NGINX
Part of F5

WHITEPAPER

A Stress-Free Roadmap to Application Modernization

By Chris Grundemann

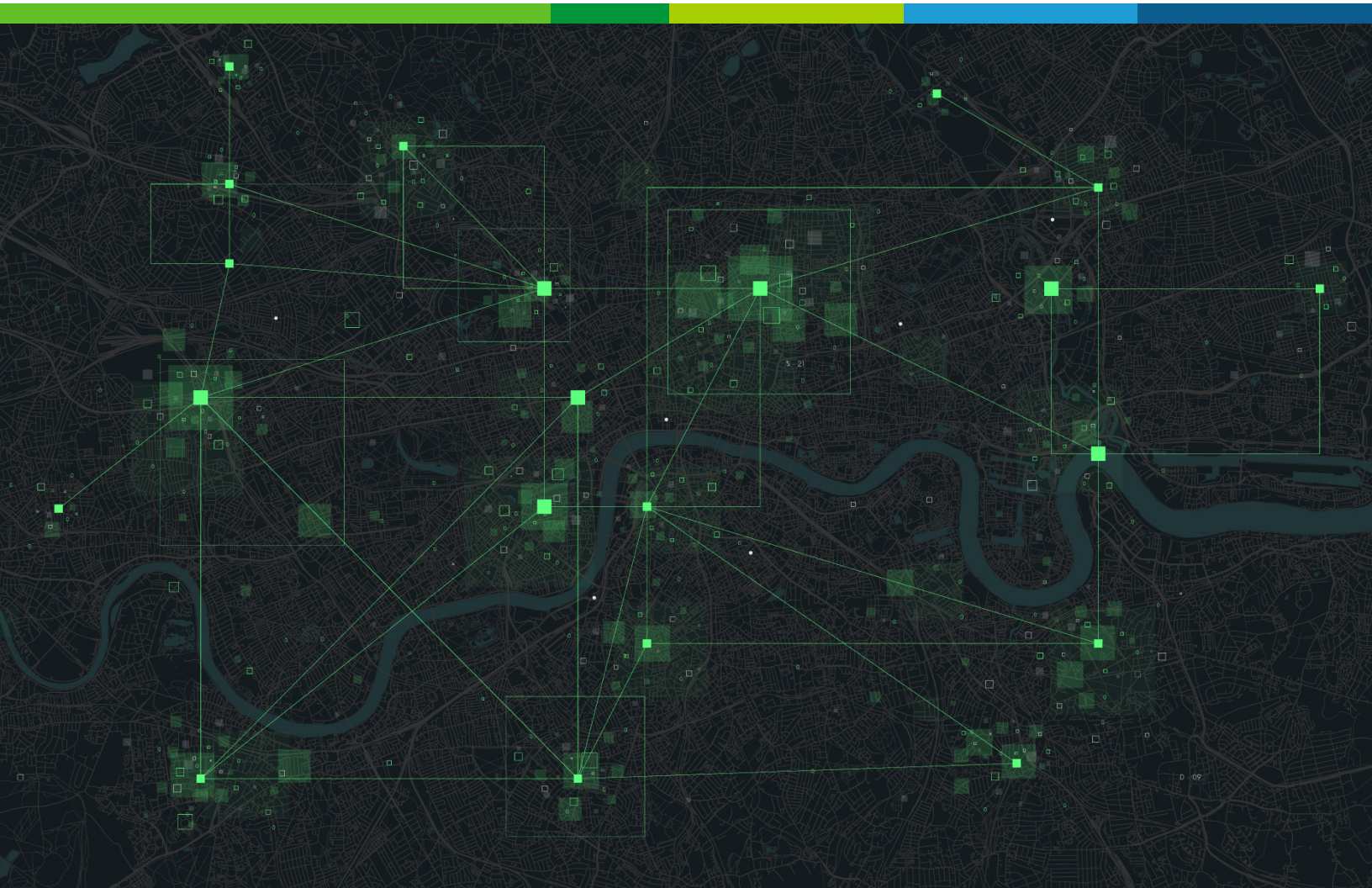


Table of Contents

Executive Summary

Microservices are both the means and the method for application modernization. A microservices architecture creates a modular application in which each service can be developed, tested, deployed, scaled, and ultimately replaced independently of any other service. What's more, the agility, flexibility, velocity, and efficiency that a microservices architecture enables is exactly what is fueling successful digital transformation for organizations and more importantly, the digital experiences that customers now expect.

Unfortunately, most digital transformation, cloud adoption, and application modernization efforts are failing. The difficulties typically coalesce around one of three main areas: culture (new operating models often require new operational structures), complexity (modularity comes at a cost, particularly in network requirements and cloud provider peculiarities), and security (the attack surface becomes larger and more dynamic).

THE PATH TO SUCCESS STARTS WITH PLANNING AROUND THE FOUR PILLARS OF SCALE, SECURITY, OBSERVABILITY, AND GOVERNANCE

The key to avoiding these common challenges, it turns out, is all about preparation. Successful application modernization starts long before the first container is ever created. The path, and thus the roadmap, starts with careful documentation, planning, cultural shifts, and process improvements around the four pillars of scale, security, observability, and governance. The bulk of this whitepaper is focused on explaining each of these pillars, including why they are important and what the common challenges are, as well as providing practical and actionable advice for your own application modernization journey.

Introduction

While some companies have made great strides in their overall digital transformation efforts, and application modernization projects in particular, that's simply not true for everyone. With all the hype surrounding "cloud native", microservices, containers, CI/CD, service mesh, and more, it's easy to feel like you're behind. The problem with that view is that it leads many intelligent folks to jump forward into a situation they don't understand without a clear plan for success.

THE FUTURE IS HERE; IT'S JUST NOT EVENLY DISTRIBUTED

The fact is that most organizations are in a similar position. It's worth remembering this William Gibson quote: "the future is here; it's just not evenly distributed". Rather than racing to catch up, you're far better off slowing down and doing some serious planning as the first step in your application modernization efforts. But even for those who know this, it can be hard to know where to start and what to focus on.

This paper answers those big questions with a roadmap crafted through knowledge from those who have gone before. We'll explore the common pitfalls, gotchas, and other dangers along your path. And we'll carefully document how to avoid them with proper planning and documenting up front. All of this revolves around the four pillars of scale, security, observability, and governance. Within each of these areas, this roadmap provides specific actionable advice to lower your stress as you embark on this journey.

PRIMER

APPLICATION MODERNIZATION IS NOT SYNONYMOUS WITH CONTAINERIZATION OR WITH DEPLOYING KUBERNETES

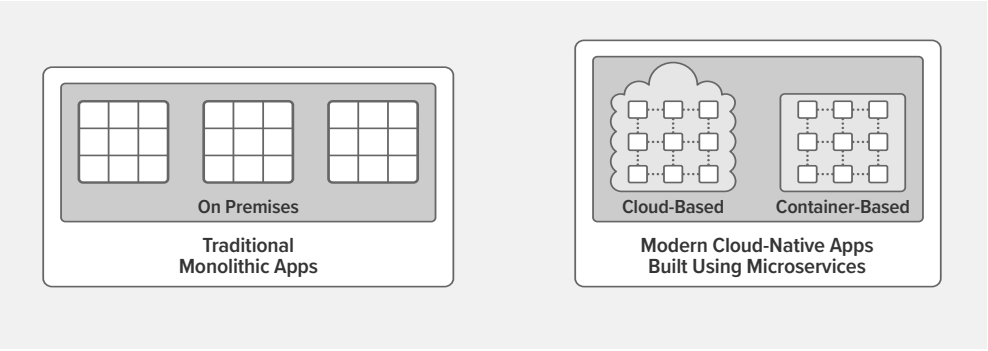
If you're reading this, you already have a pretty good idea of what application modernization is, at least at a high level. However, the idea of a "modern application" is worth exploring before we dive deeper.

First, the wrong answer. Contrary to what some flippant comments may lead you to believe, application modernization is not synonymous with containerization or with deploying Kubernetes. This is an easy mistake based on the current state of the art. The truth is a layer deeper. Kubernetes itself is not nearly as important as what it enables: microservices.

MICROSERVICES ARE BOTH THE MEANS AND THE METHOD FOR APPLICATION MODERNIZATION

Traditional monolithic applications are defined by layers, such as front-end, back-end, web controller, and database, which communicate in memory. A modern application built using microservices is defined instead by a collection of individual services – each performing a complete but singular atomic or composite function – which communicate over a network. A microservices architecture creates a modular application in which each service can be developed, tested, deployed, scaled, and ultimately replaced independently of any other service. In short, microservices are both the means and the method for application modernization.

Figure 1: Microservices power modern apps



MICROSERVICES ARCHITECTURE PROVIDES THE AGILITY, FLEXIBILITY, VELOCITY, AND EFFICIENCY THAT ENABLE THE DIGITAL EXPERIENCES THAT CUSTOMERS NOW EXPECT

Microservices architecture provides the agility, flexibility, velocity, and efficiency that lead to successful digital transformation for organizations and, more importantly, enable the digital experiences that customers now expect. In addition to automatic and elastic per-service scaling, cloud-native applications built using microservices enable and facilitate new modalities such as per-service A/B testing, blue-green deployments, and canary releases. If you must refactor entire monolithic applications to respond to a user preference or trend, you'll simply never keep up. The need for this highly responsive, digital-first approach to business (supported by cloud-native applications using microservices) has become even more clear than ever before through the global response to the COVID-19 pandemic. So, there's no time to waste.

CHALLENGES

Of course, knowing that you must modernize your applications does not guarantee success. As early as January 2018, McKinsey reported that "only 8 percent of companies we surveyed recently said their current business model would remain economically viable if their industry keeps digitizing at its current course and speed". However, 18 months later, research by Bain & Company found "that only 8% of global companies have been able to achieve their targeted business outcomes from their investments in digital technology". And the stats have not really improved. In March of 2020 another McKinsey survey found that digital progress had stalled at 76% of organizations, and as recently as June 2021 an Enterprise Management Associates survey showed that less than a third (28%) of respondents could categorize their cloud investments as "very successful".

WE'VE KNOWN FOR YEARS THAT THINGS ARE CHANGING AND YET WE'RE STILL FAILING TO MAKE THE NEEDED CHANGES

So, what's going on? We've known for years that things are changing and yet we're still failing to make the needed changes. What's standing in the way? For starters, many of these failed digital transformation and cloud adoption efforts have attempted to "lift-and-shift" existing monolithic applications onto new platforms. That's simply not going to produce the desired outcomes. But even those who focus on application modernization can,

THE GOOD NEWS IS THAT ALL THESE CHALLENGES ARE WITHIN YOUR CONTROL, AND MOST SIMPLY REQUIRE A BIT OF PREWORK TO SET YOURSELF UP FOR SUCCESS

and often do, struggle. These struggles typically coalesce around one of three main areas: culture (new operating models often require new operational structures), complexity (modularity comes at a cost, particularly in network requirements), and security (the attack surface becomes larger and more dynamic). Another factor is the lack of standardization among cloud platforms, making each one peculiar in its own ways and complicating hybrid and multi-cloud deployments.

The good news is that all these challenges are within your control, and most simply require a bit of prework to set yourself up for success.

ROADMAP

Luckily, you're not the first person to set down this path. Less luckily, many of those who tread before you have failed or are currently stuck. And while many folks will argue all day and night about which configuration management platform or automation server to use, very few are willing to admit that the real secret to success is as boring as proper preparation. The key lies in mindful and intentional documentation, planning, cultural shifts, and process improvements around the four pillars of scale, security, observability, and governance. Scale is our goal and security our mandate, observability gives us insight, and governance provides flexible control. Spending time up front to consider your current and ideal state across all four pillars will pay dividends down the road.

THE REAL SECRET TO SUCCESS IS AS BORING AS PROPER PREPARATION

The bottom line is that successful application modernization starts long before the first container is ever deployed. In the following sections, we explore each of these pillars – why they're important, what can go wrong, and how you can set yourself up for success.

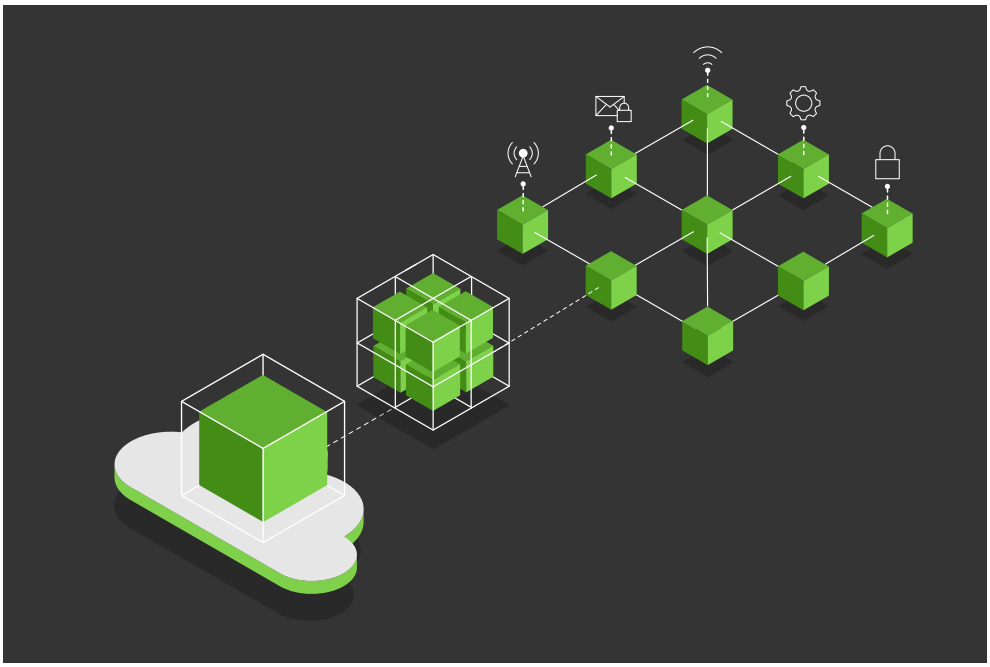
Pillar 1: Scale

THE ABILITY TO FLEXIBLY SCALE IS AT THE HEART OF MODERN APPLICATION DESIGN

The ability to flexibly scale is at the heart of modern application design. It's actually one of the driving forces, which is exactly why we've seen the so-called 'hyperscalers' (companies like Amazon, Google, and Netflix) take the lead on implementing service-oriented architectures (SOA), microservices, and containers/Kubernetes. While a legacy monolithic application can only scale vertically and "all-at-once", a microservices application can scale vertically, horizontally, and elastically – and can do so at either the cluster or service level.

But, like so many things, scale doesn't just happen by default – it must be carefully planned for. Additionally, while a microservices architecture can enable amazing scale by modularizing applications and decoupling functions, it does so by introducing additional operational complexity that must be dealt with. A modern cloud-native application built using microservices can easily have hundreds or thousands of services with dozens or hundreds (or more!) instances of each. And those instances are often spread from data center to cloud to edge. It can boggle the mind just to think about, let alone manage, that kind of distributed scale.

Figure 2: Microservices enable apps to scale easily



FIRST DETERMINE SERVICE BOUNDARIES, THEN BUILD DEVELOPMENT TEAMS THAT MAP TO THOSE BOUNDARIES

To set your organization up for successful scale down the road, the first thing you need to address is organizational structure. Conway's law applies here, which is an adage stating that "organizations design systems that mirror their own communication structure". To leverage this knowledge, you should first determine (and document) service boundaries. Then build development teams that map to those boundaries. And don't forget that many changes will need to be accomplished across these boundaries, so your teams need a way to collaborate and communicate with each other just as your services do. Beyond organizational structure, you need a robust data governance model as well, to ensure proper concurrency and data consistency across services.

Of course, you also need the technical ability to manage scale directly. NGINX is a great example of scaling efficiently. Originally developed to solve the C10K problem, NGINX helps you to scale and manage traffic across your estate of web applications, monoliths, APIs, and microservices-based apps. Lightweight and platform agnostic, NGINX can be deployed as a web server, load balancer, reverse proxy, content cache, API gateway, Kubernetes Ingress Controller, and service mesh.

A tool like F5 NGINX Controller not only makes management of an estate of NGINX instances more tenable, but also enhances the cross-team communication and collaboration we identified as essential above. Key things to look for in a management platform are app-centricity and multi-role self-service. Everyone involved in application development and deployment should be able to manage and monitor the bits of the lifecycle that are important to them through modular workflows.

Pillar 2: Security

THE POTENTIAL DAMAGE CAUSED BY EACH THREAT IS REACHING CATASTROPHIC LEVELS

Security is likely already top of mind across the industry. Not only are attacks becoming more common, the potential damage caused by each threat is reaching catastrophic levels. Modern applications are not immune and their architecture can actually compound the difficulties. Cloud-native applications built using microservices are inherently distributed, heterogeneous, and network-reliant. This means that the attack surface is multiplied. There is no single or often even identifiable perimeter. Each service and every API must be both individually and collectively secured.

WE ALL KNOW THAT WE NEED TO "SHIFT SECURITY LEFT"

In addition to the technical challenges, there is almost always cultural friction between developer (and DevOps) teams moving at the new speed of business and security teams hamstrung by policies and tools designed and deployed for legacy IT methodologies. At this point we all know that we need to "shift security left" because incorporating security testing and controls into the daily work of development, QA, and operations results in better outcomes. But if that was easy, we wouldn't still be talking about it so much, would we?

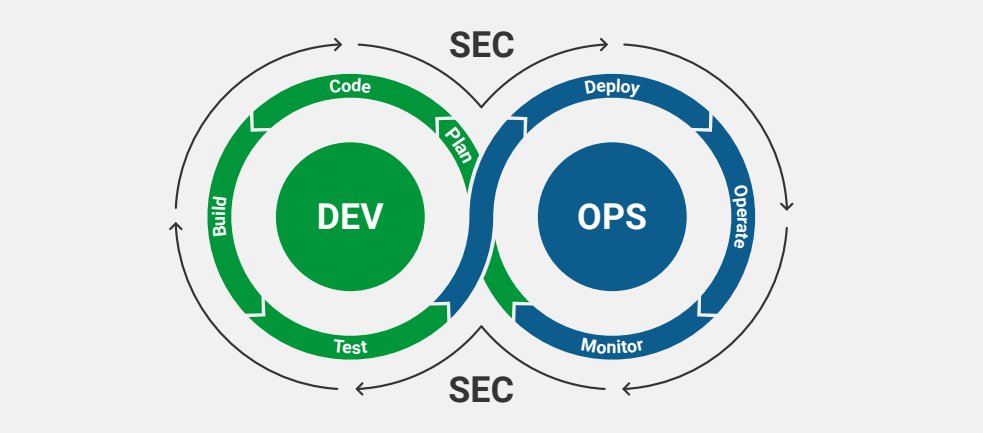


Figure 3: Security is taken into account earlier in the software development lifecycle

One of the first things that must be addressed is this friction between the legacy security methodology of centralized, inflexible, after-the-fact, perimeter-based security protections and rapid, agile, continuous modern application development and deployment. The first step is a thorough examination of security policy and how it is enforced. Document security goals and policies in a layered approach. What controls need to be centralized, and which can be self-service? How can you evolve your security procedures to be more application-centric?

SELECT SECURITY TOOLS THAT CAN BE INSERTED DIRECTLY INTO YOUR CI/CD PIPELINE AS CODE

Another key action needed before you begin deploying services is selecting security tools that can be inserted directly into your CI/CD pipeline as code. One example is finding a lightweight web application firewall (WAF), such as F5 NGINX App Protect, that deploys easily into multiple environments while automating security configuration and policies. This allows you to test for security efficacy at every stage of development without ever needing to stop or even slow down.

One more security question to ask up front: “How will we orchestrate security across all of our applications, services, and APIs?” Pay particular attention here not just to layering in security across the estate, but also providing self-service access to Dev and DevOps teams while providing the needed visibility and insights back to security teams. Leveraging an application-centric control plane such as the NGINX Controller App Security add-on can be a great way to meet these demands.

Pillar 3: Observability

Technically, observability is the ability to infer the internal state of a system based on its external outputs. In the reality of modern application development, it’s more complicated. A useful comparison is traditional monitoring, which is based on collecting pre-defined information (“known unknowns”) with the hope of catching problems you’ve seen previously, before they happen again. Observability, on the other hand, is broader in scope and is aimed at uncovering “unknown unknowns” with the goal of catching previously unseen issues and trends. While monitoring was suitable for many simple legacy systems, the world of modern distributed systems demands observability.

OBSERVABILITY IS AIMED AT UNCOVERING “UNKNOWN UNKNOWNS”

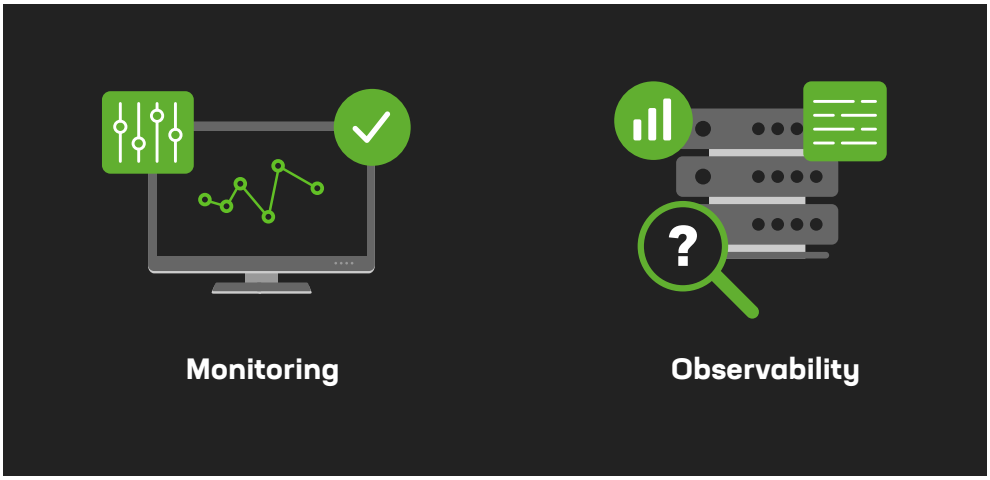


Figure 4: Quickly detecting both known unknowns and unknown unknowns is critical for delivering exceptional digital experiences

Based on what we’ve already learned about cloud-native applications built using micro-services, you likely can see the inherent challenge. Microservices are both interdependent and typically scattered across multiple hosts, or even sites and clouds. As the application scales dynamically to meet demand, new instances of these services are often created and destroyed automatically in real time. This means that in order to provide observability you need to be able to quickly discover and start collecting logs, metrics, and traces from service instances and other components. You must be able to collect and correlate a massive amount of granular data. And do all of this without losing sight of the ultimate goal: ensuring a high-quality experience to all users, everywhere.

When planning for observability, it is important to consider at least three key factors:

- **Instrumentation** – How you will discover, collect, and set baselines for telemetry from every system component
- **Correlation** – How you will structure your instrumentation to provide the needed context and topology, to understand causal dependencies
- **Artificial intelligence** – How you will leverage AI to remove dependencies on time-consuming human trial and error and surface those unknown unknowns reliably

One way to make observability (along with scale and security) easier to achieve is to use consistent components when building your distributed application delivery system. Extending the same set of network features across multiple systems (like using NGINX for proxying, load balancing, caching, API traffic management, and so on) means you can monitor just one solution rather than a number of tools that each perform one function. In this example you can then layer on a monitoring tool like NGINX Amplify, which monitors both NGINX and other components (for example, underlying OS, application servers like PHP FPM, databases, and so on).

Network traffic itself is another great source of insight. For full visibility into all north-south traffic entering and leaving a cluster of microservices, along with total traffic management and enhanced security, a production-grade Ingress controller, such as F5 NGINX Ingress Controller, is ideal. A more advanced approach to making your distributed application more observable is to deploy it with a service mesh, such as F5 NGINX Service Mesh, which essentially adds a common control plane and data plane to your Kubernetes environment. This universal data plane inherently "knows about" all network traffic within a cluster and can therefore easily provide excellent insight to your observability plane. In addition to observability, there are many other advantages to using a service mesh regarding resilience, routing, scalability, and security.

NETWORK TRAFFIC ITSELF IS ANOTHER GREAT SOURCE OF INSIGHT

Pillar 4: Governance

Finally, we need to discuss the fourth pillar of application modernization: governance. Technical governance isn't about politics or politicians, but it is about policies (rules). Specifically, it refers to the ability to enforce policy across an organization and/or across a technology estate. Proper governance leads to a common policy framework being implemented within each team, department, cluster, service, or application. More specifically, we can break governance down into two vectors: *policy scope* refers to where a policy is applied (such as within a specific cloud provider or namespace) and *policy targets* refer to which policies are applied within the given scope. Targets can be related to security, networking, configuration, and image management.

TECHNICAL GOVERNANCE ISN'T ABOUT POLITICS OR POLITICIANS, BUT ABOUT POLICIES

GOVERNANCE DONE WRONG CAN HINDER YOUR DEV AND DEVOPS TEAMS

As you can probably guess, governance is one of those areas that becomes more necessary and more complicated as you modernize your applications. Distributed applications built by distributed teams at the pace of modern business are notoriously hard to govern. And what's worse, governance done wrong can hinder and hurt the ability of your Dev and DevOps teams to execute. Many organizations are starting from a place with multiple governance frameworks, often overlapping and conflicting. And even if your current governance model works great, it probably wasn't built to interface with modern application development tools and modalities.

That makes a documentation and rationalization exercise the first step. You need to understand how and where you are managing policy today. How many governance frameworks do you have? When was the last time they were evaluated? What tools are you using to track, apply, enforce, and verify these policies? Which targets do you have robust governance frameworks for, and which are new? For example, you probably have formalized security policies (you do, don't you?) and you may already have a plan for managing cloud costs, but you likely haven't thought yet about how to manage container images or cluster configuration. Once you have a handle on current policies and their level of enforcement, you can work on creating a more universal governance framework that takes modern application development and hybrid/multi-cloud environments into account.

FOCUS ON GUARDRAILS RATHER THAN GATES

Once you've thought through the needed policies, it's time to think about enforcement. The key here is to focus on guardrails rather than gates. You don't want to slow your developers down, but you do want to keep them on track. Just as water flows downhill, people also take the easiest path. Don't put up obstructions that your teams will naturally want to circumvent. Instead make it easy to follow your guidance. We spoke above about needing security tools that can be inserted into your CI/CD pipeline as code and about providing self-service wherever possible – think about all targets this way, not just security.

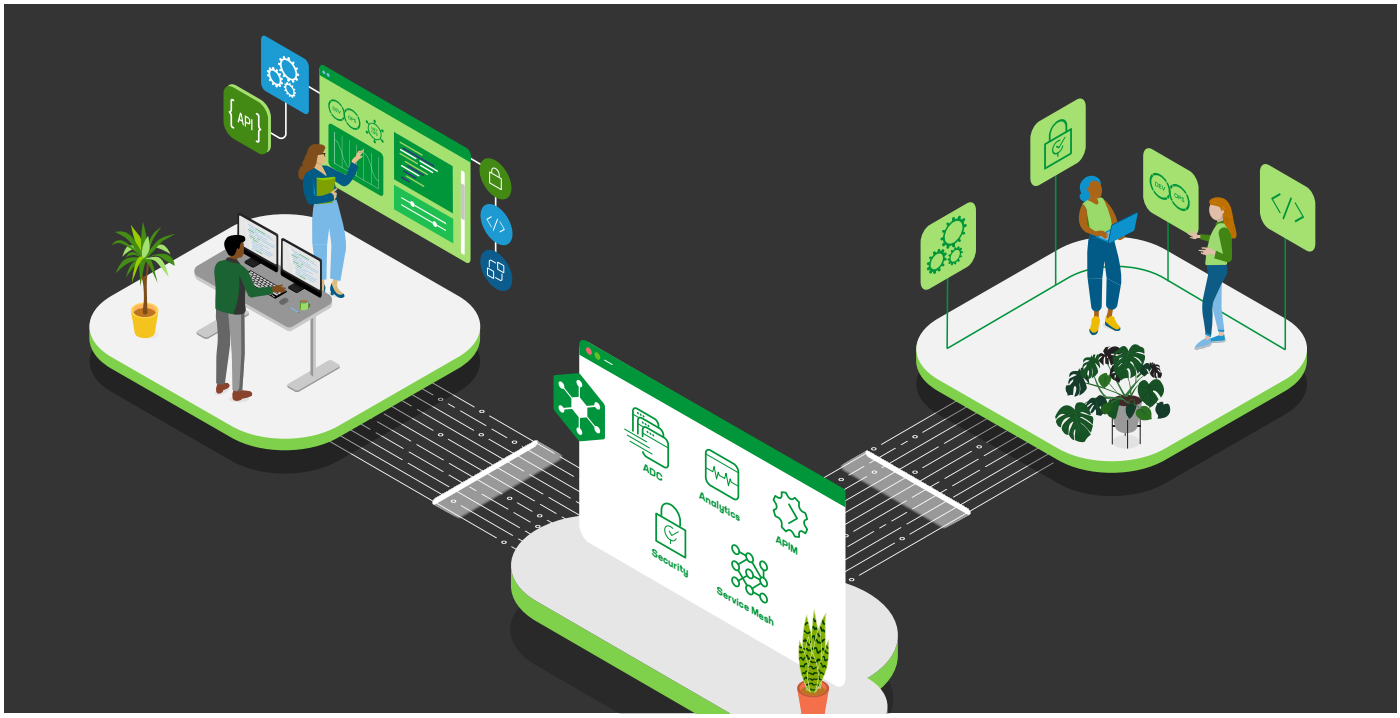


Figure 5: A governance framework that focuses on guardrails and removes friction for developers accelerates innovation

AN APP-CENTRIC DELIVERY SOLUTION EFFECTIVELY BALANCES DEVELOPER PRODUCTIVITY WITH OPERATIONS COMPLIANCE

From a tooling perspective, the best way to enable governance is with a platform approach. In fact, many companies are now standing up "Platform Ops" teams to help manage the supported suite of application-delivery tools available to teams across the organization. Of course, having a single app-centric delivery solution, like the previously mentioned NGINX Controller, can make everyone's lives easier by providing the right set of features to effectively balance developer productivity with operations compliance.

What Next?

We've explored the promise and potential of application modernization with microservices, as well as many of the most common challenges. We hope you've come away with plenty of practical advice for avoiding the worst perils with careful documentation, planning, cultural shifts, and process improvements around the four pillars of scale, security, observability, and governance.

That means it's time to take the next step by digging deeper and learning even more about the journey ahead of you in this Fundamentals of Microservices webinar.

Happy travels!

NGINX
Part of F5