LEARNING MADE EASY

Extreme Networks Special Edition

Cloud Managed Networking

for dummies®
A Wiley Brand

Cloud services and deployment models

Modern software architectures and technologies

Data science, machine learning, and AI

Compliments of

Extreme™
Customer-Driven Networking

Marcus Burton, CWNE #78

About Extreme Networks

Extreme Networks, Inc. (EXTR) creates effortless networking experiences that enable all of us to advance. We push the boundaries of technology leveraging the powers of machine learning, artificial intelligence, analytics, and automation. Over 50,000 customers globally trust our end-to-end, cloud-driven networking solutions and rely on our top-rated services and support to accelerate their digital transformation efforts and deliver progress like never before. For more information, visit www.extremenetworks.com or follow us on Twitter, LinkedIn, and Facebook.

Cloud Managed Networking

Extreme Networks Special Edition

by Marcus Burton, CWNE #78

for dummies®
A Wiley Brand

Cloud Managed Networking For Dummies®, Extreme Networks Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN 978-1-119-59219-8 (pbk); ISBN 978-1-119-59221-1 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Development Editor: Kim Wimpsett, Jennifer Bingham
Project Editor: Jennifer Bingham
Technical Editor: David Coleman
Editorial Manager: Rev Mengle
Business Development Representative: Karen Hattan
Production Editor: Tamilmani Varadharaj

Acknowledgments

I would like to thank a few people who made this book what it is.

David Coleman, thanks for asking me to write this book and managing so much of the process. You saved me many gray hairs.

Markus Nispel, thanks for having questionable judgment by hiring me at Extreme and for encouraging me to write.

GT Hill, thanks for cracking me up and storyboarding with me. It brings clarity.

Donagh Horgan, thanks for patiently sharing your data/ML/AI brilliance with me. Chapter 4 came together because of you. If it's bad, it's my fault.

Brendan Bond, thanks for the sparkling cover design.

Jen Bingham and Kim Wimpsett, thanks for being super fab development editors.

And a special shoutout to Lakshmi, See Ho, and Phal for hand-holding me on my own journey toward cloud and data science.

About the Author

Marcus Burton is a Wi-Fi and networking veteran who's currently working at Extreme Networks as a cloud technology-adoption architect with a focus on solving customer headaches using data science, machine learning, and AI. You may recognize him from previous work writing books, white papers, blogs, and exams as the technology and product lead at CWNP, or from Ruckus Wireless, where he held roles in product leadership. Marcus has a BA in English, psychology, and theology. He's a CWNE, and he's always happy to talk about fly-fishing, hunting, Shakespeare, adoption, politics, faith, and other squirmy topics.

Contents at a Glance

Introduction

As I look back at my own career progression, it fascinates me to see how skills and interests evolve. I started my career in technology editing technical exams because of my writing/editing and psychology background. So I was forced to pick up the technology in a fury — Wi-Fi was my first focus. I love the bits and bytes, the invisible RF magic, and the protocol details. Then I branched toward switching, routing, and network systems to fill in some of my knowledge gaps. Then along came virtualization. Then cloud. And then APIs. Then data science, machine learning (ML), and artificial intelligence (AI). And software development processes and tools. And there were soft skills in between, like public speaking, tuning up my writing, UX design, how to use PowerPoint like a boss, and others.

What I am trying to say is that technology does not leave room for stagnation. Technology keeps evolving. I suspect that many readers of this book are coming in from a networking domain perspective — network fabrics, routing and WAN, datacenter, wireless, and maybe others. You probably see the vast momentum of the cloud, but may not yet have a handle on cloud systems and architectures.

If that resonates with you, this book is for you. It was written from the perspective of someone with a strong network-domain heritage. My goal is to cover all the modern cloudy buzzwords and terms that we so often hear in marketing but don't necessarily understand. This book may not make you a cloud expert, but it will give you perspective to evaluate cloud systems critically. And, it will give you language to understand and talk more intelligently about cloud systems, software service and deployment architectures, data science and machine learning, and so much more.

I am glad you picked up this book (or perhaps opened the PDF). Thanks for joining me on this journey. Your future self thanks you.

Icons Used in This Book

Throughout this book, I occasionally use special icons to call attention to important information. Here's what to expect.

REMEMBER

This icon points out information you should commit to your non-volatile memory, your gray matter, or your noggin — along with anniversaries and birthdays!

TECHNICAL STUFF

You won't find a map of the human genome here, but if you seek to attain the seventh level of NERD-vana, perk up! This icon explains the jargon beneath the jargon!

TIP

Tips are appreciated, never expected — and I sure hope you'll appreciate these tips. This icon points out useful nuggets of information.

WARNING

These alerts point out the stuff your mother warned you about (well, probably not), but they do offer practical advice to help you avoid potentially costly or frustrating mistakes.

Beyond the Book

If you find this book interesting and want to dive in on more, or follow ongoing content developments, here are some links to more resources:

- **Extreme's Cloud Technology web page:** `www.extremenetworks.com/cloud-technology`.
- **My Blog Page at Extreme Networks:** `www.extremenetworks.com/extreme-networks-blog/author/mburton`.

  There are other great blog authors at Extreme Networks, so please check them out as well.
- **A video podcast series discussing cloud and data science:** `https://bit.ly/Cloud-Data-Science`.
- **Amazon's AWS Cloud Practitioner Training:** `https://aws.amazon.com/training/path-cloudpractitioner`.

  Note that Google and Microsoft have training programs and vast amounts of information on their websites, but I would argue Amazon's are still the best.

IN THIS CHAPTER

- Simplifying the network operations paradigm
- Changing expectations for network reliability and scalability
- New cost dynamics and data-first applications

Chapter 1
Why Cloud for Networking?

The cloud has already radically changed just about every industry and its use of technology. Classrooms have adopted Google Docs; enterprises have migrated to Office 365; customer management has shifted to Salesforce. While the actual user applications have shifted to the cloud, some network components need to remain on-premises to support user traffic. But, there's no reason that IT teams and network operators shouldn't enjoy the same benefits from the cloud that they've extended to their users.

The cloud offers an alternative model for consuming IT services in which companies move away from on-premises models and toward hosted services where cloud providers offload IT operations on their behalf.

Many enterprise IT departments have already made the transition and are reaping the fruit of this decision, but other companies may still be waiting for an upgrade cycle and weighing the benefits of moving to the cloud. These benefits stretch across industries, but networking applications stand to benefit in several distinct ways. This chapter gives you an overview of the benefits.

Simplicity

Traditionally, customers managed all the installation and operation of hardware, power, cabling, upgrades, backups, migrations, and redundancy. They did this for routers, switches, controllers, access points, management tools, gateways, concentrators, and other services. They ensured proper connectivity and bandwidth, security, and integration among all these systems. And then they monitored, troubleshot, inventoried, managed support issues, and planned lifecycle for all of it, end to end. Across a large or distributed deployment, it's a lot of work. This is where cloud networking comes to the rescue.

REMEMBER

The cloud eases the demands of traditional deployment models by shifting the burden of operation to the cloud provider. The operation of all these services is largely transparent to the customer, and the cloud provider offloads the complexity and overhead, so IT groups can focus on solving business problems instead of hassling with servers.

Customers enjoy faster feature velocity and the latest functionality without the pain points of system migration, upgrades, surprising bugs, overnight maintenance windows, rollbacks, and other headaches. From an infrastructure perspective, it becomes someone else's responsibility — and they are server and datacenter specialists.

Cloud simplicity also improves daily workflows, such as logging in without needing a VPN, registering new devices, configuring and applying policies centrally, license management, and enjoying centralized visibility across distributed deployments.

Resiliency

Not only does the cloud provider simplify system management, but it will also typically make a guarantee to provide 99.99 percent (or more) uptime. Cloud solutions are built on top of hyper-reliable infrastructure and platform architectures offered by trusted cloud operators like Amazon, Microsoft, and Google.

Of course, there are many different versions of cloud architecture, and not all clouds are created equal (see Chapters 2 and 3); but in general, the user experience of the cloud is highly reliable because the cloud has inherent resiliency against failure and redundancy in case of failure.

Scalability

When you hear about scalability, you may think about vast infrastructure networks, and you may tune out if you don't manage such a network. Scale is not just about growing big; it's also about any changes that drive a network into new tiers of operation, like expanding or changing client device or user populations, new business requirements, spikes in events/alarms, adding locations/sites, new data visibility or reporting requirements, or simply logging in more frequently. Little stairsteps in growth eventually add up and create scale pinch points.


TECHNICAL STUFF

The cloud replaces the fixed resourcing paradigm of on-premises hardware and addresses scalability by using an elastic and flexible resource paradigm. Cloud resources can be closely monitored and integrated with automation tooling that can dynamically scale up and down as capacity requirements change.

Vendor solutions continually evolve based on new applications and customer requirements, but traditional solutions bind up that evolution with resource limitations. There are always challenges with adding more boxes to scale. Even with virtual machines, the hardware resource pool may be exhausted, which might require a new budget cycle to justify additional spending.

In the cloud, these scale transitions are much easier to handle for network admins, and it's often transparent. The cloud scales linearly, simply by adding licenses and connecting devices. Cloud operators can also more easily manage system evolution because cloud services are modular, and scale limitations can be addressed either with additional resources or by architecture shifts to solve specific pain points.

Cost

By leveraging economies of scale, cloud providers have dramatically dropped the street price of compute, memory, and disk resources, and made those lower prices available to users.


TIP

Costs are also easier to predict and manage because the cloud avoids those transitional growth stairsteps, like when your existing appliance supports 1,000 devices, but your new design pushes you up to 1,025. Cloud licenses are typically sold in linear units as subscriptions, avoiding unexpected CAPEX costs for infrastructure at known or unknown thresholds. Perhaps the most important cost aspect is that the cloud enables customers to transition into operational expenditures (OPEX) instead of high upfront capital expenditures (CAPEX), which are common to on-premises solutions.

And beyond that, cloud services increasingly leverage on-demand computing models (for example, serverless) that streamline operational costs, which can be passed on to customers in several forms.

Data-Driven Applications

As customers become increasingly data-driven, and as machine learning (ML) and artificial intelligence (AI) automate networks, cloud solutions are outpacing on-premises options because the platforms enable faster development. Cloud makes high-scale data processing and storage easier, and provides a more robust and modular toolset for data pipelines. Cloud systems also make ML and AI more accessible because cloud toolkits are better for model training and maintenance, and of course the training dataset is richer in a big data cloud architecture. (I talk about this more in chapter 4.)

With all these benefits, cloud has become a fast favorite deployment model for networking solutions.

IN THIS CHAPTER

- Breaking down the cloud service models
- Looking at cloud deployment models
- Exploring cloud geography models

Chapter 2
Cloud Fundamentals

Cloud services are an on-demand model for computing, storage, security, networking, and other IT resources that are provided as a service, hosted in remote datacenters, accessed via the Internet, and paid by usage.

REMEMBER

Even as I attempt to craft that definition in a universal way, not everyone will agree. There are many variations of cloud technology. This chapter breaks down the cloud by looking at three ways to describe it: cloud service models, cloud deployment models, and cloud geography models.

Cloud Service Models

A few common models drive the way that cloud services are offered, and they are called *as a service* models. Figure 2-1 shows the three primary types of service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). You can think of them as progressive stages of readiness for a final cloud application.

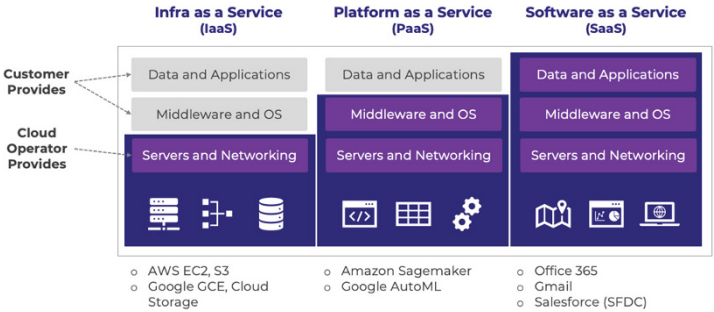


FIGURE 2-1: Cloud service models.

Infrastructure as a Service

Infrastructure as a Service (IaaS) is a cloud service model in which cloud providers offer compute, storage, networking, security, and other infrastructure services in a hosted offering. Customers pay for these services based on usage, and the customer is responsible for building everything on top of the infrastructure.

The customer decides what operating system to install, what data to store, or what application to run on those resources. For example, AWS provides elastic cloud computing (EC2), which is a hosted server. You can pay for a general-purpose compute instance with two vCPUs and 4 GB of memory, then connect that instance to a hosted 100 GB SSD volume for storage. Then you can run Linux, Windows, containers, databases, or whatever you'd like on those resources.

REMEMBER

In other words, IaaS is a way to pay for the raw infrastructure, which the cloud operator oversees on your behalf. You're responsible for everything that will run on the infrastructure, but they ensure that the infrastructure itself is reliable.

Serverless

Some computing workloads that were traditionally performed with virtual cloud instances (IaaS) have been replaced with a new form of computing called Serverless. Serverless is a way to use cloud computing on an event-driven basis. You can think of serverless as a method of running code with complete transparency to the computer beneath it — at least for the one utilizing serverless services, not for the cloud provider (Hint: There is still a server under there).

Why serverless? Because administering servers can be a pain in the neck. And for workloads that are temporary, unpredictable, occasional, intermittent, or scheduled (in other words, event-driven), it doesn't make much sense to have a dedicated compute instance (even in the cloud) running all the time, just to wait for these occasional tasks.

Even with modern orchestration, it's still inefficient to bring up and down compute instances to run lean snippets of code for a specific event transaction. Instead, serverless offers a way to do away with all the underlying responsibility of servers, OSes, and operational overhead, and simply run code on an as-needed basis. The cloud provider runs the code when you tell it to, autoscales the systems running the code, and simply charges for the computing time necessary to run the code. A 5-second compute task uses 5 seconds of computing resource, and not 24 x 7 computing time. For the right workloads, serverless provides tremendous efficiencies.


TIP

Serverless is also known as Functions as a Service (FaaS). If you want to read more about serverless, Amazon's serverless technology is called Lambda; Google and Microsoft call theirs Google Functions and Azure Functions, respectively.

Platform as a Service

The Platform as a Service (PaaS) model is a way for cloud providers to make application development easier. A PaaS bundles together computing services and other solutions that provide a more accessible framework for application development. Think of PaaS as a set of tools and middleware services layered on top of IaaS that take you one step closer to deploying code in the cloud.

For example, imagine that your application requires an analytics engine and database, and you plan to leverage Elastic Search, an open-source package. You could pay a cloud provider for compute and storage (IaaS) and then install, optimize, control, migrate, patch, and maintain Elastic Search yourself. Or, cloud providers can offer Elastic Search as a hosted service, which you pay for directly — they give you a ready-to-use Elastic Search instance. Avoid all those operational steps of IaaS, and just integrate your data into the hosted instance and start building your app faster with less overhead.

Any number of platforms can be offered in this model, like databases, container services, machine learning toolsets, messaging services, orchestration, and more. PaaS removes several operational burdens from application developers because the cloud operator directly manages those components.

REMEMBER

The large cloud operators like Google, Amazon, and Microsoft often build software tools to solve their own internal company problems — for example, machine learning toolkits, scalable databases, or orchestration tooling. Then, they take that same underlying technology and offer it as a service in the cloud. It's really difficult for in-house IT teams to compete with this kind of innovative scale and momentum, which is why PaaS is completely revolutionizing the pace at which cloud services are built.

But keep in mind, PaaS is cloud operator specific. If you build on PaaS, it may lock you into one cloud provider's ecosystem.

Software as a Service

Finally, the Software as a Service (SaaS) model is effectively a finished application, built on a cloud infrastructure, and offered as a complete packaged product. A few SaaS examples include Microsoft Office 365, Google Docs, Adobe Creative Cloud, and Salesforce.

In the computer networking industry, vendors that offer cloud-hosted network management, location services, data analytics, or other applications are effectively providing SaaS. SaaS offerings are built on top of IaaS and/or PaaS frameworks, often leveraging both in the final application. In fact, the robust PaaS toolkit is largely responsible for the explosion and ease of development of SaaS offerings. For many consumers and IT groups, it's much easier operationally and more cost-effective to buy a finished product instead of building it.

Cloud Deployment Models

Cloud still has a muddy meaning in a lot of contexts, primarily because it can be deployed in many different ways. In network diagrams, cloud icons are used to represent anything that is accessible via the Internet, even if it represents an organization's own datacenter. Nonetheless, cloud deployments are typically categorized into three groups known as public, private, and hybrid. There's also a fourth pseudocloud model, which is local cloud.

Public cloud

When people say "cloud," public cloud is what you should think of first. The public cloud is effectively a set of services built in a datacenter that is hosted and operated by someone else (for example, Amazon, Google, and Microsoft). The public cloud can utilize any of the service models discussed previously (IaaS, PaaS, or SaaS). But the key point is that the cloud is operating as a public service in a datacenter environment.

REMEMBER

As a broad statement, public cloud is open for use by the general public as a free or paid service. For example, Google's G Suite is a free public cloud suite of applications, and Microsoft's Office 365 is a paid public cloud suite of applications. Payment methods for public cloud services are usually subscription-based.

Private cloud

Private cloud is the most broad and multifaceted deployment model because it has many potential flavors. In simple terms, a private cloud is a dedicated set of computing resources used exclusively by one organization.

Private cloud is a preferred deployment approach for governments, financial industries, service providers, and large enterprises because it provides a higher level of control, security, and data privacy.

The following sections discuss some common variations of private cloud.

Third-party private cloud

This type of private cloud is hosted and operated for an organization by a third-party on dedicated resources that are not shared with other companies. A private cloud solution can be offered by a public cloud company, a SaaS provider, or any other hosting entities operating the service in a dedicated model.

Privately operated public cloud

This concept might seem counterintuitive, but it can also be labeled as a private cloud when an organization operates its workloads in a public cloud setting for the company's internal uses. For example, virtual machine images can be deployed in public cloud instances (for example, AWS, GCP, Azure) and then managed by the customer as a "private cloud" instance, instead of either traditional on-site models or public SaaS models.

End-to-end private cloud solutions

End-to-end private clouds are bundled solutions that integrate hardware (compute, memory, network, storage) and software with a management suite, which provides an easy path to private cloud. This model is essentially a prepackaged computing stack that is typically installed on the customer's premises.

Traditional on-premises datacenter

Some people use the phrase private cloud to refer to an organization operating its own datacenters for internal use. This could also be called local cloud. The term local cloud helps to distinguish private datacenters from cloud services provided and operated by a third-party, but the semantics and terminology debate rages on because it's difficult to define what exactly constitutes cloud as opposed to datacenter.

Hybrid cloud

The definition for hybrid cloud should be fairly straightforward: Hybrid cloud is a combined usage of both public and private clouds. Every application is a little bit different, so there is nearly infinite variety in hybrid cloud arrangements.

In the typical meaning, the hybrid cloud refers to an integrated computing and orchestration model in which some workloads remain intentionally private while others are offloaded to public clouds. For example, some businesses may have steady-state computing workloads that can be cost-effectively addressed with on-premises hardware. Then, unpredictable bursts of load beyond the on-premises capacity can be pushed to public cloud compute, where dynamic resource allocation is easier to manage. Similarly, privacy-sensitive data processing or storage may be performed with private resources, while less security-sensitive workloads reside in public cloud resources.

Occasionally, organizations will use the term hybrid cloud to refer to their migration period when they are moving from a private datacenter towards a public cloud. Some organizations use hybrid operation in order to diversify their technology investments and to keep a foot in both camps in case an operational pivot is necessary.

It's more difficult to build a hybrid model; nonetheless, a well-built hybrid solution can extract benefits of both computing models — the simplicity, scale, velocity, and OPEX of public cloud alongside the privacy, control, and long-term cost efficiencies of private cloud.

For an example hybrid cloud solution, check out AWS Outposts.

Local cloud

Local cloud is a reference to on-premises deployment models. In some ways, this could be branded as non-cloud, but there is plenty of debate and confusion about what exactly constitutes cloud, whether it be location, host, architecture, or combinations therein.

So, there is a lot of variety within cloud deployment models. The cloud is not just someone else's server. The cloud isn't always remote. The cloud is not just AWS, GCP, or Azure.

Edge compute

Although it's not a cloud deployment model, edge computing intersects with the public, private, hybrid, and local cloud discussion. Across industries, the cloud has tremendous benefits, but it is important to recognize that all cloud all the time isn't necessarily the right answer. Edge computing surged as a topic of interest within the networking community in part as a reaction to cloud adoption.

While some networking management services have shifted to cloud, networking is, by definition, a means of connectivity to devices and users. While cloud datacenters can be regionally adjacent to users, regional proximity may still be too far. Some drivers of edge computing are to move computing closer to the device or user to reduce latency, leverage existing points of presence and assets, and lower transport costs. Edge compute becomes a complementary location to cloud and core.

The edge of the network looks different to each network operator. For some, it may be LAN devices or the sites that host them. For others, like mobile, fixed, or cable operators, edge refers to cell towers, WAN points of presence, or data aggregation points. Or, edge could be alternate datacenter options in smaller nearby cities to deployed sites.

Applications also vary. Edge compute is often closely tied to the collection and preprocessing of data prior to shipment across the WAN into the cloud for further analytics or visualization. Edge compute is also used for real-time workflows like inline data security and device posture inspections or location-based analysis that may trigger alerts, advertisements, or health and safety responses. Edge compute can also bring machine learning models nearer to network traffic after the ML algorithm has built a model in the cloud. And finally, you might see VPN concentrators or other data aggregators billed as edge compute, simply for marketing mindshare of a trendy topic.

Cloud Geography Models

Despite the ethereal nature of the cloud ("computers in the sky"), it might be overly obvious to state that cloud resources operate in physical datacenters in real places. If you do a simple web search, you will find maps showing the hosting regions for each cloud provider. All the top-tier providers follow similar geographical design principles, which allows them to offer cloud resources globally with robust layers of redundancy built into each region. Although each provider may use slightly different terms, the concepts are generally the same.

As shown in Figure 2-2, cloud deployments can be broken into four tiers of resource:

- Global resources
- Regional resources
- Zonal resources
- Clustered resources

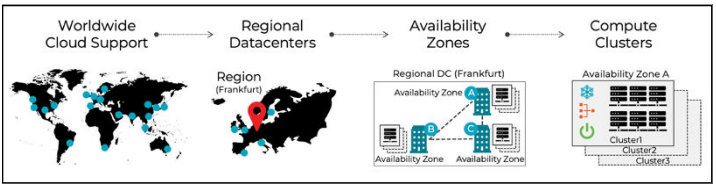


FIGURE 2-2: Four tiers of geographical cloud architecture.

Global cloud

By operating datacenters across the world, cloud providers offer the flexibility of choice to customers, which allows services to remain close to service consumers. This global coverage is important for a few key reasons:

- **Regulatory compliance:** Each region (for example, European Union) or country (for example, China) may have certifications and operational requirements for IT services, such as data encryption and storage. Each cloud geography can potentially have its own set of operating practices that are specific to the operating region.
- **Latency:** For many applications, the service delay from user to cloud is an integral part of quality of experience, which is easier to deliver with physical proximity and reduced network hops.
- **Georedundancy and backup:** Cloud regions are designed for high availability and resiliency to failure. But, some operators and mission-critical use cases expect not just intraregional resiliency/backup, but also interregional geographical resiliency. By operating and backing up an application and its data across several regions, service providers are insured against natural disasters or other outages or incidents that may affect an entire region.

Regions

Regions are sets of interconnected datacenters in a geographical area that are managed as an integrated group. Regions are typically labeled by the location of the datacenters, such as us-east-2, asia-southeast2, or EU-Central.

Cloud services are frequently provisioned and administered on a regional basis, and you will often see cloud region names as a part of the domain name in a URL when you browse cloud-hosted services.

Availability zones

Each region is composed of a group of separate facilities, which are called availability zones. Each availability zone is at least one physical facility with independent infrastructure for networking, power, cooling, and facility operations. These regional facilities are physically proximal to one another (for instance, within or near the same city) and are interconnected via low-latency, high-bandwidth links. There are usually at least three availability zones within a region. This type of datacenter separation and interconnection isolates the fault domains, which allows each region — and the services operating there — to have tremendous resiliency.

Clusters

Each availability zone is typically composed of many computing clusters, which are simply groups of computing resources, managed as a unit by the cloud operator. Typically, the cloud operator itself provides a level of abstraction for these computing clusters, which makes them transparent to customers. To make life simpler, the cloud operator ensures that each customer's workloads operate within the same computing cluster of an availability zone.

Global, regional, and zonal resources

As cloud architectures build on these geographical tiers, it is important to understand the resource allocations that may belong to each tier.

REMEMBER

In simple language, there are no global resources per se because computing is specific to a region. However, applications can be designed to failover and/or backup from one region to another in the case of outages. In this sense, the resources are still regional, but the application design takes into consideration a multiregion approach.

Most resources are configured either regionally or zonally. For example, because availability zones are network-connected datacenters, it is common for IP addressing to be considered a regional resource, which means IP addresses are shared across the entire region for a tenant. Other resources like virtual machines and

some types of storage may be zone-specific, which means you select which region (for example, us-east-1) and which zone within the region (for example, us-east-1-a) to deploy your VM.

Availability zones are one of the fundamental architectural building blocks of the modern cloud, which is why multizone application designs are common. Database instances often use interzone replication to stay in sync, while other application components use load balancing, elastic IP addresses, and domain names to provide fault-tolerant services across an availability zone. This method allows the entire service to remain available to users and devices, even if an entire availability zone of a region were to go offline.

You may not need in-depth knowledge of resource planning for cloud applications, but even as a potential user of a cloud service, it's useful to know how all the magical benefits of cloud resiliency and reliability are actually achieved.

IN THIS CHAPTER

- Examining the rise of cloud
- Looking at technology that supports cloud

Chapter 3
Microservices, Containers, and Orchestration

Software architectures are like building foundations. You never think twice about good ones, but you will notice (and pay for) bad ones. And, it is difficult to change the foundation once the structure is constructed. A general sentiment exists within the networking industry that software problems (unreliable code releases, buggy products, and slow velocity) are the fault of bad quality assurance (QA). However, the reality is that QA organizations face an impossible task because the product architecture itself leads to these problematic software artifacts.

Functional cloud architectures enable continuous integration (CI) and continuous deployment (CD) with automated quality assurance. Modern cloud architectures enable scalability, reliability, and velocity at the same time. Containers and microservices change the game for software foundations. I can keep talking benefits, but it might help to explain how the technologies work and why they drive benefits as well as cost savings.

A Story of Architectural Evolution

In the beginning, there were boxes. For the longest time, network management software and services were integrated with hardware servers and sold as appliances. But, boxes didn't scale well. Customers needed more or bigger appliances if they outgrew their hardware; they needed to install boxes for every site (and redundancy!); and the software running on the box was locked into a fixed form factor with a very hard resource ceiling.

REMEMBER

To deal with some of those challenges, virtual machines (VMs) became a popular alternative to hardware. By shifting to software, the scale ceiling was lifted, and new operational tools opened up with hypervisors. But, there were still hurdles, like inability to dynamically scale individual services, and the need to interwork with other IT organizations (like a VM infrastructure team) to plan for resources, changes, and perform troubleshooting.

But thankfully, by virtue of the software-first architecture, VMs were portable. As cloud and hosted datacenter costs dropped, many network operators saw advantages in offloading the VM infrastructure to cloud providers. This alleviated many operational pressures, but some of the core software design issues remained. For example, VMs were essentially a repackaging of the hardware appliances, so they had software scale issues, needed redundancy, and the VM itself remained a monolithic piece of software with inflexibilities.

As cloud development evolved, software architectures were built with considerably more flexibility and modularity, with loosely coupled services that are easier to monitor and scale independently. Together with this new distributed and scalable design, new software tools were invented to manage the deployment, monitoring, and dynamic change of the system. These concepts are known as containers, microservices, and orchestration. They are the software building blocks of the modern cloud.

Microservices

Microservices are a software design technique in which an application is broken down into small operating pieces with well-defined boundaries of functionality. The individual pieces (in

other words, services) are woven together via application programming interfaces (APIs) in a loosely coupled environment.

REMEMBER

I think of microservices a little bit like a car. The car is the application. But the car (application) is made up of component parts (services) that provide some function. Each part is optimized for its own specific purpose. Additionally, each part has a well-defined interface to the rest of the car, and in most cases, you can modify, upgrade, or replace parts in a modular fashion without tinkering with the rest of the vehicle. Each part represents an isolated fault domain so that component issues and failures have a limited impact on the rest of the system. Replace headlights or tires without thinking about engines or exhaust. Or enhance the audio without worrying about the alternator. Your car is analogous to a single unified application with many loosely coupled parts.

In software, a microservices design does the same. By decoupling services into small manageable parts (by the way, those parts are usually containers), each service can be designed with a software stack that is ideal for its function. As application requirements change over time, the necessary services can be modified as needed in a modular fashion. Authentication frameworks can be modified without breaking control plane functionality. Reporting mechanisms can be added without modifying events code. License enhancements can be added while leaving compute engines alone.

TECHNICAL STUFF

A microservices architecture seems pretty logical, so you might be wondering why it has not always been this way. One answer is that most traditional networking applications were built for on-premises fixed-footprint servers. So they are monolithic by design and bounded by scale constraints by necessity. They are built as a single-tier program (called monolithic applications). Multiple unrelated features were built into the program with tight coupling, often reusing underlying software components for footprint efficiencies. Management and control plane, admin role-based access control (RBAC) and security, events and logging, reports and user interface (UI) all become wed together in one complex piece of code. Over time, it becomes spaghetti, and every new addition or modification becomes increasingly complex to write and error-prone to test.

When you think of monolithic software, think of wired earbud headphones. When you lay them down on a table, they always get tangled up. When you put a pile of cords, cables, and earbuds

together, the tangle becomes such a mess that it's difficult to tease one part out from the other. This is what happens with tightly coupled monolithic architectures as they evolve over time. Another term for this is spaghetti code. And when you picture spaghetti, the reason becomes clear.

Figure 3-1 illustrates the difference between a monolithic application versus a microservices-based application. On the surface, a monolithic application can have roughly similar features as a microservices-based application, but the software architecture makes it difficult to maintain over time. And, it becomes impossible to scale features independently. Microservices, on the other hand, are segmented out into their own contained functional blocks with well-defined interfaces. Microservices have linkage within the application to other services where necessary. Just as importantly, microservices do not interface where it is not needed.

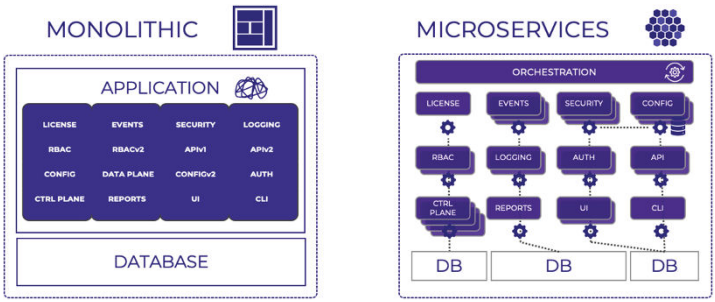


FIGURE 3-1: Monolithic versus Microservices.

Microservices have emerged as the de facto software architecture for cloud-based applications that are dynamic and complex, with many functions and moving parts — like scalable networking systems.

MICROSERVICES AND MONOLITHS

Microservices are often spoken of as a sort of binary classification of a software architecture and compared to monoliths. However, it is important to realize that the two terms function on a continuum. Microservices are a software organization technique, and there can be parts of an application that are broken down into isolated services, while others exist in larger functional blocks.

Containers

Containers are lightweight, stand-alone executable software packages that include everything needed to run, including code, system tools, binaries, libraries, frameworks, and settings. Docker is the open-source standard platform for running containers.

REMEMBER

If you are not a developer, that definition may not make sense to you, but stick with me. Containers are simply a way to bundle together some code and all of its operating dependencies so that the entire package is all-inclusive. By packaging everything as a bundle, the software components become self-contained. Therefore, the software package can function reliably in any container environment, which makes it more predictable and easier to maintain and operate.

The problems that containers solve are not very obvious when you think about version 1.0 of an application. The initial release has clean software and works just fine. However, new features added over time, together with existing modified features, will share underlying code and components. The environment keeps evolving to fit new unique requirements while maintaining backward compatibility. However, the interoperation matrix grows exponentially with added features, underlying software components and versioning, and dependencies. For example, Feature-A needs Component-Y to be on Version-N; and Feature-B needs Component-Y to be on Version-Q, and so on. The end result is that quality assurance (QA) teams have issues keeping up with all the changes and dependencies. Containers, as a part of a microservices architecture, solve this complexity problem for both developers and QA.

TECHNICAL STUFF

Containers are commonly compared to virtual machines because their function is roughly similar — to run multiple isolated workloads on shared hardware resources. You can think of a container a little bit like a lightweight virtual machine (VM). As illustrated in Figure 3-2, virtual machines deliver workload sharing by virtualizing hardware (via the hypervisor) and then running multiple guest operating systems (OS) on top. Container systems "virtualize" the host operating system (OS) itself so that workloads can operate within containers without requiring a full guest OS. In other words, containers provide a more direct route to the

processes and subsystems of the host, without requiring all the overhead of another OS. In that sense, containers "dock" to the host through the container engine.

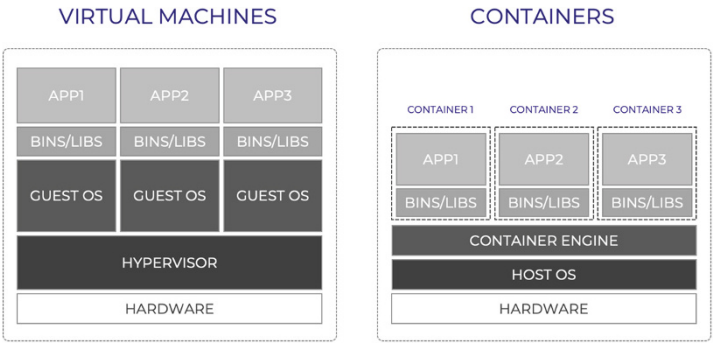


FIGURE 3-2: Comparison of virtual machines versus containers.

Compared to other software design options, containers provide several benefits:

- **Lightweight:** Containers reduce the resource footprint by running directly on the host's kernel without requiring a guest OS, which takes up CPU, memory, and disk.
- **Predictable:** By packaging an application with all of its dependencies, containers are predictable and portable because they work the same way in any container environment.
- **Fast and scalable:** In a production application with dynamic real-time requirements, containers start up very quickly and do not rely on an underlying VM to boot. Once started, they can also be scaled up or down independently from other services in other containers.

Orchestration

A microservices architecture often comprises many containerized services to deliver the application. It's not uncommon to have over 30 different services, which could each be a single container,

a collection of containers, or a distributed set of containers. To control the dynamic deployment, operation, and scaling of those containers, these architectures rely on an orchestration layer.

Kubernetes is the best-known open-source system for container orchestration, but other cloud options, hosted Kubernetes services, and tools built on top of Kubernetes also exist. This layer provides the automation and operational oversight that cloud providers and customers both want from the solution.

Orchestration solves the following operational challenges:

- Deploy containers and containerized applications
- Monitor services for operation and automate actions (for example, restart a failed container, delete containers, replicate containers on new hosts)
- Scale container deployments as application load changes
- Manage container updates

BRIDGING CLOUDS WITH KUBERNETES

Orchestration with Kubernetes is helpful for any type of deployment model where centralized orchestration of containers will benefit the operating team, whether that deployment be public, private, local, or hybrid clouds.

Hybrid deployments are made possible with a Kubernetes service (see Chapter 2 for more on hybrid cloud). As an orchestration layer, Kubernetes is made aware of all the hardware resources available for the application, and the use of those resources can be centrally controlled and automated with this orchestration layer — whether the resources are hosted in the cloud or on-premises. In short, Kubernetes becomes the engine that optimizes the workload sharing across on-premises and cloud resource pools, based on the unique requirements of the application and the business — such as security, load bursts, and cost optimization.

TECHNICAL STUFF

These containerized services can then scale vertically or horizontally by adding clustered container nodes, which are load balanced for maximum effectiveness. In this way, the orchestration layer functions like a conductor in an orchestra. The orchestration layer knows about all the services and their operating requirements, it performs monitoring while those services are running, and it adjusts the service composition dynamically to ensure that the end-to-end service architecture of the application is performing as desired.

Microservices with Containers and Orchestration

When you put all of these architecture concepts together, you end up with an application composed of many loosely coupled services that interact via well-defined APIs, operating as agile containers, orchestrated with an intelligent engine on top. The benefits are tremendous. Figure 3-3 depicts these technologies in a conceptual cloud architecture in which the orchestration layer manages the build-out of services via containers, and the application also incorporates multiple databases of different types as well as serverless compute functions.

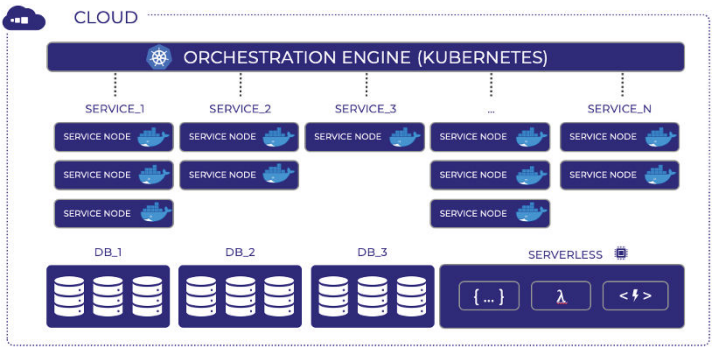


FIGURE 3-3: Containers, microservices, and orchestration in a conceptual architecture.

Scalability

By decoupling services into containers, you gain a massive benefit in scalability. As a result, you can support large-scale deployments,

while dynamically adjusting the application to right-size services based on unique customer deployments, network changes, and data usage over time. Management exists at the orchestration layer but is made possible by the service composition. For example, if control plane functionality (terminating AP/switch connections, decrypting tunnels, and packaging data into a queue/bus) is delivered as a dedicated service within the application, you can scale the control plane functionality up and down based on the number of unique nodes needing control services. Meanwhile, if the API (or any other unrelated service) is very lightly loaded, you can keep a lower resource API service, which scales up if you see demand grow. All of this can be managed automatically with monitoring as part of an orchestration flow.

Resilience

Microservices also provide an innate level of reliability because each service is maintained independently. For example, perhaps there is a problem with a node providing the events service and it needs to restart for some reason. You can restart the problematic events container/service individually without affecting the other nodes or services (authentication, admin login, licensing, control plane, API, reports, and so on). Compare that to a heavily integrated monolithic hardware or VM appliance and it likely does not work that way. You normally will have to reboot the entire platform manually to fix one service.

Modularity

Again, the microservices architecture often leverages containers, which means the service software is very modular. Each service can use a software stack that is specific and optimized for the requirements of that unique service. There is no issue if a software stack is different from the rest of the services in the application. If the software stack needs to change, then you do not need to overhaul the entire application and rewrite a bunch of features to upgrade or replace a software component.

API-driven

APIs are often discussed from a management automation perspective. In other words, public APIs can be used to build automations into your management flow. But in this case, I'm talking about the internal workings of an application with private APIs. Services

rely on APIs to integrate together. Conveniently, microservices allow each service's API to be well-defined and specific to its function. This creates integration within the application only where necessary, and avoids the spaghetti effect.

Generations of Cloud

At the beginning of this chapter, I talk about the evolution of cloud from on-premises hardware to virtual machines and eventually into the cloud. Figure 3-4 illustrates the ongoing progression from cloud toward newer generations of architecture. Many cloud systems in the networking industry have evolved (or are evolving) along this progression, where more advanced cloud systems will be leveraging all the benefits of cloud with new technologies.

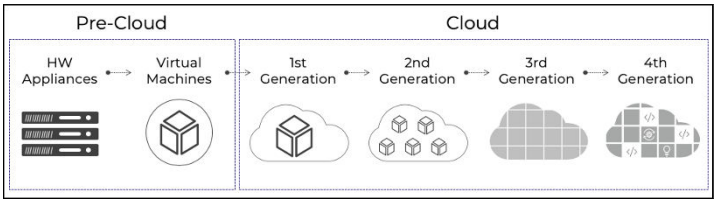


FIGURE 3-4: Generations of the cloud.

The cloud generations can be summarized as follows:

- **1st Generation:** Virtual machines (VMs) are installed and operated from a public cloud using Infrastructure as a Service (IaaS). Virtual machines are managed in a more manual way by a DevOps team in a single-tenant environment with some amount of continuous integration.
- **2nd Generation:** The services offered by monolithic virtual machines are deconstructed into groups of services (as a Service Oriented Architecture or SOA) that remain as independent VMs. The DevOps team operates the VMs with orchestration tooling, supports multitenancy, and utilizes continuous delivery of new software.

- **3rd Generation:** The SOA is replaced with a loosely coupled microservices architecture, either in VMs or containers. Services scale with horizontal clustering and orchestration oversight. Data pipelines support more advanced analytics, and the DevOps team is leveraging continuous deployment and continuous operation.
- **4th Generation:** The architecture is completely containerized microservices, and some microservices may even be replaced by serverless computing. The data stack has evolved to include machine learning and AI, and the application is cloud-agnostic so it can run in any environment. Finally, DevOps leverages a dynamic resource pool and provides a no-downtime operating environment.

IN THIS CHAPTER

- Examining the big data ecosystem and its components
- Looking at an end-to-end data pipeline and data stack
- Understanding data science, machine learning, and artificial intelligence

Chapter 4
Cloud Data Architecture

In our hyperconnected world, every action, transaction, and interaction becomes data. Businesses strive for deeper technology integration and workflows, so data often becomes the differentiating asset in competition. For that reason, all things data-related are relevant to modern cloud services, architectures, and applications.

Most big data discussions gravitate towards machine learning and artificial intelligence, but that treatment bypasses a wealth of important topics that make up a data ecosystem. So, I start this chapter by looking at the evolution that brought us to existing data architectures. Then I dive into the concepts of an end-to-end data pipeline, and finally circle back to machine learning (ML), and the ever-controversial artificial intelligence (AI). Be advised that this chapter is heavier on jargon, but don't panic. I tackle definitions and terms as I go.

The Convergence of an Ecosystem

Big data, cloud, and mobile computing exploded in the same decade because they're interdependent technologies. Mobile devices create the data, big data systems harness the data, and cloud applications expose data into meaningful services for which people will open their wallets.

In this section, I walk through these three ecosystem pieces, which are summarized in Figure 4-1.

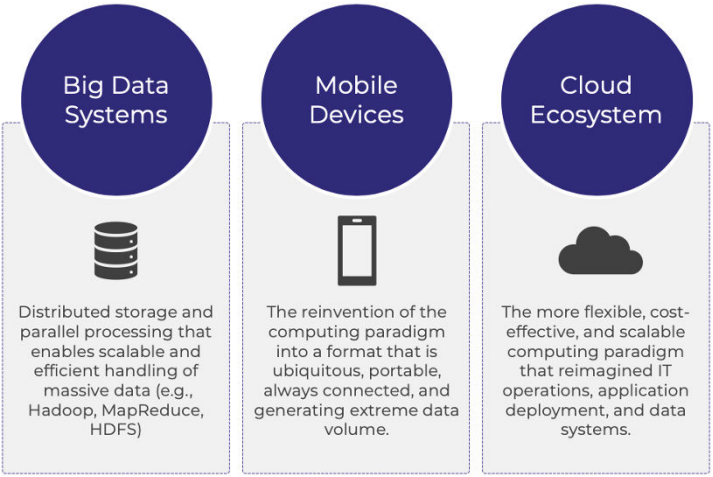


FIGURE 4-1: Converged data ecosystems.

Big data systems

The first piece of the ecosystem is the big data systems that allow data storage and processing at scale. This comes down to a number of tools that solve for the three V's of big data:

- **Volume:** Previous database technologies couldn't handle today's ridiculous data volumes. They were too expensive to scale vertically with high-end hardware and weren't designed to scale horizontally across commercial off-the-shelf (COTS) platforms. The Apache Hadoop ecosystem (Hadoop Distributed File System [HDFS] and MapReduce) effectively solved this core volume issue by distributing the data and the processing algorithms.
- **Velocity:** A new system was also needed to handle the constant flow of data telemetry used for real-time analytics applications. Platforms like Apache Kafka, Apache Spark, and Apache Storm address the velocity challenge with scalable message brokering as well as stream and microbatch processing.

- **Variety:** Finally, we're now dealing with very diverse types of data, from text, voice, and video to time-series metadata, events, and health telemetry. Each data type introduces new processing and storage challenges. So variety is handled by new data processing formats and a variety of database designs.

I get into the details of these solutions throughout this chapter. You may also notice that the Apache open-source community is a core enabler of these big data systems (kudos to the open-source Apache projects!!).

The rise of mobile

The second piece, and probably most revolutionary, is the rise of all things mobile. Mobile devices continuously multiply data because they are constant digital companions that capture our digital and physical activities. Everything captured by a mobile device becomes data. Mobile devices certainly have impressive computational capabilities today, but rely almost entirely on cloud-driven applications, credentials, sharing services, content, and backups. Furthermore, mobile devices often drive value throughout lines of business, which are hungry to harvest data in order to understand users and behavior to maximize revenue.

The cloud ecosystem

The third piece is the entire cloud ecosystem, which has evolved in a way that makes data wrangling infinitely easier, more scalable, and cost-effective:

- **Cost:** As it relates to storage cost, the daily cost per GB has been driven down by economies of scale.
- **Operations and scalability:** Cloud also alleviated the operational aspect of big data management, which, despite popular opinion, is difficult. Most modern big data tools were designed by very large technology firms to solve their own unique challenges. These tools were meant for DevOps teams thinking about massive datacenters and data volume that makes petabytes look small.

» **Flexibility:** Cloud also enables data pipelines (how data flows through an application) and software stacks to change with tremendous flexibility and agility. The cloud offers a more modular approach to application development and hosted services. Use cases continuously evolve and the stack must evolve with it. The cloud provides the most rich and robust toolset to solve data needs.

Understanding the Data Pipeline

A data pipeline is effectively the architectural mechanism for collecting, transporting, processing, transforming, storing, retrieving, and presenting data. These stages are functional steps in achieving the end goal of a data application. To deliver a data pipeline, engineers often use the term stack in order to define the software, systems, and tools that are used.


TIP

The data pipeline is not the same sexy marketing stuff as data science, ML, and AI (all covered later in this chapter). Nonetheless, all data science is part of a complete data ecosystem. That's why it's important to understand how the entire data framework works.

Figure 4-2 illustrates a generic version of a data pipeline, from collection through to visualization.

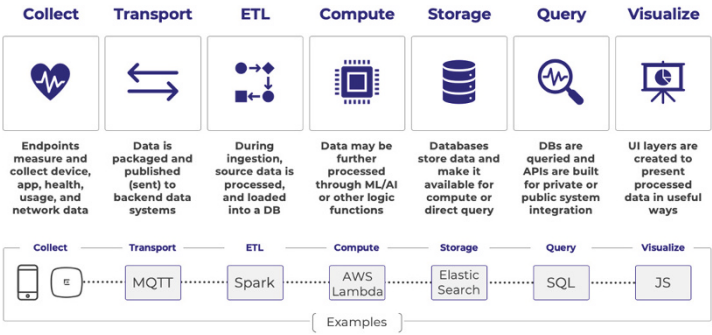


FIGURE 4-2: The data pipeline.

Collect

The data collection stage encompasses how data is captured at the beginning of an application data flow. In networking, data can be collected by network infrastructure (access points, switches, gateways), client devices or apps, collection engines, or dedicated capture tools like sniffers.

REMEMBER

The end goal of a data pipeline varies and could include monitoring and alarms, reporting, visualization, troubleshooting, automation, and others. The data use case dictates the collector's model — type of data, frequency of collection, preprocessing requirements, and more.

But, collectors may be limited by their role in the network, such as processing power, visibility to data, storage capacity, and telemetry bandwidth.

TECHNICAL STUFF

Telemetry is just a fancy word for a regular stream of data.

Collection happens over time, and there are three terms that describe the way that collectors process data over time:

- **Collection interval:** The rate at which data is sampled and collected for processing
- **Measurement interval:** The rate at which collected data is measured or processed and packaged into summarized data
- **Reporting interval:** The rate at which the summarized (or raw) data is communicated to other systems

Transport

In a cloud pipeline, after data is collected, it must then be packaged up and transported to the backend infrastructure for ingestion. Transport should use network bandwidth efficiently, handle large data volumes, and do so with security and reliability.

Transport could be broken down into a few sub-stages:

- **Data serialization:** Data serialization is a process used to take data from one system and package it into a data format that other systems can use. After collecting and measuring

data, the collection endpoints repackage the data and send it along for further processing. Modern data processing systems are increasingly moving away from languages that are difficult for humans to read, like extensible markup language (XML), and moving toward other data serialization alternatives like JSON (JavaScript Object Notation) and YAML (YAML Ain't Markup Language). Other options like Google protocol buffers (GPB) and Avro were designed to improve packaging efficiency.

- **Tunneling and encryption:** For network endpoints to send their serialized data to other systems, they need a connection, tunnel, or remote procedure call. This is known as data in motion. Tunneling takes place over a variety of protocols with different session and messaging requirements as well as security standards, including Secure Shell (SSH), Hypertext Transfer Protocol Secure (HTTPS), and Internet Protocol Security (IPsec), but there are a number of others, like Control And Provisioning of Wireless Access Points (CAPWAP), Generic Routing Encapsulation (GRE), Simple Object Access Protocol (SOAP), Remote Procedure Call (RPC/gRPC), Websockets, and more, each with its own advantages and disadvantages in terms of supportability, flexibility, and security.
- **Message transport:** Message transport (also known as a message queue or message bus) is the technology that controls the flow of data from producers (or collectors) to consumers (processing endpoints) throughout a system. A message bus is necessary for a few reasons. First, the producer and consumer may not process data in the same way or at the same rate. For example, producers may provide data in 1-second intervals, while consumers may process the data in 10-second batches, so some system needs to keep the data during that window, and solve for reliability. Second, the producer-to-consumer relationship may not be one-to-one. It is common to have thousands of distributed producers sending data to just a few consumers. And finally, consumers may not be interested in the data from all producers. The message queue brokers this process so that producers can publish and consumers can subscribe to specific data (often organized by topics) of interest.

SOME TRANSPORT DEFINITIONS

- **Producer:** The entity that generates data and sends data into the message queue
- **Consumer:** The entity that receives and uses the data from the message queue
- **Publish/Subscribe (Pub/Sub):** A data queuing model in which producers send data to a message queue (publish), and consumers receive data from a message queue (subscribe), usually by topic
- **Ingestion:** The process of gathering and importing data for storage in a database

The message bus functions much like a circulatory system in the body, providing a data exchange from one system to another. A few common message transport protocols and tools are Message Queuing Telemetry Transport (MQTT), Apache Kafka, Advanced Message Queuing Protocol (AMQP), RabbitMQ, and Eclipse Mosquitto.

ETL

Extract, transform, and load (ETL) is a process in which a system retrieves data from a database or queue (extract), manipulates it into a different format (transform), and then installs it into another database (load). Or more simply, ETL is a step in the data pipeline that receives data as input, performs some function on the data, and then sends data as output. ETL is the conversion process that readies data for use. This includes data preparation, importing, cleaning, normalizing, transforming, aggregating, and other data functions that many of us have heard but never quite understood.

Here are some ETL examples to make it more concrete. In some architectures, data may be initially stored as flat files in a generic storage volume. The ETL process may load those flat files, convert them to a format more efficient for query, and then index them. ETL may also be used for joining together data from multiple sources (for example, tables from different databases) and storing this joined data in a different database to accomplish

some task. Finally, ETL could also be used to take streaming data and populate a database with various aggregations of the stream across different time intervals, like 10-second, 60-second, and 10-minute summaries.

REMEMBER

These are generic examples, but the general idea is that ETL massages the data into a different format to make it more ready for storage or for further compute, like machine learning (ML).

There are two general types of data processing in a pipeline:

- **Batch:** Batch processing is a more traditional approach in which data processing happens over blocks of data that have accumulated over time. For example, the ETL process might run on a schedule every 30 minutes to process the last half hour of data. Or perhaps the system runs nightly jobs to backup an entire day of data activity. Apache Hadoop and Apache Spark are common tools used today for batch processing.
- **Stream:** Conversely, stream processing is for analytics applications where data needs to be processed in real-time, which can be useful for applications that need instant detection of problem conditions. Data can also be processed near real-time, which is often called microbatch. Stream processing innovations have also enabled us to deal with the velocity of data coming into systems from many different sources, such as in large-scale Wi-Fi deployments. Apache Flink, Apache Storm, and Apache Spark are popular tools for stream or micro-batch processing.

Compute

Compute and ETL are similar stages in a data pipeline, but I separated them in this discussion for clarity. ETL is typically concerned with massaging data so that it can be readily consumed by other processes. The compute stage is more focused on model training, which is common in ML applications. In order to train a model, the application will pull data from some source (for example, a message bus or a database), possibly perform some lightweight ETL operations on the data, and then pass it to a compute algorithm (statistical, ML, deep learning, and so on). The term data model refers to the artifact that is produced by an algorithm as it learns patterns that describe the data.

There are entire toolsets dedicated to this ML process, and I get into this later in this chapter.

Storage

Data applications wrestle with competing data demands relating to storage cost versus data volume, which requires an evaluation across three dimensions:

- **Types of collected data:** What data sources and metrics are needed?
- **Granularity of collected data:** How often does data need to be sampled/collected, and in what level of detail?
- **Duration of collected data:** For what time length and granularity does data need to be kept?

Because of decreasing costs and evolutions in tooling, this discussion now biases heavily towards more data. The Hadoop framework, with HDFS and MapReduce, was a core enabler of these growing data volumes. HDFS helps solve the storage equation by distributing the file system across a cluster of commodity servers. It solves for reliability and fault-tolerance by storing multiple copies of each data block. But the challenge with distributed storage is pulling data across networks to a centralized processing algorithm. So, MapReduce helps solve this problem by distributing the algorithms across worker nodes where the data exists, so the processing can be parallelized. Hadoop is one way to solve volume problems, but there are other ways as well.

The next important point with storage is that data comes in many formats, which is why there are many types of databases designed to handle different data formats and use cases. Therefore, you'll see that data-driven networking applications often support three or more different databases.

Relational versus nonrelational

Databases are typically classified into two categories: relational databases and nonrelational databases. Relational databases use Structured Query Language (SQL) and nonrelational databases are often referenced as NoSQL.

Relational databases use a familiar table format with columns and rows, where data is highly structured with strict formatting. They're useful for data that follows a well-known and consistent format, such as for inventory, configurations, account management, and fixed settings in systems. MySQL, Oracle, MS SQL Server, MariaDB, and PostgreSQL are popular relational databases.

Conversely nonrelational databases are dynamic, less structured, and often more document-oriented, which gives them flexibility to evolve and scale over time. Their flexibility is well suited for bulk data that may require more processing, so they are inherently well suited to cloud systems and distributed processing. NoSQL effectively means "anything that is not SQL" (or "not only SQL"). There are many variations of NoSQL databases, including key-value stores, document stores, graph databases, search engines, and even time-series databases can fit here as well. Some popular examples are MongoDB, CouchDB, Cassandra, Elastic Search, and Druid.

Types of data store

In the big data hype, you'll also hear about different conceptual ways to store data, such as data lakes, warehouses, marts, and streams. The terms and analogies hint at the meaning:

- **Data lakes** are databases storing large volumes of raw unstructured data that is not ready for query, which is usually lower cost than processed data ready for query.
- **Data warehouses** are similar to data lakes, but they are data volumes that have been processed and indexed to make them ready for query.
- **Data marts** refer to data warehouses that contain specific subsets of data that may only apply to a specific interest of a data-driven business.
- **Data streams** (or rivers) relate to data that is sent in the form of ongoing telemetry, and it implies an intent to process data in a streaming flow. A data stream is used for real-time analytics or events, instead of storing large data in bulk for future processing/analytics.

Query and APIs

Each database type has a specific query format to extract data. But, applications are often defined with backend logic that facilitates query via APIs. An application programming interface (API) is an interface for interaction between software endpoints.

There are two general types of APIs: public and private. If you're not a software engineer, most of your focus is on public APIs. The internal services that make up an application are often communicating and interacting with one another via private APIs. Conversely, public APIs expose some subset of an application's functionality for integration and usage by public, or external entities. In this case, APIs are the mechanism by which network engineers, admins, integrators, and operators build programmable integrations or automations between systems.

The most popular API architecture today is Representational State Transfer (REST), which is used for interactions between modern web services (and very cloud-friendly). REST is a way for applications to expose service endpoints (or resources) as Uniform Resource Identifiers (URIs) (for example, https://x.y.z./API/service_endpoint1). These URIs (URIs are similar to URLs) can then be called or requested by systems using HTTPS.

In other words, REST APIs are services (or features) of an application that can be exposed as a URI and then requested by peer systems with common HTTP operations like GET, POST, DELETE. When an API endpoint is called, the service responds with an HTTP response, which includes some information, which could be formatted as HTML, plain-text, JSON, XML. APIs can be used for many different features, such as onboarding new devices into a management system, modifying configurations, checking status, polling inventory, or requesting data.

GraphQL is another popular API language that was developed more recently, and is seeing strong adoption. GraphQL is commonly compared to REST, but is more focused on streamlining the query and efficiency of data transfer as a part of API calls.

WEBHOOKS

Webhooks are similar to APIs, but they flip the model on its head. REST APIs are service endpoints on the server that can be requested by a client (in other words, they are a pull model), while webhooks are HTTP POST actions performed by the server based on a trigger (in other words, a push model). Webhooks allow server-side systems to automatically send updates to client endpoints based on event triggers. One common networking use case for webhooks is to automatically create service tickets based on fault conditions, and then update those tickets as conditions change — where the network management software is the webhooks server and the ticketing system is the webhooks client.

Visualize

The visualization stage of a data pipeline is commonly the only data stage that the application user ever experiences. In some data pipelines, there is no visualization at all, and the output may simply be API calls to other systems, automations, email messages, or text alerts.

But by and large, the user interface (UI) is one of the most important parts of a cloud data system because it is the translation of data value to the user. User interfaces fit into a few general categories:

- **Internal/Development UIs** are usually very quick and functional UIs that are focused on events or data monitoring by internal teams (for example, DevOps), with less focus on beautiful layouts and perfect styling.
- **Generic Dashboards,** such as Tableau, PowerBI, or Looker, are a type of query and reporting tool that can integrate with databases and provide flexible access to data. The stock UI then provides tools to run queries, filter data, or slice-and-dice the data in different manual ways to create custom visualizations. These tools solve some data requirements but may be insufficient for many use cases because the tool itself is designed to be generic.

» **Custom User-Facing UIs** are the most common, and are completely customized visualizations that attempt to solve the customer use cases of a product. They are unique to each product and take the most time to develop and maintain. Network management UIs fit into this category.

Data Science and ML

Data science is the technology discipline that combines the disciplines of math and statistics with computer science, with the goal of harvesting value from data. As shown in Figure 4-3, data science is a broad field that also encapsulates the subfields of ML and deep learning. Deep learning is often used today for some of the most advanced ML use cases that best approximate human intelligence. As a result, deep learning is often conflated with AI, which is when computer systems or algorithms mimic humanlike intelligence.

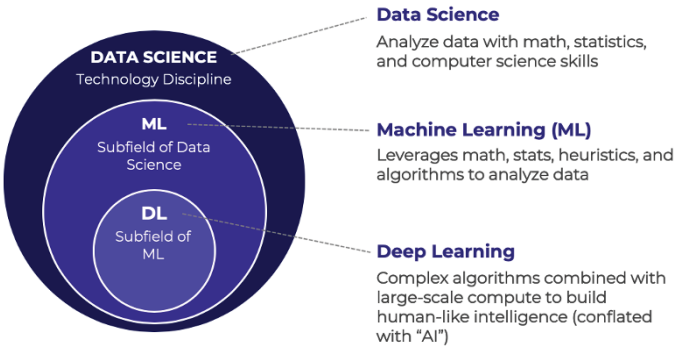


FIGURE 4-3: Data Science, ML, and deep learning.

As a subfield of data science, ML is focused on algorithms by which computers learn about data and data patterns without being explicitly programmed with strict rules. ML algorithms use training data to build mathematical models that describe the data; the models are then applied to working data to solve some problems, like classification, prediction, or pattern mining.

The fourth piece of the modern data ecosystem is ML. Before the mass availability of the open-source and cloud ML toolkit in the 2010s, ML was largely an academic pursuit except for only a

handful of very big companies — even then, it was a very limited pursuit. Today, the toolkit makes it possible for even nonexperts to access the richness of ML processing.

Some software tools that opened the doors (and are still opening them) are TensorFlow, PyTorch, Scikit-learn, Keras, Apache Spark, and Jupyter Notebook, in addition to the complete ML development engines and services from the leading cloud providers: Amazon Sagemaker, Google's AutoML or AI Platform, Azure ML, and others.

Data modeling

A good starting point to understand how ML works is to understand data modeling. A data model is an abstract way to represent data, its structure, and relationships to real-world things that it represents. That might sound confusing, but it makes more sense as you look at three types of data models (surprise, the final one is ML!):

- **Rule-based models** are simple ways to represent data using rules. For example, when the outside temperature is over 70°F (21°C), it is considered to be warm. Or, if a network switch's CPU usage is over 95 percent, the system is considered to be stressed. Rule-based models are simple to implement and easy to understand, but they're not sophisticated enough to capture more nuanced representations of data. And they are typically domain-centric.
- **Statistical models** are a next step in describing data using math and statistics. In the same example as before, people might say that it is warm if today's temperature is 20° higher than the 50-year daily average for a given calendar day. In that sense, even a winter day could be warm if the model uses statistics to build its classifications. Because statistics are typically descriptive of the actual data, they rely less on domain-specific knowledge. But, even statistics are relatively simple and can only represent certain aspects of data.
- **ML** is a way to model data using a combination of statistics as well as algorithms and heuristics. ML is similar to statistical models in some ways; however, ML captures more complex relationships in the data by allowing the algorithm itself to adapt. As a result, ML has become useful because it can model complex data relationships using mathematical models that are too difficult to solve or describe simply by rules or simple statistics.

REMEMBER

To understand the benefits of ML models, think about the description of a cat. The human brain can recognize a picture of a cat as something distinct from a picture of a dog (unless it's a wimpy dog), but the number of strict rules necessary to define the cat/dog distinction for a computer are very complex to write. With statistics, it's difficult to accomplish the kind of accuracy necessary for solving real problems — think about statistical descriptions of a cat, like approximate size and contour of the body, legs, face, and tail; fur length, color, and texture; ear and eye size and shape, and so on. It all gets complex quickly, and there are a lot of exceptions and special cases where the rules don't apply. ML helps us solve these problems by learning a model that captures this complex logic. Using ML, a computer learns to recognize a cat.

COMMON MISCONCEPTIONS ABOUT ML

The most common misconception about ML and AI is the terrifying concept of sentient robots and the all-knowing government spy systems. The original term AI referred to this concept of an all-intelligent computer that thinks and reasons as well or better than humans in all aspects of intelligence. This AI definition is known as artificial general intelligence, or strong AI. Contrast this with modern-day AI, which is classified as narrow AI. Narrow AI is when computers reflect very specific aspects of human-like intelligence — interpreting the meaning of human speech, playing board games, finding patterns in text, recognizing objects in images, or driving a car. No doubt, it is impressive in many respects, but also very narrow. Despite the impression we get from movies, modern AI is nowhere near human-like intelligence, other than for tasks where the rules are somewhat fixed and easy to describe.

The second misconception is that ML is a magic wand. Throw the data into the ML algorithm, and the output results are perfect intelligent insights, not to be doubted. ML remains driven by math, computer science, domain knowledge, and a certain kind of alchemist's art. There is a lot of imperfect ML, or mathematically accurate ML models that are biased, poorly trained, misleading, and practically not very useful. The conventional wisdom is that within data science efforts, your outputs are only as good as your inputs; therefore, if dirty data

(continued)

(continued)

(inaccurate, incomplete, estimated, buggy, and so on) is used for training or modeling, the results may be questionable. Keep in mind that all data is incomplete or imperfect in some way, and ML outputs come with levels of confidence. Some "insights" are high-confidence guesses, and some "insights" might be low-confidence guesses too.

The third misconception, often due to overzealous marketing, is that any modern data presentation is naturally based on ML. It is quite possible to have an elegant data visualization that solves complex problems without using a drop of ML/AI — and that is perfectly okay.

Finally, ML/AI is not necessary for every data problem. Many problems can (and should!) be solved with simpler versions of data analysis and processing. ML always gives an approximate answer to a question, so if there is a simpler and less costly solution, use it. Some data questions are simply not that important, so more naïve and low-cost solutions are perfectly suitable to the task. Non-ML solutions can also be faster to build, easier to deploy, manage and understand, and ultimately more cost-effective. Some problems certainly benefit from additional intelligence, but many do not.

Types of ML

Within ML, there are many different types of algorithms that can be used to model and extract information from data. As shown in Figure 4-4, ML is generally classified into three types.

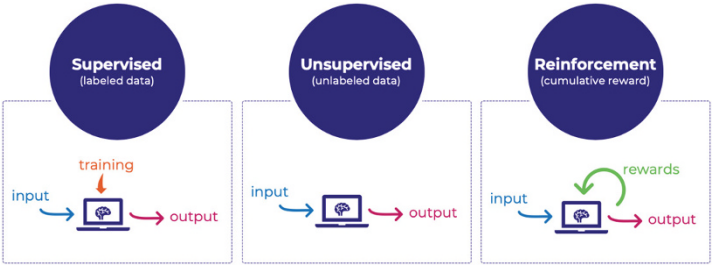


FIGURE 4-4: Types of ML.

Supervised ML

Supervised learning is a type of ML in which the data model is learned with explicit labeled data. In other words, the learning is done with direct teaching by a training dataset. A straightforward example is image classification. By feeding a supervised learning engine a series of pictures with labels, it can learn the association between a label and picture contents. For example, if you input many pictures of dogs, cows, and ducks (with labels), after sufficient training, the model will be able to correctly classify a picture for which there is no label.


TIP

Supervised learning is excellent for problems like classification (identify the category/label of an unknown) and regression (predict a value).

Unsupervised ML

Unsupervised learning is the creation of an ML model in which the inputs are not labeled nor explicitly trained. The system learns about the data on its own, and is tasked with grouping the otherwise unsorted data according to similarities/differences and patterns.

For example, when you shop online, unsupervised learning can identify patterns in your behavior to learn which items should be recommended based on search terms, page views, and history. Similarly, unsupervised learning is commonly used for television or movie recommendations on media platforms. Unsupervised learning allows for intelligent recommendations based on unlabeled inputs. At first glance, it might seem like your watch history would be considered labels, but it is actually the relationships that are being learned, and those relationships are unlabeled. No one explicitly teaches the system that if users like *My Little Pony,* they will also like *Dora the Explorer.* However, an unsupervised ML model can learn this type of relationship.


TIP

Unsupervised learning is used for clustering, association, and dimensionality reduction. Clustering discovers groups within data; association describes relationships in data; and dimensionality reduction simplifies the number of variables to find the most important variables.

Reinforcement learning

Reinforcement learning is a slightly different type of learning by which the system uses a feedback system to progressively train a model by cumulative reward. In simple terms, the learning system is designed to work towards a goal and is told "bad" and "good" along the way. The model provides an output, the output is evaluated, and a reward or punishment is supplied back to the system indicating whether the output moves the system closer to the desired result. After enough feedback, the system builds a model that gravitates to the outputs that were positively rewarded, and thus, move progressively closer to the goal of the system.

TIP

Reinforcement learning is effectively a way for systems to learn by experience, which makes it excellent for ML applications related to progressive skill acquisition. Some applications are self-driving cars, gameplay, or robotic automation that can be learned through a series of trial and error until a skilled model is built.

Deep learning

I know I said there are three types of ML; however, I like to treat deep learning as a special subfield of more advanced learning architectures that could be supervised, unsupervised, or semisupervised. In semisupervised models, some data is labeled while some is not.

REMEMBER

Deep learning uses artificial neural networks, which are data processing models based loosely on the functionality of nerve cells (neurons) in human biology. In neural networks, a deep series (thousands or millions) of processing nodes are interconnected in layers (the "deep" part refers to the number of layers of "neurons"). Each node is connected to previous and subsequent layers, which is illustrated in Figure 4-5.

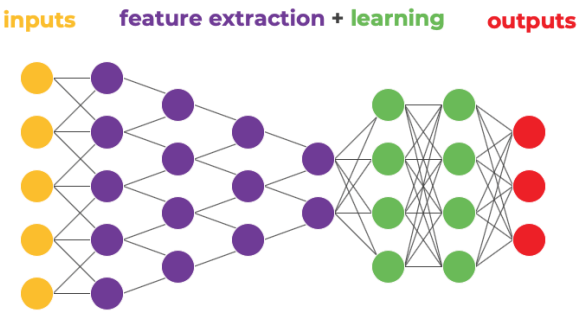


FIGURE 4-5: Deep learning node layers.

TECHNICAL STUFF

Data is fed into the first layer as input, and then moves through the layers in steps, where each step provides some additional processing of the data. The parameters in the learning layers (weights, summation and bias, activation functions, thresholds, and so on) are randomly tuned and tweaked until the model reaches the goal.

That explanation may be confusing, so think of it like this: The system starts in a random state, and the system is also given a measurable target to work towards. Over time, the algorithm makes small random adjustments in each of the layers, and measures the impact of those adjustments, only accepting the adjustments that move the system as a whole closer to the target.

Deep learning has already proven very useful for building and expanding some of today's best AI, including applications in computer vision (image recognition) and natural language processing (speech recognition and translation). Deep learning is a high performance type of ML, which makes it well suited to very complex problems involving huge datasets (where other ML might be better for simpler tasks and datasets).

WARNING

Despite its strengths, deep learning is also more compute-intensive (and thus, more costly) than simpler types of ML. It has another strong drawback, which is the lack of transparency. It truly operates like a black box, which makes it impossible for data scientists to explain why the system's layers of nodes and millions of random guesses eventually start accurately identifying cat pictures (or not identifying them). The "why" behind its outputs may be impossible to explain or justify. This behavior may be fine if everything is working perfectly, but it makes for difficult troubleshooting when it doesn't work perfectly. This black box issue also raises ethical concerns when deep learning is used to influence decisions, like college admissions, hiring candidate selection, insurance or finance approvals, policy decisions, and more.

TIP

Keep in mind, despite the cool factor of deep learning and neural networks, each type of ML model has its own advantages and disadvantages (as well as costs), which makes each type well suited to solving a specific set of problems. ML algorithms are commonly used in cooperation with one another, or with other data modeling strategies, to solve complex data problems.

For cost, complexity, and efficiency reasons, simpler data models might be chosen in order to get "near enough" answers. Keep in mind the business considerations. So, while we still can, use your human brain to supplement what the machine tells you.

IN THIS CHAPTER

- How DevOps practices work with cloud delivery models
- Understanding security best practices in the cloud
- Introducing cloud operating certifications and compliance audits

Chapter 5
Cloud Operation and Security

Cloud technologies rely on operational teams and processes that enable cloud applications to change over time while remaining highly available and secure. I discuss some of those teams, processes, and compliance requirements in this chapter.

Continuous Integration and Delivery

Applications have changed in the modern cloud because of the adoption of microservices architectures in a container or serverless execution model. But, these architectures aren't a one-and-done build-and-deploy effort. Because technology and businesses are tightly integrated, the expectations are that as business requirements change rapidly, technology must change rapidly as well. The applications that businesses use require ongoing development, maintenance fixes, new features, and architectural evolution. To address that need, the software delivery framework for cloud applications must accommodate regular, reliable, and automated updates for deployment.

This release management framework is known as a release pipeline. Cloud teams frequently use an agile process called continuous integration (CI) and continuous delivery (CD) to execute the release pipeline.

CI is a practice by which developers regularly integrate their code into a central code repository so that the code can be tested and approved for faster rollout into the customer-facing production application. The pace might vary by organization (hourly, daily, weekly), but the goal of CI is to encourage developers to submit code in more frequent intervals. CI improves the pace of new software updates, primarily by automating testing and weeding out bugs and other software issues as early as possible in the development process.

CD is the second part of that pipeline that automatically prepares new code for use in the production system. This occurs by automating the process of build and unit tests, scale and load tests, integration tests, and so on. In CD, the actual approval for deployment to production is performed by an administrator or DevOps engineer, after the code has been approved through the CD process.


TECHNICAL STUFF

There is a slight alternative to continuous delivery called continuous deployment (also referred to as CD), wherein the deployment to production is also automated. From a terminological perspective, the variation between continuous delivery and continuous deployment is somewhat minor. Still, for some code deployment pipelines, the addition of automatic deployment can streamline daily (or more frequent) enhancements to an application.

Some CD enhancements to cloud applications happen in a soft launch (or soft release) workflow. Soft launch is a mechanism by which changes or new features are released incrementally, starting with a small audience. Soft launching allows for extended beta testing, customer/user feedback, and development iteration prior to full release to an entire customer set. It also mitigates the scope and impact of potential issues from new releases and software. There are many ways to perform soft launches, including custom URLs for hand-selected partners/customers, opt-in customer participation, geography-based testing, and random sampling.

Agile and Waterfall Processes

Prior to cloud adoption of microservices, most network administrators were accustomed to a release cadence for feature enhancements in monolithic software applications at a pace of a few times a year. This cadence is typical of a project management approach using a waterfall model. With waterfall, development occurred in linear stages toward a final release to customers. Once or twice a year, a slew of new features were planned, scoped, designed, developed, tested, modified, beta-tested, and then delivered all-at-once via a final release.

These giant waterfall releases introduced many new features at once, which also meant that they introduced many new issues at once. Each customer has its own process for adopting new code (and tolerance for the problems that come with it), but many customers would wait several months before adopting new releases. Those additional months provide time for the software provider (and its customers) to determine whether the release was reliable, as well as to find and fix defects from actual deployments. Then the provider would release maintenance versions or patches of the releases.

TIP

However, cloud management platforms are better suited for a project management approach using an agile model. With agile development, features are defined, scoped, built, and tested in small regular batches. Agile development works well with CI and CD because ongoing iterative change is much easier for the provider to deploy using the CI/CD automation. And it is easier for customers to adopt and integrate because the maintenance windows are much shorter and less disruptive (if there are maintenance windows at all).

With cloud applications built on microservices, many new features are built in phases. The new feature may not have all the bells and whistles in the first phase, but it evolves in a fast and continuous deployment timeline, based on customer and business requirements.

DevOps and CloudOps

DevOps (short for developer operations) is a function within IT systems that bridges the gap between development and operations of systems. DevOps combines people, processes, and tools to make sure that applications are well-managed across the entire lifecycle, from development through to production systems. The DevOps function is performed by a DevOps team, which is responsible for software integration, scaling, testing and quality assurance (QA), security, change management, internal monitoring/reporting, and more.

REMEMBER

As the modern cloud evolves, it is increasingly common to see a stronger distinction between DevOps and something called CloudOps. DevOps is focused more narrowly on the tools and systems used by developers to build and integrate code, whereas CloudOps is focused on the tools, systems, and infrastructure that are used to build cloud systems. CloudOps increasingly revolves around orchestration toolkits that are involved in the end-to-end ecosystem oversight.

Another layer of this evolution is a concept called Infrastructure as Code. In brief, what infrastructure as code means is that CloudOps teams are increasingly managing and overseeing the infrastructure with software tools (and skills) in the same way that teams treat application code. In the past, DevOps played a more integral role in hardware management and operation, whereas the cloud has shifted hardware operation to the cloud provider. Cloud has moved almost entirely to an API-centric model of interaction that is completely driven by code.

Operational Reliability

Any time an IT organization shifts the responsibility of operation onto a third party, that responsibility should come with financial commitments for service level agreements (SLAs). In cloud services, all cloud operators provide service level commitments to their customers as a part of a monthly or annual billing cycle. Most services fall into a service-level guarantee between 99.9%

and 99.99%, and there are predefined discount structures for failures to meet those SLAs.

If you're interested in the SLAs of the big three providers, here are some links:

- https://cloud.google.com/terms/sla
- https://azure.microsoft.com/en-us/support/legal/sla
- aws.amazon.com/legal/service-level-agreements

Organizations that build and offer SaaS products also provide an SLA. For the end-customer, the SaaS operator's SLA is the only contractual agreement. The SaaS operator is responsible for the monitoring and accountability of the underlying cloud host's service reliability (assuming the SaaS is built on IaaS and PaaS).

WARNING

In many cases, even though SLA agreements may legally commit to 99.9% or better, the actual service delivery exceeds the commitment. Also, keep in mind that these service commitments may not include planned outages for maintenance and upgrades. Be sure to read the fine print to determine what is and is not included in the agreement.

Security and Privacy

By involving third parties in IT systems and operations, you will also introduce questions related to security and data privacy.

TIP

If you are not an industry standards and compliance junkie, and you don't want to read the other info on this topic, I encourage you to lock in on one principle: Not all clouds are the same. Certifications and compliance are validated measures to ensure your cloud provider is following best practices, has proper integrity controls in place, and abides by regulatory standards. It's not sexy, but it is essential. Always confirm your cloud provider's stance before purchasing its solution.

Multitenancy

REMEMBER

Multitenancy is a security, administration, and privacy concept that is fundamental to how the cloud (and even many non-cloud) systems work today. Multitenancy is a way to separate the logical operations of computing systems such that software and hardware instances can serve multiple users/tenants while keeping each user partitioned or protected from other users. A properly designed multitenant system prevents each customer from knowing about other customers or being impacted by other customers, in every respect — data visibility, performance impact, and security vulnerability.

Multitenancy should be a core design foundation of any modern cloud system. However, there are use cases where less strict multitenancy may still be advantageous. Managed service providers (MSPs) may leverage public or private cloud systems to operate networks on behalf of their customers. The MSP is then responsible for the oversight of the tenancy, user creation and permission, and licensing and inventory. MSP use cases might represent just one more tier in the ecosystem of operator/customer relationships in the cloud.

Data Privacy

The data pipeline process includes both data in motion (transport functions) as well as data at rest (storage functions) (for more on this topic, see Chapter 4). Both stages are critical from a data privacy perspective. For many years now, it has become standard practice and customer expectation for data to be encrypted at all times, both in transit and at rest. Transport mechanisms rely on SSH, HTTPS, IPSec, TLS/SSL, and other standard security protocols that support encryption for data privacy while in motion.

TECHNICAL STUFF

Similarly, reputable cloud providers natively support encryption for the majority of all storage types using either AES-128 or AES-256, at either the disk, block, or file level. Some types of high transience data (buffers, caches, and so on) may not be encrypted by default, but may have encryption options, which may come with performance penalties. Encryption keys may also be flexibly controlled and managed by either the IaaS/PaaS operator, the SaaS providers, or the end-customer in a bring-your-own-key model.

Certification and Compliance

In the following section, I discuss a few of the cloud operating and security standards that address various aspects of operating cloud infrastructure and applications. In this review, it is very important to evaluate both the cloud infrastructure provider (the host of IaaS/PaaS) as well as any SaaS providers as separate entities. In other words, just because the cloud provider (for instance, Amazon, Microsoft, Google) has some certification and operating practices, it does not mean that the SaaS product follows the same practices or has the same certifications. When software is added to an IaaS/PaaS stack to create a SaaS, that software should also undergo robust controls and security diligence. For example, a cloud provider such as Amazon might hold numerous security certifications for its operations and datacenters, but the company that provides the hosted cloud application may not possess any comparable security certifications.

REMEMBER

Several industry compliance initiatives should be considered alongside cloud adoption and architecture. These compliance initiatives are important for any deployment where industry or organizational policies mandate their support. I cover them briefly in the remaining sections of this chapter.

HIPAA, PCI, and FIPS

The Health Insurance Portability and Accountability Act (HIPAA) is a healthcare act in the U.S. that defines protections and policies related to security, privacy, and breach notification. The primary goal of HIPAA is to protect patient information. This information is also known as personally identifiable information (PII), or as protected health information (PHI). In the cloud, HIPAA interests include the creation, maintenance, transmission, storage, and sharing of all PII data. HIPAA also protects from unauthorized sharing of information between service providers or cloud entities.

The Payment Card Industry Data Security Standard (PCI DSS) is a security standard focused on protecting information related to credit card information and processing. PCI addresses requirements for data encryption, controls for administrator system access, change management processes, and much more. You will find a heavy focus on PCI compliance in cloud deployments wherever retail transactions occur.

The Federal Information Processing Standard (FIPS) 140-2 is a security processing standard for the U.S. federal government that specifies requirements for security and cryptographic modules in hardware and software. The FIPS specifications for minimum security requirements include access control, configuration management, authentication, risk assessment, auditing, and much more. When evaluating FIPS compliance within the cloud, keep in mind that FIPS is both a security standard as well as a certification. End-to-end FIPS compliance requires cloud hardware and software to be certified. However, cloud systems can implement FIPS-compliant security and best practices without completing the certification process. Although FIPS 140-2 is a standard and certification required by many federal agencies within the U.S government, other countries also recognize the FIPS 140-2 standard or have similar regulations. Additionally, FIPS compliance may only be available in a subset of cloud datacenter regions.

GDPR

General Data Protection Regulations (GDPR) is a recent data privacy regulation passed by the EU that has garnered vast amounts of attention for cloud and IT systems. GDPR is the first-of-its-kind broad privacy mandate for the protection of Personally Identifiable Information (PII) on behalf of end-users in the EU and European Economic Area (EEA). GDPR requires appropriate technical and organizational steps to secure PII, and provide disclosure about its collection and transparency about its uses. The law also requires data collectors/processors to delete or anonymize PII data upon request by the user. Finally, GDPR also defines restrictions on the distribution and storage of PII data, ensuring that data collected within the EU/EEA is not sent and stored to other geographies. For this reason, many cloud architecture solutions are designed to provide regional cloud datacenters that collect and store the data in a specific country.


TIP

Though GDPR is EU-centric, it has also influenced many similar privacy policies in other countries and regions of the world, which will continue to gain momentum throughout the 2020s.

ISO/IEC certifications

If you evaluate 100 different companies and their cloud service's operational and security practices, you might find 100 different approaches, with varying degrees of security assurance and

diligence. To address the lack of predictability and transparency, the International Organization for Standardization (ISO), along with the International Electrotechnical Commission (IEC), offers several different programs to standardize these practices. Here are just a few of the essential standards for cloud security:

- ISO/IEC 27001 specifies a formal process for implementing and maintaining security controls for an information security management system (ISMS). The framework covers organizational processes for identifying, analyzing, and addressing security threats that are consistent with the organization's software and systems.
- ISO/IEC 27002 and 27017 are two security standards that specify operational guidelines for security best practices and controls. ISO 27002 has existed for a few decades, with revisions along the way, while 27017 is an enhancement to 27002 that specifies cloud-specific security controls.
- ISO 27018 and 27701 are standards for cloud service providers that focus on security controls (27018) and privacy controls (27701) for protecting personally identifiable information (PII).

Figure 5-1 displays the ISO/IEC 27001 certification for ExtremeCloud IQ.

Service Organization Control (SOC)

Accountants like to account for things, so the American Institute of Certified Public Accountants (AICPA) defined a set of audit reports for organizations that offer information systems as a service to their customers. These reports are called System and Organization Controls (SOC). SOC reports validate that the service organization meets standards for internal controls, which may be necessary for customers that rely on the operators' services. There are three general classes of SOC report:

- SOC1 describes the internal control environment for financial reporting.
- SOC2 describes the internal control environment across operating functions (also known as trust services principles): security, availability, processing integrity, confidentiality, and privacy.

» SOC3 provides a public report demonstrating that the service operator met the trust services principles. This report is similar to SOC2 but does not describe the control environment in detail because SOC3 is for public use instead of detailed customer validation.

For cloud computing services, SOC2 is the primary report of interest.

CERTIFICATE

This is to certify that

Extreme Networks, Inc.

6480 Via Del Oro
San Jose, CA 95119
United States of America

has implemented and maintains an
Information Security Management System.

Scope:
The Cloud Ops business processes that develop, support and manage ExtremeCloud IQ (formerly known as HiveManager or HiveManager Next Generation (NG)) services and all underlying assets that support those processes.

With reference to Statement of Applicability (SOA): Version 1.0, Dated 10/31/2019.

Through an audit, documented in a report, it was verified that the management system fulfills the requirements of the following standard:

ISO / IEC 27001 : 2013

Certificate registration no.	10017825 ISMS13
Date of original certification	2019-11-05
Date of revision	2020-02-15
Date of certification	2019-11-05
Valid until	2022-02-22

DQS Inc.

Brad McGuire
Managing Director

Accredited Body: DQS Inc., 1500 McConnor Parkway, Suite 400, Schaumburg, IL 60173 USA

FIGURE 5-1: ISO/IEC 27001 certificate - ExtremeCloud IQ.

Cloud Security Alliance (CSA)

The Cloud Security Alliance (CSA) is a nonprofit organization focused on security assurance for cloud systems. CSA promotes best practices and certifies the operation of cloud systems and their security postures. CSA's certification framework is called STAR (Security Trust Assurance and Risk) and aligns with General Data Protection Regulations (GDPR) requirements for data privacy and PII. There are three levels of CSA's STAR program:

- **Self-Assessment:** Organizations can perform self-evaluation and submit self-assessment and documentation demonstrating cloud security controls.
- **Third-Party Certification:** Organizations can receive a validated third-party audit for privacy and security controls.
- **Continuous:** Organizations can automate security and privacy controls with always-on monitoring and validation at all times.

IN THIS CHAPTER

- Examining a 4th generation cloud architecture
- Shifting network services to cloud management
- Introducing Co-Pilot

Chapter 6
ExtremeCloud IQ

This chapter introduces you to ExtremeCloud IQ, the enterprise cloud-driven networking management solution from Extreme Networks. ExtremeCloud IQ provides management from the edge to the datacenter and helps organizations automate network operations, gain insights from analytics, and optimize the end-user and application experience.

4th Generation Cloud Architecture

ExtremeCloud IQ is a 4th generation networking management platform that is built on containerized microservices and orchestrated by a Kubernetes infrastructure.

Centralized and unified management

REMEMBER

The IT group of an enterprise company may be responsible for supporting potentially hundreds to thousands of locations across the business footprint, but the organization can't guarantee or afford onsite IT resources at each property. With ExtremeCloud IQ, companies can offload the need for dedicated IT resources with centralized network visibility and management for all wired, wireless, and remote-networking assets. ExtremeCloud IQ management and monitoring capabilities provide network automation, insight, and assurance for your wireless and wired network.

Ultimately, cloud-based management from ExtremeCloud IQ reduces upfront business costs, automates deployments and centralizes support to ease IT burden and operational expenditure.

Just as important is unified management of APs, switches, branch routers, VPN gateways and other networking devices. An administrator does not have to monitor the wireless network separate from the wired network. All networking devices can be onboarded, configured, managed, and monitored from one console.

Furthermore, ExtremeCloud IQ allows an administrator to create a single network configuration policy that can span an entire enterprise network while keeping management simplified. Classification rules and cloud-configuration group objects make this possible even though network devices have unique settings at different locations.

Data

As shown in Figure 6-1, ExtremeCloud IQ currently ingests telemetry from over 5 petabytes of traffic every day from customer networks. To explain what that means, 1 petabyte of data is equal to 13.3 years of HD video. Therefore, every day, ExtremeCloud IQ analyzes health and behavioral patterns of the data equivalent of 66 years of HD video. Data science and machine learning algorithms are applied to such large datasets to drive accurate and high-performance data insights within the system.

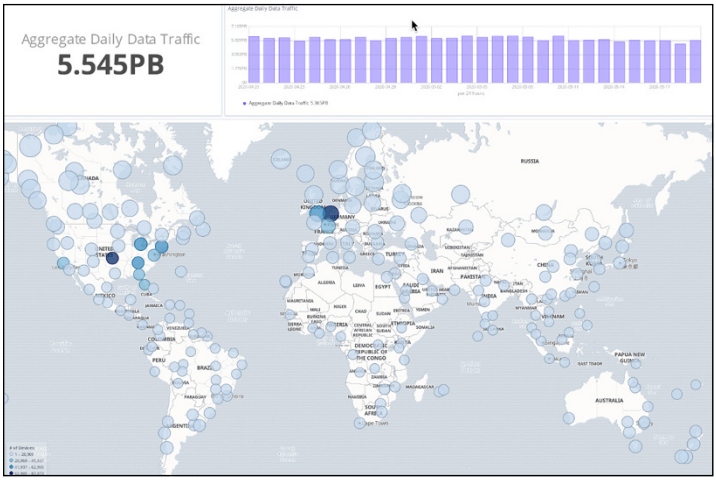


FIGURE 6-1: ExtremeCloud IQ daily data.

ExtremeCloud IQ offers rich data metrics for IT administration and business operations with an unlimited data horizon, meaning their entire dataset is available for the full duration of service. Network infrastructure, client, and application data are available both in real-time and from a historical viewpoint. Timeline slider bars can be used to focus in on custom time granularities that are of interest to an administrator.


TIP

With a 4th generation architecture, Extreme's cloud solution provides 99.9999999% of data durability. The eleven "9s" of data durability ensures the loss of only one object in 10,000 after 10 million years of constant read/write processes.

Elasticity

By building ExtremeCloud IQ using a modern web architecture, Extreme is able to scale services dynamically. Each regional datacenter in the global network serves the needs of many customers simultaneously in a multitenant design. Then a Kubernetes service oversees the usage and monitoring of containerized service clusters, scaling up or down depending on network load. Each regional system can service tens, hundreds, or thousands of customers with this elastic design. And all of this is completely transparent to customers, in a good way.

Flexibility

Different customers have different organizational needs. Although many customers today opt for public cloud deployment with its many advantages, other customers choose on-premises deployment due to organizational or legal requirements. With one underlying cloud networking platform, the feature set is consistent across deployment models, and customers can choose the option that works best for them without having to compromise on supported functionality.

Public, private, and local

Extreme offers a broad range of cloud deployment options for network management: public cloud, private cloud, and local cloud. The Extreme public cloud streamlines network operations with continuous updates, high availability, advanced analytics and insights, and anytime anywhere portal access.

Extreme partners and customers with large scale requirements above five thousand devices can deploy their own private cloud instance of ExtremeCloud IQ, capable of managing millions of connected access points, switches, and routers. A private cloud regional datacenter (RDC) provides the same benefits and features as the public cloud, however, a private RDC can be located within a customer's or partner's own infrastructure.


TIP

Private hosting is an ideal solution for managed service providers (MSPs) or enterprises who want the scalability and elasticity of the cloud with the added control of hosting it in their own datacenter or within their own segmented RDC instances within a cloud provider datacenter such as AWS.

Extreme's local cloud offers the same flexible architecture, but in a simplified and highly cost-efficient infrastructure deployed on-premises. This solution is ideal for small to mid-size organizations that want the power of the cloud in addition to complete control over their local deployment. And this same flexibility with public, private, and local cloud is being extended to traditionally on-premises services to hybridize the offering, for even more choice. This enables some platforms to have a foot in the cloud and a foot on-premises. These hybrid solutions are leveraging the modern toolsets of cloud (Kubernetes, Docker, and so on) and balancing the workloads across public/private and local resources.

Cloud agnostic

Some cloud solutions are only available via a single cloud provider. For a variety of reasons, certain cloud providers might not be the best option for a customer. For example, retail companies often prefer Google Cloud or Microsoft Azure because they view Amazon as a competitor. Additionally, a company might already have some cloud applications running on a specific cloud platform provider and might want their cloud networking management on the same platform. ExtremeCloud IQ is a fully agnostic Cloud-Driven networking management solution. ExtremeCloud IQ currently is available in the Amazon AWS, Google Cloud, and Microsoft Azure platforms.

Programmability

Like other web services, cloud-managed networks are inherently well-suited to management via a webUI as well as an API. Extreme IQ supports a number of RESTful API endpoints for monitoring,

configuration, and event visibility via a third-party system. It also supports Webhooks for event notifications into third-party systems.

Resiliency

REMEMBER

Customers expect reliability and uptime guarantees when it comes to access of network systems. Simply put, network management must always be available. Many cloud providers strive for 99.99999% availability which is also known as the seven "9s" of uptime, while they might only commit to 99.9% availability in an SLA. Providers must also clearly state their resiliency plans including redundancy and backup measures to ensure continued availability of networking services in case of outages and disasters.

ExtremeCloud IQ is engineered to meet these requirements of continuous availability by ensuring forward and backward compatibility. Every line of code in the application that interacts with a database is designed to anticipate potential database schema changes, and thus gracefully process it. In addition, the databases involved have to be backward compatible such that during the upgrade process the legacy application can still function. This is a ballet of code, conditional processing, and operating processes that come together to create a situation of zero downtime updates. With an application as large and complex as ExtremeCloud IQ, there are lots of moving pieces; however, when those pieces move in a well-orchestrated fashion, zero-downtime upgrades are accomplished.

Scalability

ExtremeCloud IQ currently manages over 1 million networking devices and can scale infinitely because of the 4th Generation cloud architecture. The geographically distributed public cloud architecture consists of datacenters in North America, Europe, and Asia. The cloud topology consists of several *global data centers (GDCs)* and multiple *regional data centers (RDCs)*. The GDCs manages customer and admin accounts. The GDCs are effectively landing pages for admin logins. The distributed RDCs are the workhorses in ExtremeCloud IQ. The RDCs are where all customer network management occurs and all customer network monitoring data resides. RDC-level management allows customer data to reside in-region and in-country, which is often necessary for compliance with local data security and privacy regulations.

Extreme is always adding more RDCs and its CloudOps team can stand up a new regional datacenter (RDC) in under 30 minutes.

Simplicity

One of the best aspects of management from the cloud is simplicity. ExtremeCloud IQ streamlines every aspect of network management with a user interface (UI) that is friendly and easy to use. The UI of ExtremeCloud IQ is sufficient for 98 percent of configuration tasks as well as both manual and automated troubleshooting tasks. Figure 6-2 shows the user-friendly and intuitive UI of ExtremeCloud IQ.

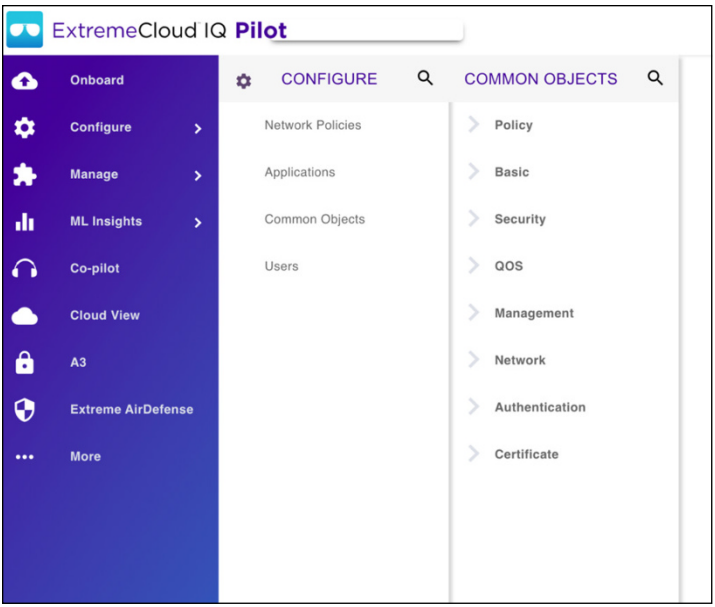


FIGURE 6-2: User interface - ExtremeCloud IQ.

But rest assured, Extreme still offers a means of remote command line interface (CLI) capabilities from the cloud for the power-user. Supplemental CLI configuration objects can be appended to any network policy or device-level policy. Furthermore, the SSH Proxy diagnostic tool provides for advanced CLI troubleshooting to any remote networking device via ExtremeCloud IQ.

Security

To ensure the highest levels of information systems and data protection, management, and compliance, ExtremeCloud IQ is ISO/IEC 27001 certified by the International Standards Organization (ISO).

The cloud providers where ExtremeCloud IQ is hosted implement their own set of activities and controls to keep datacenters and the customer data in them secure. These include business continuity and disaster recovery processes, comprehensive controls, and facility monitoring.

Extreme takes additional measures to secure cloud-based applications, including firewalling, continuous threat monitoring, DDOS prevention, daily backup, penetration testing, DevOps activity audits, and much more.

Zero-touch provisioning

Manual configuration at local sites is labor intensive, prone to errors, costly, and time consuming. With ExtremeCloud IQ, there is no need for local configuration. No matter where in the world a networking device is deployed, zero-touch provisioning is a reality. You can connect an AP, the AP auto-discovers your cloud account, and the AP then automatically downloads needed firmware and the device's unique configuration. Within minutes, you have an AP broadcasting an SSID to provide wireless access for your employees and devices.

Cloud-Speed Innovation

ExtremeCloud IQ is built on an internal process of continuous integration and continuous deployment that enables what Extreme calls *Cloud-Speed* innovation. (For more on this topic, see Chapter 5.)

The Cloud-Speed deployment pace provides customers with unprecedented feature velocity and bug fix availability, operating in a continuous delivery mode and ensuring that ExtremeCloud IQ is automatically updated with the latest capabilities. Last year alone, more than 100 core features were added to ExtremeCloud IQ.

Continuous cloud updates provide a steady flow of new functionality, but administrators still remain in control of device updates, ensuring minimal downtime and disruption. Hardware OS updates can be scheduled for all or part of the network devices, at a time convenient for the organization.

To learn more about ExtremeCloud IQ innovations, be sure to check out the monthly Cloud-Speed Innovation video series on Extreme Network's YouTube video channel at `https://bit.ly/CloudSpeed`.

Cloudification

The microservices design technique breaks an application down into small operating services, each with well-defined boundaries of functionality (in Chapter 3, I discuss the concept of microservices software design). The individual services are woven together via application programming interfaces (APIs) in a loosely coupled environment. Containers and microservices change the game for applications that live in the cloud.

Extreme Networks has been very busy with the cloudification of the entire networking product portfolio. The term *cloudification* refers to the conversion and/or migration of data and application programs in order to make use of cloud computing. All of these networking solutions and applications from Extreme Networks have a clear path of management via the cloud:

- Cloud IQ Engine APs
- ExtremeWireless WiNG
- ExtremeSwitching
- Extreme Remote Networking
- Extreme AirDefense
- ExtremeNAC
- ExtremeGuest

- ExtremeLocation
- Extreme Defender for IoT

Co-Pilot: Network Optimization via ML

Co-Pilot is the next phase of machine-learning analytics found in ExtremeCloud IQ. As shown in Figure 6-3, Co-Pilot enhances network management by leaning into a cloud data architecture and applying various machine learning techniques to extra value for users.

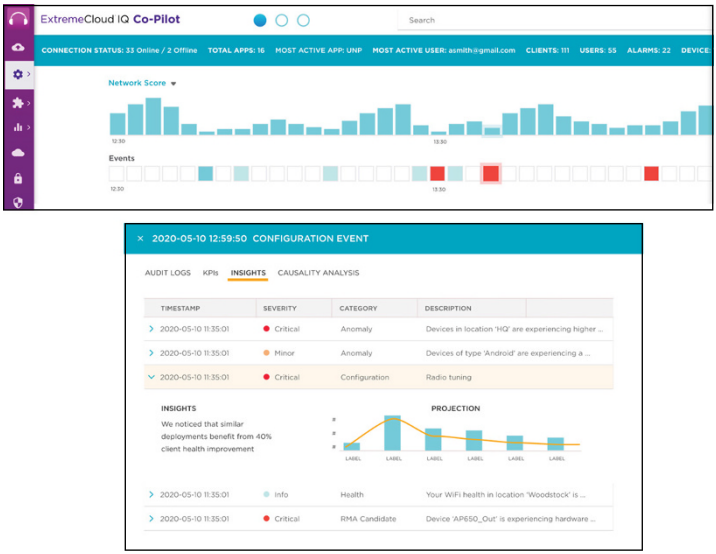


FIGURE 6-3: Co-Pilot.

For example, Co-Pilot enhances search and provides the data warehouse to users in a more comprehensive format for analysis. Anomaly detection is layered into health metric monitoring to determine when network nodes or locations are experiencing

poor client-impacting service, such as high connection failures or low capacity. Also, it's common for administrators to scratch their heads as they look at all the seemingly random events and alarms popping up in a system, so Co-Pilot adds event pattern mining algorithms to extract out meaningful sets of events, root causes, and connected issues with configurations, other events, or health data that might seem unrelated.

Do you want to learn more and experience the benefits of cloud-driven end-to-end enterprise networking with ExtremeCloud IQ? Register today for a free 30-day trail account with ExtremeCloud IQ:

`https://www.extremenetworks.com/cloud-networking`

Chapter 7
Top Ten Reasons for Cloud Supremacy

For this chapter, I put together a top ten list summarizing the virtues of the cloud. Some organizations may never shift their computing to the cloud for privacy or latency reasons — and of course, networks will always have some device footprint on-premises. But if there's any doubt, cloud has already won the battle against on-premises computing models for the majority of workloads. Here are some reasons why.

Cloud Operations Are a Game Changer

Every IT organization faces a question about the financial and technical benefits of owning and operating systems in-house or using third parties. Cloud providers have dramatically influenced that conversation by offering the most reliable computing platforms, easy-to-use packaged services, low-overhead operating model, and favorable cost structures. The performance, scale, and reliability are tough to beat, and it comes with almost zero effort.

Cloud Costs Are Compelling

For many businesses, it's impossible to match the operating benefits of cloud with insourced systems, and it's also challenging to turn down the cost benefits of cloud economies of scale. In some situations, long-term total cost of ownership doesn't play out in cloud's favor, but the flexibility benefits of OPEX budgeting are often more compelling than a long-term tradeoff. This is particularly true when you factor in the development velocities of cloud, and the corresponding impact this has on addressing business objectives.

Cloud Has Many Flavors

Though cloud is a universal term, it offers many different consumption options for each user type. Cloud has a model for hosted infrastructure services (IaaS) on which you can freely build whatever you dream up. Cloud has a model where platform services (PaaS) make it easier and faster to build applications. And cloud has a model where you can shop around and buy ready-to-use turnkey services (SaaS) that have already been built by someone else.

Cloud Has Many Places

Cloud services are offered in many ways, and with modern software architectures and packaging, clouds can be deployed in a variety of places. Public cloud provides remotely hosted services available to anyone. Private cloud provides dedicated services for a specific organization or entity. And hybrid cloud combines both public and private computing worlds into a blended architecture. There's even a local cloud if you want it that way.

Cloud Is Still Evolving

Cloud technologies have changed quite radically in the last decade, and are still changing. Modern software architectures in the cloud provide for dynamic orchestration and resourcing with

Kubernetes, streamlined containerized code with Docker, and microservice architecture construction for resilient operation and modular services. In the next decade, we'll see even more systems taking advantage of the operating and cost efficiencies of serverless operation.

Cloud Is Ready for Data

Driven in part by mobile devices, cloud services have evolved to become the preferred choice for data-focused applications and architectures. Cloud systems solve for the three V's of big data. Cloud solves for volume with low-cost scalable distributed systems (without the complex operational overhead); cloud solves for velocity with new cloud-native data messaging, telemetry, ETL, and stream processing systems. And cloud solves for variety by making it easier to implement multiple databases in an application, including high-performance data warehouses, hot cached data, inexpensive cold storage, and distributed data backups for resiliency.

Cloud Has the Toolset

As IT and development teams turn to cloud for infrastructure, they're also turning to cloud for platforms and services that make it easier and faster to build applications. Cloud providers have responded by providing a robust set of fully managed services (for example, ready-to-use databases, machine learning toolkits, serverless computing, data streaming services, and container management solutions). These hosted service offerings and development toolsets are largely responsible for the dramatic, and ever-growing, advantages of cloud.

Cloud Enables Machine Learning

Machine learning and artificial intelligence applications have grown exponentially in the last decade because of cloud toolkits. Cloud providers now have end-to-end ML services for building, training, and implementing ML models, so even small teams with

less expertise can build automation and learning solutions. And, the power of deep learning and neural networks is possible for AI use cases because of the limitless data volumes and data processing paradigms of the cloud.

Clouds Can Be Certified

Organizations follow different processes for cloud development, security, and operation. And those processes come with varying degrees of integrity in terms of documentation, audits, security, and compliance. Cloud providers should provide operating SLAs with financial guarantees; they should operate according to regulatory requirements like GDPR and industry requirements like PCI-DSS; and they should provide certifications validating their process, like the ISO 27000 family and CSA's STAR compliance programs.

Not All Clouds Are the Same

Despite the ubiquity of the cloud, not all clouds are created equal. Consumers should carefully evaluate clouds across several criteria:

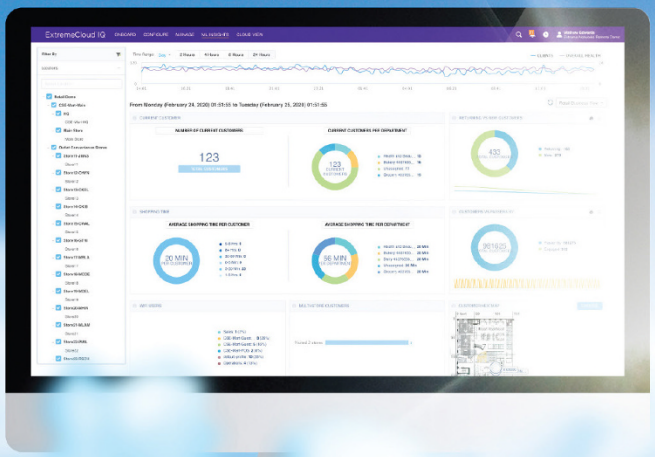
- **Generations of cloud technology:** Is the cloud built on modern architectural principles like dynamic orchestration, containerized microservices, and flexible resource provisioning?
- **Cloud hosting options:** Is the cloud built in an agnostic way that makes it available in public, private, hybrid, and local models? And is the cloud available via different cloud hosting providers, such as Amazon, Google, and Microsoft?
- **Cloud basics:** Is the provider offering a no-downtime operating environment, following regulatory compliance standards, and certified by a third party audit?

WELCOME TO

ExtremeCloud™ IQ

Powered by the industry's only 4th generation cloud platform with unlimited data retention, ExtremeCloud IQ delivers ML-driven data analytics for user, device, and network 360 insights, centralized and unified orchestration of thousands of wired and wireless devices, and advanced health monitoring KPIs to keep highly distributed networks up and running.

Learn more at **extremenetworks.com/extremecloud-iq/**.



ADVANCE WITH US

The next generation of cloud-managed networks

Network architectures have shifted in the last 20 years from hardware appliances and virtual machines toward a software- and cloud-first paradigm. With that shift, network professionals are exposed to an entirely new set of technologies, like microservices, containers, serverless, and orchestration. Along with the architectural shift, modern technologies are heavily focused on automation and APIs as well as new the language of data, machine learning, and artificial intelligence.

Inside...

- Understand modern cloud architectures and software building blocks
- Recognize the as-a-service models
- Get a handle on big data, machine learning, and AI
- Learn to evaluate cloud systems and generations of technology

Marcus Burton is a Technology Architect at Extreme Networks where he works on the adoption of cloud, machine learning, and other technologies to solve networking problems. He is CWNE #78 and has a BA in English and Psychology.

Go to Dummies.com™
for videos, step-by-step photos, how-to articles, or to shop!

ISBN: 978-1-119-59219-8
Not For Resale

for dummies®
A Wiley Brand

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.