White Paper: **April 2022**

Go from 0-60 When Automating Multi-Cloud App Delivery with Load Balancers

Multi-cloud automation, self-service, security, and elastic scale

vmware®

Table of contents

Introduction

As more enterprises embrace storing data across multiple public and private clouds using infrastructure that includes bare-metal servers, VMs, and containers, it's clear that networking teams need to prioritize adopting continuous integration / continuous delivery (CI/CD) practices.

These application-centric enterprises need networking teams to automate and deliver self-service to lines of business. They simply cannot afford the operational inefficiencies and management challenges of fleets of legacy load balancing appliances.

They need intelligent automation and consistent networking services across multi-cloud environments that eliminate silos caused by load balancing appliances in the data center, virtual ADCs in the cloud, or open-source solutions for container networking. Appliance-based (physical or virtual) load balancers and most cloud load balancers do not offer real-time visibility into end-user experience or app performance, have no automated failure recovery capabilities, lack the ability and automation to scale up or down based on traffic patterns, and require extensive manual configuration and separate management for each instance. See Figure 1.

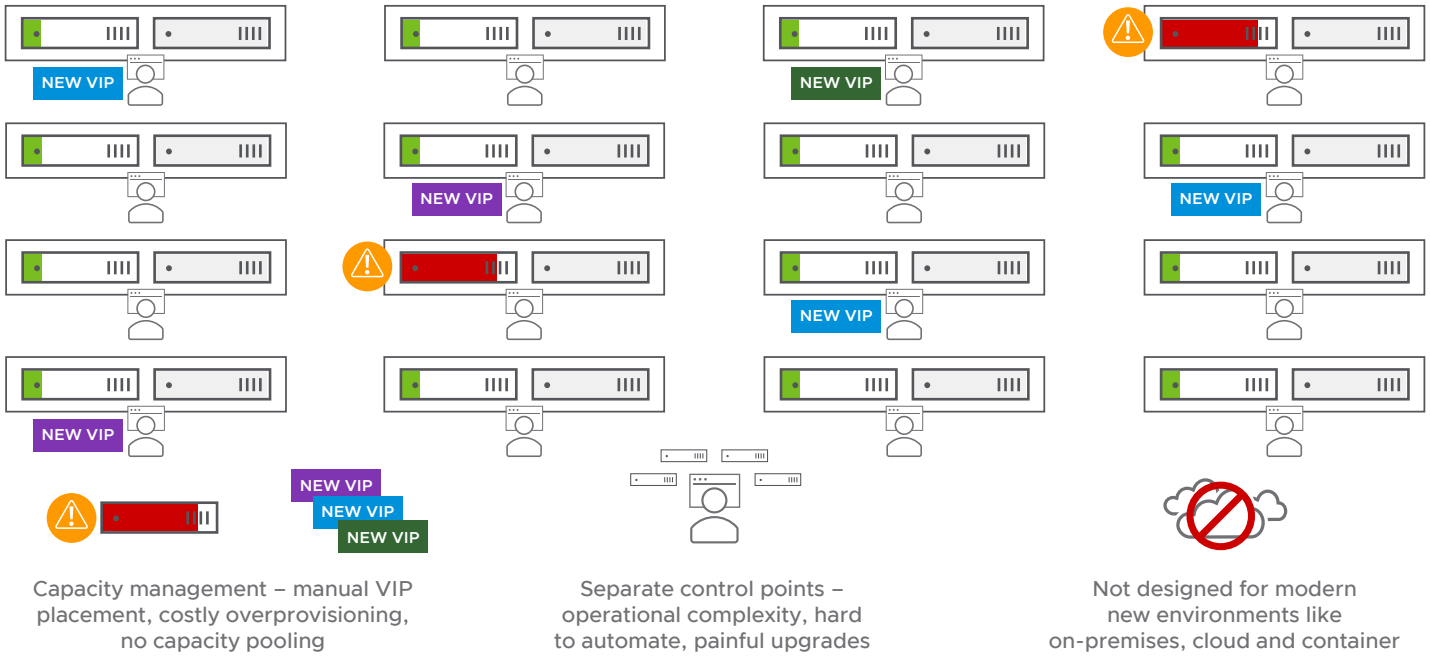


Figure 1: Legacy deployment creating new virtual service

API ≠ Decision Automation

All too often, vendors eager to jump on the automation bandwagon tell enterprises that their products are fully automated because they have REST API. However, there is a vast chasm between providing APIs, and delivering automated load balancing, WAF, GSLB, and container ingress networking. It comes down to writing scripts using the APIs versus letting the system automate mundane "decisions" using closed-loop intelligence about the environment and real-time conditions. Correctly implemented, such declarative "Decision Automation" simplifies operations, improves agility, and improves security.

Decision Automation lets administrators determine a desired outcome of a network implementation, then automates provisioning, configuration, and on-demand availability of services. Decision Automation covers the full lifecycle of the

application services including automated failure recovery. The plans and policies specified in a declarative model are automatically implemented, delivered, and enforced with the help of machine learning. This avoids laborious manual coding of scripts to cope with specific scenarios. Instead, the admin states the desired outcome (e.g., deploy new application), and the system then decides how best to meet it. Decision Automation also allows an IT generalist or even business owner to specify their intent, without having to rely on an IT specialist to manually provide input or code specialized scripts. See Figure 2.

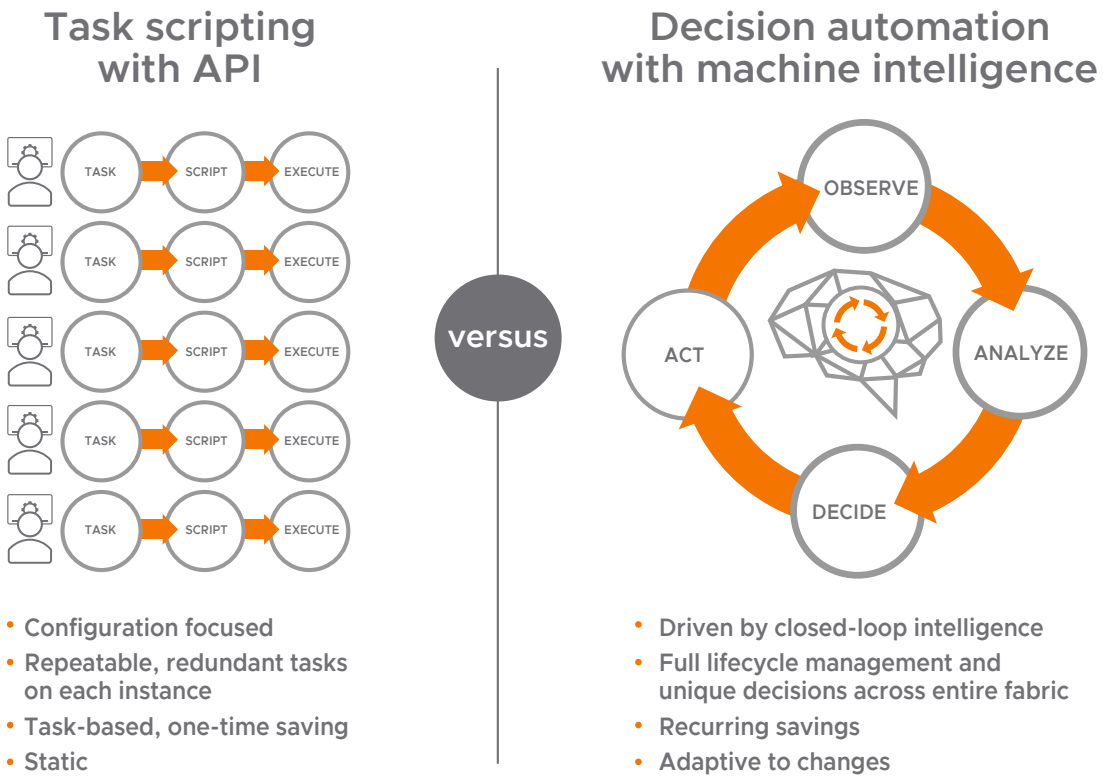


Figure 2: Task scripting vs Decision Automation

Decision Automation requires a fundamentally different architecture for applications services—one that is software-defined, agnostic to the underlying environment, and built on cloud-native principles. This architecture moves the focus to applications over infrastructure, software over hardware, and centralized policies over manual configurations. It allows for programmability but more importantly leads to better automated, system delivered outcomes and reduced operational costs.

VMware NSX Advanced Load Balancer and WAF (Avi Networks)

Avi uses a software-defined architecture that separates the central control plane (Avi Controller) from the distributed data plane (Avi Service Engines). Avi is 100% REST API based, so it can be fully automated and integrated into the CI/CD pipeline for application delivery.

The Avi Controller is the “brain” of the entire system. It acts as a single point of intelligence, management, and control for the distributed data plane. The Avi SE represent full-featured, enterprise-grade load balancers, web application firewall (WAF), and real-time analytics that provide traffic management and application security. The Avi Controller provides comprehensive observability based on closed-loop telemetry and presents actionable insights based on application monitoring, end-to-end timing, searchable traffic logs, security insights, log insights, client insights, and more.

Avi delivers Decision Automation, autoscaling, and full lifecycle management—all this is made possible through in-depth ecosystem integration, 100% REST API and, analytics fed to the Controller. The Avi Controller can natively talk to the APIs of private and public clouds such as vCenter, Microsoft Azure, Amazon AWS, Google Cloud Platform to deploy load balancers and WAF in an any of those environments. See Figure 3.

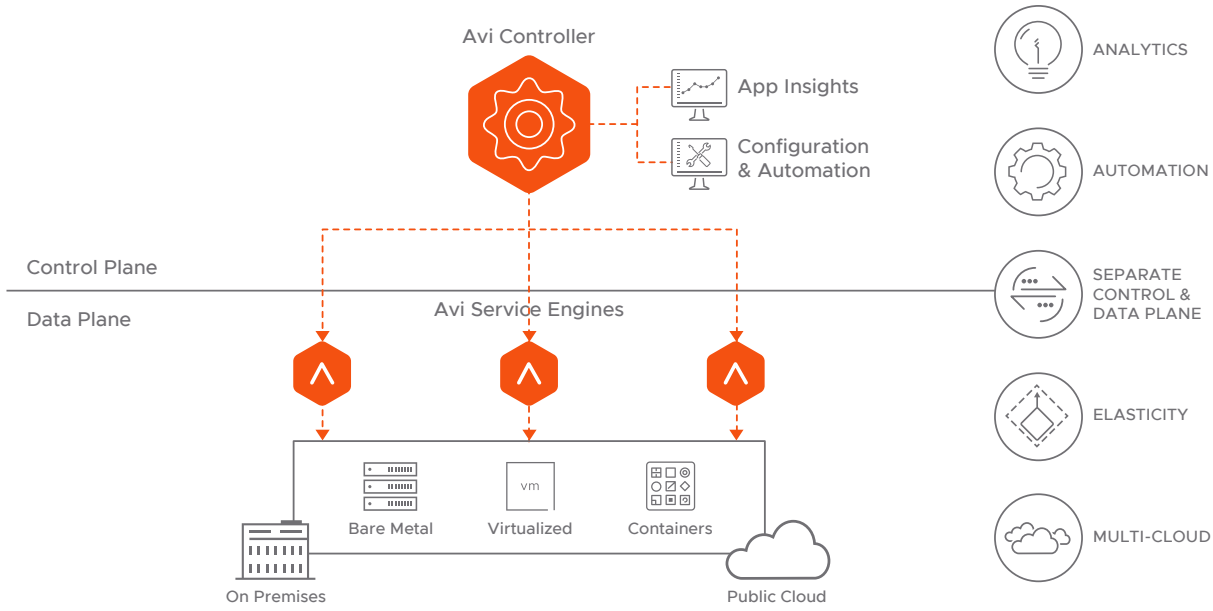


Figure 3: Avi High-level Architecture

Automated Application Delivery

Deploying applications and ensuring consistent end user experience requires decision-based automation across the entire lifecycle of an application. Starting with the initial deployment, the application can automatically scale capacity based on demand or automated failure recovery across on prem, multi cloud, and multi zone deployments. See Figure 4.

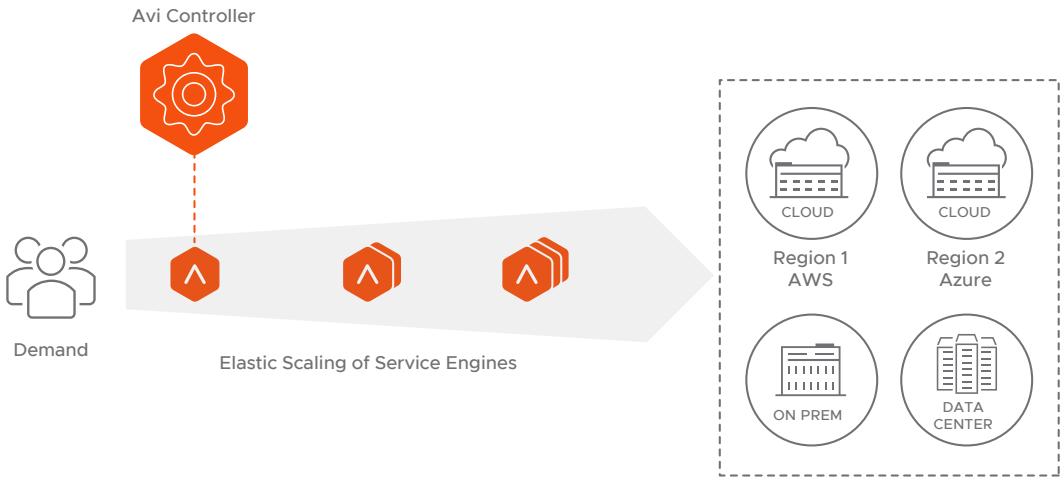


Figure 4: Avi elastic capacity

While monitoring real-time data from the Service Engine, the Avi Controller can:

- Create a new virtual service for a new application
- Recover from Service Engine failure
- Migrate a virtual service to an unused Service Engine
- Scale out the virtual service across multiple Service Engines across Multi Region and/or Multi Availability Zone deployments

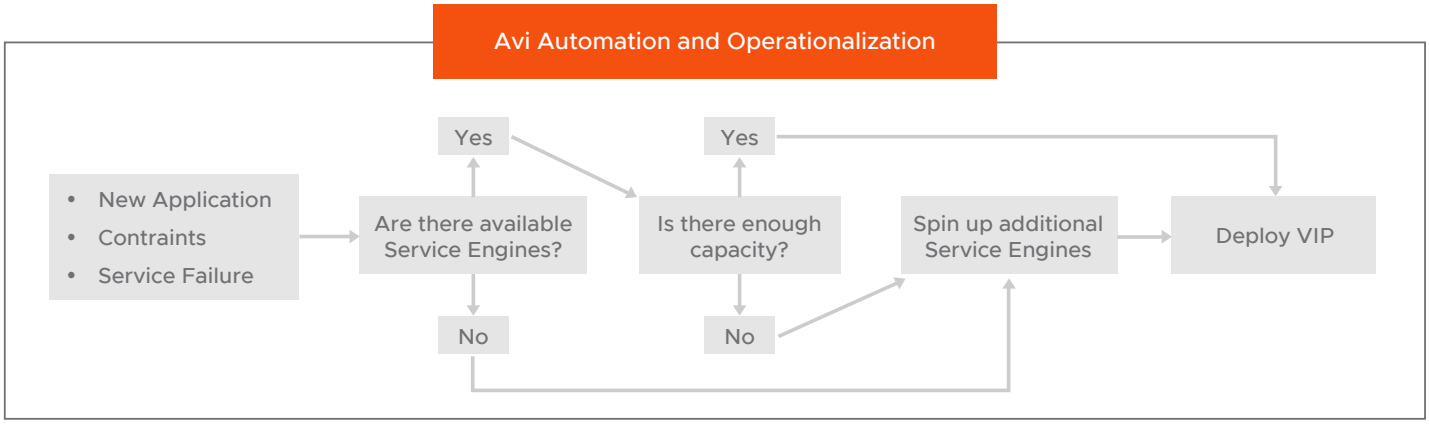


Figure 5: Avi Automation and Operationalization

Decision Automation for VIP creation with Avi

What happens when an administrator wants to deploy a new application?

With hardware-based legacy systems, the administrator must identify if a load balancer exists, if there is available capacity, if the environment variables are correct, and whether there any dependencies. Once all these questions have been answered, every load balancer, firewall policy, authentication, IP, and DNS setting has to be configured manually. See Figure 6.

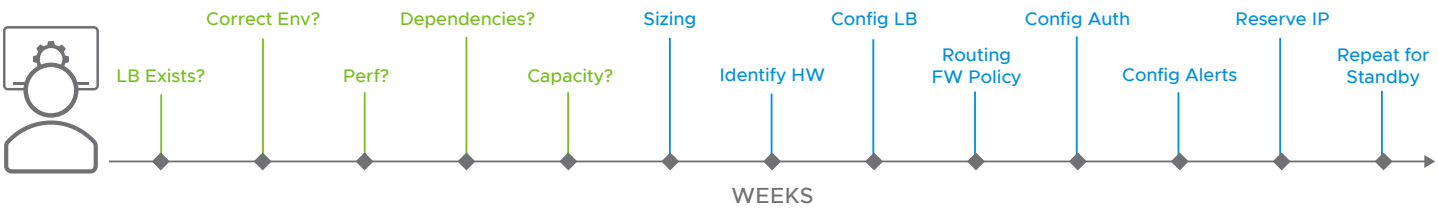


Figure 6: Legacy deployment creating new virtual service

In contrast, Avi will find the best location(s), the best host(s) to automatically deploy a new Service Engine, configure the Nics, virtual services, WAF, security policies, authentication and authorization, register the DNS, and automatically configure and pick up SSL certificates. See Figure 7.

Upon initial application deployment the Controller will determine if Service Engines are available and if the Service Engines have sufficient capacity available to service the new application. Based on these criteria the Controller will automatically configure a new virtual service for the application on existing Service Engines or spin up new Service Engines as needed. See Figure 8.

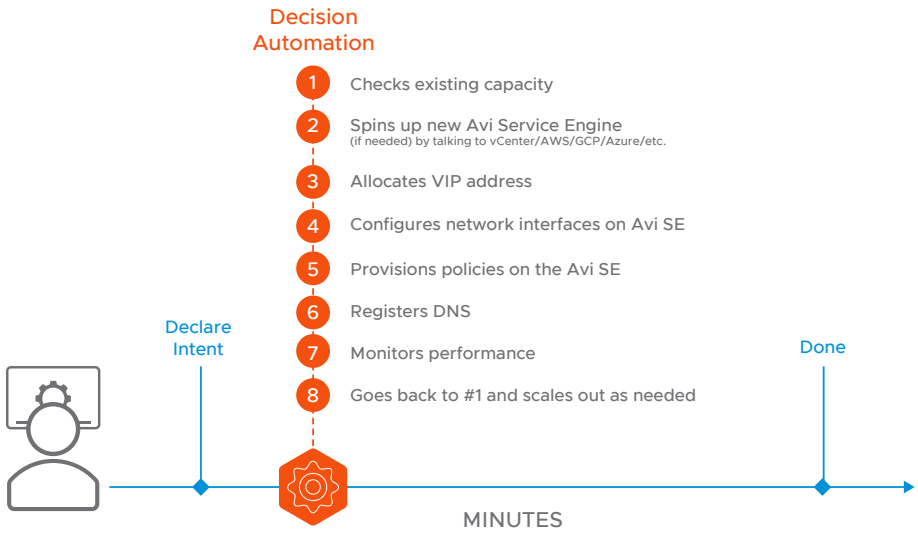


Figure 7: Avi automated deployment creating new virtual service

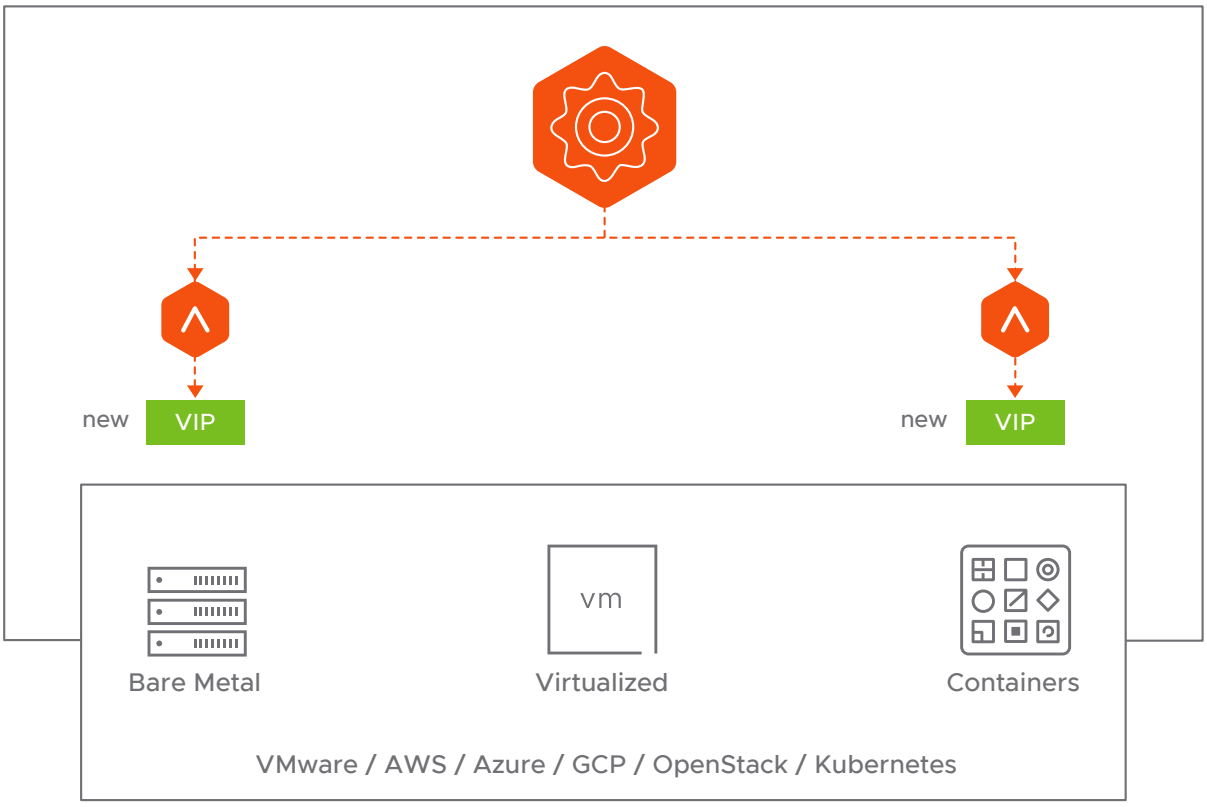


Figure 8: Avi New Virtual Service creation

Increasing capacity and rebalancing

While performing application delivery tasks, Service Engines can experience resource exhaustion. This may be due to a new application deployment, high CPU or memory utilization, or traffic patterns. Monitoring several application and network telemetry real-time data feeds from the Service Engine, the Avi Controller can automatically migrate a virtual service to an unused Service Engine or scale out the virtual service across multiple Service Engines to increase capacity. This allows multiple active Service Engines to concurrently share the workload of a single virtual service.

In addition, Avi learns application access patterns and can perform intelligent, predictive autoscaling based on the learned traffic patterns and application usage, making services highly available before demand causes any service exhaustion or disruption. See Figure 9.

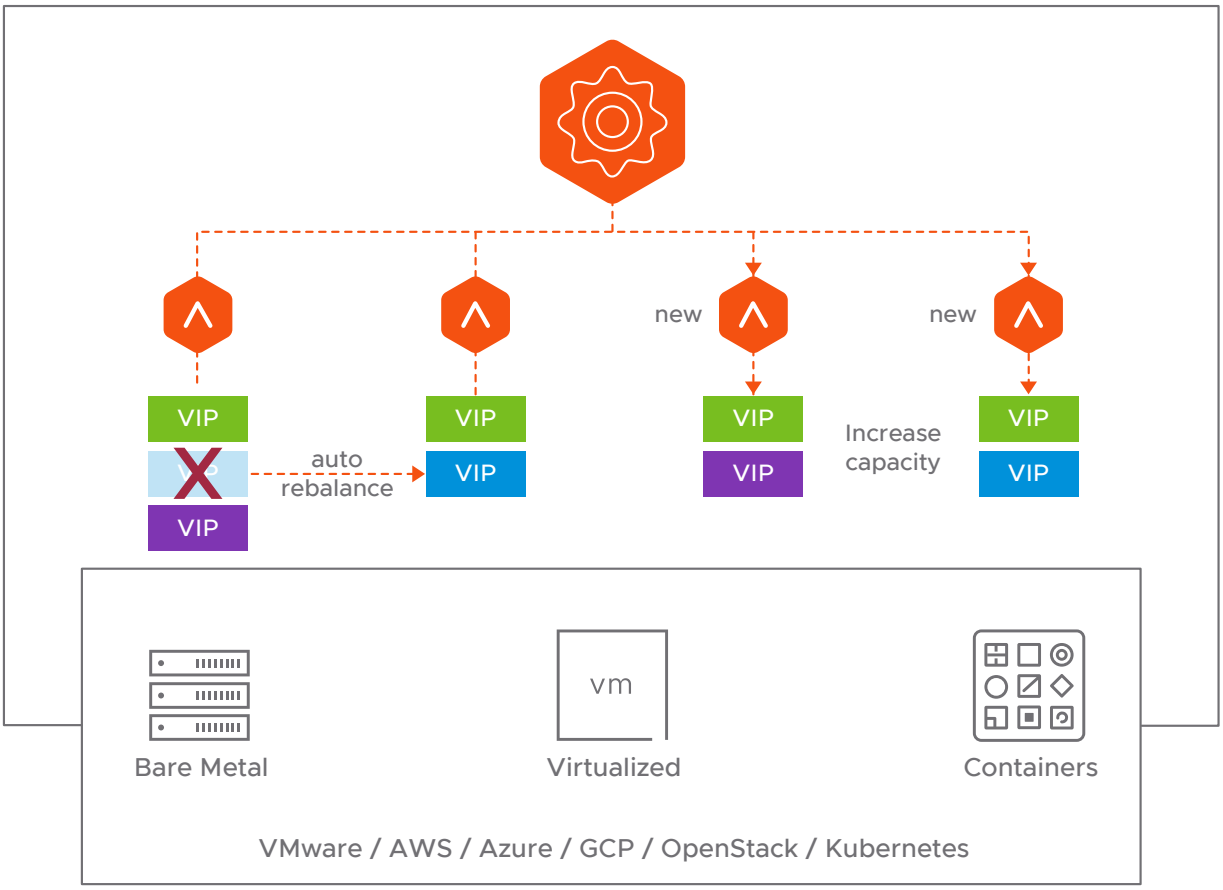


Figure 9: Avi Auto rebalance and capacity adjustment

Automated failure recovery

Automated failure remediation is essential in achieving consistent application availability. Avi relies on a variety of methods to detect Service Engine failures.

Controller ➔ Service Engine Failure Detection:

In all deployment modes, the Avi Controller sends heartbeat messages to all Service Engines in all groups under its control. If there is no response from a specific Service Engine for six consecutive heartbeat messages, the Controller concludes that the

Service Engine is down, and moves all virtual services to an available Service Engine with sufficient capacity or spins up a new Service Engine.

Service Engine ➔ Service Engine Failure Detection Method:

In the first example, the Controller detects a Service Engine failure by sending periodic heartbeat messages over the management interface. However, this method will not detect data path failures for the data interfaces on Service Engines. To ensure holistic failure detection, Service Engines send periodic heartbeat messages over the data interfaces and if the Controller concludes that a Service Engine is down it moves all virtual services to an available Service Engine with sufficient capacity or spins up a new Service Engine.

BGP Router ➔ Service Engine Failure Detection Method:

With BGP configured, the Service Engine to Service Engine failure detection is augmented by Bidirectional Forwarding Detection, which detects failures and prompts the router not to use the route to the failed Service Engine for flow load balancing and by using BGP protocol timers. See Figure 10.

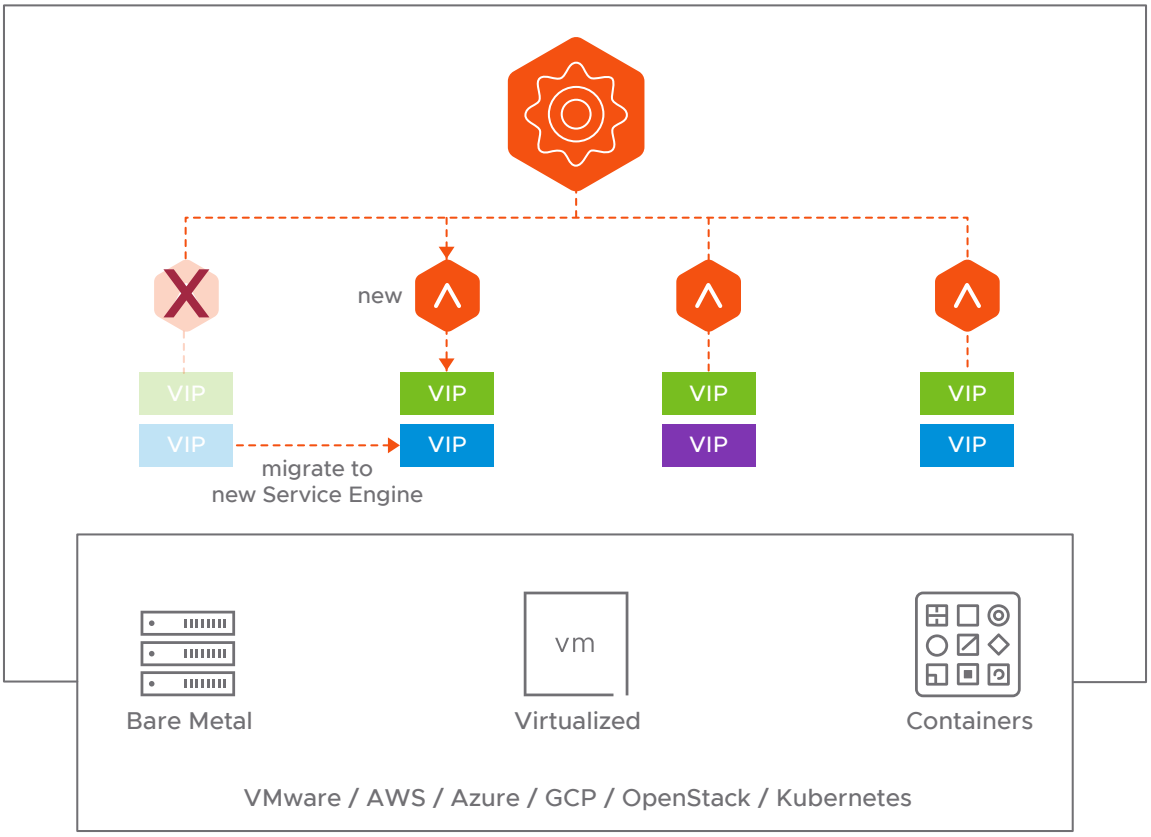


Figure 10: Avi automated failure recovery

Automated Load Distribution

With Avi, enterprises can move workloads across multiple clouds effortlessly. Avi enables enterprises to use AWS/GCP/Azure as a natural extension to their data centers by automatically overflowing to the cloud during traffic peaks. Avi can automatically create app resources in public clouds to absorb traffic surges and scale them back down.

For operational automation, Avi is natively integrated with the surrounding ecosystem to enable:

- Firewalls to automatically initiate
- IP addressing
- Automatic DNS configuration with Infoblox, Amazon RAW 53, etc., with some customization through REST APIs

See Figure 11.

Figure 11: Avi ecosystem integration

Automated Services to Manage Security, Support, and Anomalies

VMware Avi PULSE services provides an automated and central facility to manage and operate security intelligence and case management for globally distributed Avi Controller clusters run by customers.

Any Avi Controller in any cluster can optionally (based on customer consent) connect to Avi PULSE Services to retrieve threat intelligence updates including IP reputation, application signatures, WAF CRS rules and more. Avi PULSE Services automatically contact VMware support giving operations the choice to create a case manually from any Avi Controller or configure the Controller to automatically create a support case if a system failure is detected. See Figure 12.

These new services provided by Avi PULSE will reduce the complexity, management, and cost of your network operations.

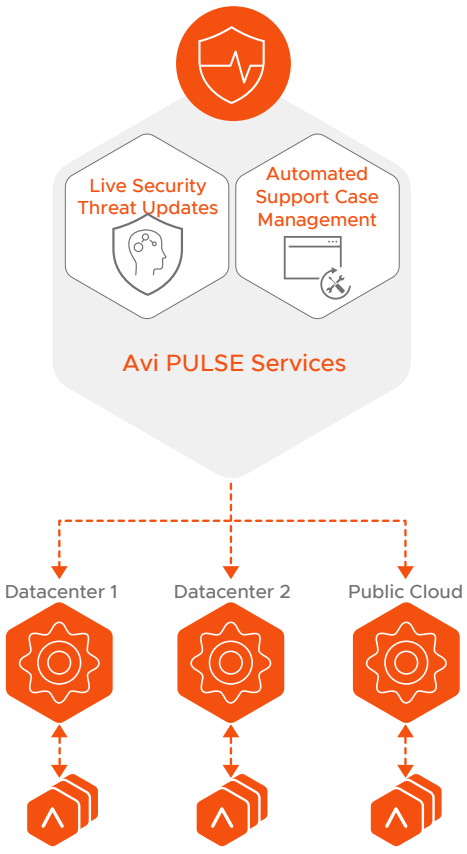


Figure 12: Avi PULSE Services

Automated Service Discovery

Service discovery is the process of automatically detecting devices and services on a network. Like Kubernetes service discovery, it works by connecting devices through a common language on the network without manual intervention.

There are two types of service discovery: Server-side and Client-side. Server-side service discovery allows clients applications to find services through a router or a load balancer. Client-side service discovery allows client applications to find services by querying a service registry, in which service instances and endpoints are all within the service registry.

The Service Registry is a database that contains the network locations of service instances. The service registry needs to be highly available and up to date so clients can go through network locations obtained from the service registry. Microservices service discovery is a way for applications and microservices to locate each other on a network. Service discovery implementations within microservices architecture discovery includes a central server (or servers) that maintain a global view of addresses for clients that connect to the central server to update and retrieve addresses. The "global state" (available service IP addresses) of the application across sites and regions also resides in the service discovery database and is accessible by DNS. Users of all services (users using browsers or apps or other services) use well-known DNS mechanisms to obtain service IP addresses.

Avi service discovery automatically maps service host/domain names to their Virtual IP addresses across multiple clusters and availability zones and updates the service discovery database as services are created and disabled. Avi provides an authoritative DNS server for users' devices and other services including a variety of DNS configuration options as well as integration with third-party DNS and IPAM services such as Infoblox. See Figure 13.

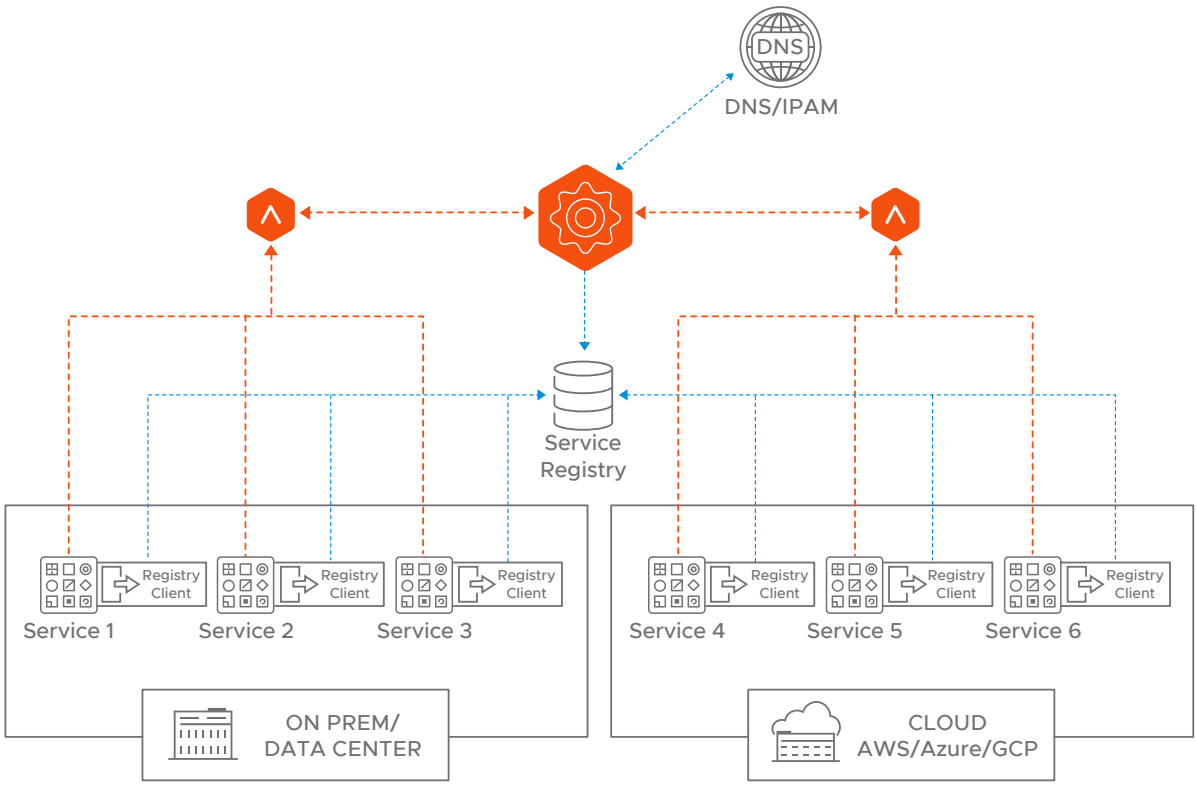
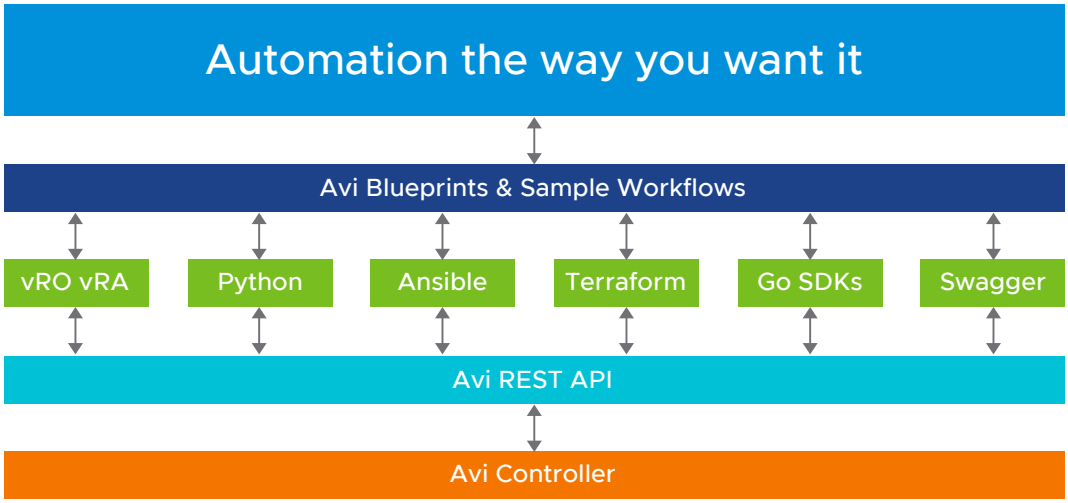


Figure 13: Avi third party services integration

Integration with Automation Tools

Many of these automation elements are something that Avi has natively baked in, but because Avi is built on top REST API, it allows for customization. That means that what is displayed in the UI and the CLI are just wrappers for the API. This is what makes it easy to build out some customization through Terraform, Ansible, VRO VRA, or custom Python scripts.



In Summary

Avi delivers load balancing with integrated app monitoring and analytics, security, predictive autoscaling, and multi-cloud load balancing while offering operational simplicity through automation for enterprises that have their apps deployed in a mix of private data centers and multiple public clouds delivering uniform architecture and user experience, regardless of the environment.

Item No: VMW_WP_AppDelivery_DecisionAutomation_v2 4/22