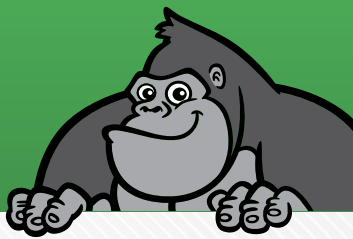
ActualTech Media

PLATFORM9

THE GORILLA GUIDE TO...®

EXPRESS EDITION



Kubernetes Operations

Joep Piscaer

Inside the Guide

- A Vendor-Neutral, Cloud-Agnostic Approach to Managing Containers
- What a Managed Kubernetes Service Is, and Advantages of Using One
- The New Distributed Architecture: Edge Computing

THE GORILLA GUIDE TO...

Kubernetes Operations

Express Edition

By Joep Piscaer

ACTUALTECH MEDIA

6650 Rivers Ave Ste 105 #22489
North Charleston, SC 29406-4829
www.actualtechmedia.com

PUBLISHER'S ACKNOWLEDGEMENTS

EDITOR
Keith Ward, ActualTech Media

PROJECT MANAGER
Wendy Hernandez, ActualTech Media

EXECUTIVE EDITOR
James Green, ActualTech Media

LAYOUT AND DESIGN
Olivia Thomson, ActualTech Media

WITH SPECIAL CONTRIBUTIONS FROM PLATFORM9:
Bic Le, Roopak Parikh, Kamesh Pemmaraju

TABLE OF CONTENTS

CALLOUTS USED IN THIS BOOK

The Gorilla is the professorial sort that enjoys helping people learn. In the School House callout, you'll gain insight into topics that may be outside the main subject but are still important.

This is a special place where you can learn a bit more about ancillary topics presented in the book.

When we have a great thought, we express them through a series of grunts in the Bright Idea section.

Takes you into the deep, dark depths of a particular topic.

Discusses items of strategic interest to business leaders.

ICONS USED IN THIS BOOK

DEFINITION

Defines a word, phrase, or concept.

KNOWLEDGE CHECK

Tests your knowledge of what you've read.

PAY ATTENTION

We want to make sure you see this!

GPS

We'll help you navigate your knowledge to the right place.

WATCH OUT!

Make sure you read this so you don't make a critical error!

TIP

A helpful piece of advice based on what you've read.

The Forgotten Innovation in Kubernetes Deployments

The recently developed application architecture of containers, microservices, and cloud computing can deliver immense benefits in terms of scalability, response time, rapid innovation, and cost savings—but it makes organizations rethink their strategy on many levels. Your developers are thinking about how to modularize feature development and conduct continuous integration. Your cloud and infrastructure operators are thinking about how to use modern tools—especially Kubernetes—to run these complex applications. But too few companies apply creative thinking to tools and processes that tie everything together.

Kubernetes, the industry-standard container orchestration tool, is powerful. It can do so much, including: monitoring, upgrading, restoring, troubleshooting, security patching, logging, alerting, load balancing, exception handling, DNS services, and more.

Whew! That's a whole lot of functionality, but comes at the cost of simplicity and ease of management. Fortunately, there are solutions.

The Gorilla Guide To...® (Express Edition) Kubernetes Operations illuminates three major areas of your infrastructure that require standardization and consolidation:

- Monitoring and observability
- A service mesh for communication
- Automated management and updates of multiple distributed locations

In each of these areas, you need one or a few modern tools that meet your organization's needs. Platform9 Managed Kubernetes (PMK) provides a unifying management system that can work across all your systems, whether they're on-premises, multiple cloud environments, or scattered to the edges of the network.

In this Gorilla Guide, you'll learn where many companies go wrong and get trapped in suboptimal solutions or miss out on the ability to manage their systems consistently in all environments. You'll see what each tool contributes to solutions that guarantee performance and up time without requiring heroic efforts from operations staff. And you'll get a glimpse into computing on

5G cell towers and other edge locations—a promising new environment for highly responsive applications. Rev up and take a tour!

CHAPTER 1

SaaS Managed Kubernetes: the Effective DIY Alternative

Kubernetes is here to stay. But its operational complexity is a major hurdle for many organizations, creating a barrier to entry that's hard to solve: qualified engineers are expensive, and DIY solutions have a long lead time and are very complex.

This means organizations are missing out on the advantages that Kubernetes provides for development teams, like accelerating software releases with more control over the infrastructure to optimize performance and cost.

This chapter helps you understand how to remove complexity and decrease lead time for Kubernetes using a cloud-agnostic, managed solution approach.

Kubernetes is a powerful infrastructure platform for developers. Its self-service nature allows developers to take control of releasing software to production without the direct involvement of Ops teams. This helps development teams increase their velocity, enabling them to release more often and more quickly, with

more control over the infrastructure than ever to optimize cost, performance, and resilience.

The downside is increased complexity. With all of its advantages when it's up and running, Kubernetes is notoriously hard to deploy and manage. Its open, pluggable architecture is complicated and can be overwhelming for those new to Kubernetes.

This creates high barriers to entry for Ops teams wanting to design and implement production-grade clusters that provide high resilience and good performance at a reasonable cost.

In the architecture diagram in **Figure 1**, it becomes clear that Kubernetes is a complex solution with many moving and interchangeable parts.

Even after the initial learning curve is conquered, new challenges await. Kubernetes clusters tend to have a shorter lifespan than virtual infrastructure clusters, and are often built for a very specific function, such as a single application. This is especially true in environments with ephemeral compute needs, like cloud computing. Here, clusters here are constantly spun down, recycled, and replaced by new clusters.

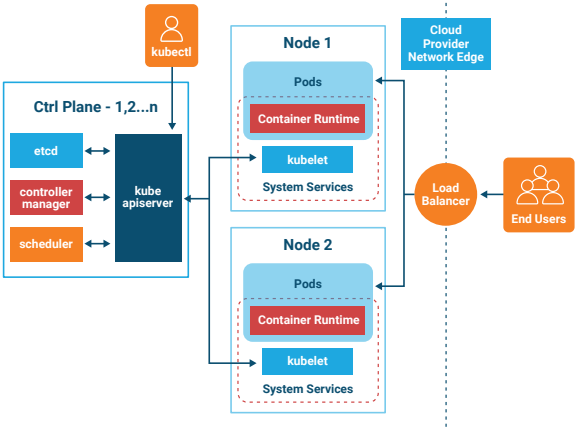


Figure 1: Kubernetes architecture is difficult to understand and master

This poses major challenges in the way Ops teams work, requiring new operational processes. It also requires new technical skills, which use up team resources that could have been better spent on other, more business-focused initiatives.

Kubernetes, Microservices, and Modern Development

The powerful benefit offered by container technologies such as Kubernetes is also the source of the difficulties that containers present. They're part of an architecture for computer applications that combines:

- Multiple services, such as databases, that communicate by exposing APIs
- A microservice architecture, which is a modern form of the classic programming practice of breaking programs into independent modules
- Cloud deployment, which makes it easy to scale up or down in response to demand

This type of architecture is becoming increasingly popular for several reasons First, it allows a company to spin up new instances—virtual representations of a physical computer—as demand increases, such as if it advertises a sale and thousands of people suddenly visit its web site. The company saves money by terminating those instances when the spike ends.

The architecture is also fault-tolerant. If a physical computer blows a fuse, you have to power up a new computer. In a virtual or containerized environment, you just press a button on the cloud provider's graphical interface—or even better, run a monitoring system to detect failures and to automatically launch new instances through the cloud provider's API.

Less obvious, but equally important, the containerized architecture supports a faster, more distributed development cycle. Programmers in different places can work on different microservice components and indulge in their own release cycles, without having to communicate frequently with the developers of other components. Because many new features require changes to only a single component, developers can keep the users happy with fast-changing stream of new features, or can react quickly to a change in the business model.

Indeed, chances are that Ops teams are already stretched thin, and putting in the time to master the day-to-day chores of managing a cluster's lifecycle is simply another added burden. Ops is already busy maintaining existing infrastructure, from WAN and LAN networking, to virtual compute infrastructure, client management systems, and many others.

Outsourcing Makes Sense—on the Surface

It is therefore logical to outsource Kubernetes to a managed services provider. Luckily, there are many options to choose from, across a spectrum of different types of solutions.

On one end of this spectrum are software distributions that provide a framework, but leave you alone to figure out how it works. While better than rolling out your own Kubernetes distribution, these options still require you to do all of the heavy lifting yourself, and don't actually solve the problems outlined earlier.

The hyperscale public cloud vendors go one step further, by offering these software distributions as an easy-to-consume service. These services take care of initial cluster deployment, using their own design best practices and implementation tooling.

While saving massively on initial deployment lead time, these solutions have a number of downsides. Public cloud vendors' service portfolios are designed to lower the barriers for developers to start using additional services, often without an upfront cost. It's an elegant, integrated portfolio of services that developers love.

Lock-in Blues

It's not without its drawbacks, though—the biggest one being vendor lock-in. These hyperscale public cloud vendors have many services, and they'll try to persuade you to use their other services, as well. Lock-in increases steadily, drawing you in as a customer with each step.

While, operationally, the level of integration between services and products is naturally very high, the cost and the operational and strategic risk increase exponentially.

Lock-in

Software vendors—and now cloud providers—have devised many barriers over the years to leaving their offerings. Some of the dangers of lock-in are:

- The vendor going out of business, which means stopping updates and bug fixes or (in the case of cloud offerings) terminating all services. Customer data can also be lost.

- More subtly, a change in the vendor's plans and priorities may take the product in a direction that doesn't help some of its own clients. They may find that the product no longer offers the original value they found in it, but that their data and processes are trapped in it.
- Similarly, the vendor may decide to remove a feature that the client relies on. Perhaps the feature isn't important to the clients the vendor wants to attract. Often, the vendor sees the feature as a hindrance to selling some other services, and removes it to force clients to spend more for a new service.
- Bugs of high priority to the client may be of low priority to the vendor, so they may go unfixed.
- Vendors have been known to raise prices so much that they make the offering unaffordable to many users. Sometimes they suddenly start charging high prices, without warning, for offerings that were previously free of charge.
- Thus, professional administrators and operators are wary of lock-in.

The public cloud provider may lock you into using their authentication, monitoring, or data services (like databases and object storage). For instance, to use their managed Kubernetes services, you have no alternative but to use their compute instances, block storage service, and monitoring services, as well as their authentication service. This lack of choice increases the lock-in.

In the Kubernetes realm, another lock-in is more obvious: the managed Kubernetes service often dictates what compute instances can be used, in the sense that you can only use their compute nodes. This can prevent you from using a third-party service or bringing your own compute.

And while technically there's nothing wrong with using their compute nodes as Kubernetes worker nodes, they do make up the vast majority of cloud costs. And what happens if they alter their terms or hike prices? If you're locked in, you may feel you have to accept these unwanted changes.

But maybe not. And that leads to an often-hidden, but frustrating and expensive, issue—breaking the lock and finding another provider can be incredibly frustrating. The time it takes to move away from a specific public cloud service when locked into that ecosystem

can take many months, and may have a significant impact on your projects and budgets. Sometimes providing your own infrastructure is cheaper and offers more agility.

While lock-in is often associated with the risk of cost increases, the strategic risk of not being able to move and adapt to changing circumstances in your business could also be crippling, especially when the services are used for customer-facing digital transformation projects.

This means a loss of agility in the marketplace, since you're no longer able to adjust requirements in response to changing circumstances. That increases your Total Cost of Ownership, or TCO, when using the public cloud vendor's entire service landscape.

Mitigating Risk

Intelligent planning demands that organizations consider solutions that don't have economic and operational lock-in, while still offering Kubernetes as a service. The value proposition is clear: the enterprise receives all the benefits, without the downside.

With SaaS, you're essentially hiring the best consultants to assist with the architecture design, configuration, and operational processes to optimize your

Kubernetes environments for availability, resilience, security, and cost. But you don't pay the high price of a specialized consultant. Instead, the SaaS provider creates the automated workflows and the back-end automation that allow hands-off initial deployment, upgrades, monitoring, alerting and more. Since it's all done with software instead of labor-intensive manual processes, it scales elegantly.

Smaller organizations simply can't justify the cost of a dedicated Operations team with the appropriate Kubernetes knowledge and experience, which requires expensive staff—nor can they run the risk of being dependent on a single worker or a small number of employees for their specific knowledge.

Instead, cloud-agnostic, managed Kubernetes services, like Platform9 SaaS Managed Kubernetes service (PMK), are indifferent to the location of your Kubernetes cluster: on-premises, in a private or hosted cloud, across any of the public clouds, or in a combination of all of these.

Workloads are moving increasingly to the network edge. This dynamic creates hundreds or even thousands of new locations. In that scenario, operational overhead ramps up massively, leading to huge management nightmares. But SaaS provides central

management of those widely distributed clusters with the simplicity of a single pane of glass console. That means formerly labor-intensive operations like software updates are as easy as the click of a button.

PMK offers quick onboarding to Kubernetes for developers, allowing them to use the service without any re-training, and includes many of the moving parts that usually accompany Kubernetes for monitoring, logging, networking, and storage.

Focus on Business Outcomes

Managed Kubernetes services are invaluable in other ways, too. Not only do they remove the operational complexity of designing, implementing, and operating Kubernetes, they allow organizations to focus their staff's time on things that directly impact their bottom line.

Instead of ITOps staff focusing on daily IT operations, they will have time to spend on business projects, which increasingly have an IT or tech component.

Thus, a managed Kubernetes platform has two key advantages. First, your organization will have a proper Kubernetes infrastructure, which is a driver for many digital transformation, digitization, and e-commerce

projects. It allows organizations to quickly develop, release, and iteratively improve customer-facing applications.

Second, freeing up IT staff from their day-to-day task will accelerate those projects by adding invaluable tech skills and experience into the mix, without having to risk being pulled back into yet another operational fire that needs their immediate and undivided attention.

Many companies underestimate this latter aspect of using a managed Kubernetes service. Freeing up technical staff that know the organization's technology stack and all of its subtleties, technical debt, and quirks can have a massive impact on the quality of the software delivered as part of those innovative projects.

Improving Time-to-Value

In addition, using a managed Kubernetes service allows organizations to hit the ground running. Instead of slowing down a transformation project to hire the right staff, design, install, and configure a Kubernetes environment, a managed Kubernetes service helps speed up projects by decreasing lead time for the technical aspects of building a Kubernetes environment.

This newly unencumbered IT staff can be the difference between a successful digital transformation and

a failed one. IT staff have a crucial role in non-functional aspects. While functional characteristics define specific behavior and functionality, non-functional aspects define qualitative aspects of a system, including stability, availability, resilience, security, performance, manageability, upgradeability, cost, and more. With IT staff safeguarding those attributes, these projects will deliver a better end result, and more quickly.

Now that we've examined the value of a dedicated service to manage your containers, we'll look at what such a service offers. The first of such a service is to monitor what's going on so you know when there are problems. Thus, monitoring and observability are the topics of the next chapter.

CHAPTER 2

Tackling Observability in Your Kubernetes Environment

We're on our way to discovering a robust and vendor-neutral way to manage Kubernetes instances. In this chapter we look at the foundation of management: monitoring.

There are many tools in the cloud-native and microservices tool chest. Kubernetes is the go-to for container management, giving organizations superpowers for running container applications at scale. However, running an enterprise-grade, production-level Kubernetes deployment is more than running just Kubernetes by itself.

Because containers are ephemeral and transient, monitoring, security, and data protection are fundamentally different from their counterparts in virtualized or bare metal applications. Optimizing the tooling that supports a Kubernetes deployment is not a trivial task. In many cases, this means that tooling aimed at virtualized environments doesn't translate well into containerized platforms. Replacing these tools may be better than retrofitting legacy tooling.

In fact, more modern tooling created to support container environments may help you get the most out of your container platform. In this chapter, we'll look at how to optimize observability in your Kubernetes environment. We'll define the types of observability, offer a path for starting and expanding the process, and describe the advantages of cloud-based or Software as a Service (SaaS) monitoring.

Types of Observability and Their Value

Let's go back to basics first. Looking at the *observability* space for container-based microservices landscapes, we can distinguish three separate types of tooling:

1. Monitoring (or metrics): collecting operational telemetry about applications, application services, middleware, databases, operating systems, and virtual or physical machines
2. Logging: collecting error messages, debug or stack traces, and more detailed messages
3. Tracing: collecting user transactions and performance data across a single or distributed system

KUBERNETES

Watch the Platform9 webinar on how Kubernetes has transformed monitoring. Another great resource to check out is the Platform9 blog, "Logging & Monitoring of Kubernetes Applications: Requirements & Recommended Toolset."

In a DevOps or Site Reliability Engineering (SRE) world, these three disciplines collectively make up observability.

Each discipline provides valuable insights in all layers of the layer cake that make up the increasingly complex application and infrastructure landscape of containers. DevOps engineers and SREs use the insights from these tools to improve resilience and performance, as well as triage errors, fix bugs, and improve availability and reliability.

Finally, they use these tools to gauge how users are interacting with the system. The tools help figure out which functionality visitors use or don't use, and where performance bottlenecks lie.

As application landscapes expand due to digital transformation, the number of microservices and individual containers explodes, making it harder to see the inner workings of systems. So, it shouldn't be a surprise that executing a good observability strategy is one of the deciding factors of a successful Kubernetes deployment.

While enterprise IT is more important than ever, digital transformation has led many more organizations to create digital and online applications. IT has often become a critical business function, vital for the survival of your business.

Layers of Monitoring

A good place to start with monitoring is by collecting metrics and operational telemetry of the Kubernetes constructs like clusters and pods, as well as collecting metrics on resource usage like CPU, memory, networking, and storage. Starting with the bottom two layers (see **Figure 2**) for monitoring is relatively easy and a good way of becoming comfortable with observability tooling.

Infrastructure monitoring and logging are key capabilities because it's important to know the activities of your physical infrastructure. A substantial amount of

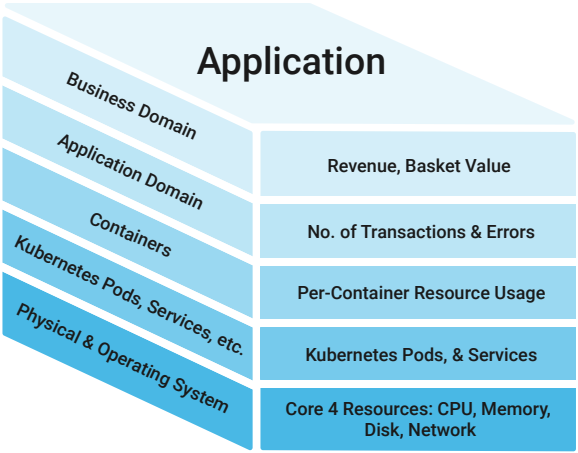


Figure 2: An application layer cake with monitoring examples

your application's performance and resilience comes from correctly functioning servers and networking.

As the application landscape expands, a well-executed infrastructure monitoring and logging strategy also builds a shared understanding of application performance across teams, preventing miscommunication between application development, cloud platform, and other teams.

Visibility into infrastructure and the shared understanding it builds is crucial, but of course doesn't give

the entire picture. For that, you need to move up the stack, and start with application performance monitoring (APM). For many organizations, the application monitoring journey starts with monitoring (or metrics collection) and logging containerized workloads. For Kubernetes-based environments, there are natural combinations to start with, like the open source Fluentd and Prometheus, which make it easier to run monitoring and logging.

Making Observability Work for Your Business

To gain benefits from observability, you need to think about your business requirements over the next few years and choose a platform that meets your needs. This section describes how to think about requirements, tools, and the IT organization.

Align Monitoring to Business Objectives

This chapter has presented a natural progression of how teams use observability, starting with the infrastructure basics, working their way up the stack into the realm of applications, and even tracing users across the application landscape, monitoring their behavior and transactions.

This journey up the stack is an opportunity to align monitoring, logging, and tracing to business objectives, mining more insights from the increased visibility. It allows teams to gain visibility into more than just technical metrics, generating business-oriented metrics, too.

By measuring business-oriented metrics (such as the dollar value of the shopping basket, the number of abandoned baskets, and metrics on popular or even disused features), product owners can align development priorities to what their users really want, optimize performance in areas where it actually matters, and fix technical debt to accommodate further growth. Naturally, these insights fuel business growth and revenue.

When tooling is aligned to the business and customer experience, the tools can be used by more than just IT teams, allowing business teams to gain insights into their applications and its users.

Think Mid-Term to Long-Term

The tools you choose for observability should serve your needs for several years. This requires you to think about how your business is changing and how that will change your observability requirements in the long-term.

The cost of migrating to a new, more capable APM platform can be significant, but won't immediately give you additional functionality. This additional functionality requires additional engineering and implementation before these capabilities are fully unlocked.

And let's not forget that moving to another APM platform requires you to retrain staff and needs time to regain confidence in the metrics and insights, all of which reduce the value the APM platform brings in the short term. So, it makes sense to choose your tooling wisely from the start, keeping the long-term goals in mind.

In other words, while you won't need the most complex or feature-rich solution now, look at what features you'll need to support evolving requirements in the future. Invest in your team and people and start with the APM capabilities you need now. You don't need to enable, implement, and incorporate every feature the tooling provides from the get-go. It's OK to start simple, build up confidence along the way, continuously evolve your knowledge of the tool, and expand its use in-sync with changing requirements.

The Observability Platform Team

As your organization grows and your use of metrics becomes commonplace, it makes sense to create a dedicated team focused on the continual improvement of the observability platform, and help application development teams gather the metrics that matter to them.

The dedicated team owns the platform and executes an observability strategy across many applications and teams. The observability platform team's expertise speeds up troubleshooting, helps with application architecture optimization and can help teams to quickly pinpoint and solve bottlenecks and fragile areas prone to failure.

With their expertise and knowledge of the environment, they can "travel" from application team to application team to grow each team's knowledge and expertise more efficiently, preventing re-inventing the wheel and other local optimizations within each team, instead letting all teams learn from the collective knowledge, driving up maturity more quickly.

With this team structure, the application teams can focus on gathering metrics, refining the metrics they collect, and improving the signal-to-noise ratio inherent in gathering metrics, so the telemetry gathered is optimally serving business objectives and the team isn't spending any time on managing or operating the observability platform.

Cloud-Based Observability (SaaS)

If your teams are busy setting up and managing servers, they'll be less effective at their real job: improving the usage of APM across the organization. By adopting a SaaS offering, the observability platform team can focus on the functional side of observability instead of the day-to-day toil of managing and operating the platform.

That means they're not bogged down with installing yet another security patch or forced to think about scaling the observability platform. Instead, they have more time to help application development teams with their observability challenges or to improve the platform itself.

SaaS also helps an organization get started with an observability platform. Instead of having to make the

hard design choices at the start of a project (with little to no experience), you can use the choices offered by the SaaS vendor, confident that its experts have vetted the choices.

Using a SaaS service lets the vendor take on the operational burden of upgrades, scalability, and performance for the APM software, freeing up your team to work on broader and deeper implementation of APM across application teams.

This will speed up the pace at which your observability platform matures, allowing teams to gain a deeper and more user-oriented understanding of the application landscape more quickly. This increases the value of the investments made in the observability space in two ways: You spend less time installing, configuring, and getting started with the tooling, and the insights gained from the tooling can be applied more quickly to optimizing revenue and fueling growth.

In effect, you can leapfrog your APM journey, skipping the traditionally hard first steps in getting started with monitoring.

This chapter laid out the essential role of monitoring in the management of Kubernetes clusters. In the next chapter, we turn to the networks on which you run your Kubernetes containers.

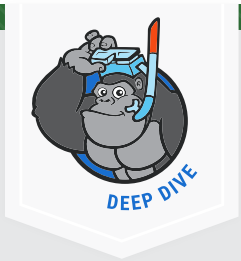
CHAPTER 3

Best Practices for Selecting and Implementing Your Service Mesh

As applications are being broken down from monoliths into microservices, the number of services making up an application increases exponentially. And as anyone in IT knows, managing a very large number of entities is no trivial task.

Service meshes solve challenges caused by container and service sprawl in a microservices architecture by standardizing and automating communication between services. A service mesh standardizes and automates security (authentication, authorization, and end-to-end encryption), service discovery and traffic routing, load balancing, service failure recovery, and observability. Just as virtualization abstracted the hardware layer of computer systems and containers abstracted the operating system, a service mesh abstracts away communication within the network.

Why a Service Mesh?



As monoliths are pulled apart into their smallest constituents, the resulting microservices are usually distributed across multiple systems and communicate over HTTPS, so they become heavily dependent on network communications.

A service mesh manages the network communications by setting up standards and automating their implementation. It frees developers from defining and implementing the communications for every service, over and over again (see **Figure A**).

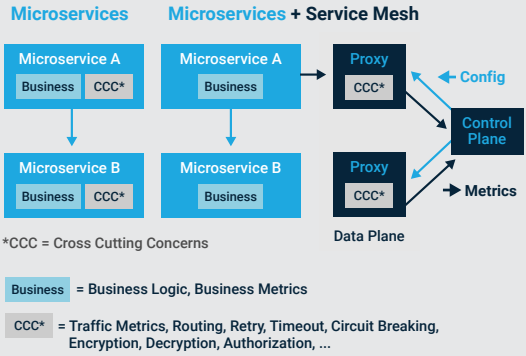


Figure A: Service mesh architecture

This is much more scalable, more automated, and less error-prone. The service mesh also improves security and reliability by standardizing the interface between services. The service mesh acts like an automatic walled garden for each service on the network.

This is done by making sure other services know the service exists (called "service discovery"), managing authorization and authentication between services, taking care of load balancing, and adding security policies for communication to and from the service by the outside world.

Because a service mesh has control over the network communication between all services in the mesh, it unlocks some advanced deployment and release strategies, such as canary releases, blue/green releases, and rolling upgrades.

This improves the reliability of the services in production. In some cases, the service mesh can react to changes in the traffic patterns, adding circuit breakers and rate limiters between services to prevent cascading failures. In order for teams to gauge the performance and quality of each release, a service mesh often has observability tooling (for collecting telemetry and metrics, as well as building in distributed tracing capabilities).

In short, a service mesh acts like an operational mission control to determine the behavior of microservices at scale, making sure the landscape of microservices is communicating securely, and monitoring performance and service quality. It removes much of the manual work from the developer's plate, so they need to focus only on the business logic, not the network, security, and communication plumbing.

The result is not only higher quality in business logic code, but also a reduction in variations and human errors in the plumbing, by standardizing and automating much of that work.

Reducing Service Mesh Complexity

Although a service mesh is very useful to development teams, implementing the service mesh itself still takes some work. Because there are many moving parts, a service mesh leaves a lot of flexibility and room to customize it to your specific needs. As always, flexibility comes at the cost of complexity.

Balancing the features, functionality, and value of a service mesh with its inherent complexity it is highly

challenging, and requires expertise, but is well worth the effort. With an experienced team in place, organizations can overcome the complexity associated with running a service mesh at scale.

The best way to start developing the necessary skills and experience is no different from any other technology: start early, and start simple. You don't need to accelerate from 0 to 60 miles per hour instantly. Instead, start small, and incrementally add more features and functionality as you build trust in the service mesh.

It's recommended to start developing service mesh skills in tandem with your microservices architecture, because adding service mesh features to a relatively simple microservices architecture is much easier than when it's already complex and large. Let the service mesh grow organically alongside your ever-evolving microservices architecture. This keeps services secure and compliant, and helps maintain visibility.

The Service Mesh Team

As your organization grows and your use of the service mesh increases, it makes sense to create a dedicated team focused on the continual improvement of the service mesh, as well as helping application

development teams make the most of the features and functionality it offers.

The dedicated team owns the service mesh platform and is responsible for the adoption of the service mesh across application teams and the entire microservices landscape. With this team structure, application development teams can focus on building business logic and microservices.

As you'll see in the following sections, having a dedicated team keep tabs on service mesh use cases (like multicloud and heterogenous workloads) may save you from an expensive, intrusive, and complex migration project because reality got in the way.

The Service Mesh Catch-22

Choosing the right service mesh technology, and nailing the implementation details, are crucial factors in your service mesh success. But how do you make the right decisions and do the right things when you don't have the right knowledge and experience yet? This is the catch-22 for the initial deployment and configuration of every new technology, including a service mesh.

This is a common pitfall for organizations, as engineers start designing and implementing a new technology

enthusiastically. The inefficiencies and sub-optimal decisions due to lack of experience don't immediately come to light, but often surface only weeks, months, or even years later, when it's too late to drastically change anything.

How do you prevent these mistakes? And how do you kickstart the learning process without the associated risk and possibly massive impact down the road? Turning to a simpler, less feature-rich alternative carries its own risk, as you introduce a future point in time where your own maturity outpaces the limited feature set, forcing you to do a forklift upgrade of the mesh, introducing a migration not only of the mesh itself, but a migration of all the microservices in the mesh, too.

Instead, choosing the right mesh technology with the end-goal in sight makes more sense. Currently, there are three leading, mature options available in the Kubernetes ecosystem: Consul Connect, Istio, and Linkerd.

While there are differences, all three are battle-tested, production-ready, and enterprise-grade solutions. It's a matter of finding the right one given your unique context, requirements, and goals.

Istio has the most functionality and flexibility, but is also the most complex, making the first steps harder. Linkerd is Kubernetes-only, making it easier to implement and use. If you need to support virtual machines (VMs) alongside Kubernetes, Consul is a good choice.

The paradox here is knowing which level of flexibility you need a few years down the line when you have zero experience and expertise to make that decision now. Let's dive into an overview of these three options to start building a picture of which one is right for your organization. This will help you make the right decision and prevent obvious pitfalls as you build trust and increase your service mesh proficiency.

HashiCorp Consul Connect

Connect is Consul's service mesh feature. It provides service-to-service networking and security (authorization, encryption). As seen in **Figure 3**, applications can use a sidecar proxy deployment model.

These sidecars handle the inbound and outbound TLS connections, with the application completely agnostic of Consul. Consul also has a native integration deployment model. In Kubernetes environments, Consul uses a per-host DaemonSet agent and Envoy sidecar proxies per application that handles application traffic.

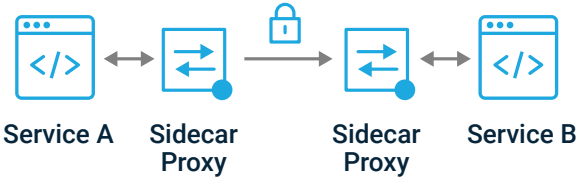


Figure 3: Consul Connect in a sidecar proxy model

Consul applies a zero-trust security model, is platform agnostic, and supports multi-cluster deployments.

As with other HashiCorp tools, Consul Connect is easy to get started with. Its deployment and initial configuration are a little less daunting than other options, making it a good solution for those very new to the service mesh space.

Istio

Istio is the darling of the cloud-native space. Like many projects before it, it was open sourced by an end-user company (Lyft, in Istio's case), as they built a solution to handle complexity and scale.

Istio has seen massive adoption, especially as the basis of various public cloud offerings.

Istio's complexity is its downside for newcomers to the field, but also what makes it so powerful; one example

is the addition of telemetry and analytics. As **Figure 4** shows, its architecture is much like Consul Connect.

A notable fact about Istio is that it is not part of the Cloud Native Computing Foundation (CNCF) landscape map, even though it's the most popular service mesh option for the CNCF's Kubernetes ecosystem.

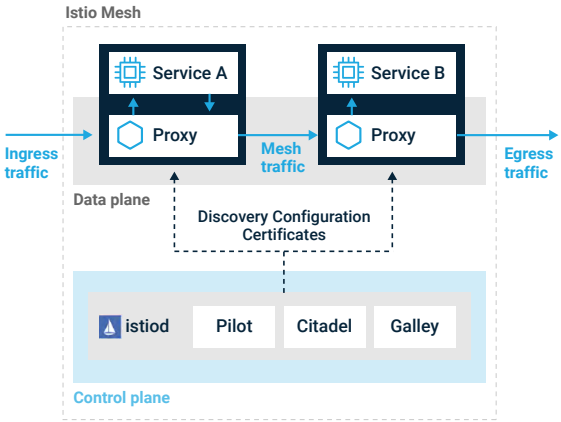


Figure 4: The Istio architecture

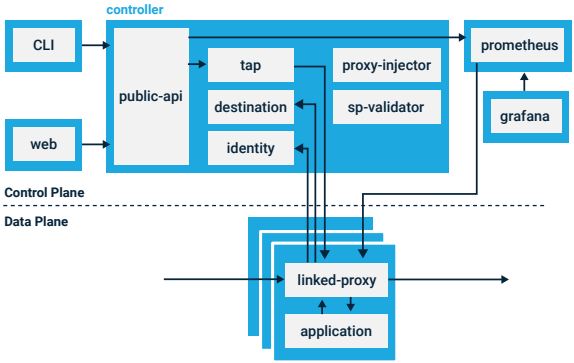


Figure 5: The Linkerd architecture

Linkerd

Linkerd is the CNCF answer to a service mesh (see **Figure 5**). Its v2 architecture mimics Istio, but favors simplicity over features and flexibility.

Where Consul and Istio work with Kubernetes and VMs, Linkerd exclusively works on Kubernetes. This means its architecture has fewer moving parts and fits into the Kubernetes architecture more seamlessly, with deeper integration into many other CNCF projects like Prometheus.

To get the full details, the Platform9 blog has a post called "Kubernetes Service Mesh: A Comparison of Istio, Linkerd and Consul." It compares these three feature-by-feature.

A Service Mesh Choice Is Not Forever

Even though you should now have the knowledge to make an initial choice, remember that requirements and circumstances change, so your service mesh will need to evolve, catering to those changes.

In some cases, a different technology is needed. If you're using the sidecar deployment model, applications and microservices running as part of the mesh are not aware of the mesh, nor do they have any special customization or integration with any specific mesh. The sidecar model makes it easier to migrate between technologies.

For more deeply integrated service mesh approaches, the Service Mesh Interface, or SMI for short, may prove useful. SMI offers a set of common, portable APIs that provide developers with interoperability across different service mesh technologies including Istio, Linkerd, and Consul Connect.

Conquering Multi-Cloud

Reality is messy, and IT is no different. Migration from old technologies to new ones is always happening, whether from VMs to containers, from on-premises to public cloud, or from one public cloud to another. What use is a service mesh that helps you control traffic, security, permissions, and observability when it works for only a sub-set of workloads in just one environment?

Multi-cloud in a service mesh context means more than just multiple public clouds. It also needs to support on-premises deployments and support VMs. Last, the service mesh should span all these environments and have multi-cluster support.

This multi-cloud reality is often not explicitly designed by the organization, but "just happens." For instance, a group of developers starts using yet another public cloud, because it has the specific functionality they need to do their work. Whatever the cause, making sure your service mesh can handle this guarantees you can take a proactive approach to supporting the endless variety of multi-cloud scenarios in production. It gives you the piece of mind that you're in control of security in the untrusted world of public cloud, and have visibility into the entire microservices landscape.

In other words, if chosen correctly, a service mesh can serve as an abstraction layer on top of the public cloud, abstracting away the cloud and giving back control over traffic, security, permissions, and observability in a multi-cloud reality.

Looking at the three options shows that while Linkerd's simplicity sounds great on paper, reality may get in the way, requiring you to use a service mesh technology that works across containers and VMs. And again, SMI may help you migrate service mesh technologies if you need to—accepting and acknowledging that reality is messy may save you from a painful service mesh migration project.

With the tools we've described in the first two chapters of this guide, you can set up a robust architecture for your applications. But where should you actually run those systems? In the final chapter we'll look at a promising new environment for highly responsive containerized applications.

CHAPTER 4

Distributed Edge with Managed Kubernetes

This guide has laid out a strategy that can run in any modern environment, including multiple third-party clouds. This chapter presents a high-performance environment that has evolved recently, part of a movement known as *edge computing*.

EDGE COMPUTING

As Internet speeds grew and SaaS services became popular, applications moved a lot of their intelligence into the cloud. The popularity of mobile devices with limited storage and compute power accelerates this trend toward centralizing most processing in a remote hub. But for performance reasons, application developers are trying to do more processing with local data in the device itself, or in a cell or local system located near the device. This is called edge computing.

Each generation of networking presents unique challenges that are met by unique solutions. Current application architectures connect small computers at the edge—mobile apps, Internet of Things (IoT) devices, retail point-of-sale systems, and so forth—with hubs in the cloud. These environments are characterized by:

- Compute-intensive requests sent by mobile devices into the cloud, where a vendor is expected to handle the requests and return results quickly
- Cellular networks to handle most requests
- Modular, scalable worker nodes that handle requests and are controlled by Kubernetes, the most popular tool currently for distributing requests across worker nodes

This network architecture calls on companies hosting applications to place worker sites on network endpoints, so they can quickly accept requests from mobile device users, calculate the answers to the requests, and return them to the mobile devices.

RUNNING COMPUTATIONS IN CELL TOWERS

Don't cell towers contain just radios and antennae? Nowadays, towers are building in compute power for their clients, including GPUs. They offer this compute power so that applications can run close to the clients instead of in far-away data centers, saving time and network costs.

A New Architecture for Responding to Compute-Intensive Applications

During the past 10 to 15 years, companies have gotten used to accepting data from mobile users or edge devices into a centralized data center. This data center run by the company owning the app determines the proper response and returns it to the edge.

But this simple model has turned out to be inadequate for delivering the performance needed as the complexity of calculations grows. Cellular networks, especially those employing some form of 5G, have alleviated

bandwidth limitations between the cellular tower and the edge, whether the edge is an end user on a mobile device or an IoT device reporting conditions in the field. But a significant bottleneck remains between the cell tower and the company's systems.

A new architecture has therefore developed, based on distributing the work to intelligent systems at the collection points provided by enterprises. Much of the information and work is never seen by the sites run by app developers. Instead, enterprise companies launch local containers to handle requests. Kubernetes instances are also launched at the endpoints to start up and monitor worker processes.

This chapter describes the new architecture and how to take advantage of it to offer applications that respond gracefully to user requests or reports from devices in the field.

Bandwidth and Compute Power in a Distributed Architecture

Edge computing brings compute power closer to the end users and their devices, essentially decentralizing some of the compute capabilities of centralized public cloud offerings.

Recent developments in connectivity, such as the increased bandwidth of 5G networks, have increased the opportunities for communication between edge locations and end users. This increase in connectivity allows an enormous growth of data and unlocks many new use cases, from image and video processing and voice recognition to running factories and retail locations over 5G instead of Wi-Fi. Fast response time is critical in the new applications. Mobile users are impatient, and IoT devices must make real-time decisions.

While the connection between the end user and the edge location enjoys ample bandwidth, responses from the centralized cloud or data center can't keep up in speed. Edge locations in many cases are remote and have limited connectivity, but require complex processing for huge amounts of locally generated data. That in turn creates an architectural challenge to bring compute resources where they can exploit the increases in bandwidth at the edge.

A Distributed Architecture Tailored to Kubernetes for the Edge

Transporting data from the edge to the central cloud or core data center for processing doesn't make sense, especially if there's a large amount of data to be transferred and fast response times are required. Processing

at the edge is more cost-effective. In this architecture an edge location, such as a 5G radio tower, runs one or more clusters of worker nodes. The edge location sends only processed data that's useful for business-related analytics to the central repository.

This new architecture processes data close to where it's generated, with a few core data centers or cloud regions acting as the brains of the operation.

In this scenario, the edge locations themselves need sophisticated task management for thousands of simultaneous processes that are set up and torn down quickly. Kubernetes is the current industry standard for this kind of process management. That means starting up Kubernetes worker nodes at the edge locations to run the data processing applications locally. By running only the absolutely necessary workloads at the edge, companies can reduce costs associated with maintaining centralized data centers and transferring data from the edge.

Application providers are now dealing with hundreds to thousands of edge locations, or even more. With an architecture that requires less hardware at the edge, savings scale linearly with the number of edge locations.

Of course, this concept applies to more than just data-processing applications. 5G, as well as faster and

more cost-effective endpoints (consumer and industrial IoT devices alike), are creating new use cases that generate far more data than ever before, such as video feeds from CCTV systems, telemetry information from industrial IoT devices, and interactions generated by apps and games on consumer phones.

Central Management Is Still Needed

Although the modern, distributed applications described in this chapter process data at the edge, application providers still need central control and visibility into the processing. Platform9 Managed Kubernetes (PMK) delivers Software-as-a-Service (SaaS) Kubernetes cluster management and simplicity of operations like native public cloud services but using upstream open source stacks that are deployed and operated on a wide range of on-premises (VMware, bare metal), public cloud(s) (AWS and Azure), and edge infrastructures.

The federated, distributed architecture of PMK provides a consistent experience across regions, while being centrally managed and resilient against connectivity and bandwidth issues. The architecture supports multiple regions in a hub-and-spoke model. **Figure 6** shows an overview of the federated architecture supported by

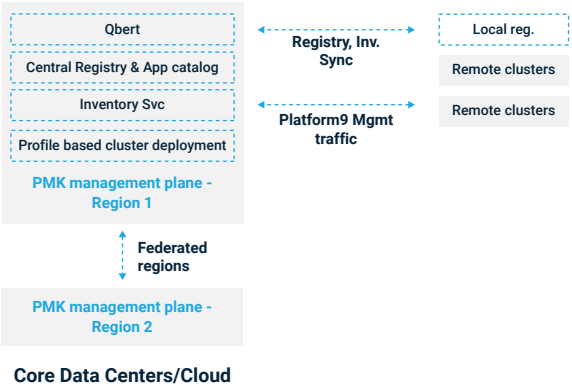


Figure 6: Distributed Edge Platform overview

PMK. Multiple regions work independently. Each management plane region acts as the central hub and brains of a region. The management plane defines policies for all the edge processors, with the federation of configuration templates and apps.

The management plane is where DevOps engineers manage the entire operation. There, they store container images and inventory caches of remote locations. Synchronization ensures eventual consistency to regional and edge locations automatically, regardless of the number of locations.

Profile-Based Management

Platform9 facilitates the scaling of edge computing to thousands of edge data centers by grouping them so that similar data centers can be managed centrally through a single policy known as a *profile*. This relieves IT staff from managing each data center individually. Instead, the staff just defines a small number of profiles and indicates exceptions to policies where needed for a particular data center. Each edge location, such as radio towers, warehouses, and retail locations, runs its own worker nodes and containers.

Profile-based cluster management makes it easier to deploy identical remote clusters and configurations, from a single profile, instead of managing each remote cluster separately. That minimizes configuration drift, while still being able to apply unique configurations where needed.

This feature helps standardize application and container deployment across clusters, making the onboarding of new edge regions easy and consistent. This allows the day-to-day operations work to scale non-linearly, making the most of each engineer's time. The feature allows administrators to manage a large number of edge locations without additional work.

As you saw in **Figure 6**, the "core" registries and catalogues are synced with remote locations, which cache this information to reduce bandwidth and remove any connectivity dependencies they might have to the core data centers.

As a result, deploying and scaling applications at the edge isn't dependent on the core data centers, but can be handled locally while still receiving periodic policy-based changes to keep configurations consistent and in sync with the centrally defined policies.

The architecture deep dive in **Figure 7** shows the entire architecture, from endpoints to core data centers. We'll dive into each area of this diagram in the next sections.

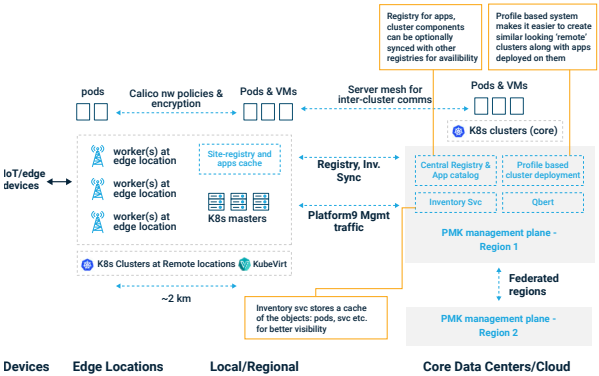


Figure 7: Distributed Edge Platform deep dive

Core Data Centers Are the Brains of the Operation

The core data centers run the management plane, central container registry, and App Catalog, as well as inventory services and policy-based deployment and configuration tools for cluster management.

The profile-based system makes it easier to create identical remote clusters and to deploy applications to remote clusters consistently, keeping configuration drift to a minimum and ensuring maximum application compatibility.

Core data centers are federated across regions for consistent deployment in large scenarios. A single management plane region is able to handle up to 100 Kubernetes clusters with up to 100 nodes per cluster. The central inventory service caches a representation of the remote sites for better visibility.

Consistent Networking and Granular Security

Networking and security policies are deployed consistently from core to edge, making sure application deployments are secure and compliant. Compliance can be enforced even in untrusted physical environments through service mesh and micro-segmentation technologies. A distributed Kubernetes architecture

combines all of these locations—public cloud VPCs, core data centers, edge data centers, and edge locations—in a single, interconnected mesh.

Local Data Centers Ensure Independent Operation

Each local data center can deploy and scale applications independently of the core data centers, creating geographically separated "cells" that can run without a continuous connection to the core data centers. Each local or regional location runs one or more Kubernetes master cluster nodes, which manage worker nodes across that location. This way, each edge location runs only the absolute necessary hardware—often low-cost, low-power, and low-maintenance machines—while regional hubs coordinate application deployment and resilience across their local region.

Applying This Architecture to Retail, Manufacturing, and SaaS

A large number of use cases can benefit from the distributed Kubernetes architecture described in this chapter. Many enterprises see the same growing need for edge computing as their revenue streams become more and more digitally focused. The enterprises see more need for connecting cloud services to where their

users are, regardless of whether those users are consumers, other businesses, or IoT-enabled devices.

Retailers are innovating and transforming the shopping experience to be seamless across online and in-store visits. This requires connecting cloud and on-premises locations to work together, offering buyers a consistent experience. The new architecture unlocks new revenue streams and increases existing value streams by accelerating the rollout of digital store concepts and by increasing in-store automation and the use of innovative retail software.

The distributed Kubernetes architecture helps retailers deploy new stores quickly and consistently. It reduces per-store operational IT costs both for onboarding new stores and for continuous operation, including lifecycle management and seamless software upgrades of running containers.

Manufacturers are replacing Wi-Fi and legacy wired networks with 5G wireless connectivity for factory floors and manufacturing plants to connect IoT and edge devices. That means they need to connect 5G endpoints with worker nodes for data processing, central management, and factory process engineering. Putting the compute power at the endpoints minimizes costs and operational burden.

Similarly, SaaS and independent software vendors (ISVs) are starting to use edge computing to improve their users' experience, decrease time to market, and reduce costs and operational burdens. The Platform9 Managed Kubernetes distributed architecture helps them deploy their software to edge locations, decreasing latency and bandwidth requirements, while consistently deploying the application to many locations simultaneously.

Especially with many single-tenant application deployments across edge locations (such as customer sites), upgrades and other operational and lifecycle tasks are automated and consistently executed across the board. This reduces support costs and effort, and allows developers to spend more time on delivering new features and software.

Solve Your Kubernetes-at-the-Edge Challenges

In this Gorilla Guide, you've seen how Platform9 Managed Kubernetes is suitable for any kind of application at the edge across telco, retail, manufacturing, enterprise, ISV, and SaaS use cases. Its ability to manage deployments on any infrastructure based on centralized policies is a huge time saver. It lowers time-to-fix when

outages occur, lowers support costs, and improves customer satisfaction.

Make sure to look at Platform9 Managed Kubernetes and its distributed architecture to solve your Kubernetes-at-the-edge challenges. Try it for free or download the updated buyer's guide.

Thanks for reading, and stay safe out there!

ABOUT PLATFORM9

Platform9 enables freedom in cloud computing for enterprises that need the ability to run private, edge, or hybrid clouds. Our SaaS-managed cloud platform makes it easy to operate and scale clouds based on open source standards such as Kubernetes and OpenStack, while supporting any infrastructure running on-premises or at the edge.

ABOUT ACTUALTECH MEDIA

ActualTech Media is a B2B tech marketing company that connects enterprise IT vendors with IT buyers through innovative lead generation programs and compelling custom content services.

ActualTech Media's team speaks to the enterprise IT audience because we've been the enterprise IT audience.

Our leadership team is stacked with former CIOs, IT managers, architects, subject matter experts and marketing professionals that help our clients spend less time explaining what their technology does and more time creating strategies that drive results.

For more information, visit
www.actualtechmedia.com