

LEARNING MADE EASY

Avi Networks Special Edition

# Multi-Cloud Load Balancing

for  
**dummies**<sup>®</sup>  
A Wiley Brand



Deliver multi-cloud  
applications

Derive rich application  
insights with analytics

Support new app  
architectures

Brought to you  
by:

**vmware**<sup>®</sup>

Lawrence Miller  
Ranga Rajagopalan

## **About VMware NSX Advanced Load Balancer by Avi Networks**

VMware NSX Advanced Load Balancer (Avi Networks) provides multi-cloud load balancing, web application firewall, and container ingress services with application analytics across on-premises data centers and any cloud. The software-defined platform delivers applications consistently across bare metal servers, virtual machines, and containers to ensure a fast, scalable, and secure application experience. Learn how customers like Adobe and Deutsche Bank enjoy 90% faster provisioning and 50% lower total cost of ownership.



# Multi-Cloud Load Balancing

Avi Networks Special Edition

by **Lawrence Miller and  
Ranga Rajagopalan**

for  
**dummies**<sup>®</sup>  
A Wiley Brand

# Multi-Cloud Load Balancing For Dummies®, Avi Networks Special Edition

Published by: **John Wiley & Sons, Ltd.**, The Atrium, Southern Gate Chichester, West Sussex, [www.wiley.com](http://www.wiley.com)

© 2019 by John Wiley & Sons, Ltd., Chichester, West Sussex

*Registered Office*

John Wiley & Sons, Ltd., The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted by the UK Copyright, Designs and Patents Act 1988, without the prior written permission of the Publisher. For information about how to apply for permission to reuse the copyright material in this book, please see our website <http://www.wiley.com/go/permissions>.

**Trademarks:** Wiley, For Dummies, the Dummies Man logo, The Dummies Way, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates in the United States and other countries, and may not be used without written permission. Avi Networks and the Avi Networks logo are trademarks or registered trademarks of Avi Networks. All other trademarks are the property of their respective owners. John Wiley & Sons, Ltd., is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: WHILE THE PUBLISHER AND AUTHOR HAVE USED THEIR BEST EFFORTS IN PREPARING THIS BOOK, THEY MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS BOOK AND SPECIFICALLY DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IT IS SOLD ON THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING PROFESSIONAL SERVICES AND NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. IF PROFESSIONAL ADVICE OR OTHER EXPERT ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL SHOULD BE SOUGHT.

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact [info@dummies.biz](mailto:info@dummies.biz), or visit [www.wiley.com/go/custompub](http://www.wiley.com/go/custompub). For information about licensing the *For Dummies* brand for products or services, contact [BrandedRights&Licenses@Wiley.com](mailto:BrandedRights&Licenses@Wiley.com).

ISBN 978-1-119-54702-0 (pbk); ISBN 978-1-119-54728-0 (ebk)

Printed in Great Britain

10 9 8 7 6 5 4 3 2 1

## **Publisher's Acknowledgments**

We're proud of this book and of the people who worked on it. For details on how to create a custom *For Dummies* book for your business or organization, contact [info@dummies.biz](mailto:info@dummies.biz) or visit [www.wiley.com/go/custompub](http://www.wiley.com/go/custompub). For details on licensing the *For Dummies* brand for products or services, contact [BrandedRights&Licenses@Wiley.com](mailto:BrandedRights&Licenses@Wiley.com).

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

**Project Editor:** Jennifer Bingham

**Acquisitions Editor:** Katie Mohr

**Editorial Manager:** Rev Mengle

**Business Development**

**Representative:** Frazer Hossack

**Production Editor:**

Mohammed Zafar Ali

**Special Help:** Ranga Rajagopalan,  
Chandra Sekar, Joanne Ayres,  
and Lei Yang

# Table of Contents

INTRODUCTION .....	1
About This Book .....	1
Foolish Assumptions .....	2
Icons Used in This Book .....	2
Beyond the Book .....	3
Where to Go from Here .....	3
CHAPTER 1: <b>Losing Your Balance in a Multi-Cloud World</b> .....	5
Hardware Versus Software: The Hard Facts .....	5
Virtualized and Software-Defined Aren't the Same Thing .....	7
The Seven Requirements of Highly Effective Load Balancers .....	7
CHAPTER 2: <b>Rediscovering the "Center of Balance" for Your Application</b> .....	11
A Software-Defined Application Services Architecture – Defined .....	11
The Three Pillars of a Modern Application Services Platform .....	13
Multi-cloud .....	14
Intelligence .....	14
Automation .....	15
CHAPTER 3: <b>Attaining Security Nirvana with Modern Application Services</b> .....	19
Using a WAF to Inspect and Protect the Web Application Stack .....	19
Creating an Elastic Service Mesh for a Microservices Architecture .....	22

**CHAPTER 4: Exploring Multi-Cloud Load Balancing Use Cases**..... 27

- A Refreshing Approach to Load Balancer Refresh ..... 27
- Clearing the Air about Load Balancing in the Cloud ..... 28
- Going Global with Server Load Balancing ..... 30
- SSL Is So SSlow on Your Legacy Load Balancer ..... 31
- Putting Microservices under a Microscope..... 35

**CHAPTER 5: Ten Things You Can Do with Multi-Cloud Load Balancing** ..... 39

# Introduction

Software-defined architectures have transformed enterprises seeking to become application-centric, with many modern data centers now running a combination of cloud-native applications based on microservices architectures alongside traditional applications. With application owners seeking public-cloud-like simplicity and flexibility in their own data centers, IT teams are under pressure to deliver services and resolve application issues quickly, while simultaneously reducing provisioning time for new applications and lowering costs for application services.

Legacy load balancing solutions force network architects and administrators to purchase new hardware, manually configure virtual services, and inefficiently overprovision these appliances. At the same time, new container technology choices are also enabling IT teams to rearchitect applications into microservices from monolithic or n-tier constructs.

These transformations are forcing organizations to rethink load balancers, also known as application delivery controllers (ADCs), in their infrastructure. Your IT operations demand agile, cost-effective solutions for any cloud because modern applications can be deployed anywhere. In this book, you'll learn about a more innovative approach to modern application delivery — one that focuses on business outcomes, instead of manually defined inputs, to support digital transformation initiatives and position enterprise IT for the future in a multi-cloud environment.

## About This Book

This book consists of five chapters that explore:

- » Traditional load balancer challenges in a multi-cloud world (Chapter 1)
- » The requirements and benefits of modern multi-cloud load balancing (Chapter 2)
- » How web application firewalls and an elastic service mesh securely enable modern application services (Chapter 3)



- » Multi-cloud load balancing in the real world (Chapter 4)
- » What you can do with multi-cloud load balancers (Chapter 5)

## Foolish Assumptions

It's been said that most assumptions have outlived their usefulness, but we assume a few things nonetheless!

Mainly, we assume that you're someone responsible for application delivery in your organization. Perhaps you're an IT executive/vice president/director, application owner, cloud architect, network engineer, or network administrator. As such, this book is written primarily for technical readers with at least a basic understanding of cloud computing, networking, and IT operations management. But don't worry, we promise not to get too technical and we'll be sure to explain any techie concepts and jargon along the way.

If any of these assumptions describe you, then this book is for you! If none of these assumptions describe you, keep reading anyway. It's a great book and when you finish reading it, you'll be a multi-cloud load balancing leader!

## Icons Used in This Book

Throughout this book, we 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 we 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

There's only so much we can cover in 48 short pages, so if you find yourself at the end of this book, thinking "where can I learn more?" just go to [www.avinetworks.com](http://www.avinetworks.com).

## Where to Go from Here

If you don't know where you're going, any chapter will get you there — but Chapter 1 might be a good place to start! However, if you see a particular topic that piques your interest, feel free to jump ahead to that chapter. Each chapter is written to stand on its own, so you can read this book in any order that suits you (though we don't recommend upside down or backwards).

## IN THIS CHAPTER

- » Comparing hardware and software-based load balancers
- » Differentiating between virtualized and software-defined
- » Reimagining the modern load balancer for a multi-cloud world

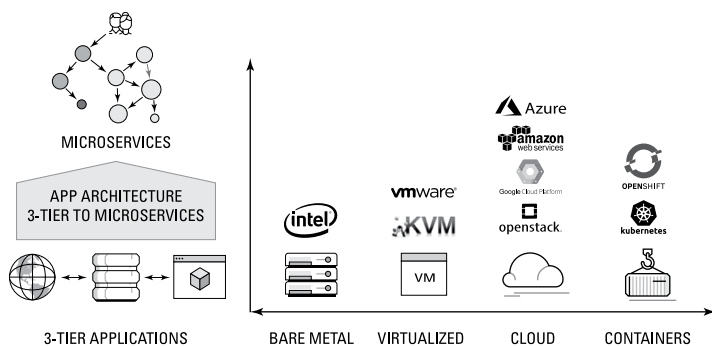
# Chapter **1**

# Losing Your Balance in a Multi-Cloud World

In this chapter, you explore the differences between legacy hardware and software-based load balancers, including legacy virtualized appliances and modern software-defined application services platforms. Finally, you'll review a list of capabilities you might include if you could design your next load balancer.

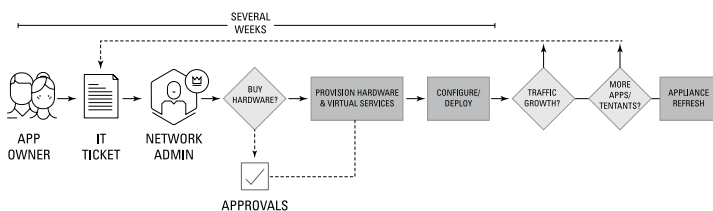
## Hardware Versus Software: The Hard Facts

Flexible infrastructure choices and application architectures are changing the way that modern enterprises run their computing environments (see Figure 1-1). Enterprises have become application-centric, investing significant effort and resources on continuous delivery goals and DevOps practices to achieve automation of routine IT and operations tasks.



**FIGURE 1-1:** Computing today: Evolving app architectures and infrastructure heterogeneity.

Hardware-based application delivery controllers (ADCs) have been the staple of application delivery in data centers for the last two decades. However, these legacy load balancing solutions aren't keeping up with the changes in modern, dynamic computing environments. Legacy hardware-based ADCs have become inflexible in the face of changing requirements, delaying application rollouts and causing overspending and overprovisioning in many cases. Most enterprises experience a process similar to the workflow shown in Figure 1-2 when it comes to rolling out new applications or updates, which can take weeks.



**FIGURE 1-2:** Legacy hardware-based load balancing solutions are not keeping up with the modern pace of business.



**WARNING**

With aggressive continuous delivery goals and ever-greater customer expectations, businesses are pushing back against delays due to hardware provisioning and manual configurations of ADCs that slow time to market for application deployments and updates.

# Virtualized and Software-Defined Aren't the Same Thing

As virtualization technology has advanced and matured over the past decade, many ADC vendors have begun to offer a virtualized version of their hardware-based appliances. Virtualized appliances enable ADC vendors to offer a lower cost alternative to their hardware-based appliances in a form factor that can be deployed on x86 servers. However, virtual appliances inherit most of the architectural debt (control plane and data plane tightly coupled in an appliance) and are no better at solving many of the fundamental issues — such as inflexibility, overprovisioning, and management challenges — with hardware-based ADCs in modern data centers.



TIP

Instead, enterprises need flexible and agile application services that align with their continuous integration and development (CI/CD) processes. Advances in the power of commodity x86 servers are enabling software-defined principles to be applied to modern application services platforms.

Unlike a virtualized load-balancing appliance, a software-defined application services platform separates the data and control planes to deliver application services beyond load balancing, real-time application analytics, security and monitoring, predictive auto-scaling, and end-to-end automation for Transport (Layer 4) to Application (Layer 7) layer services. The platform supports multi-cloud environments and provides software-defined application services with infrastructure-agnostic deployments on bare metal servers, virtual machines (VMs), and containers, in on-premises data centers and private/public clouds.

## The Seven Requirements of Highly Effective Load Balancers

The late Dr. Stephen R. Covey once said, “True effectiveness requires balance.” Okay, certainly not the same context, but his words of wisdom are no less true here — true application

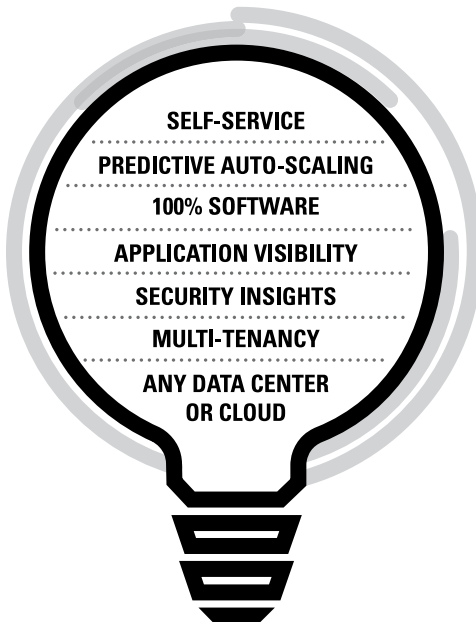
effectiveness requires a modern load balancer! So, with our apologies to Dr. Covey, the seven requirements of highly effective load balancers (see Figure 1-3):

- » **Be proactive — with self-service.** Application owners and DevOps teams can't wait several weeks for application services to become available. Despite several automation initiatives, and adoption of agile application development models, legacy load balancers force application developers to "create a ticket and wait." A modern load balancer provides a single point of management and automation for administrators to quickly provision the capacity required by their applications. Administrators can integrate the platform with their own orchestration services, and provide self-service and role-based access control (RBAC) for owners to provision, troubleshoot, and monitor their own applications.
- » **Begin with the end in mind — with on-demand autoscaling.** Today's enterprises must understand and plan for peak usage times, with the capability to scale down when demand recedes. Consider Black Friday and Cyber Monday, for example. The infrastructure needs to be scaled up and down elastically, without causing overprovisioning and idle capacity during other times of the year. At the end of the day, a smooth user experience in the application is the outcome you want to achieve, but it needs to be handled by the infrastructure automatically.
- » **Put first things first — with a 100 percent software solution.** The need to support on-premises data center and multi-cloud deployments means that enterprises, first and foremost, need to choose applications that work consistently across different environments with a single point of orchestration. A software-defined solution can be deployed on bare metal servers, VMs, or containers in private data centers or public clouds.
- » **Think win/win — with real-time insights and visibility into application traffic.** Agile enterprises need to troubleshoot application issues while minimizing disruptions and mean-time-to-innocence (MTTI) — the time that it takes network teams to prove that it isn't the network! In all seriousness, getting to the bottom of application performance and latency issues can sometimes involve multiple teams and take several

hours or days — which disrupts business productivity. That's a lose/lose situation. Okay, actually it's more of a catch-22, but either way it's a no-win situation! A modern load balancer should collect real-time application telemetry and provide insights into each transaction and end user patterns. The load balancer is best positioned to “record and replay” traffic events, enabling administrators to troubleshoot and resolve application issues in a matter of minutes.

- » **Seek first to understand, then to be understood — with security insights.** Or in the words of Sun Tzu, “Know the enemy and know yourself.” Many application vulnerabilities stem from incorrect access control policies or misconfigurations of application delivery controllers (ADCs, also known as load balancers). And attackers are all too ready to exploit any vulnerabilities you expose. Although ADC security features have improved over time, legacy solutions still can't offer a simple way to understand the applications' security postures at a glance, and can't work across multiple data center or cloud environments.
- » **Synergize — with multitenancy.** Large enterprises supporting multiple applications and groups on shared infrastructure often end up spending more than necessary on lines of business (LoB). Legacy ADCs fuse control and application services into a single appliance. When IT teams deploy applications on these appliances it typically shares this infrastructure across multiple LoBs. To support this shared model, many leading appliance-based ADCs provide some rudimentary form of multitenancy to isolate tenants. However, this model doesn't provide true per-app isolation and security; a distributed denial-of-service (DDoS) attack or a single busy application could still hog resources to the detriment of others (the “noisy neighbor” problem). System upgrades, workload movements, or the addition of new tenants can also be disruptive events for all tenants.
- » **Sharpen the saw — with a software-defined approach to application services.** Okay, this one's a bit of a stretch, but bear with us. Sharpening the saw is about self-renewal so that you're always balanced. With a software-defined application services architecture (discussed in Chapter 2), you can automate IT operations and integrate with existing and evolving technologies like OpenStack, virtualized

environments, public cloud, software-defined networking (SDN), containers, Kubernetes, and more. This flexibility ensures that you can deliver application services effectively and reliably in any data center or cloud, now and in the future. So, a software-defined application services architecture is like a self-renewing resource — not such a stretch after all!



**FIGURE 1-3:** The big idea: Re-imagining the modern load balancer.



- » Laying the foundation for a modern application services architecture
- » Enabling multi-cloud, intelligent, and automated application services

# Chapter 2

## Rediscovering the “Center of Balance” for Your Application

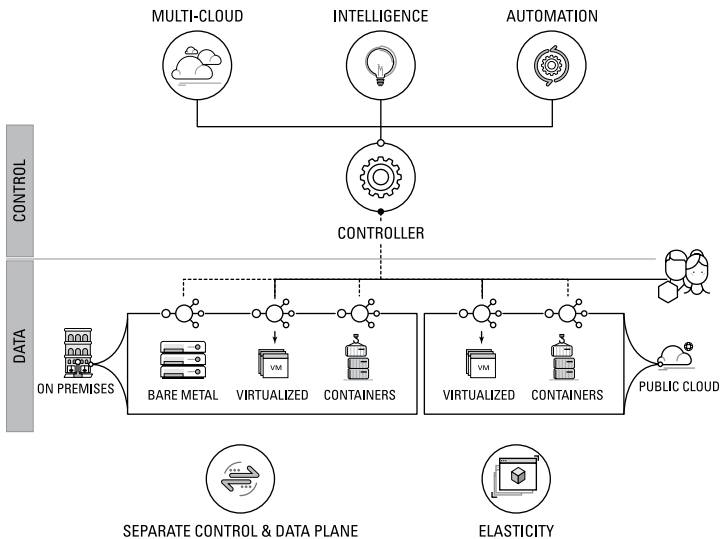
In this chapter, you learn about the key components and capabilities of a modern, software-defined application services platform.

### A Software-Defined Application Services Architecture – Defined

Application services built on software-defined principles need to mirror the automation goals for your data center and cloud initiatives. As more enterprises adopt web-scale IT strategies, application services that don't depend on proprietary hardware or environments (data center or cloud platforms) become a requirement. In order to address both current and emerging requirements in application-centric enterprises, a software-defined application services architecture needs to be defined.



A key architectural principle of a software-defined application services architecture is the separation of the control and data planes, which enables the system to have programmable “service engines” to deliver more than load balancing services (see Figure 2-1). For example, a modern application services platform can deliver real-time application analytics, security and monitoring, predictive autoscaling, and end-to-end automation for Transport layer (Layer 4) to Application layer (Layer 7) services. It also needs to integrate with the underlying network infrastructure (Layer 2 and 3), to deliver full-stack automated application services.



**FIGURE 2-1:** High-level modern distributed application services platform.

Application delivery in a software-defined application service architecture is provided by a distributed *data plane* deployed alongside application instances that can run across any commodity x86 servers, virtual machines, containers, or public cloud environments. The load balancers in the data plane sit in line with application traffic and collect and relay application telemetry continuously to the *control plane*. The load balancers represent a pool of resources that can be deployed on-demand across multi-cloud environments wherever application services are desired. Compared to legacy appliance-based application delivery controllers (ADCs), software load balancers take advantage of this flexible and modern architecture to provide elastic, high-performance, and high availability services at lower costs.

The software load balancers can be deployed to deliver services right between the end users and actual application or even on a per-application basis. This eliminates the need to backhaul (or trombone) traffic to the edge of the network as is the case with hardware appliances. The approach also enables services for east-west traffic among containerized applications in addition to the traditional north-south transactions between end users and data center applications. Newer application architectures, including microservices, benefit from this approach due to dynamic and granular application services enabled by the software-defined architecture.



The *control plane* represents a central point of management, orchestration, and policy control for the distributed software-defined application services. The control plane is the brain behind the services delivered by the data plane. It receives and analyzes the continuous stream of application telemetry sent by the distributed load balancers across the environments to decide on service discovery, placement, autoscaling, and high availability for each application.

## The Three Pillars of a Modern Application Services Platform

The high-level components that make up the control plane in a modern application services platform comprise the three pillars that deliver the key capabilities of the platform: multi-cloud, intelligence, and automation (see Figure 2-2).

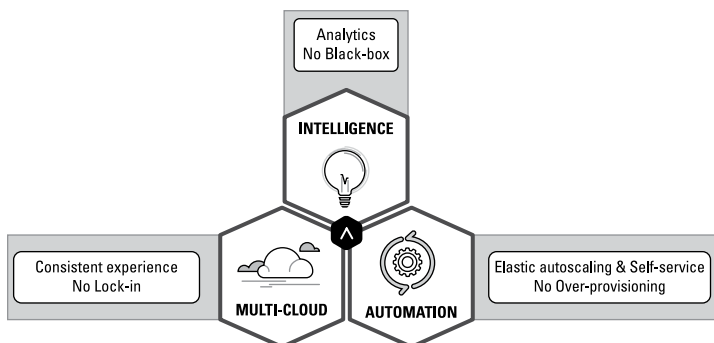


FIGURE 2-2: Key capabilities.

## Multi-cloud

The RightScale *2018 State of the Cloud Report* found that organizations are using an average of five clouds to run their application workloads today. This hybrid, multi-cloud environment is comprised of public and private clouds, with enterprises prioritizing migrations to the public cloud in 2018. Thus, a modern software-defined application services platform must support multi-cloud deployments.

*Cloud connectors* in the application service platform's control plane enable integration and multi-cloud capabilities. The control plane's cloud connectors facilitate turnkey deployments of application services in any on-premises data center and private/public cloud environments. Cloud connectors natively integrate with the control points of cloud platforms such as OpenStack, software-defined networking (SDN) controllers such as Cisco Application Policy Infrastructure Controller (APIC), virtual machine managers such as VMware vCenter, or public cloud providers such as Amazon Web Services (AWS) and Microsoft Azure.

In newer microservices environments, the platform integrates with the container orchestration platforms to deliver application services such as service discovery, service proxy, microsegmentation, and service mapping across multiple environments. The control plane initiates the appropriate script or application programming interface (API) call to the cloud or data center orchestration platform to instantiate additional application workloads.



TIP

Cloud connectors can help future-proof your application services strategy in the face of the continuous evolution of application architectures and infrastructure choices.

## Intelligence

Load balancers are deployed in a prime location — in the path of traffic from end users to the application. The most powerful value of the software-defined application services platform is the capability to derive rich application insights from the inline analytics provided by the data plane. The machine learning based *analytics engine* enables application performance monitoring, troubleshooting, and operational insights.

A big data engine analyzes the application telemetry information received from the distributed data plane to provide detailed transactional information, including round trip times from the end user all the way to the back-end application.

The system records all the transactional data and plays it back on-demand to administrators or application owners through the central management console, to provide pinpoint analysis of application issues or failures over specific periods of time.



TIP

The analytics engine also provides detailed views of the security posture including:

- » Distributed denial-of-service (DDoS) attack status
- » Secure sockets layer/transport layer security (SSL/TLS) certificates used
- » Security health scores

Client insights include data such as:

- » Connecting devices
- » End user geographical location
- » Transaction durations

All of these capabilities make the services much more valuable to administrators who otherwise have to spend significant time and effort, and use multiple point solutions to troubleshoot or monitor application performance.

## Automation

Automation capabilities in the modern application services platform are delivered through three main components:

- » **Lifecycle manager** (automated service creation, placement, autoscaling, and high availability)
- » **Policy repository** (configurations and policies)
- » **Representational State Transfer (REST) APIs** (automation for IT and self-service for developers)

Because the data plane elements are deployed on commodity hardware, they can be spun up or down dynamically wherever services are needed or when services need to be scaled out or scaled in elastically. The lifecycle of the load balancers in the data plane are managed by the *lifecycle manager* in accordance with policies set on the control plane by administrators. The control plane also monitors the health of the system and reacts to data plane component failures or any application changes. With the power of software load balancers, active-active high availability configurations can be set up at a fraction of the cost of traditional load balancers. Also, when load balancers need to be scaled in response to application needs, the control plane simply spins up additional resources and automatically configures them.

The *policy repository* holds all of the system configurations, including the details of all virtual services and pool members. Today, many network administrators painstakingly document virtual service configurations (sometimes using spreadsheets) in a futile effort to maintain an accurate representation of their deployed services. The policy repository eliminates the need for such manual steps and enables an administrator to readily see a tree view of their deployment and specify their intent. The repository is automatically updated when application workloads move or IP addresses change.

Finally, a modern software-defined application services platform must make all of its services available through a complete set of *REST APIs*. DevOps teams favor the flexibility and programmability of public clouds like Amazon Web Services (AWS). REST APIs enable application developers to build in services and policies, such as autoscaling thresholds, during application development. These APIs also allow developers and testers to understand how their applications will perform in production by provisioning enterprise class load balancers together with rich analytics in development and test environments. This results in faster application rollouts by enabling IT to automate repetitive tasks and set self-service policies to empower application teams and drive outcomes.



REMEMBER

The benefits of a modern application services platform include:

- » **Improving responsiveness to business units:** Application-centric enterprises rely on rapid application delivery to achieve revenue targets and maintain competitiveness. Many lines of business now leverage DevOps and continuous integration/

continuous delivery (CI/CD) development processes for quick prototyping and deployments of application updates. In these fast-paced operating environments, the networking team can't be a bottleneck taking weeks to provision new application services (such as virtual services for load balancing) or to debug network issues.

- » **Supporting new application architectures:** Application teams in many enterprises are developing disaggregated applications using microservices architectures. In the compute layer, containers and container orchestration platforms such as Kubernetes and OpenShift are enabling microservices-based development. Supporting these applications without automating as much of the service chain as possible is challenging, if not impossible.
- » **Eliminating manual configuration errors:** Adding the wrong pool member to the wrong virtual IP (VIP), taking an application server out of service inadvertently, or configuring incorrect server certificates are examples of potentially costly errors that can be caused by manual configuration processes. These tasks can be easily automated, mitigating security risks due to misconfigurations and saving significant time lost to troubleshooting and debugging later issues that arise, particularly in complex application environments.
- » **Ensuring consistency across applications:** Automating configuration tasks is the easiest way to ensure consistency across applications. It enables network engineers to turn infrastructure configurations into code, apply version control to the configurations, and ensure that system configurations are always stamped out from a single point of truth.
- » **Enabling multi-cloud and hybrid cloud use cases:** Many cloud and network architects are effectively leveraging public cloud environments as an extension of their private cloud or on-premises data center. They deploy applications in their data center or private cloud, then automatically burst to the public cloud to take advantage of flexible capacity when they experience unexpected traffic spikes, then automatically scale back to their data center when traffic levels return to normal.
- » **Using skilled IT resources more strategically:** IT departments are always going to face shortages of skilled technicians to handle requests. By automating these tasks, IT leaders can engage their limited staff in more strategic value-added initiatives for the business.

- » Strengthening your security posture with a web application firewall
- » Stitching monitoring, control, and security together in an elastic service mesh

# Chapter 3

## Attaining Security Nirvana with Modern Application Services

In this chapter, you learn about using a web application firewall (WAF) to push the security perimeter to each web application with point-and-click simplicity and how an elastic service mesh enables monitoring, control, and security in container-based microservices applications.

### Using a WAF to Inspect and Protect the Web Application Stack

Web applications are the backbones of many businesses today and unfortunately, they're often soft targets for debilitating cyberattacks. Modern enterprises that rely on web applications to access, collect, process, and transmit sensitive data to execute business logic must fortify their application backbones to improve their security postures.



Like a network firewall, a web application firewall (WAF) protects online services from malicious Internet traffic by detecting and blocking threats that could degrade, compromise, or knock out online applications. However, traditional port-based network firewalls only examine traffic at the Network layer (Layer 3) and Transport layer (Layer 4) of the Open Systems Interconnection (OSI) model, including source and destination IP addresses and protocol information. WAFs inspect web traffic at the Application layer (Layer 7) to detect attacks that attempt to exploit common security flaws in web applications. See the following list:



- » **Application layer (Layer 7) denial-of-service (DoS):** Attackers use valid HTTP requests in typical URL data retrievals to create an HTTP flood.
- » **Buffer overflow:** A bug exploit that overwrites adjacent memory locations while writing data to a buffer.
- » **Cookie poisoning:** Cookies are modified to gain unauthorized information about the user for purposes such as identity theft.
- » **Cross-site scripting (XSS):** Client-side scripts are injected into web pages viewed by other users.
- » **Forced browsing:** Resources that aren't referenced by the application but are still accessible, such as directory listings, are accessed.
- » **Hidden field manipulation:** Hidden data fields are manipulated to alter data stored in those fields.
- » **Parameter tampering:** Parameters exchanged between a client and server are manipulated to change application data.
- » **SQL injection:** Malicious code is inserted or injected into a web entry field that allows attackers to compromise the application and underlying systems.
- » **Unvalidated input:** Attackers tamper with HTTP requests (including the Uniform Resource Locator [URL], headers, and form fields) to bypass the site's security mechanisms.
- » **Web scraping:** Valuable data is extracted from websites using scraping techniques.

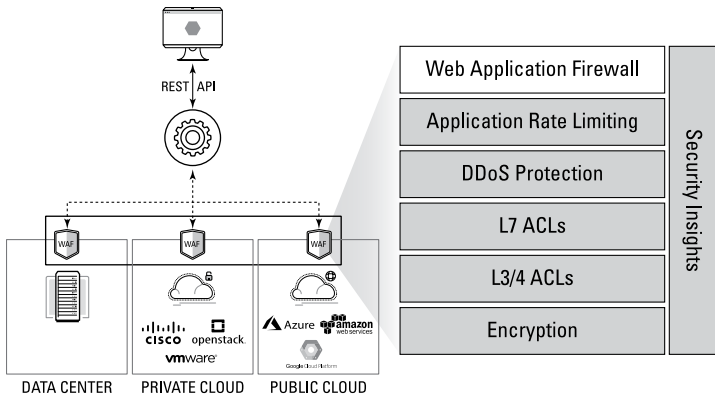
Despite the security benefits of WAFs, many enterprises tend to shy away from implementing appliance-based WAF solutions for three key reasons:

- » **Poor performance and scalability:** Traditional WAFs consume significant processor (CPU) and memory resources and don't provide the performance and scalability required for variable loads, often resulting in costly overprovisioning.
- » **Complex rule writing:** Creating rule sets and policies to implement OWASP best practices is complex, and tuning rules to optimize their performance and effectiveness is even more challenging.
- » **No visibility or intelligence:** Most traditional WAFs are a "black box." Once rule sets are defined, it's difficult to monitor their effectiveness and nearly impossible to react to changes or new security threats in real-time.

To address these challenges, the WAF needs to be reimagined for today's multi-cloud environments. Enterprises need an intelligent software-defined WAF that leverages machine learning and the strategic network location of the WAF to gain real-time application security insights. An intelligent WAF provides application security teams with an elastic and analytics-driven solution that scales across on-premises data centers and private and public cloud environments with capabilities that include (see Figure 3-1):

- » **Web application firewall:** OWASP ModSecurity Core Rule Set (CRS) protection and attack analytics
- » **Application rate limiting:** Control and restrict by application or tenants
- » **Distributed denial-of-service (DDoS) attack protection:** DDoS detection and mitigation with elastic scaling
- » **Layer 7 access control lists (ACLs):** Content-based security rules
- » **Layer 3 and 4 ACLs:** IP-port based security rules
- » **Encryption:** Transport Layer Security (TLS) – discussed in Chapter 4

- » **Centralized point-and-click management:** Simplifies policy customization and administration with a representational state transfer (REST) application programming interface (API) that enables a multi-cloud elastic fabric, automation and programmability, and real-time visibility and analytics



**FIGURE 3-1:** Key capabilities of an “intelligent” multi-cloud WAF.



TIP

Benefits of an intelligent WAF include:

- » On-demand elasticity based on an automatic scale-out architecture to address evolving security threats
- » Real-time visibility into application performance and end-user experience
- » Granular security insights on traffic flows and rule matches to create concise policies and minimize false positives

## Creating an Elastic Service Mesh for a Microservices Architecture

Common services — such as load balancing, network performance monitoring, and security — for legacy monolithic applications are often delivered via separate physical or virtual appliances that are deployed in line between the applications and clients (north-south traffic). However, these same services often need to be implemented or approached differently with container-based

microservices applications (east-west traffic). Some challenges in providing application services for container-based microservices applications include:

- » **Granularity:** In container-based microservices applications, a single service is often represented at the infrastructure level by multiple containers residing on multiple hosts (servers). From a networking standpoint, each of these containers is a unique application endpoint that requires the same load balancing and traffic management services as conventional applications. However, this granularity and its associated east-west traffic (communication between microservices) means that traditional appliance-based load balancers don't work at all since they can't be decomposed down to the level of individual services/containers.
- » **Visibility and security:** Application visibility is especially important in the context of container-based applications. Application developers and operations teams alike need to be able to view the interactions between services to identify erroneous interactions, security violations, and potential latencies.
- » **Performance and complexity:** Disaggregation of services into multiple microservices causes many more dynamic network endpoints, which increases complexity and makes performance monitoring more challenging.
- » **Elasticity:** Application service elasticity is even more important in the context of ephemeral container-based applications, given that containers can be spun up or taken down at a much faster pace than traditional infrastructure. Applications running on containers are able to take advantage of this agility in the compute layer, but network services still remain a bottleneck with application services not designed for cloud-native use cases.
- » **Automation:** Application-centric enterprises are looking to automate repetitive networking operations and remove error-prone manual steps. They want to achieve agility through continuous integration and continuous delivery (CI/CD) practices for rapid application deployments. Application networking services need to be API-driven and programmable without the constraints of hardware appliances that limit their flexibility and portability across environments.

An elastic service mesh is a new paradigm that addresses these challenges by delivering a layer of integrated application services within the compute cluster itself. A service mesh provides monitoring, scalability, and high availability application services through APIs instead of requiring discrete appliances. This flexible framework removes the operational complexity associated with application service for modern applications.



TECHNICAL  
STUFF

An elastic service mesh provides an array of network proxies alongside containers and each proxy functions as a gateway for any interactions between containers and between servers. The proxy accepts the connection and spreads the load across the service mesh. A central controller orchestrates the connections. While the service traffic flows directly between proxies, the control plane knows about each interaction and tells the proxies to implement appropriate access controls and collects performance metrics.

Service proxies within a service mesh can be deployed in several ways, including (see Figure 3-2):

- » **Per node:** Every node in the cluster has its own service proxy. Application instances on the node always access the local service proxy.
- » **Per application:** Every application has its own service proxy. Application instances access their own service proxy.
- » **Per application instance:** Every application instance has its own “sidecar” proxy.

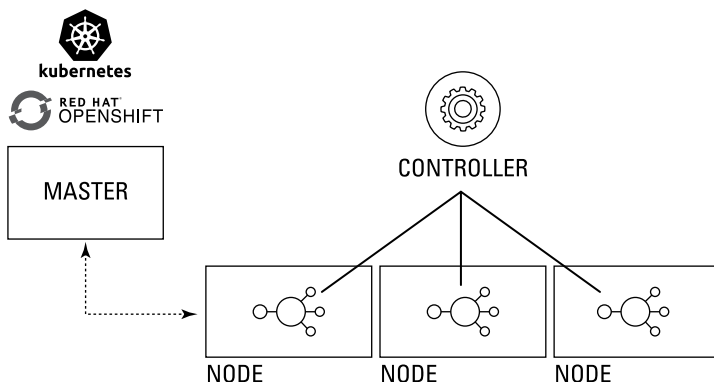


FIGURE 3-2: An elastic service mesh architecture.

With an elastic service mesh, smaller companies can create application features that only larger companies could afford under the traditional model of using customized code and reconfiguring every server. Other benefits of an elastic service mesh include:

- » Faster development, testing, and deployment of applications
- » More efficient and quicker application updates
- » More freedom to create truly innovative apps with container-based environments

## IN THIS CHAPTER

- » Taking a fresh look at load balancer refresh
- » Demystifying cloud load balancing
- » Enabling load balancing in multiple locations with global server load balancing (GSLB)
- » Balancing the need for speed and security
- » Evolving to a modern, software-defined data center

# Chapter 4

## Exploring Multi-Cloud Load Balancing Use Cases

In this chapter, you learn how to load balance across multiple locations with global server load balancing (GSLB), take a look at some multi-cloud load balancing use cases, and read a few real-world customer success stories.

### A Refreshing Approach to Load Balancer Refresh

Modern data centers use web-scale technologies to optimize and automate compute and networking infrastructure. These environments use standard x86 servers for compute, centrally manage infrastructure as a fluid collection of resources, and enable seamless scaling by adding compute resources dynamically. However,

load balancing and application services are lagging behind (see Chapter 1).

Traditional application delivery controller (ADC) vendors barely address the needs of modern applications and cloud-native use cases. Instead, they force you down the path of expensive and inflexible hardware refreshes or clumsy low performance virtual appliances, without addressing the fundamental challenges of elasticity, automation, multi-cloud use, and cost:

- » No central management — inefficient operations with each device managed separately
- » Not architected for cloud-native applications with lots of east-west traffic
- » Proprietary hardware leads to expensive overprovisioning without elastic scalability
- » Unable to address per-application or per-service load balancing needs
- » Don't offer any visibility to the application or network to help resolve issues
- » Can't scale up or down in response to traffic and without manual intervention
- » Lack of consistent architecture for multi-cloud and hybrid-cloud use cases



TIP

Enterprises looking to refresh their legacy hardware-based or virtual load balancing appliances now have an opportunity to get an enterprise-grade, software-defined load balancer with better capabilities at significant cost savings.

## Clearing the Air about Load Balancing in the Cloud

Traditional load balancers were built for use in data centers prior to the rise of public and private clouds. As virtualization and public and private clouds emerged, ADC vendors began to force fit their products for these environments with virtual editions of their appliances. Although these virtual versions are offered as software that can run on virtual machines, they inherit the same disadvantages of the hardware appliances including disparate management, limited automation, and lack of elasticity.



# EBSCO MODERNIZES ITS LOAD BALANCERS TO ENABLE A MULTI-CLOUD APPLICATION STRATEGY

EBSCO's high-performance environment demands speed and agility to deliver services to a global customer base that submits an average of 4.5 million transactions per day without overprovisioning server capacity.

## CHALLENGES

EBSCO searched for a modern load balancing solution to solve challenges in:

- **Application rollouts:** OpenStack helped with faster provisioning of the compute layer, but development teams building applications had to wait for configuration changes and new virtual services to be provisioned.
- **Isolation:** Multiple development teams need their own isolated slices of load balancing resources to enable continuous innovation.
- **Seamless integration:** EBSCO needed high performance load balancing solutions with seamless integration for OpenStack components such as Horizon, Nova, Keystone, and Neutron.

## SOLUTION

EBSCO chose Avi Vantage to deliver key capabilities including:

- **Self-service and automation:** Avi Vantage simplifies provisioning of application resources with a single point of control for distributed load balancing resources and native integration with OpenStack components
- **Analytics and insights:** Deep insights provided by Avi Vantage enable developers to define predictive autoscaling requirements to mirror traffic needs, and reduce troubleshooting time



REMEMBER

Virtual load balancers offer poorer performance than their hardware-based versions and are generally not recommended by ADC vendors — except when their customers need a solution for the cloud.

Unfortunately, these solutions are often cost prohibitive to run in the public cloud and don't offer centralized management or enable hybrid cloud operations. Enterprises thus often end up with completely different application delivery solutions for their data center and cloud applications.



WARNING

Traditional appliance-based load balancers and their virtual editions don't offer the automation and elasticity necessary to meet SLAs in the cloud. They're also difficult to set up and maintain. And, while open source tools appear less expensive, they're not feature-complete, and setup and support are very do it yourself, which requires a deep bench of engineers.

## Going Global with Server Load Balancing

Global server load balancing (GSLB) balances an application's load across multiple instances of the application that have been deployed to several locations (typically, multiple data centers and/or public clouds). Application load at any one of those locations is usually managed by a local load balancer. GSLB is usually implemented in one or more of the following application use cases:

- » Provide optimal application experience for geographically distributed users
- » Ensure resilience in the event of a data center or network failure
- » Enable a hybrid cloud with cloud bursting to the public cloud to absorb peak loads

To achieve these goals, GSLB performs the following functions:

1. It chooses the location (data center/cloud) to which to steer the client's requests

2. It monitors the health of virtual services so that it can choose the best location and rule out services in unhealthy locations
3. It synchronizes configuration and state information across GSLB sites, so that the first two functions can continue despite certain failures

GSLB uses the Domain Name System (DNS) to provide the optimal destination information to user clients. When a client (typically a browser) performs a DNS query for a fully qualified domain name (FQDN), GSLB responds with the virtual IP address (VIP) of the optimal application instance. The optimal address changes based on the load balancing algorithm, current health of the application instances, and the location of the client.

## SSL Is So SSLow on Your Legacy Load Balancer

Although increased attention has been focused on application and web security recently, many websites still have weak implementations of Secure Socket Layer (SSL)/Transport Layer Security (TLS). Reasons for the dearth of stronger SSL implementations include:

- » Lack of infrastructure and browser support
- » Performance penalty
- » Implementation complexity

However, with recent advances in the SSL protocol, as well as significant performance improvements of SSL on commodity x86 platforms, stronger SSL can be — and should be — implemented everywhere.



TECHNICAL  
STUFF

Technically speaking, SSL has been deprecated and should no longer be used. As of August 2018, TLS — which is still commonly referred to as SSL — version 1.3 is the most current version of SSL/TLS.

Legacy hardware load balancers can't scale elastically and are capped at speeds that are punitively tied to licensing and

acquisition costs. Additionally, many legacy hardware load balancers can't support advanced SSL features such as:

- » **Server Name Indication (SNI):** Virtual hosting with SSL is a chicken-and-egg problem. The client sends an SSL hello, and the server must send back the SSL public key. If there are multiple domain names attached to the same IP address, a client that supports SNI sends the hello along with the requested domain name. The server can now send back the proper SSL response. Other workarounds have existed for this challenge, such as wildcard certificates, but all have had limitations. However, SNI is now prevalent enough among client browsers to safely enable it on servers.
- » **HTTP Strict Transport Security (HSTS):** This feature indicates to client browsers that they should only access a particular site via HTTPS, not HTTP. This is used to mitigate man-in-the-middle attacks, particularly from open Wi-Fi connections. HSTS is sent to client browsers via an HTTP header, along with a length of time HSTS should be considered true.
- » **Elliptic Curve Cryptography (ECC):** Historically, SSL/TLS has been performed using Rivest-Shamir-Adleman (RSA) certificates. Elliptic Curve Cryptography (ECC) certificates are based on newer algorithms — which are significantly more secure, less computationally expensive, and now supported by every current browser. Most hardware-based crypto accelerators don't support ECC, though, so legacy environments may be required to upgrade before utilizing the newer certificates.
- » **Perfect Forward Secrecy (PFS):** SSL ensures that communication between a client and server is secure and encrypted. But if a session is recorded, and an attacker acquires the server certificate after the fact, the attacker can replay and decrypt recorded sessions. With PFS, a unique key is used to generate a unique, one-time-use key for each session. After the session, the ephemeral key is discarded, ensuring that a recorded session can't be decrypted later. PFS has been around for a while, but using legacy RSA certificates imposes a heavy computational cost. Adoption of the newer ECC certificates allows PFS at near zero additional CPU cost.

- » **SPDY:** SPDY (pronounced “Speedy”) is now a deprecated open-specification networking protocol that was developed to improve the performance of HTTP by reducing web page latency. Although SPDY was primarily intended to improve performance, it required SSL to negotiate over HTTP/1.1.
- » **HTTP/2:** HTTP/2 is a technical extension of SPDY, with a few key differences. One important difference is that HTTP/2 doesn't require SSL encryption. However, many of the browser vendors are unsatisfied with this decision, so browsers such as Chrome and Firefox will only negotiate HTTP/2 if the site is encrypted, otherwise they will downgrade to HTTP/1.1. This will lead to a fair bit of confusion over whether SSL is required for HTTP/2 or not. To ensure that all clients can take advantage of the performance improvements of HTTP, best practice is to encrypt all HTTP communications.

Historically, SSL TPS (transactions per second) were a critical bottleneck for the performance and sizing of a load balancer, and required custom hardware for crypto acceleration. However, with new crypto algorithms, advances in x86 CPU performance, and next-generation software architectures, SSL performance on commodity x86 servers is no longer an issue and a single server can support up to 10 gigabits per second (Gbps) of SSL throughput (see Table 4-1).

**TABLE 4-1**

### Commodity x86 platforms (2x8-core CPUs) provide better performance for ECC certificates than for RSA certificates

Certificate	SSL transactions per second (TPS)
RSA 2K	8,000
ECC	32,000



TECHNICAL  
STUFF

The Avi Networks ADC leverages instructions within current-generation CPUs for Advanced Encryption Standard New Instructions (AES-NI). AES-NI provides better performance using instruction sets native in the CPUs. Following Moore's Law, with dramatic increases in CPU capabilities, new servers introduced into a server farm will continue to dramatically improve SSL TPS.

# MULTINATIONAL SECURITY PROVIDER SCALES OUT CYBERSECURITY SOLUTIONS

One of the largest antivirus software and Internet security services vendors recently chose Avi Networks to drive its data center innovation.

## CHALLENGES

The company's next-generation flagship security solutions needed more than 100,000 connections per second to access backend representational state transfer (REST) APIs. The challenges included:

- **Cost of expanded load balancer footprint:** The company's data center transformation initiative meant that the team needed to consider Layer 4 (L4) to Layer 7 (L7) services across more applications in the data center.
- **Scaling wide versus scaling up:** To support the over 100,000 SSL connections per second that new applications generated, the company needed a solution that could scale out across the environment.
- **Load balancing a large volume of SSL/TLS transactions:** When the team implemented SSL/TLS transactions on its existing appliance-based load balancers, they experienced a 10x drop in performance.

## SOLUTION

The team chose the Avi Vantage Platform for its software-defined, application-centric load balancing capabilities.

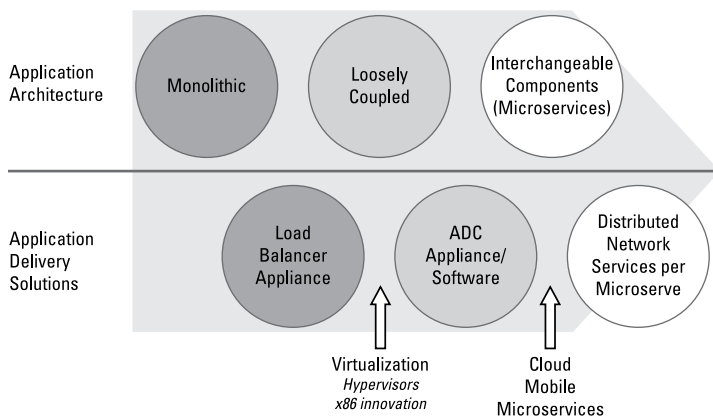
Key capabilities include:

- **Simplicity and ease of use:** With the Avi Vantage Platform, the team is able to build custom data scripts with point-and-click simplicity.

- **Scale out:** Avi load balancers can be spun up seamlessly in a matter of minutes and scaled out horizontally to support throughput requirements of approximately 100,000 SSL connections per second.
- **Analytics:** Granular visibility into round-trip time for each hop in a transaction helps the team pinpoint rogue transactions without combing through logs, packet dumps, and traces to resolve an issue.

## Putting Microservices under a Microscope

Applications have evolved from client-server based to service-oriented architecture (SOA) based to a microservices-based architecture used in most modern web apps today (see Figure 4-1). This evolution has greatly impacted how applications are created, scaled, secured, and managed. Unfortunately, most ADCs in the industry are still lagging. An application that previously needed a single service now requires ten or more different services. Unless modern ADCs are “microservices aware” and have the right hooks (via APIs, automation, and orchestration frameworks) to provision, configure, and manage each microservice, IT teams will struggle to keep up with the lifecycle management of applications.



**FIGURE 4-1:** Evolution of application architectures versus application delivery systems.

Benefits of using microservices architecture include:

- » **Each microservice is relatively small.** The code is easier for developers to understand. The small code base doesn't slow down the IDE, and each service typically starts a lot faster than a large monolith. Overall, this allows developers to be more productive and speeds up deployments.
- » **Each service can be deployed independently of other services.** If developers responsible for a service need to release local changes, they can deploy their changes without coordinating with other developers. A microservices architecture makes continuous deployment feasible and attractive, and REST offers a lightweight mechanism for communicating between services.
- » **Each service can scale independently.** With monolithic architecture, components with wildly different resource requirements (for example, large CPU versus large memory requirements) must be deployed together. In contrast, microservices architecture deploys each service on hardware that is best suited to its resource requirements, and scales each service independently of other services.

Evolving application delivery for microservices architecture in a modern data center is as easy as 1, 2, 3 . . . 4:

- » **Step 1:** In terms of architecture, the control and data planes need to be separated within the ADC. Data plane resources need to be dynamically distributed across different hardware platforms and public/private clouds.
- » **Step 2:** ADCs must achieve the concept of *application affinity*, which is based on the app world concept of *processor affinity*, where resources are aligned/pinned for specific functions. There are two major benefits to this approach. First, with ADC resources side-by-side, microservices improve the application response time. Second, this tight alignment (affinity) enables ADC resources to achieve automatic lifecycle management of microservices without manual intervention, significantly reducing management complexity.



- » **Step 3:** Achieving data plane independence (isolation) enables multitenancy, especially in cloud environments. Multitenant features enable microservices to operate and change independently without disrupting microservices. This is also called the no-noisy-neighbor impact.
- » **Step 4:** ADCs must fulfill the self-service programmability and efficiency promises of SDN. Most, if not all, ADC vendors today support REST, the protocol of choice in hyperscale web services. However, ADCs can only achieve SDN promises and enable one-to-one communication between applications' controller and control elements through RESTful APIs when control and data planes are separated.



REMEMBER

Application architecture has evolved from purpose-built, monolithic (“shrink wrapped”) code and products to a tightly federated collection of microservices that are both modular and reusable. It’s as if app developers began using a common set of building blocks to build any number of web-based apps, limited only by their imagination. For networking teams, the move to microservice-app development means that existing assumptions around traffic patterns, load balancing scale, and service requirements are no longer valid. Instead, what’s needed is a greater level of network-wide intelligence and a new application services architecture that mirrors microservice apps.

- » Leveraging automation
- » Keeping web applications secure
- » Supporting robust networking for microservices applications

# Chapter 5

## Ten Things You Can Do with Multi-Cloud Load Balancing

A modern software-defined load balancer platform provides multi-cloud, intelligence (visibility and analytics), and automation capabilities to enable intent-based application services that go far beyond load balancing alone. With intent-based application services, declarative (intent-based) models replace imperative (input-based) models so you can focus on outcomes. You specify your intent and the infrastructure does the heavy lifting using automation and analytics. Here are ten innovative things you can do now with intent-based application services including multi-cloud load balancers:

- » **Elastic load balancing:** A software-defined, scale-out architecture provides on-demand autoscaling of elastic load balancers. The distributed software load balancers and the backend applications can scale up or down in response to real-time traffic monitoring. Application load balancing becomes more adaptable and intelligent. Key benefits include:
  - No more overprovisioning of appliance-based (virtual or hardware) load balancers

- Elastic scale based on learned traffic thresholds
- Per-app, elastic load balancing to maintain SLAs

» **Automation for IT, self-service for developers:** Match the continuous application delivery goals and DevOps principles of modern enterprises by automating routine tasks such as virtual IP (VIP) provisioning and configuration changes. Important capabilities include:

- Application service automation through 100 percent representational state transfer (REST) application programming interfaces (APIs)
- Visibility with application maps to monitor and troubleshoot applications
- Policy-driven self-service for modern web application developers

» **Customizable centralized security management:** Administrators and security analysts are equipped with point-and-click tools to enforce the right security policies through central management of all distributed web application firewall (WAF) instances. Important capabilities include:

- IP protection: GeolP blocking
- Data leakage protection: Error message suppression, leakage of personally identifiable information such as credit card or Social Security numbers
- Input protection: SQL injection, cross-site scripting, local/remote file inclusion, remote code execution, hypertext preprocessor code injection, path traversal, and session fixation
- HTTP validation: Limit HTTP allow method, encoding bypass detection, HTTP response splitting, HTTP parameter pollution

» **Multi-cloud elastic security fabric:** Protect public-facing and internal web applications and APIs with a high-performance solution that addresses the most persistent WAF challenges for enterprises. Important capabilities include:

- Per-app deployments and elastic scaling across data centers and multi-cloud environments
- Distributed web security fabric enforces security through closed-loop intelligence

- » **Achieving compliance:** Web application attacks are on the rise. WAF helps organizations achieve General Data Protection Regulation (GDPR), Health Insurance Portability and Accountability Act (HIPAA), and Payment Card Industry (PCI) compliance with a scalable and distributed software fabric.
- » **Set-and-forget security policy:** Granular security insights on traffic flows and rule matches enable precise policies and one-click customization of rules and exceptions to sharply reduce the problem of false-positive fatigue. Important capabilities include:
  - Automated attack blocking, such as scanner detection and brute force attacks
  - Zero-day attack protection, such as Shellshock and HTTPoxy
  - Application-specific security, for example, Drupal and WordPress
- » **Real-time visibility and security analytics:** Granular, actionable security details from the distributed service proxy fabric to secure web applications in real-time allow visual policy checks prior to enforcement and comprehensive web application security testing. Important capabilities include:
  - Visible security details including information about Secure Sockets Layer/Transport Layer Security (SSL/TLS) versions and ciphers used in transactions, real-time distributed denial-of-service (DDoS) attack data, and system health scores
  - Protection from SQL injections, cross-site scripting, and other threats as identified by Open Web Application Security Project guidelines
- » **Global and local traffic management:** With an array of network proxies on each node in the container cluster, each proxy serves as a gateway between containers and servers. The interactions are visualized in an application map. Enterprise-class application services for containerized applications, orchestrated by platforms like OpenShift and Kubernetes, include:
  - Load balancing, health monitoring, TLS/SSL offload, session persistence, content/Uniform Resource Locator (URL) switching, and content modification

- Directing requests to the appropriate site/region based on the availability, locality of the user to the site, site persistence, and load
- Content/URL switching, redirection, error page, caching, compression, and on-demand autoscaling

» **Dynamic service discovery:** Service discovery bridges the gap between a service's name and access information (IP address) by providing dynamic mapping. An IBAS platform provides an authoritative domain name system (DNS) server for users' devices and other services to map host/domain names to virtual IP addresses (VIPs), including:

- Built-in IP address management (IPAM) for virtual IP address allocation
- A variety of DNS configuration options and the ability to add static address mapping (A) and canonical name (CNAME) records to the DNS server
- Continuous integration and delivery (CI/CD) and application upgrades using Blue-Green or canary deployment models

» **Application monitoring and analytics:** Analytics is the foundation for intent-based systems to collect, aggregate, accumulate, store, and rollup metrics and logs, especially for containerized applications based on microservices architecture. A modern application services platform provides the following features without having to instrument each application:

- Application map: Real-time dynamic map of communications between microservices available as a dependency map
- Analytics dashboard: An end-to-end latency view of all transactions in addition to real-time and historic views of critical metrics
- Log file analytics: Logs of every significant transaction including errors and excessive latencies
- Client analytics: Aggregated page load times and dimensional analytics for every page in the application



Networks®

Now part  
of VMware

---

## MULTI-CLOUD APPLICATION SERVICES

**Software Load Balancer**  
**Intelligent WAF**  
**Container Ingress Services**



**Across Any Cloud**

**x86 / VM / Container / Public Cloud**



**90% Faster Provisioning**

**Built-in Analytics and Automation**

Legacy

Avi

50%  
SAVINGS

**50% Lower TCO**

These materials are © 2019 John Wiley & Sons, Ltd. Any dissemination, distribution, or unauthorized use is strictly prohibited.

**On-demand Auto Scaling**

# Deliver software-defined application services

It's time to reinvent the load balancing market. Legacy hardware-based load balancers don't meet modern enterprise application requirements in a multi-cloud world. A software-defined application services platform separates the data and control plane of a load balancer to deliver application services beyond load balancing, such as real-time application analytics, security and monitoring, predictive autoscaling, multi-cloud global load server balancing, and end-to-end automation. In this book, you'll learn how multi-cloud application services go far beyond load balancing alone.

## Inside...

- Enable elastic on-demand autoscaling
- Automate routine IT tasks with intent
- Empower developers with self-service
- Simplify advanced security management
- Gain real-time visibility and analytics
- Modernize microservices app delivery


vmware®

**Lawrence Miller** has worked in information technology in various industries for more than 25 years. **Ranga Rajagopalan** is CTO and co-founder of Avi Networks. He architected several high-performance distributed systems such as SAN-OS and led development of data center products including Cisco Nexus 7000.

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

ISBN: 978-1-119-54702-0  
Not For Resale

for  
**dummies**®  
A Wiley Brand

 Also available  
as an e-book



# **WILEY END USER LICENSE AGREEMENT**

Go to [www.wiley.com/go/eula](http://www.wiley.com/go/eula) to access Wiley's ebook EULA.