Section

Solving the Edge Puzzle

A breakdown of the expected and unexpected building blocks required to build **optimized distributed systems**

Table of Contents

Introduction: The Edge Puzzle 3

The Complexities of Replicating the Cloud Developer Experience at the Edge 4

The Context: Moving from the Cloud to the Edge 4

The Challenge: The Many Complexities of the Edge Developer Experience 5

- Code and Configuration Management 5
- Developer Workloads Moving to the Edge 5
- How Complexity Factors into Edge Runtimes 5
- Observability for Distributed Systems: Diagnostics and Telemetry 6
- Managing the Application Lifecycle 7
- Leveraging GitOps and CI/CD Workflows 7

The Solution: Consistency is Key 9

How to Approach Application Selection, Deployment, and Management for the Edge 10

The Context: Complexities Enhanced by Placing More Parts of the Application at the Edge 10

The Challenge: The Complexities in Moving Diverse Workloads to the Edge 11

- Web Application Firewalls (WAFs) & Bot Management Tooling 11
- Image Optimization 11
- Modern Testing & Experience Optimization 11
- Load Balancing Solutions 12
- Containers: Challenges of Orchestration 12
- Serverless Computing for Edge Computing 12

The Solution: Partner with an Edge as a Service Provider 13

The Complexities of Building and Operating Edge Networks and Infrastructure 14

The Context: Edge Networks and Infrastructure are Changing 14

The Challenge: Managing the Network, Infrastructure, and Operations in a Distributed Compute Environment 14

- DNS 14
- TLS: Provisioning, Management, and Deployment Across Distributed Systems 16
- DDoS: Protecting Layers 3, 4, and 7 16
- BGP/IP Address Management 17
- Edge Location Selection and Availability 17
- Edge Workload Orchestration 17
- Load Shedding and Fault Tolerance 18
- Compute Provisioning and Scaling 18
- The Messaging Framework 18
- Observability for Distributed Systems 18
- NOC Integration 18

The Solution: The Right Expertise 19

Conclusion: Overcoming the Complexities 20

Introduction: The Edge Puzzle

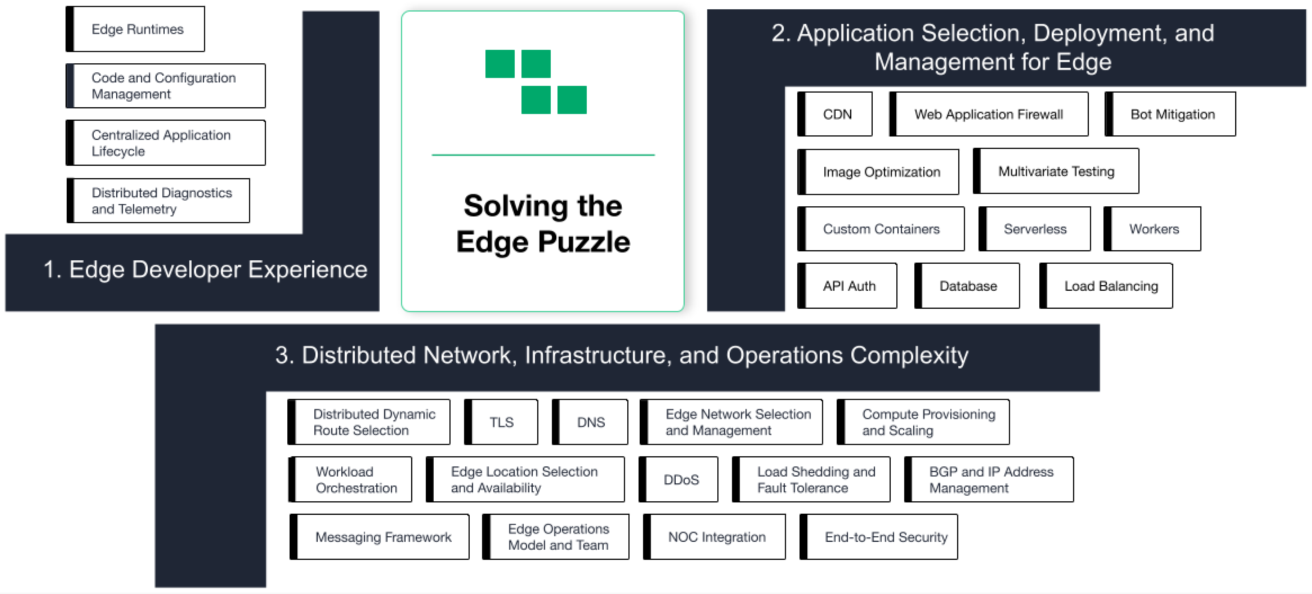
While most developers have grown familiar with cloud deployment, running workloads across a distributed edge introduces an entirely new set of complexities and considerations. Moving from managing a single deployment endpoint to hundreds or more, requires a different knowledge base and skillset, particularly if different microservices also need to be served from different edge locations.

Nonetheless, developers are increasingly being asked to program in a new computing paradigm - both for edge computing in its own right and as part of a hybrid cloud infrastructure. Further, the parts of the application being moved closer to the end user are also changing. Initially, simple, static assets like images were the main items, but in recent years, application owners have started to migrate more advanced logic, security, and persistent data stores out to the edge. However, the adoption curve has been slow, mainly due to the extreme complexities involved in designing, building, and operating a distributed deployment model.

For the edge to achieve the $800B mark by 2025 (as predicted in The Linux Foundation's State of the Edge Report 2021), developers must master the edge puzzle to learn to build optimized distributed systems, or partner with an Edge as a Service (EaaS) platform provider.

What are the main complexities involved in running workloads at the edge? Let's break down the complex and always shifting puzzle into component pieces. What do developers and Dev Ops teams need to know to reap the rewards of edge computing?

Building for Edge is a complex and always shifting puzzle to be solved and then managed



Edge Runtimes
Code and Configuration Management
Centralized Application Lifecycle
Distributed Diagnostics and Telemetry
1. Edge Developer Experience
Solving the Edge Puzzle
2. Application Selection, Deployment, and Management for Edge
CDN
Web Application Firewall
Bot Mitigation
Image Optimization
Multivariate Testing
Custom Containers
Serverless
Workers
API Auth
Database
Load Balancing
3. Distributed Network, Infrastructure, and Operations Complexity
Distributed Dynamic Route Selection
TLS
DNS
Edge Network Selection and Management
Compute Provisioning and Scaling
Workload Orchestration
Edge Location Selection and Availability
DDoS
Load Shedding and Fault Tolerance
BGP and IP Address Management
Messaging Framework
Edge Operations Model and Team
NOC Integration
End-to-End Security

The Complexities of Replicating the Cloud Developer Experience at the Edge

The Context:

Moving from the Cloud to the Edge

Over the last fifteen years or so, developers have become very familiar with cloud deployment. There are many reasons for its popularity which can mainly be summed up by saying that cloud simplifies the management of delivery of services to end users, with functions that span compute, storage and delivery.

Whether using AWS, Azure, GCP, Digital Ocean, or a more niche provider, the dev experience is fairly similar, no matter which cloud you're on. For a developer, cloud workloads typically include:

- Identifying where your highest concentration of users are and selecting a single cloud location that will deliver the best performance to the maximum number of users.
- Connecting your code base, hosted in code repository tooling (e.g. Github, GitLab, Bitbucket)
- Automating build and deployment through CI/CD tooling (e.g. Jenkins, CircleCI, Travis CI).

These processes are fairly straightforward when all code and microservices are feeding into a single deployment endpoint. However, what happens when you add hundreds of edge endpoints to the mix, with different microservices being served from different edge locations at different times? In this kind of environment, how do you decide which edge endpoints your code should be running on at any given time? More importantly, how do you manage the constant orchestration across these nodes among a heterogeneous makeup of infrastructure from a host of different providers?

The Challenge:

The Many Complexities of the Edge Developer Experience

Some of the complexities surrounding the edge developer experience include:

Code and configuration management

Edge runtimes

Distributed diagnostics and telemetry

Application lifecycle management

Code and Configuration Management

Every application is unique. Developers need granular, code-level control over edge configuration to fit the requirements of their applications. At the same time, they require simple, streamlined workflows to continue to push the pace of innovation and maintain secure, dependable application delivery.

With various industry players competing for a share of the edge computing market, from hyperscalers to CDNs, there are many considerations for evolving the developer experience to adapt to edge nuances. Many traditional CDNs, for example, have hard-coded proprietary software into their solutions (e.g. web application firewall technology), offering very limited configuration options. Thus, developers can find themselves backed into a corner with legacy CDNs offering edge services, forcing them to bolt on additional solutions that inevitably erase some of the benefits they were seeking to solve with the CDN solution in the first place.

Additionally, developers are increasingly looking to migrate more application logic to the edge for performance, security, and cost-efficiency gains.

Developer Workloads Moving to the Edge

The types of workloads being considered for edge deployment are many and diverse. Examples of developer workloads moving to the edge include:

- **Micro APIs -** Hosting small, targeted APIs at the edge for use cases such as search or fully-featured content exploration with GraphQL (which enables faster responses on user queries while lowering costs).
- **Headless Commerce -** Decoupling the presentation layer from back-end eCommerce functions allows you to push more services to the edge to create custom user experiences, achieve performance gains, and improve operational efficiencies.
- **Full application hosting at the edge -** More and more developers are exploring the idea of hosting their entire application at the edge. Rather than beaconing back to a centralized origin, hosting databases alongside apps at the edge and then syncing across distributed endpoints is quickly becoming a reality that has the potential to become the new normal as edge computing matures.

In order for developers to progress towards migrating more advanced workloads to the edge, they require flexible solutions that support distribution of code across programming languages and frameworks.

How Complexity Factors into Edge Runtimes

As more developers adopt edge computing for modern applications, edge platforms and infrastructure will need to support different runtime environments. Runtime describes the final phase of the program lifecycle, involving the machine executing the program's code.

Complexity factors into runtimes at the edge in particular in relation to interoperability, i.e. the complexity of managing runtimes across distributed systems. Developers need to be able to run applications in their dedicated runtime environment with their choice of programming language. Systems that support diverse developer needs must be able to support this to be useful to all.

One of the most widely used runtime environments for JavaScript is Node.js, used by many businesses, large and small, to create applications that execute JavaScript code outside a web browser. Other well-known examples of runtime environments include the Java Runtime Environment, a prerequisite for running Java programs, .NET Framework which is required for Windows .NET applications, and Cygwin, a runtime environment for Linux applications that allows them to run on Windows, macOS, and other operating systems.

The Challenge:

The Many Complexities of the Edge Developer Experience

With developers building across many different runtime environments, Edge as a Service offerings need to be able to support code portability. Developers can't be expected to refactor their code base to fit into a rigid, pre-defined framework. Instead, multi-cloud and edge platforms and services must be flexible enough to adapt to different architectures, frameworks and programming languages.

"And since we're in a DevOps world, not only do we need to think about the application development lifecycle and the CI/CD workflow, but we need to think about the observability and management of the application at the edge."

Stewart McGrath

Co-founder and CEO,

Section

Observability for Distributed Systems: Diagnostics and Telemetry

It is critical for developers to have a holistic understanding of the state of their application at any given time. Observability becomes increasingly complex when you consider distributed delivery nodes across a diverse set of infrastructure from different providers.

As reported in a recent Dynatrace survey of 700 CIOs, "The dynamic nature of today's hybrid, multicloud ecosystems amplifies complexity. 61% of CIOs say their IT environment changes every minute or less, while 32% say their environment changes at least once every second." To add to the complexities of trying to keep up with dynamic systems, that same report revealed that: "On average, organizations are using 10 monitoring solutions across their technology stacks. However, digital teams only have full observability into 11% of their application and infrastructure environments."

An effective solution for multi-cloud/edge observability should be able to provide a single pane of glass to draw together data from many locations and infrastructure providers. This kind of visibility is essential for developers to gain insight into the entire application development and delivery lifecycle. The right centralized telemetry solution will allow engineers and operations teams to evaluate performance, diagnose problems, observe traffic patterns, and share value and insights with key stakeholders.

The Challenge:
The Many Complexities of the Edge Developer Experience

Managing the Application Lifecycle

Along with configuration flexibility, control, and comprehensive observability tooling, developers need to be able to easily manage their application lifecycle systems and processes. With a single developer or small team overseeing a small, centrally managed code base, this is fairly straightforward. However, when an application is broken up into hundreds of microservices that are managed across teams, coupled with a diverse makeup of deployment models within the application architecture, this can become exponentially more complex and impact the speed of development cycles.

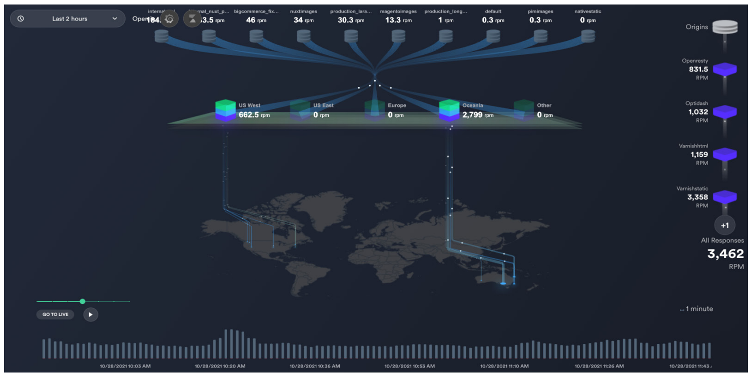
When we add the additional complexities of pushing code to a distributed edge and maintaining code application cohesion across that distributed application delivery plane at all times (even during an application deployment cycle), application lifecycle management for the edge becomes more complex than the centralized approaches used to date with cloud. This is where management solutions such as leveraging GitOps and CI/CD workflows come into play, so that the benefits of edge can be leveraged without increased team overhead.

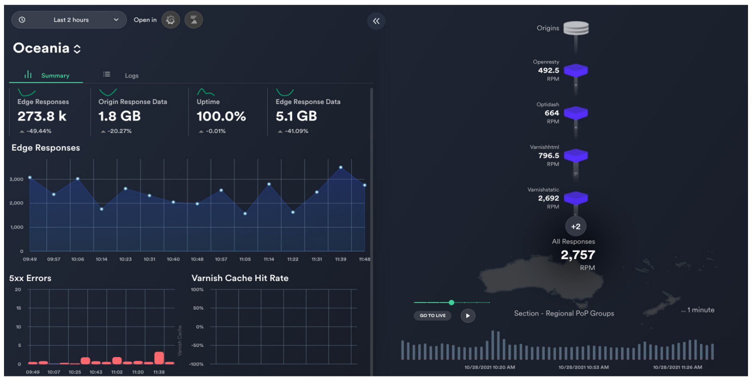
Leveraging GitOps and CI/CD Workflows

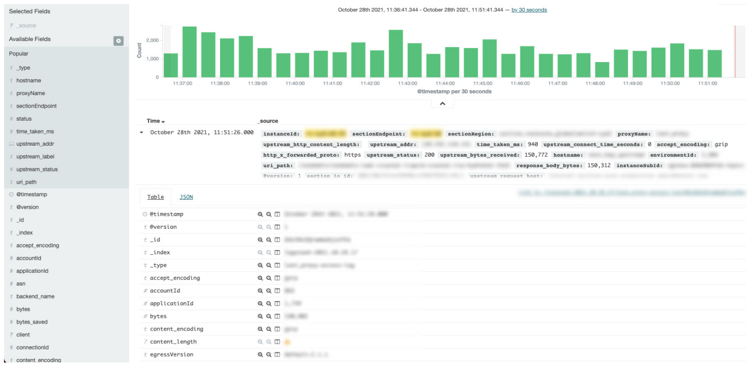
To streamline operations, many teams take advantage of GitOps workflows. GitOps is a way of implementing Continuous Deployment for cloud native applications. It focuses on a developer-centric experience when operating infrastructure, by using tools developers are already familiar with, including Git and Continuous Deployment tools.

While responsibilities and oversight of different parts of an application's code may be siloed within an organization, all of the code needs to feed into a unified code base. In order for developers to be able to move more services to the edge, they need tooling that is underpinned by GitOps principles for a fully integrated edge-cloud application lifecycle.

As an integral part of GitOps workflows, developers need flexibility and control when it comes to integrating edge deployment processes into existing continuous integration/continuous delivery (CI/CD) pipelines.







The Challenge:

The Many Complexities of the Edge Developer Experience

There are three main principles worth following when managing changes to your edge configuration through a CI/CD pipeline:

- **Optimize for fast feedback.** Identify steps within the pipeline that need optimizing by tracking execution time on individual stages. From the time you push a change to version control to making the change live should take no longer than five minutes. Fast feedback is important for quickly ensuring your changes meet business needs and cutting out technical debt and unnecessary costs.
- **Chunk your changes, test immediately.** Instead of changing multiple things in batches and then testing for the effect, interleave the changes and tests, and stop execution immediately if the tests fail. By turning changes into small, verifiable units, you lessen the risk factor.
- **Push all changes through the CI/CD pipeline.** You lose the benefits you're striving for if you accommodate changes outside the process.

Delivery and maintenance of code to distributed edge infrastructure can be more difficult to execute with the speed and consistency required to achieve the same application experience for end users at all times as if they were all connecting to one centralized application in the cloud. But using the right tools can make all the difference.

The Solution:

Consistency is Key

The key to accelerating edge computing adoption is making the experience of programming at the edge as familiar as possible to developers, explicitly drawing on concepts from cloud deployment to do so. The added complexities that a distributed edge deployment brings introduces new challenges to achieving consistency across these experiences.

"When we look at the challenges of scale and operational consistency, the edge cannot be seen as a point solution that then needs to be managed separately or differently across hundreds of sites – this would be incredibly complex. In order to be successful, you need to manage your edge sites in the same way you would the rest of your places in the network – from the core to the edge. This helps minimize complexity and deliver on the operational excellence that organizations are striving for."

Rosa Guntrip
Senior Principal Marketing Manager
Cloud Platforms, Red Hat

The right Edge as a Service platform will help minimize complexity, enabling developers to focus on innovation and executing mission-critical tasks instead of juggling all the pieces involved in managing edge/multi-cloud workloads.

Furthermore, growth in adoption of container technology and serverless functions has completely changed the game, leaving many legacy CDNs unequipped to support modern applications. With Kubernetes becoming the preferred container orchestration platform, edge solutions built on Kubernetes are significantly better positioned to support the needs of modern developers.

How to Approach Application Selection, Deployment, and Management for the Edge

The Context:

Complexities Enhanced by Placing More Parts of the Application at the Edge

During the cloud computing era, many application creators turned to complementary CDN technology to boost performance, security, and scalability. Today, they are turning to the edge for the next logical iteration. Placing parts of an application at the edge has obvious performance benefits but it also adds complexity to the simplicity of the cloud by adding an additional and discrete delivery layer.

With the demand for faster user experiences being driven by emerging and evolving use cases, application creators are increasingly looking to offload more services to the edge. At the same time, application operations teams are looking to simplify their delivery stacks. Bringing more of the application delivery cycle into a single cohesive edge delivery solution can achieve both of these goals concurrently.

While cloud providers have the flexibility to support a diverse range of workloads, developers working in the cloud are limited to a single provider's network, or are responsible for managing workload orchestration across multiple providers. CDNs, meanwhile, may have expansive global networks of infrastructure, but they are typically unable to support general purpose workloads beyond basic content delivery.

Content Delivery Networks (CDNs) are often thought of as the first evolution of edge computing. However, content delivery encompasses only a small subset of all edge workloads. As the diversity of edge workloads has expanded beyond content delivery, existing solutions fall short in terms of what they're able to support.

Many CDNs were built around open source technologies, such as Varnish Cache and ModSecurity. Typically, they have customized the code base so much over the years that developers using them are locked into "black box", proprietary solutions that don't offer the flexibility and control necessary to fit the unique requirements of each application.

Furthermore, growth in adoption of container technology and serverless functions has completely changed the game, leaving many legacy CDNs unequipped to support modern applications. With Kubernetes becoming the preferred container orchestration platform, edge solutions built on Kubernetes are significantly better positioned to support the needs of modern developers.

Emerging and Evolving Edge Use Cases:

- The Internet of Things (IoT)
- The Industrial Internet of Things (IIoT)
- Hospitals and health infrastructure
- Remote learning
- eCommerce and Retail
- Video conferencing
- Storage gateways

The Challenge:

The Complexities in Moving Diverse Workloads to the Edge

Now, let's take a deeper dive into some of the complexities involved in moving more diverse workloads to the edge, including selection, deployment and ongoing management.

Web Application Firewalls (WAFs) & Bot Management Tooling

DevOps teams are increasingly choosing to deploy WAFs and bot mitigation tools across a distributed architecture, with the goal of detecting and mitigating threats faster. Managing a WAF or bot mitigation deployment across a multi-cloud/edge network is no simple feat, however.

While many best-in-class WAF and bot management technologies have emerged - providers such as Wallarm, Snapt (WAF), ThreatX, Signal Sciences, Radware Bot Manager, and PerimeterX - most legacy CDNs still don't give developers the option of deploying third-party solutions. Fastly, for example, recently acquired Signal Sciences, recognizing the need for more advanced WAF technology beyond their own proprietary solution.

We often speak with developers who are frustrated with the "black box", built-in solutions of legacy CDNs, and demand more choice and flexibility.

Image Optimization

Beyond the simple caching of images, developers, especially in the e-Commerce sector, are increasingly seeking out image optimization solutions, such as Optidash, that optimize and transform images on-the-fly.

Image optimization benefits include:

- **Faster page load times for end users**
- **Improvements to operational efficiency**
- **Removing the load on centralized infrastructure**

Just as with security solutions, most legacy CDNs don't support third-party software that specializes in point solutions. What's more, if you're operating a multi-cloud/edge environment, you will have to install and manage these types of image optimization tooling across the entire network.

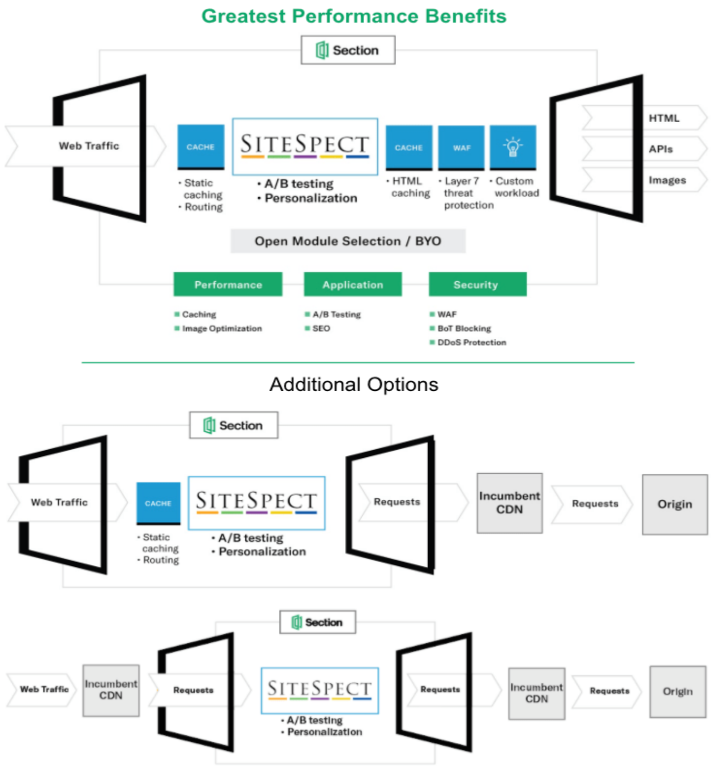
Modern Testing & Experience Optimization

Marketers, product managers, developers and others need the ability to effectively test and optimize applications across the client-side, server-side, single page application (SPA), mobile, redirects, and so on. Conventional A/B testing solutions use JavaScript tags to manipulate content on applications, which reduces site performance with flicker and increased latency.

Modern tools like SiteSpect, however, rethink this model by sitting in the flow of HTTP traffic. This allows them to support multiple user experience optimization techniques, including client-side, server-side, redirects, and SPA optimization.

Legacy CDNs can't support this new architectural model and therefore require extra hops in the HTTP delivery chain, ironically negating many of the performance benefits they are aiming to solve.

Quick and Flexible Deployment Options



The Challenge:

The Complexities in Moving Diverse Workloads to the Edge

Load Balancing Solutions

While most hyperscalers and edge providers offer load balancing, these solutions are often restricted to their own environments. Therefore, if you migrate your application to a different cloud or data center, the hyperscaler or edge provider's proprietary load balancer won't be able to follow.

In the instance of a traditional load balancer that is being deployed to the cloud, you need to use a virtual appliance. If you then decide to use a load balancer in a second cloud, that virtual appliance will need to be re-configured again… and so on for every cloud or data center it operates in. There is no communication between these two appliances. In this instance, you are operating two (or more) separate clouds that your teams will need to manage separately.

Organizations that use multi-cloud/edge networks are then faced with having to separately configure, monitor and manage delivery and security for each distinct environment. Similarly, for any application that changes hosting location, adjustments must be made on an individual basis. This not only increases complexity, but takes up valuable resources and limits much of the flexibility that is supposedly a key benefit of a multi-cloud/edge model.

Containers: Challenges of Orchestration

In a small environment with only a handful of systems, managing and automating orchestration is fairly straightforward, but when an enterprise has thousands of individual systems that interact with each other on some level, orchestration automation is both powerful and essential.

Containers are lightweight by definition with a low footprint, making them perfect candidates for running on edge devices. The main reason machine learning models leverage containers is because legacy devices can still interact with cloud services like AI/ML to achieve fast computation in-place.

Containers can be deployed to the device of your choosing and can be built using the architecture of your choice so long as it can run the container runtime. Updating containers in-place is simple, particularly when orchestration solutions like Kubernetes are used.

Consider SaaS providers who traditionally offered on-premise or single point of presence installations. As customers increasingly demand distributed deployment models, SaaS providers are faced with the build vs. buy dilemma.

The management of these complex clusters of devices, services, and networks can get highly complicated very quickly.

Serverless Computing for Edge Computing

Serverless computing, also called function as a service (FaaS), enables the execution of event-driven logic without the burden of managing the underlying infrastructure. The name 'serverless' is characterized by the freedom that it gives developers to focus on building their applications without having to think about provisioning, managing, and scaling servers.

The concept of serverless was originally designed for cloud environments, eliminating the 'always-on' model to save on resource consumption, among other benefits. In recent years, advances in edge computing technology have led more developers to migrate serverless workloads to the edge. The benefits of serverless at the edge, when compared to alternatives like containers and VMs, include lighter resource consumption, improved cost efficiencies, code portability, and speed of deployment.

However, not all workloads are suitable for serverless models and it's important to understand the requirements of a given workload when determining the most appropriate deployment model. Considerations such as code dependencies, cold starts and their effect on performance, security, and resource requirements are critical when designing edge architectures.

The Solution:

Partner with an Edge as a Service Provider

An Edge as a Service (EaaS) provider can help overcome many of the complexities involved in application deployment and management at the edge.

The right EaaS provider, for instance, won't lock developers into specific software choices for security or image optimization tooling, allowing for "best of breed" selection. EaaS can also support distributed deployment of more advanced workloads, making it easier to integrate testing and experience optimization solutions like SiteSpect into your edge stack.

One of the most important areas that Edge as a Service can make more straightforward is load balancing across multi-cloud/edge networks. EaaS providers may offer automated load balancer options, which automatically migrates traffic and workloads so that developers don't have to do this on an individual basis.

Similarly, Edge as a Service providers can help streamline containerization and serverless deployments by (i) containerizing applications and accelerating the developer path to the edge and (ii) offering flexible language support that allows developers to simply ship code and offload the responsibilities of deployment, management, and scaling of the underlying infrastructure to the edge compute platform.

The Complexities of Building and Operating Edge Networks and Infrastructure

The Context:

Edge Networks and Infrastructure are Changing

We are in the midst of a foundational technological shift in communications infrastructure. IDC predicts that by 2023, more than 50% of new enterprise IT infrastructure will be deployed at the edge and the edge access market is predicted to drive $50 billion in revenues by 2027. To interconnect this hyper-distributed environment, which spans on-premise data centers, multi-clouds, and the edge, the network is in the process of evolving and becoming more agile, elastic, and cognitive.

The Challenge:

Managing the Network, Infrastructure, and Operations in a Distributed Compute Environment

Let's dive into a high-level overview of some of the critical components you need to consider when building and operating distributed networks.

Areas that we'll cover include:

- DNS
- TLS
- DDoS mitigation (Layers 3, 4, 7)
- BGP/IP address management
- Edge location selection and availability
- Workload orchestration
- Load shedding and fault tolerance
- Compute provisioning and scaling
- Messaging framework
- Edge operations model and team
- Observability
- NOC Integration

As you're evaluating all of these considerations, bear in mind that working with Edge as a Service can solve many of these complexities for you.

DNS: A Critical Component in Networking Infrastructure

The domain name system (DNS) is often referred to as the phonebook of the Internet since it translates domain names to IP addresses, allowing browsers to load Internet resources. DNS provides the hierarchical naming model, which lets clients "resolve" or "lookup" resource records linked to names. DNS therefore represents one of the most critical components of networking infrastructure.

DNS Services are Often Vulnerable to Threats

Other widely used Internet protocols have started to incorporate end-to-end encryption and authentication. However, many widely deployed DNS services remain unauthenticated and unencrypted, leaving DNS requests and responses vulnerable to threats from on-path network attackers. Hence, when building out DNS services, it's critical to maintain a security-first approach.

The Challenge:

Managing the Network, Infrastructure, and Operations in a Distributed Compute Environment

"Enterprise infrastructure is evolving faster than ever before. Emerging technologies make it possible to spin up microservices and cloud instances in minutes. DevOps teams are churning out code 40 times faster than legacy production environments. New edge and serverless architectures are taking computing out of the data center and closer to devices enabling global real-time applications. Those organizations not born in the cloud-native era face the additional challenge of connecting legacy applications with new technology in the never-ending race to meet user demands for performance while driving efficiency and security."

Kris Beevers
CEO,
NS1

DNS in a Distributed Computing Environment

DNS is lightweight, robust and distributed by design. However, new approaches to computing architecture, including multi-cloud and edge, introduce new considerations when implementing application traffic routing at the DNS level.

In a distributed compute environment, DNS routing entails ensuring that users are routed to the correct location based on a set of given objectives (e.g. performance, security/compliance, etc.). When routing, you need to take into account service discovery. The majority of the time, people use DNS for public-facing service discovery, but this can be challenging to pull off. Routing is complicated both in terms of ensuring that users are routed to the right location and that failover is handled - in other words, what will you do to help update routes when systems fail?

The Challenge:

Managing the Network, Infrastructure, and Operations in a Distributed Compute Environment

TLS: Provisioning, Management, and Deployment Across Distributed Systems

Transport Layer Security (TLS) is an encryption protocol that protects communications on the Internet. You can feel reassured that your browser is connected via TLS if your URL starts with HTTPS and there is an indicator with a padlock assuring you that the connection is secure. TLS is also used in other applications, such as email and usenet. It's important to regularly upgrade to the latest protocol for TLS and its predecessor, the SSL protocol.

When working with TLS and/or SSL in distributed environments, you have to either work with your own certificates using a managed service such as DigiCert or use an open source version with a service like Let's Encrypt or Certbot. The managed certificate authorities such as DigiCert will provide automated tooling to provision certificates. With the open source versions, you will find that you have to build the services to manage auto-renewal components and the provisioning of new certificates.

An added complexity is the question of how you deploy these protocols? In relation to distributed systems, you will have certificates that need to be running in multiple places. They might be running across multiple providers, for example, you might be using one specific ingress controller in one location and a different ingress controller in another. How do you ensure that your certificates are being deployed where needed in order to actually handle the TLS handshakes? And as the number of domains that you manage increases, so too do the complexities.

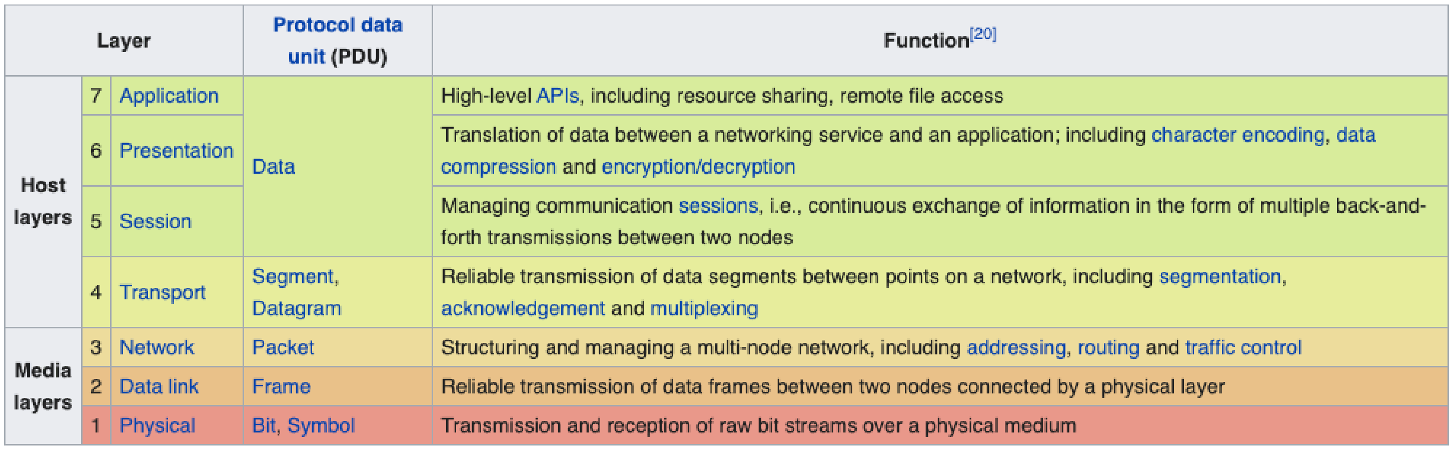
This ties directly back to DNS since you need to ensure that you're routing traffic to the correct endpoints containing the workloads where your TLS certificates are deployed. Further, you will have to take into account the state of your systems at any point in time and how you route traffic, since you never want to be servicing users incorrectly.

Ultimately, servicing your user correctly is the end goal, meaning that when implementing TLS at the edge yourself, you have to take into account all these different components.

DDoS: Protecting Layers 3, 4, and 7

When protecting against Distributed Denial of Service (DDoS) attacks across distributed systems, the first question to ask should be, where are my systems most vulnerable to attack? The primary layers of focus on protecting against DDoS attacks include Layers 3, 4, and 7 in the OSI model.

OSI model



Layer			Protocol data unit (PDU)	Function[20]
Host layers	7	Application	Data	High-level APIs, including resource sharing, remote file access
	6	Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5	Session		Managing communication sessions, i.e., continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4	Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3	Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2	Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
	1	Physical	Bit, Symbol	Transmission and reception of raw bit streams over a physical medium

The Challenge:

Managing the Network, Infrastructure, and Operations in a Distributed Compute Environment

Firms like Wallarm, Signal Sciences, ThreatX, and Snapt will provide DDoS protection for you at the application layer (i.e. Layer 7). However, in an edge computing paradigm that's made up of heterogeneous networks of providers and infrastructure, there are more questions that need asking. The most important: how do all the different providers I'm using handle network and transport-layer DDoS attacks (i.e. Layers 3 and 4)?

All major cloud providers typically have built-in DDoS protection, but when you begin to expand across a multi-cloud environment, and further out to the edge, you need to ensure that your applications are protected across the entire network. This includes knowing how each underlying provider handles DDoS protection, along with implementing safeguards for any areas in your networks that may be underprotected. This takes us back to DNS and the question of how to handle traffic routing when one (or more) of your endpoints becomes compromised.

BGP/IP Address Management

The Border Gateway Protocol (BGP) is responsible for examining all the available paths that data can travel across the Internet and picking the best possible route, which usually involves hopping between autonomous systems. Essentially, BGP enables data routing on the Internet with more flexibility to determine the most efficient route for a given scenario.

BGP is also widely considered the most challenging routing protocol to design, configure, and maintain. Underlying the complexities are many attributes, route selection rules, configuration options, and filtering mechanisms that vary among different providers.

In an edge computing environment, rather than announcing IP addresses from a single location, BGP announcements must be made out of multiple locations, and determining the most efficient route at any given point becomes much more involved.

Another important consideration when it comes to routing is load balancing at the transport layer (Layer 4). Building a Layer 4 load balancer is complicated for the following reasons:

- It must support BGP announcements.
- You need to own the IP space, which can be very costly.
- You need to understand BGP, which may require a team of network engineers to truly manage the system.
- You need to be able to announce in locations all around the world (which are also the most peered locations).
- Finally, you need to take into account where you're load balancing traffic to. The distributed system that you're running applications on must be able to support packets that are being load balanced from the load balancer in front
- BGP/IP address management

The complexities of routing across multi-layer edge-cloud topologies are perhaps the most daunting when it comes to building distributed systems.

Edge Location Selection and Availability

An effective presence at the edge is based on having a robust location strategy. By moving workloads as close as possible to the end user, latency is reduced. Selecting the appropriate geographies for your specific application within a distributed compute footprint involves careful planning.

Compliance also plays a role in the selection of edge locations. Increasingly, regulations and compliance initiatives, such as GDPR in Europe, are requiring companies to store data in specific locations.

Edge Workload Orchestration

Managing workload orchestration across hundreds, or even thousands, of edge endpoints is no simple feat. This can involve multiple components. You need to start with where you want the workload to be defined (e.g. full application hosting, micro APIs, etc.) Next, ask where will it be stored? Finally, take into account how the workload is actually deployed. How do you determine which edge endpoints your code should be running on at any specific time? What type of automation tooling and DevOps experience do you need to ensure that when you make changes, your code will run correctly?

Managing constant orchestration over a range of edge endpoints among a diverse mix of infrastructure from a network of different providers is highly complex.

The Challenge:

Managing the Network, Infrastructure, and Operations in a Distributed Compute Environment

Load Shedding and Fault Tolerance

A load shedding system provides improved fault tolerance and resilience in message communications. Fault tolerance allows a system to continue to operate, potentially at a reduced level, in the event of a failure within one or more of its components.

In regards to load shedding and fault tolerance at the edge, the primary area of concern is ensuring that the systems handling your workloads and servicing requests aren't overloaded. Essentially, how do you make sure that one location isn't set up to infinitely scale and how do you ensure that load is distributed appropriately?

Compute Provisioning and Scaling

Load shedding and fault tolerance brings us to auto scaling and configuring auto scaling systems. One of Kubernetes' biggest strengths is its ability to perform effective autoscaling of resources. Kubernetes doesn't support just one autoscaler or autoscaling approach, but three. These are:

- Pod replica count - This involves adding or removing pod replicas in direct response to changes in demand for applications using the Horizontal Pod Autoscaler (HPA).
- Cluster autoscaler - As opposed to scaling the number of running pods in a cluster, the cluster autoscaler is used to change the number of nodes in a cluster. This helps manage the costs of running Kubernetes clusters across cloud and edge infrastructure.
- Vertical pod autoscaling - the Vertical Pod Autoscaler (VPA) works by increasing or decreasing the CPU and memory resource requests of pod containers. The goal is to match cluster resource allotment to actual usage.

If you're not using a container orchestration system like Kubernetes, compute provisioning and scaling can get challenging very quickly.

The Messaging Framework

The messaging system provides the means by which you can distribute your configuration changes, cache ban requests, and trace requests to all your running proxy instances in the edge network or CDN, and report back results.

This involves two primary components:

- Workload orchestration
- Receiving updates - if your system needs to receive a message or update, how do they get it, and is it getting there reliably?

Observability for Distributed Systems

It's imperative to treat observability as a first-class citizen when designing distributed systems, or any system for that matter. Reliable and real-time information is critical for engineers and operations teams who need to understand what is happening with their applications at all times.

Observability is also a key element in disaster recovery planning and implementation. Your infrastructure needs to be observable and flexible so that you can understand what has broken and what needs to be fixed. If a critical error occurs, you need visibility into your system to keep the incident as brief as possible versus experiencing a protracted disaster.

NOC Integration

Enterprises are taking steps to unify their network operations centers (NOCs) and security operations centers (SOCs). Why? By creating alignment between these two frequently siloed teams, organizations can reduce costs, optimize resources and improve the speed and efficiency of incident response and related security functions.

You need to take into account the expertise of your team when planning a NOC and/or SOC integration. Not everyone will have the range of crossover experience necessary to pull off a successful integration.

The Solution:
The Right Expertise

To truly manage distributed network, infrastructure and operations at the Edge, you will likely need an edge operations model with an experienced team comprised of:

Network engineers

Platform engineers

DevOps engineers, with an emphasis on site reliability engineering (SRE)

If you don't have some or all of these specialists, or don't have the expertise or resources in edge computing to manage the network, infrastructure, and operations, you can work with an Edge as a Service provider whose solutions abstract away many of the complexities associated with edge computing.

Conclusion: Overcoming the Complexities

As we've just started to look at the puzzle, the reality of deploying and managing workloads at the edge is far from simple. And while we've covered a lot in this white paper, this is just a subset of the critical components and complexities that come with building and operating cloud-edge networks and infrastructure; and we really only scratched the surface on each of the considerations included.

Every organization, team, and application has unique requirements when it comes to designing distributed systems. Because of this, many teams start down the path of building their own bespoke systems, and typically become overwhelmed quickly with all of the complexities that play into design decisions and implementation.

The real power of the edge arises when we provide application developers the opportunity to seamlessly run the software of their choice at the edge and application operations teams the simplicity of a single delivery plane so they have a reduced operational footprint (even with a larger geographic delivery footprint).

As an Edge as a Service provider, Section is often pulled into projects during the early stages of research and discovery, where we're able to offload the build and management of many, if not most, of the critical components, ultimately accelerating the path to edge for organizations across a diverse range of use cases.

Section's Edge as a Service simplifies all the steps involved in deploying your application to the edge. You also gain the round the clock support of our dedicated team of expert engineers. We take care of the massive complexities and resources necessary to support distributed provisioning, orchestration, scaling, monitoring and routing, allowing you to focus on innovation.

For more information about how edge can work for you or how to get started, reach out to us at section.io/contact-us.

Citations / Related Research

"State of the Edge Report 2021", The Linux Foundation, 2021
https://stateoftheedge.com/reports/state-of-the-edge-report-2021/

"The Move Towards a Multi-Cloud and Hybrid IT Infrastructure", Section, January 20, 2021
https://www.section.io/blog/multi-cloud-hybrid-it-infrastructure/

"The Balance Between Granular Control and Simplicity in Edge as a Service", Section, February 23, 2021
https://www.section.io/blog/balance-control-simplicity-edge-as-a-service/

"Headless Commerce Drives Edge Computing Adoption", Section, October 28, 2019
https://www.section.io/blog/headless-commerce-drives-edge-computing-adoption/

"Is Edge as a Service (EaaS) the Next Big Thing in Tech?", Section, January 27, 2021
https://www.section.io/blog/edge-as-a-service-the-next-big-thing-in-tech/

"Observability, automation, and AI are essential to digital business success", Dynatrace, October 14, 2020
https://www.dynatrace.com/cio-report-automatic-and-intelligent-observability/

"Managing Changes to Your Edge Configuration Through a CD Pipeline", Section, December 9, 2019
https://www.section.io/blog/edge-configuration-management-cd-pipeline/

"Edge Computing Use Cases Driving Innovation", Section, January 22, 2019
https://www.section.io/blog/edge-compute-use-cases/

"CDNs Were a Prototype for Edge Compute", Section, September 10, 2018
https://www.section.io/blog/cdn-prototype-edge-compute/

"Kubernetes Is Paving the Path for Edge Computing Adoption", Section, May 12, 2020
https://www.section.io/blog/kubernetes-enabling-edge-computing-adoption/

"Build vs. Buy - Evaluating Edge Solutions", Section, September 16, 2020
https://www.section.io/blog/edge-computing-build-vs-buy/

"IDC FutureScape: Worldwide IT Industry 2020 Predictions", IDC, October 2019
https://www.idc.com/getdoc.jsp?containerId=US45599219

"Comparing OpEx vs. CapEx Models at the Edge", Section, April 23, 2020
https://www.section.io/blog/opex-vs-capex-edge-computing-model/

"Understanding Your Edge with Section's Next-Gen Observability Tooling", Section, October 7, 2020
https://www.section.io/blog/traffic-monitor-edge-observability-tooling/

Section

About Section

Section's Edge as a Service technologies power next-gen applications with faster, simpler, and more secure digital experiences. For application engineers, the Section platform removes the burdens associated with Edge infrastructure provisioning, workload orchestration, scaling, monitoring, and traffic routing, so they can focus on innovating their core product. The Section Edge is built on the backbone of a vendor-agnostic global network of leading infrastructure providers spanning the edge-cloud continuum and offers the most flexible Edge solutions to meet the unique requirements of any application. Founded in 2012 in Sydney, Australia, Section moved its headquarters to Boulder, CO in 2016 and continues to grow its team, partners, and customers across the globe.

Ready to Jump In?

Visit our website to learn more:

section.io