptually not much different from apps — in fact, small models are often packaged into apps! These models rely on libraries of information and are often packaged together (on the edge, at least). So, the problem is the same one that has plagued edge devices for a long time: content delivery.

You need some of the tools we discussed in earlier chapters to effectively manage content like apps and AI models on the edge. More specifically:

- **Pipelines:** AI developers use CI/CD (continuous integration/continuous delivery) pipelines to automate testing, versioning, and packaging for reliable model updates. We use the same philosophy to enable you to deliver those updated AI models to the edge.
- **Containerization:** To prevent device-specific dependencies, containerization is necessary. This makes your AI more portable, allowing it to run on the edge. SLMs are ideal for offline processing and lower latency — instead of running in the cloud, they're part of the overall container.
- **Testing and rollback:** Testing goes hand-in-hand with pipelines, but a repeatable practice for testing AI model updates is necessary before widely rolling out new models. And in the case of failure or other issues, a reliable rollout back method is critical.

Remember the Continuous Delivery discussion? That's what we're talking about! With the right tooling, delivering AI models and updates becomes a non-issue, enabling your dev team to focus on improving the model and IT team on more strategic technology choices.



Operationalizing AI at the Edge has Never Been Easier

With Esper, your AI goals become realities. We can help with every part of the deployment and update process — from testing to rollout (or rollback). We built our platform to address problems just like this. Our modern DevOps approach means you get everything you need for AI model deployment:

- **Blueprints:** This is what device management and optimization at scale look like. You can use Blueprints for everything from settings tweaks to content management and, of course, your AI/ML models on any of your devices — then enforce that state as aggressively as you want.
- **Pipelines:** CI/CD pipelines are how DevOps engineers streamline app development and updates. It's also how we enable scalable app, content, and AI model deployment. With Pipelines, you can test your new AI model on a few devices and then roll it out in stages to the rest of your fleet as needed. Esper Pipelines is the key to successfully pushing updates to your AI model at scale, no matter how big the deployment.
- **App and content libraries:** With the Esper Cloud, you can ensure that you always have the latest apps, AI models, and content on any device in your fleet at any time. When you combine the Esper Cloud with Blueprints, you can easily enforce compliance across your devices, ensuring they're always running the latest policies, app versions, and AI models.

Operationalizing AI at the edge can be challenging, but it doesn't have to be. Using Esper, you can deploy and update your AI model with confidence.

This is *why* we built Esper — to help organizations streamline device management for business-critical hardware. Now and in the future.

Blueprints Manager



Blueprint 10

3 2 3 Always Apply

Android iOS

Device Mode

Choose how users are able to interact with the device.

Kiosk Mode



Allow Local App Install

SHARED

Allow apps from unknown sources to be installed on the device.



Launcher

Select the Esper Launcher for dedicated devices and kiosk functionality.

Esper Launcher



Pipelines

Run initiated by Philip

Cancel Run

Promtion to State 2

Stage 1

Dev Environment

200
Device

Esper Agent v1.2.4



Stage 2

Dev Environment

800
Device

Esper Agent v1.2.4



Stage 3

Install updates and apps
in Southwest region

2000
Device

Esper Agent v1.2.4



Postface

I get it — that’s a lot to take in. Being proactive about “what’s next” when planning or expanding your device strategy is critical, but I don’t have to tell you that. The good news is that there’s no better time to start thinking about your AI strategy than now, regardless of where you are in your digital journey.

Using the concepts outlined in this book, especially in order, is your springboard to success — even if you don’t use Esper. Of course, we want you to give our service a try and know that you’ll love it if you do, but ultimately I want you to take the core

concepts and apply them to your current strategy.

If you’re having trouble understanding any of these steps or applying them to your device fleet, [book a demo with us](#). We can answer your questions, show you how to implement these philosophies on our platform, and show you what the future of device management looks like — no strings attached.

We’re ready when you are.

Thanks for reading!

Book a Demo with Esper

About the Author



Sudhir Reddy is a hands-on technologist who prides himself in building and scaling global engineering teams while delivering high-quality products for customers worldwide. He is an empathetic leader who passionately cares about customer success, product quality, and engineering excellence. With over 26 years of experience in leadership roles at Intuit, Oracle, Hyperion, ASML, and more, Sudhir brings a unique blend of business acumen, product innovation, development of large-scale DevOps platforms, and execution capabilities to Esper.



Yadhu Gopalan has over 25 years and 35 patents in embedded systems and security. His career includes past engineering leadership roles such as Chief Architect of Windows CE and Windows Phone. At Amazon, Yadhu designed back-end solutions for FireOS and AWS before he owned Systems Engineering for Amazon Go. Today, Yadhu is Esper's Chief Geek, CEO, and Co-Founder. The visionary behind our mission to bring next-gen device management tools and DevOps deployment principles to company-managed devices everywhere.