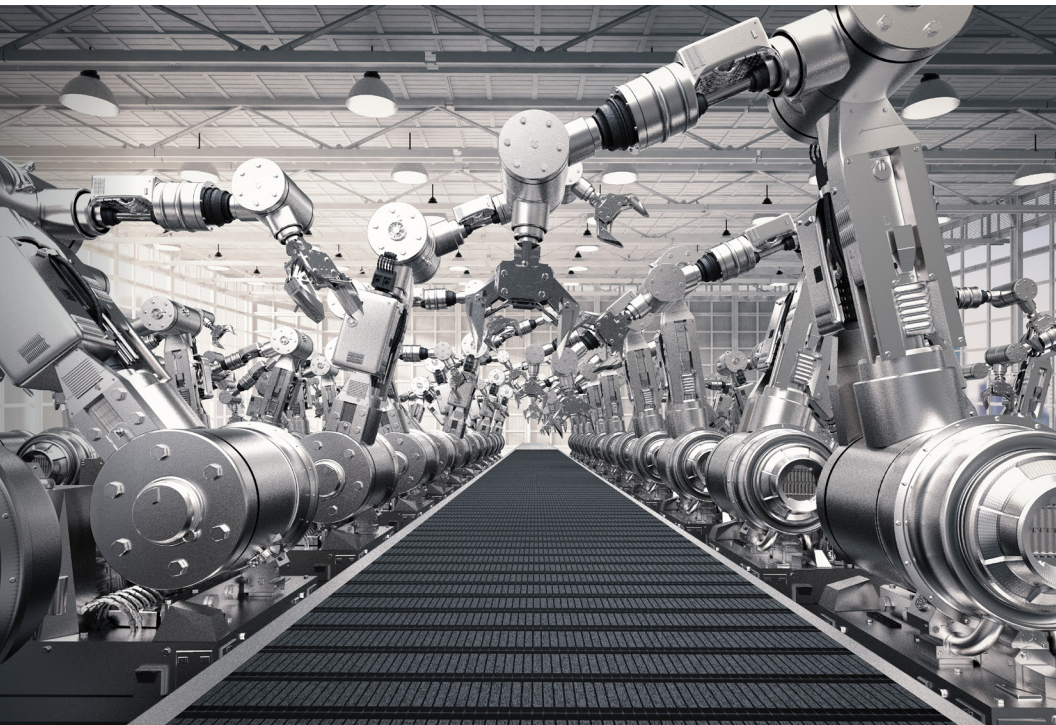
vmware PRESS

VMware NSX® Automation Fundamentals

Caio Oliveira, VMware
Thiago Koga, VMware

Foreword by Martin Casado



VMware NSX® Automation Fundamentals

Caio Oliveira, VMware
Thiago Koga, VMware

Foreword by Martin Casado

VMWARE PRESS

Program Managers

Katie Holms
Shinie Shaw

Technical Writer

Rob Greanias

Reviewers and Content Contributors

Marcos Hernandez
Anderson Duboc
Gustavo Santana
Angel Villar Garea
Andrew Voltmer
Scott Goodman

Designer and Production Manager

Michaela Loeffler
Sappington

Warning & Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors, VMware Press, VMware, and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

The opinions expressed in this book belong to the author and are not necessarily those of VMware.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA
Tel 877-486-9273 Fax 650-427-5001 www.vmware.com.

Table of Contents

List of Figures

About the Authors

Caio Oliveira
Staff Systems Engineer, NSBU
VMware Inc.

Caio Oliveira is a Staff Systems Engineer within the VMware Networking and Security Business Unit (NSBU) for Latin America region, responsible for driving networking and security virtualization across the main enterprises in Brazil. Caio has been with VMware for three years, and has over ten years of networking industry experience. Caio has presented at conferences such VMworld US, OpenStack Days Brazil, vForum Brazil, and Gartner Security Summit Brazil in addition to speaking at local VMware user group meetings.

Caio holds several certifications, including CCIE (#27569) and AWS Solutions Architect. He is a VMware Certified Implementation Expert (VCIX6-NV) for the VMware NSX product and has been recognized as a VMware vExpert for technical community involvement. Most recently, Caio was invited to be part of the VMware CTO Ambassador program for the 2018-2020 tenure. Caio holds a Bachelor's degree in Computer Engineering from the University of São Paulo in Brazil and a Master's degree in Computer Engineering from Politecnico di Torino in Italy.

You can follow Caio on Twitter @oliveirac_caio or visit his blog http://www.oliveirac.com.br.

Thiago Koga
Staff Systems Engineer, NSBU
VMware Inc.

Thiago Koga is a Staff Systems Engineer within the VMware NSBU for Latin America, serving as the first NSX pre-sales engineer for the region. Thiago has been with VMware for more than 4 years and worked with all verticals across the entire LATAM region. Prior to his time at VMware, Thiago spent twelve years working for different vendors in the networking industry. In 2016 he moved to Florida to focus on coverage of northern Latin America.

Thiago holds the VMware Certified Implementation Expert (VCIX6-NV), VCP-NV and VCP-DCV certifications for VMware and AWS Solutions Architect certification. He is also recognized as a VMware vExpert for his community engagement. Thiago holds a Bachelor's Degree in IT systems analysis and MBA at CEAG/FGV.

You can follow Thiago Koga on Twitter @ thikoga or on his LinkedIn profile https://www.linkedin.com/in/thkoga

About the Reviewers and Content Contributors

Marcos Hernandez is a Principal Systems Engineer in the NSX and OpenStack teams at VMware. He is responsible for supporting large global enterprise accounts and providing technical guidance around VMware's suite of networking and automation products. Marcos has a background in data center networking design and expert knowledge in routing and switching technologies. Marcos holds CCIE certification #8283, VCIX, and a Master's Degree in Telecommunications from Universidad Politécnica de Madrid.

Anderson Duboc is an experienced software professional working as a Sr. Systems Engineer within VMware Cloud Business Unit. He possesses a diverse set of interests ranging from technology to philosophy. Over the past twelve years, Anderson has helped IT organizations develop and deliver software in an efficient and effective fashion. He is a huge fan of how IT operations can enable developers to maximize throughput of features without causing chaos and disruption. He is particularly interested in collaborative and agile practices, helping organizations adapt to the increasing pace of change demanded by business teams who wish to utilize cloud and mobile technologies.

Gustavo Santana is the leader of the SDDC system engineering team in Latin America. With more than 20 years of experience in the data center industry, Gustavo has worked on multiple enterprise and service provider data center projects, addressing demands that require tight integration between multiple technology areas – including networking, application optimization, storage, and servers. A true believer of education as a technology catalyst, he has also dedicated himself to the technical development of many IT professionals at customers, partners, and vendors. In addition to holding three CCIE certifications, Gustavo is VCIX-NV and an SNIA Certified Storage Networking Expert (SCSN-E).

Angel Villar Garea is a VMware employee who has worked with NSX for more than three years – initially as a Systems Engineer and recently as Technical Product Manager. Before joining VMware, he enjoyed an extensive career in roles focused on traditional networking.

Additional Contributors

Andrew Voltmer
Scott Goodman

Acknowledgements

"The greater the obstacle, the more glory in overcoming it."
Molière

First, we would like to thank our families for their support during the countless hours spent writing this book – both late into the night and across many weekends. None of this was possible without your support.

It takes the knowledge and resources of multiple individuals to create a technical book successfully. We would like to thank the following people for their support in developing and reviewing the material included:

Thank you to Marcos Hernandez, Gustavo Santana, Anderson Duboc, and Angel Villar Garea for your efforts on either reviewing or contributing with inputs for this book.

Thank you to Martin Casado. Your ideas were always an inspiration for both of us and your passion about NSX and network virtualization is exactly what we were looking for in the opening of our book.

Thanks to the VMware Networking and Security Business Unit's team, including Katie Holms, Shinie Shaw, and Kausum Kumar for the encouragement in creating this book. Thank you for putting your trust in us.

Also thanks to our leaders Paul Byrne, André Andriolli, and Dom Delfino for always supporting and pushing us to "think outside the box" and, as result, this unique book was created.

Caio Oliveira

Thiago Koga

Preface

VMware NSX Automation Fundamentals offers guidance to help organizations automate their environments, leveraging familiar tools including VMware vRealize® Automation™, OpenStack, and PowerNSX. It speaks to engineers and technical decision makers responsible for selecting different sets of tools to automate their infrastructures, helping reduce the manual operation associated with each task.

VMware NSX Automation Fundamentals provides specific details on automation projects while removing removes hurdles and misunderstandings that are the primary roadblocks for organizations seeking innovation and automation for their environments.

Foreword

In the mid-2000s, I was fortunate enough to have visibility into some of the world's largest data centers. This was in the early days of the data center becoming the dominant computing platform. And at the time, the web giants and the large clouds were leading the push with respect to scale, sophistication and efficiency.

One of the architectural aspects that set these data centers apart from the traditional ones that grew out of the wiring closet is that despite catering to different use cases and often using different base technologies, they would implement as much of the infrastructure in software as possible. This didn't mean that they didn't run their data centers on top of best of breed hardware, many did. However, functionality that had traditionally been tied to an appliance such as security, network configuration, L4-7 processing and visibility were instead decoupled from the hardware and implemented fully in software.

They did this not out of ideology, but practicality. The demands on the data center were growing. They had to provision applications and associated services in the timescales the customers wanted them. And they had to scale them as customer demand across multiple applications fluctuated. They had to keep up with the rapid innovation speeds of their application developers that were implementing the functions that drove the business. And they had to do that at scale, with full debugging support, and with as much performance and cost efficiency as they could manage.

Of course, these companies were able to architect this way because in most cases they didn't have the burden of significant legacy workloads. Most of the enterprise didn't have this luxury, and so often continued to use legacy approaches to infrastructure to support their existing workloads.

This was the foundation that lead to the development of NSX. Compute virtualization had already demonstrated that a software layer exposing existing interfaces can be used to get the operational efficiency of the cloud. And so we set out to do the same thing on the network. The charter was simple, we set out to build a software layer that supported all of the interfaces and services of traditional networking. This included everything from L2 connectivity to L4-7 services such as

firewalling and load balancing, as well as standard logging and debugging interfaces. Yet we wanted it to have all the flexibility of software, allowing for the dynamic creating of services and full automation capabilities. Used right, we believed that any data center could achieve the properties of the leading data centers even while hosting legacy workloads.

Now, over a decade later, the adoption of NSX has shown this to be the case. It is being used all over the world and hosting a stunningly broad array of customer classes. From service providers, the large enterprise. to health care, to government, and many, many others.

As with many technologies, the adoption and use of NSX is often a journey. One that starts with a single use case, such as micro segmentation, or dynamic provisioning. But NSX was designed to be a platform that can encompass most network and security functionality. The most sophisticated users integrate deeply with NSX driving much or all of its functionality through programmatic interfaces.

This is exactly why this book is so important. NSX was designed as a platform, one that can be integrated into any existing workflow. It was designed with open interfaces and automation in mind. And automation is the way to fully realize its power. Many new users don't realize the tremendous amount of work and thought that have gone into making NSX the most comprehensive and versatile network hypervisor every built.

Looking back over the last decade, we've come a long way. From what started as a university research project to what's now one of the largest and most pervasive software infrastructure products on the planet. However, there is still a lot more innovation to be done in the data center. And there are few enabling technologies as powerful as compute and network virtualization. I'm delighted to be writing the foreword for this book because it contains what you need to know to be an NSX power user. And mastering NSX is a critical step to mastering modern data center design.

Martin Casado

Abstract

This book explores how VMware NSX® delivers the power of automation. *VMware NSX Automation Fundamentals* brings guidance and knowledge on designing the automation for the software defined data center (SDDC), unlocking NSX's full potential to provide the flexibility and agility needed by enterprises today.

VMware NSX improves the network and security posture of the SDDC by fundamentally changing the approach for networking and security. Through NSX's open API model, organizations can select the automation solution best aligned to their operational practices. VMware NSX has already helped over a thousand organizations design, deploy, and manage their SDDCs.

VMware NSX Automation Fundamentals delivers the roadmap to understanding networking and security automation challenges in today's data centers. It demonstrates the fundamental nature of NSX in the data center architecture while detailing integrated solutions for both VMware and third party offerings (e.g., VMware vRealize® Automation™, OpenStack, Puppet, Chef, PowerNSX) that assist in creating networking and security components on-demand.

Introduction

What does Automation Fundamentals for NSX mean?

Networking and security virtualization is one of the most common topics in IT industry today, with VMware NSX recognized as the most advanced platform in the segment.

VMware NSX has three primary use cases: automation, security, and application continuity. This book will explore automation in depth. The following concepts summarize why VMware NSX is the right answer for automation of networking and security:

1. VMware NSX is an open platform where anybody – regardless of role – can automate. Common automation tools include VMware vRealize Automation, VMware vRealize Orchestration, and OpenStack.

2. VMware NSX helps enterprises achieve modern data center standards, transforming IT from a cost center to a strategic partner of business growth. Companies using NSX can implement solutions to provide any of the following models:

    1. On-demand IT infrastructure

    2. Service-like experience

    3. Developer-centric IT

3. The VMware NSX community actively contributes to open source tools (e.g., PowerNSX, Power CLI, PyNSX). Helping enable automation use cases, these tools are especially focused on and useful in the developer-centric IT space.

Intended Audience

This book is meant for people new to NSX as well as those who have already implemented NSX and now are looking to begin their automation journey. While it is recommended for readers to have a basic understanding of the VMware NSX solution, this is not mandatory since the book will explore some basic NSX concepts.

This book is for any individual involved in transforming and innovating IT environments; someone who is interested in achieving true automation to help accelerate the pace of corporate growth. This list includes:

- Decision makers concerned about challenges of implementing private or public clouds

- Managers responsible for IT infrastructure with the need to achieve greater agility, improve business availability, and maintain security standards

- Engineers responsible for maintaining, creating, and operationalizing IT infrastructure
- Technical decision makers interested in faster and better methods to offer infrastructure to developers or other areas of the business

In short, it provides insights for everyone who should be involved in networking and security automation.

What it will teach

This guide presents an introductory perspective on what is required to automate an NSX environment. It offers a basic understanding on how to consume NSX primitives along with a deeper look into the integration between NSX and two of the main platforms used for automation today: VMware vRealize Automation and OpenStack. References to more detailed documentation are provided where applicable.

Why this subject is important

Automation is one of the most common subjects currently explored by every company across every vertical. The guidance provided in this book will help individuals understand how to achieve business value for NSX through the automation use cases. This book will help individuals build onto their career skillsets while increasing their immediate personal value to their employer.

How to proceed

This book is intended to be read sequentially; however, it was designed to also allow a reader to jump to each section separately, understanding that specific sections are more valuable to different readers.

Readers will be guided first into the definitions and concepts surrounding automation, orchestration, APIs, and related concepts. The book will then explain in more detail the challenges faced in automating the enterprise and why it is necessary to identify a networking and security virtualization platform before considering an automation platform for IT.

In the final chapters it will explore VMware NSX integration with both VMware vRealize Automation and OpenStack. It will examine the differences between the two solutions and help identify the most suitable option on based on situational requirements. It closes with a look at additional tools that leverage NSX automation (e.g., Puppet, Chef) as well as a short summary on the importance of VMware NSX for IT automation.

Disclaimer

This book was written based on NSX for VMware vSphere® (NSX-V); all architectural and technical terms are NSX-V related. However, the concepts and fundamentals explored here are universal and can be used for both VMware NSX versions: NSX-V (for vSphere hypervisor) and NSX-T (for multiple hypervisors).

Concept Definitions

IT Automation is the process of automating the configuration, management, and operation of an IT environment. It is a broad term that includes several tools, technologies, and methodologies used to automate IT processes.

Large enterprises are looking for automation to help achieve better time-to-market, speed, and agility while reducing operational costs to do basic tasks. Automation can help companies of all sizes benefit through a more consistent infrastructure, greater predictability, and a lower rate of human error.

This section explores some of these definitions to offer a better understanding of how IT organizations can leverage NSX to achieve desired benefits.

Programmability

The term programmability can have different meanings depending on user perspective. For a network engineer, programmability means interacting with a device or group of devices (e.g., driving configurations, changes, instructions, or troubleshooting) through software that interacts directly with the device and alters its behavior. To a developer, programmability means transparently abstracting the infrastructure to allow manipulation with a specialized set of tools. For a systems administrator, it is the capability to create the whole infrastructure without the need to interact with different components.

Programmability can also be driven by scripts and API consumption. Sysadmins have been doing this for many years; network engineers have rarely had the same opportunity. Their success has been limited by the variety between each version, model, or function offered by different vendors – if they offer such support at all.

The industry is pivoting to leverage programmability. Previously, RESTful APIs were available only sparingly; now, most companies offer solutions for a variety of use cases.

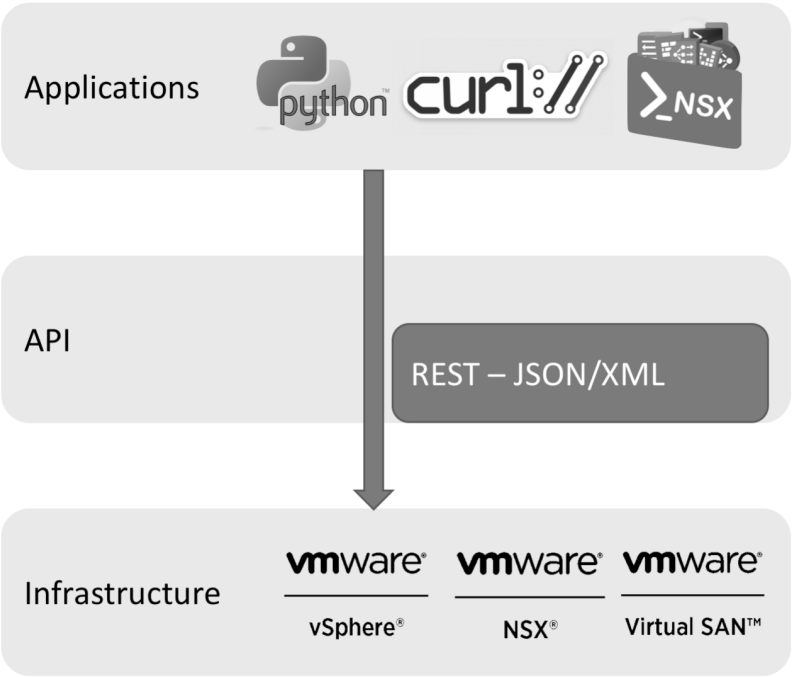


Figure 1.1 Programmability Workflow

Network programmability is a generic term, many times referred to as software defined networking (SDN). OpenFlow, as one of the first major protocol of the SDN movement, helped pave the way for more diverse implementations that were standards based rather than vendor specific. Enterprises are now looking for solutions that allow such programmability, independent of underlying hardware, and deliver better integration of the network with IT infrastructure platforms such as VMware or OpenStack.

Chapter 7 discusses tools available in the market and different perspectives on optimal selection. It also provides examples to demonstrate tangible benefits and realistic deliverables.

Automation

For some organizations, the idea of automation starts with artisan scripts that leverage the programmability offered by the infrastructure to start an automated process.

After a while, those same organizations become familiar with scripting/developer tools (e.g., Ansible, Puppet) and use them to speed up configuration tasks by removing the human interaction; if there is something that was done manually using a CLI or GUI, it can be transformed into code to scale and avoid errors. In this manner, both individual engineers and the organization can reap the advantages as they get rid of time-consuming and repetitive tasks.

The problem with individual automated systems (e.g., VLAN creation) is consistency and task-specific limitation. Enterprises can have a solution to automate the physical fabric, but what about security, compute, and storage? Would it not be better to begin with a unified language that can be leveraged to automate switches, servers, firewalls, and load balancers?

Organizations are focusing their efforts where they experience the most change, finding that most of time it is better to have something automated than nothing at all.

Chapter 3 will cover more details around automation of these tasks. It will explain the concepts and benefits of automation through APIs, REST, and XML.

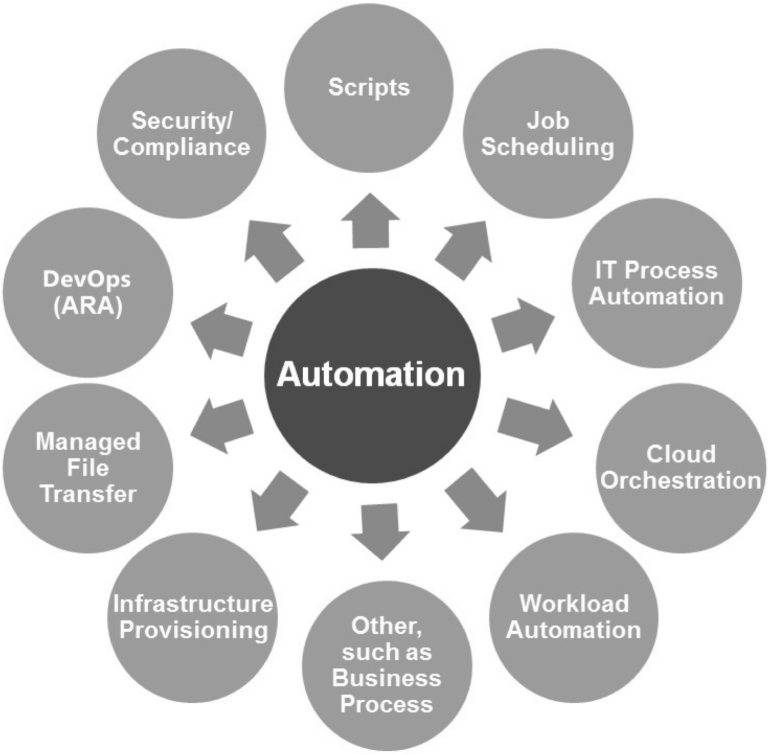


Figure 1.2 Different Automation Solutions

To deal with this singularity of automation – by vendor, type, or model – and lack of ability to view all pieces around an application, an orchestration engine serves an important role.

Orchestration

Both automation and orchestration bring massive value to the business and the end user. They provide a single platform to the IT organization to help curb the dreaded day-to-day operations, breaking them down into smaller, automated tasks that truly provide greater value. On the surface, they seem to be the same thing – but in the context of IT, they could not be more different.

In this context, automation describes tasks or functions accomplished without human intervention. Orchestration describes the arranging and coordinating of automated tasks, ultimately resulting in a consolidated process or workflow. Orchestration tools, whether native to an IaaS platform or distinct 3rd party offering, enumerate the resources that are required as well as their configuration and interconnection. IT admins can use tools such as VMware vRealize® Orchestrator™ to create declarative templates that orchestrate these processes into a single workflow, so that this new environment workflow becomes a single API call.

The subtle difference between automation and orchestration is important because automation is focused on codifying a concrete set of steps, normally handled manually, that are done to a device or component. Orchestration requires participation by and decision making on the part of the infrastructure being orchestrated. The infrastructure is an active participant, a collaborator, in orchestration – but is likely not in automation.

A good example is an orchestral performance at a theater; each musician is performing some sort of automation while playing their own instrument. On the other hand, there is a conductor who is making sure all musicians are working together, in sync, to create harmonization between each instrument/musician. In this example, the conductor acts like the orchestrator – creating servers, networking, security policies, etc. – to make sure everything is ready to the application.

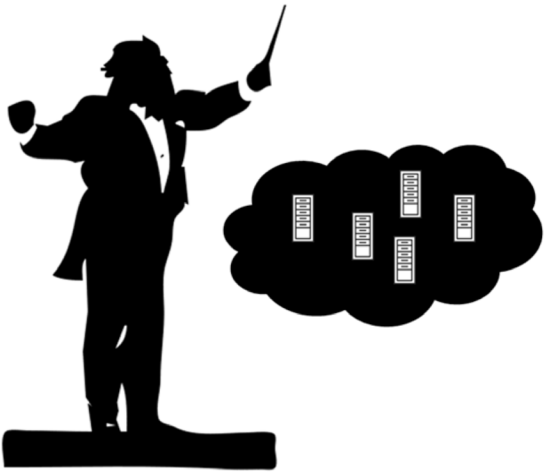


Figure 1.3 Infrastructure Conductor (Maestro)

There are many different orchestration engines in the market. Most do not have context about the underlying infrastructure, which is why a well-documented – even a self-documenting – API is important and relevant when it comes to the IT infrastructure.

Enterprises that have decided to move away from simple, automated tasks to a broader approach through orchestration must pay attention to unique situations and make sure the right configurations are part of a service catalog.

Chapters 4 and 5 cover in detail how cloud management platforms (CMPs) help organizations realize this overall approach. They provide a guide through different scenarios and customer requirements.

Data Center Automation Challenges

> *"If I had asked people what they wanted, they would have said faster horses."*
> *Henry Ford*

There are times in the history of humankind when innovation and technology become fundamental to the transition from one era to another.

The true impact of those changes may not truly be understood by most people from the outset, but they are quickly adopted to leverage the benefits of this disruptive new way of thinking.

Looking at the pictures in Figure 2.1, the car industry is a perfect example to demonstrate this concept.

Figure 2.1 Cars substitution of Horse-Drawn Vehicles

These pictures illustrate the changes that happened in fewer than 15 years with the car industry. Horse-drawn vehicles were completely replaced by cars due to the new technology and benefits that cars offered. In this specific period, cars were accepted as faster, safer, and more reliable. By challenging the status quo rather than continuing along the exact same way of doing things without thought of innovation, the world was changed. What remains of those days is the concept of how we measure the power of an engine – mechanical, hydraulic, or otherwise – with the throwback term horsepower.

What took thirteen years from year 1900 to 1913 is today happening at a much faster pace due to the evolution of technology. Uber, Airbnb, Netflix, Tesla – these are new entrants that can completely disrupt a stable and established market overnight.

It is important to note that for a new technology to make a complete disruption in any market, creating value is sometimes not enough; it is just an entry point. The new technology needs to address the challenges associated with scaling and attaining healthy growth in order to achieve disruption. This is a goal that many startups or challengers never accomplish.

New industries, and especially disruptive technologies, generally fail when the demand cannot be sustained by the supply. When the first cars started to run in the streets, the demand for cars significantly increased over the years. With the growth of the population, the available offering was not enough to meet the demand. This led to automotive industrialization, and later automation.

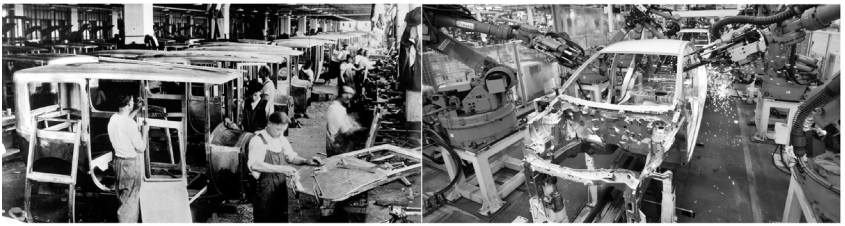


Figure 2.2 Car Industrialization

New ideas and demands are generally associated with a big challenge of meeting the demand with corresponding supply. The car industry needed that, and started to use specialized machines to substitute for the human workforce. This allowed companies accomplish the same goals in a much more efficient manner, without human error, and with repeatable standard processes. Figure 2.2 offers two images exemplifying this idea.

Moving ahead in time and observing what is happening today in the data center, enterprises are facing similar challenges that the car industry faced decades ago. The requirements of the business areas of today need a much faster infrastructure to address their goals and the demands of their customers. Developers cannot wait weeks or months to start building and developing their applications.

Several companies wisely identified that gap and started to offer services allowing a developer or interested user to build out an entire infrastructure in few minutes. This phenomenon is often called shadow IT. Most companies struggle with it, because now they need to maintain the same policies they apply for their internal workloads for cloud workloads. There is also a new set of tools and skills that should be well understood for infrastructure administrators to control and maintain both environments – on premises and in the cloud.

Enterprises are eagerly looking for solutions and products to increase their data center automation for faster infrastructure offerings for developers; however, networking and security are often forgotten in this process. These companies cannot achieve their goals and these projects fail because it is important to address all four essential pillars to every data center: computing, networking, security, and storage.

Network Automation Challenges

Since the beginning of the rise of private clouds and the quest for a fully automated data center, automation of networking components has been the most difficult challenge faced by cloud administrators and users.

There are three main reasons why it is so difficult to automate networking:

- Lack of well documented APIs, or lack of APIs altogether
- Non-standard configuration commands across vendors
- Several points of configuration

Only with the most recent network hardware systems have APIs capable of exposing the features and capabilities begun to appear. This lack of functionality creates a particular and important problem; whenever a user, administrator, or system needs to create any sort of automation, direct and full access to each piece of equipment needs to be provided. This is most often accomplished using Telnet or SSH protocols. Some may try SSH or CLI for automation with scripts, but mostly only for basic tasks; this approach is especially frustrating when more complex tasks cannot be achieved.

While NETCONF and YANG are implemented by many vendors, there is no standard configuration set of commands for network hardware equipment. Every vendor has its own commands and specific language, creating a hard job for any tool trying to automate the network. Even inside the same vendor, different switches and routers from different lines of business have their own sets of commands. Even for a straightforward example of verifying a switch serial number – how hard is to identify the right command?

One of the biggest challenges is nature of the network itself. It is composed of several pieces of equipment, almost none offering a centralized point of control. Each switch needs to be configured and

managed as a standalone component. This is one of the main reasons for network failures in the enterprises today (e.g., improper inter-switch configuration). Extend this to managing hundreds, or sometimes thousands, of networking hardware devices as standalone components, across a wide range of different models and vendors.

In summary, the provisioning of an automation solution faces a series of difficulties, from simply using the different command interfaces – without the use of a common API – of each vendor to accessing each component as a standalone element without a centralized management point.

It is also important to mention that in some big enterprises, especially cloud providers, the limits of VLANs are a common issue. There is a necessity for an overlay technology like VXLAN when offering a tenant-based solution for customers in order to avoid the use of shared subnets and networks across tenants.

There are some initiatives (e.g., OpenDaylight – www.opendaylight.org) that are trying to create a centralized controller capable of interacting with different vendors. The goals include creating a centralized point of control and exposing APIs for consumption. Despite a multi-year effort, this initiative is sparsely adopted, existing mostly in lab environments. The main impediment is the complexity of integrating with each vendor in a different way.

The two main approaches being adopted today to provide true network automation for the data centers are the SDN hardware approach and a network virtualization software approach. Both approaches rely on creating an abstraction layer and using overlay technologies for faster provisioning of network elements. They offer a centralized point of control that also exposes APIs to any external CMP to automate the creation, modification, or removal of network components. The models differ in their approach to encapsulation of the independent hardware underneath the solution.

SDN relies on proprietary hardware switches and controllers, demanding a refresh of all data center switches to ensure support for the encapsulation piece. All the work of the encapsulation piece will be done inside this new networking equipment. There are also scalability issues with this model of deployment, further reducing interest in adopting this technology.

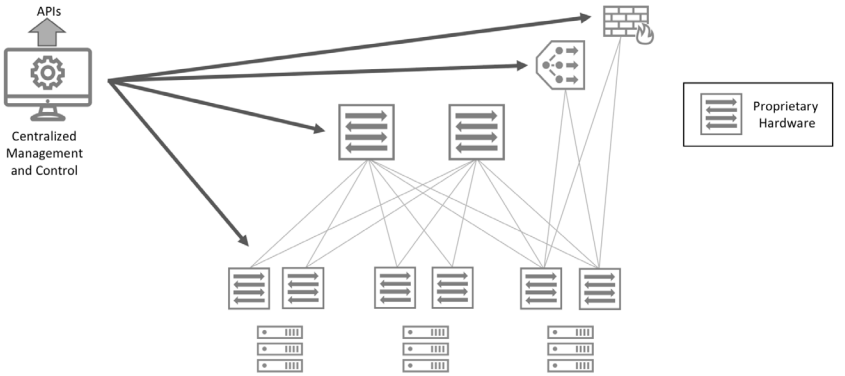


Figure 2.3 SDN - Hardware Approach

The most common approach is use of an overlay protocol. This method lets the encapsulation be handled inside the hypervisors, enabling enterprises to create abstract networks without the need to refresh the physical network. It also allows any mix of hardware vendors – not possible in the SDN hardware approach – in a simplified manner. The underlay physical network needs to meet only two requirements: MTU of at least 1600 bytes to address VXLAN requirements and IP connectivity between the hypervisor endpoints. Some newer encapsulation protocols even eliminate the requirement for jumbo frames.

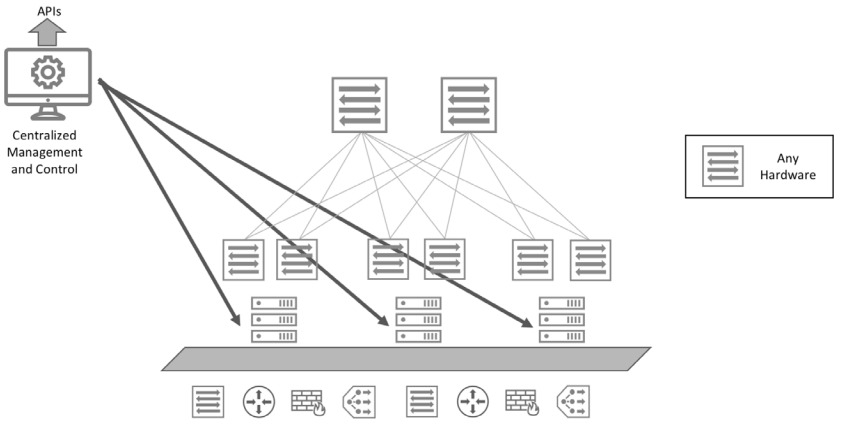


Figure 2.4 Network Virtualization - Software Approach

The network is a tightly coupled system; any misconfiguration or problem can damage or even break the entire data center environment. Hypervisors are loosely coupled systems; they work separately from each other, so if a problem arises in one hypervisor it does not affect the others. This is another reason why network virtualization based in software is more reliable than the SDN hardware approach – it can be distributed in the endpoints (i.e., hypervisors) instead of utilizing centralized components to perform the functions.

VMware NSX uses a network virtualization platform based on a software approach with thousands of enterprises around the globe. The next few chapters will discuss in more detail the benefits seen by those who have already adopted the technology.

Security Automation Challenges

Security is one of the top concerns for enterprises. Every year, the spend related to security solutions and products grows exponentially compared to other areas, led by concerns over cybercrime and cyberattacks. These are now part of the new enterprise reality, no matter an organization's size.

Before proposing solutions to solve or minimize the problems, consider the anatomy of a modern cyberattack by dividing it into four parts:

1. **Intrusion**: Attacker gains access to a specific system or component
2. **Propagation**: After gaining control of the specific system or component, the infected system infects several other systems or components
3. **Extraction**: Attacker extracts sensitive data or information from the infected systems or components
4. **Exfiltration**: Attacker erases all information related to the attack

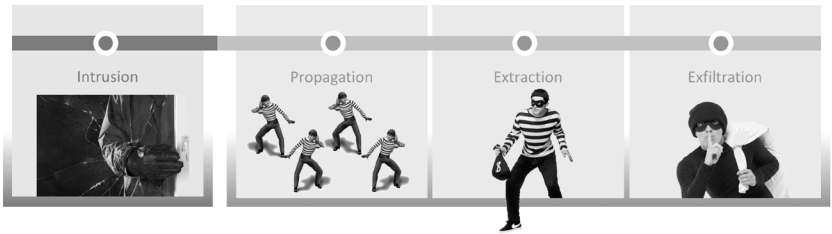


Figure 2.5 Anatomy of a modern Cyber-Attack

These four phases present significantly different risks and impact to enterprises losses through an attack. Despite of the investments in solutions and products to prevent intrusion (e.g., huge and complex perimeter firewalls with several advanced features), the losses related to the propagation, extraction, and exfiltration phases represent a much larger cost. Many customers have few or no tools to avoid these types of attacks.

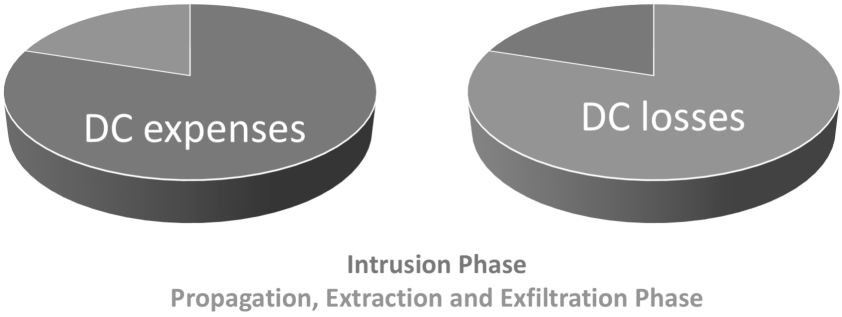


Figure 2.6 Security Data Center Expenses and Losses

Enterprises are looking for a complete end-to-end automated data center, and security is a central component of this strategy. Interacting with security components in the environment (e.g., firewalls) is a must-have for any on-demand security model. It must be enabled for every new workload or application provisioned. Direct interaction with different products and solutions makes security automation one of the most difficult tasks that administrators and users face.

Traffic patterns in the data center today vary greatly from those several years ago. East-west traffic (i.e., between the applications and workloads) inside the data center today represents more than 80% of the total traffic volume. Facing this reality, internal security is a concern that every enterprise needs to address.

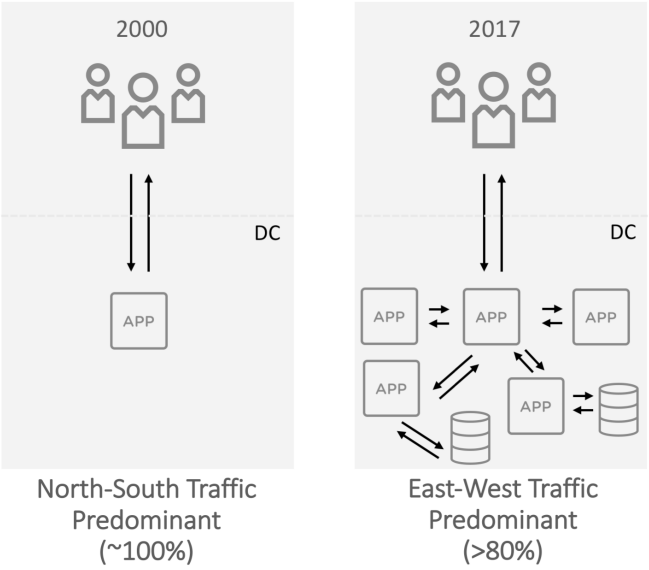


Figure 2.7 East-West and North-South Traffic in the Data Center

With applications, VMs, and containers growing at an increasingly faster rate, manual creation of firewall rules is coming to an end. It is impossible to create, maintain, and operate thousands of firewall rules in a heterogeneous environment of security vendors and solutions. As IP-based rules become harder to maintain, enterprises must look for ways to summarize these lists while finding other types of objects and groupings to help consolidate and automate firewall rules.

Every year, the traffic inside the data center continues to grow; how can security solutions address this growth? Centralization is not the optimal approach. Enterprises are looking for more robust firewall

appliances with the next-generation capabilities that are needed to handle an enormous amount of traffic. They are spending money and effort instead of looking to a new and modern approach that offers a distributed approach to protecting the workloads in an automated fashion.

In summary, enterprises today struggle with the growth of east-west traffic. They are looking for a centralized, automated, simple, and powerful tool to enforce security closer to the application. This tool should support automation through any cloud management platform in the market. In the coming chapters, this book will explore how VMware NSX can help deliver on this vision.

Tales from the Field – Gustavo Santana

Challenges of Physical Network Automation

There are several reasons why I particularly dislike the term *software-defined networking* (*SDN*). One of them is the sheer fact that young engineers are leaving university nowadays without dissociating such technologies from what they consider basic skills of *networking*.

Whereas such acceptance surely attests to how network virtualization is firmly ingrained in IT automation, it still amazes me when the following statement is issued as an afterthought during architectural discussions: "*No problem, we can automate the physical network".*

Unfortunately, the misfortunes I suffered during my earlier experiences with network automation do not allow me to remain silent in such occasions.

These experiences roughly started during the second half of the 2000s, when I was working as a solutions architect in a big network equipment manufacturer. Back then, many service providers and advanced enterprise corporations were already flirting with the concepts of cloud computing via Infrastructure-as-a-Service (IaaS) projects intended to provision virtual machine to end users, developers, or IT administrators.

As soon as network architects were engaged in these projects, they often did not feel too comfortable with a piece software configuring a physical network infrastructure already in production. Understandably for these professionals, these solutions were not yet proven in critical environments such as the ones they were designing for.

Consequently, in a way to minimize disruption and complexity, I saw many projects strictly focusing on the automated provisioning of virtual machines while the network infrastructure remained statically configured, as Figure 2.8 explains.

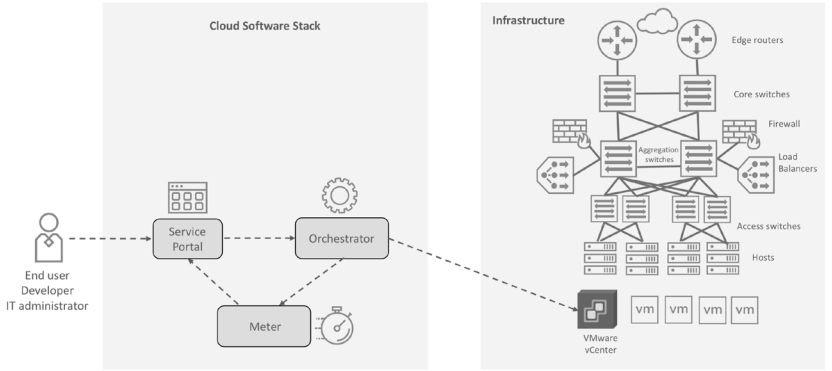


Figure 2.8 Automation with a Preconfigured Network

In this example, the cloud software stack, via its orchestrator, only interacts with the VM manager – VMware vCenter in this case – to provision VMs that will consume a carefully designed set of VLANs, IP subnets, firewall rules, and load balancing functions.

Such sensible compromise apparently satisfied both network and cloud architects for some time, until they painfully discovered together the inherent challenges associated with this approach:

If service portal users could choose the VLANs where the virtual machines would be connected, a large number of security issues could arise. For example, distinct tenants assigned to the same security segment could easily access each other's resources without proper protection from a firewall.

Assigning a set of VLANs to each tenant established a clear scalability limit to the project because a network infrastructure can only offer 4,094 VLANs as defined in the IEEE 802.1Q standard.
After deciding the number of VLANs each tenant should be connected to, the configuration of firewall rules and load balancing configurations for the all the potential tenants could become complex and fairly long.

Worst of all, the pre-provisioned network approach did not allow the offering of new services in the portal – such as applications with a different number of tiers as well as user-customizable security and content rules – without a network reconfiguration.

With time, the unstoppable force ended up winning its fight with the immovable object. As more corporations relied on the agility provided by cloud computing concepts during the early 2010s, more flexibility was also required from their network infrastructure. The era of proper physical network automation then began, as Figure 2.9 illustrates.

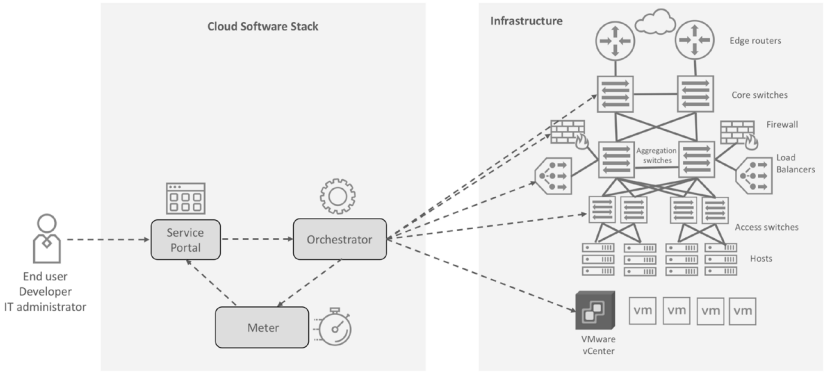


Figure 2.9 Physical Network Automation

In IaaS projects deployed as this figure shows, services selected on the portal request the orchestrator to execute *workflows* that introduce new configurations on existing data center network devices, such as core switches, aggregation switches, access switches, firewalls, and load balancers.

Still, many network teams were nervous about a piece of software executing configurations in traditional production networks. Consequently, it was not uncommon that a separate network infrastructure was acquired and built for these automated environments.

All was well on paper, until the project architects faced misgivings such as:

- Many network devices did not have APIs, making it necessary for the orchestrator to deploy configurations through SSH sessions toward each device's CLI. Differently from APIs, automation through CLI access can become extremely complex because command outputs vary immensely between different vendors, platforms from the same manufacturer, and even firmware versions

from the same platform. Resultantly, many network architects decided to standardize network equipment in an attempt to minimize orchestration coding complexity, but causing hardware lock-in.

- The addition of new devices in the network infrastructure required changes in the orchestration software, which sometimes also required a finer perspective of the network topology to differentiate similar devices with distinct roles such as core, aggregation, and access switches.

- If tenants also had the option to decommission resources, the orchestrator workflows could become increasingly convoluted with tasks such as recognition on each device of all configurations that belong to this tenant, removal of these configurations without causing noise to other tenants, and recalculation of the available pool of resources.

For such reasons, I have personally witnessed IaaS projects where the orchestration programming was actually much more expensive than the sum of computing and network hardware acquired!

A fresh breath of air to these IaaS projects came with the porting of routing, firewall, and load balancer functions into specialized VMs called *virtual network appliances.* Their use greatly simplified such projects because these appliances could relieve the physical network from the exact functions that required sophisticated programming on the orchestrator.

However, successful projects that relished the capability to deploy standalone virtual routers, virtual firewalls, and virtual load balancers felt the sour taste of the following shortcomings:

- These virtual devices are generally licensed per instance, which can cause huge costs in the IaaS project as it becomes more popular.

- Because each tenant deploys a set of virtual appliances, a large number of devices can become difficult to monitor and troubleshoot without a centralized management solution.

Enter VMware NSX… and the rest was history.

With this brief walk down memory lane, I truly hope you will not be doomed to repeat history because you did not hear about it. As Thiago and Caio will continue to explain in this book, VMware NSX became a natural fit for automated IT environments because it strongly leverages in its essence all experience gathered by the many network automation pioneers.

Tales from the Field – Marcos Hernandez

The Curious Case of the Ever-growing Firewall Table

In our dealings with customers around the world, we encountered one with a very common challenge faced by many security teams: firewall rule sprawl. While the challenge itself is not unique, in this particular case it had spun out of control. After years and years of poor firewall hygiene, multiple configuration sources, and a slew of merges/acquisitions, the configuration of the core firewalls had reached more than a million lines. Compiling this configuration took five hours, and backing it up more than two hours. This customer was forced to seek special treatment and support as their configuration far exceeded the devices maximum advertised parameters. Over the years, they had also experienced a number of outages and upgrades that prompted a hardware reboot, inevitably leading to traffic loss.

After careful analysis of their security posture, intended state, application profile, and even geographical distribution, VMware estimated that 25% of the rules in the ruleset were masked (e.g., were never going to be hit), 20% of the rules were referencing systems that no longer existed in the environment, and over 30% of the rules were duplicates! That meant that only 25% of the rules – remember, that is 250 thousand – were there for a reason, although no one knew exactly why.

As of the writing of this book, VMware is still working with this customer to identify areas of optimization. Migrating the ruleset in the perimeter to the NSX distributed firewall for more efficient segmentation is not enough, as this does not solve for the complexity problem, only moves it to a different plane. Assessing their application needs is crucial and we expect to see a tremendous reduction and rationalization in the number of rules, while enhancing, not diminishing, their security posture.

Automation Concepts

When it comes to automation, many people get the chills or are overcome with doubt and fear. This is often due to the fact the word itself means different things to different people depending on what and where they want to automate. This confusion causes misunderstandings in the expectations, but it is important to remember that automation is an essential and strategic component of modernization and digital transformation. Modern and dynamic environments need a new type of management solution that can improve speed, scale, and stability across the enterprise.

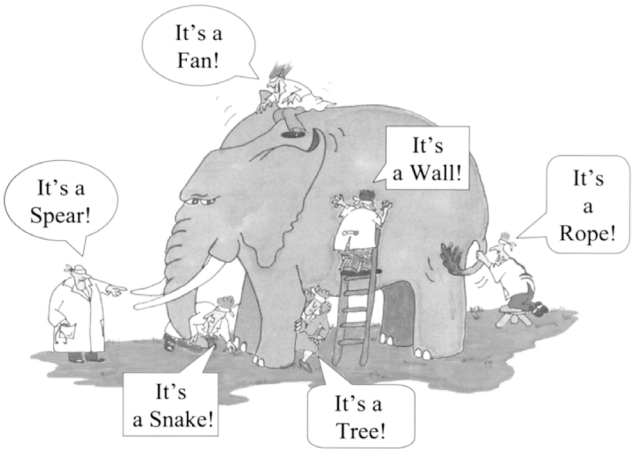


Figure 3.1 Automation may be Different for Each Person/Organization

Before the discussion around the tools and ways to automate the IT world, consider the concept itself. Automation is defined as:

1. the technique of making an apparatus, a process, or a system operate automatically
2. the state of being operated automatically
3. automatically controlled operation of an apparatus, process, or system by mechanical or electronic devices that take the place of human labor

https://www.merriam-webster.com/dictionary/automation

Interest in automation has increased over the years as demands for better security, enhanced mobility, and reduced expenses have grown and technological possibilities have multiplied. The following examples will demonstrate that automation is an integral part of daily life, not just for consumers but even simply human beings.

The food retail industry has started to apply automation to the ordering process. Some vendors have introduced touch screen ordering and payment systems in many of their restaurants, reducing the need for more cashiers. Some cafes and restaurants are using mobile apps to make the ordering process more efficient for customers ordering and paying on personal devices. Some restaurants have

automated food delivery to customers' tables using a conveyor belt system. Many supermarkets, and even smaller stores, are rapidly introducing self-checkout systems that reduce the need for employing checkout workers.

Another example is an emerging practice of increased automation of household appliances and features in residential dwellings. Of particular interest are those with electronic means that allow for services that in the past were impractical, overly expensive, or simply impossible. It is possible to start a coffee machine, request a cab, order pizza, or ask to an artificial intelligent system turn on or turn off the light.

All these examples help better understand the impact of automation on daily life; people like to order a coffee without waiting a long time in the line or prefer to pay the supermarket bill over an express self-checkout system.

Some automation advantages include:

- Increased throughput or productivity
- Improved quality or increased predictability of quality
- Improved robustness (i.e., consistency) of processes or product
- Significant reduction in operating time and work handling time
- Freeing up of workers to take on other roles
- Creating higher level jobs in the development, deployment, maintenance, and running of the automated processes

What about data center applications? Is it possible to automate their networking and security? What benefits does automation bring to the table?

Data center applications need much more than simply appropriately sized virtual machines. They also need accurately configured network connectivity, security, availability, scalability, and performance. In order to deliver these capabilities to the applications, it is necessary to automate more than just assigning IP address, DNS entries, and VLANs to the virtual machines that host those applications. As part of deploying a multi-tiered application, it is necessary to provision connectivity through deployment of logical switches and routers. In addition, it is important to securely deploy the application through intelligent placement of workloads in security groups, protected by firewall rules. To achieve these results, it is important to consider a

platform that can abstract the underlay infrastructure and automate these processes in the same manner it has been done with compute for years. There are many benefits of such an approach:

- Avoid manual configuration mistakes
- Achieve efficiency through automated processes and workflows
- Increase productivity through elimination of required manual steps
- Decrease operational expenses (OPEX) by eliminating manual tasks
- Streamline maintenance and troubleshooting through deployment of a consistent application environment

Moving forward in this journey, engineers may be concerned that automation will replace their jobs. Looking back at the convergence of wired PBX systems to voice-over-IP, some of the PBX technicians made the transition, while others retired along with the equipment they supported. In practice, this transition was more about changing job roles than eliminating them. There will always be a need for people who understand how complex technology works, how to use it in good designs, and how to diagnose problems when it does not work as intended. Using the grocery store example, it is like the cashier helping when the express self-checkout system does not work.

Having a better understanding of automation itself and the advantages it brings, the question now is how best to use that in the data center environment? What sets of tools are available to help? How can automation accelerate application development and delivery, reduce provisioning time, improve resource utilization, and eliminate error-prone processes?

Before answering these questions, it is important to understand what tools are used today and how VMware NSX can easily apply them to deliver these benefits.

What is an API?

This section will answer questions familiar to those new to the concept of an API: what does it mean; how is it used; and what are its benefits? It will explain how the concept is relevant to automation and the importance of always looking for a solid and well documented API.

The acronym API stands for Application Programming Interface. It is a particular set of rules, specifications, and tools for building software applications. It serves as an interface between different software programs and facilitates their interaction, the way the user interface facilitates interaction between humans and computers.

The API offers flexible ways of projecting capabilities to an outside audience. It enables enterprises to innovate faster without actually getting into or understanding the underlying code or software of the applications.

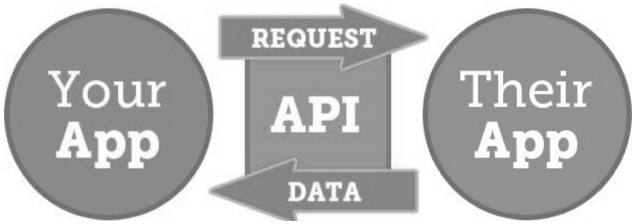


Figure 3.2 API Interactions

Some vendors consider an API as product; they carefully build and document it so other stakeholders can understand how to interact with it and create value. A good API must be open and carry a package of capabilities that is attractive to an audience, independent of any specific piece of software running on the back end.

An API is not usually a user interface. It provides software-to-software interaction, not user interaction. Sometimes, though, an API may provide a user interface widget, which an app can use and display.

There are two primary benefits that an API brings – simplification and standardization.

There are many different types of APIs for operating systems, applications, or websites. Microsoft Windows has many API sets that are used by system hardware and applications; when text is copied and pasted from one application to another, it is an API that allows that to work.

Along with program-centric APIs, there are also web APIs such as the Simple Object Access Protocol (SOAP), Remote Procedure Call (RPC), and Representational State Transfer (REST). The next few pages will discuss these types of APIs in greater detail.

API Documentation

API documentation describes what types of services an API can offer and how to use them. It aims to cover everything a consumer would need to know to use the API. Documentation is crucial for the development and maintenance of applications that use the API. It should follow some specifications to provide a consistent format to document and consume the APIs.

API specification formatting provides several benefits for developers, including:

- Developers or testers can write tests in a simpler way because the specification is well defined and clearly readable.
- Clarity leads to a dramatic reduction of errors during implementation, testing, and troubleshooting.
- Consistency allows other developers to use resources in a simple way.

Two of the most common API documentation and specification formats are RESTful API Modeling Language (RAML) and OpenAPI Specification, formerly known as Swagger. Both are widely utilized and have a large open source community.

RAML is a YAML-based language, providing all the information necessary for describing RESTful APIs. RAML makes it easy to build an API by turning specification into code with server-side generators in several different languages, including NodeJS, Java, .NET, and Python. VMware offers a community-supported RAML specification of NSX for vSphere API (nsxraml) that can be used to simplify the consumption of NSX services. For additional details please see the nsxraml page on GitHub (https://github.com/vmware/nsxraml).

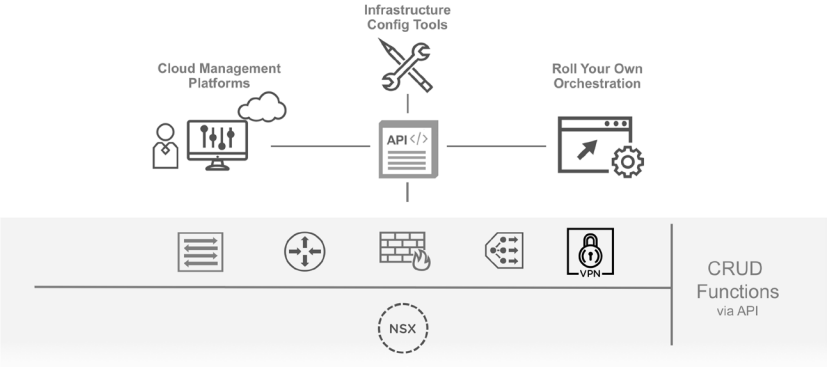


Figure 3.3 VMware NSX RESTful API

Swagger is a framework that was designed to describe, produce, visualize, and consume RESTful web services. Referred to as language-agnostic, it has been developed to be read using a common language. It is also extensible into new technologies and protocols beyond HTTP. The popularity stems from its simplicity; from its concise documentation to its ease of readability for machines and humans alike, Swagger is a framework that has been made very easy to utilize.

What to look for in a good API

The following list reviews some of the key areas to consider when looking at APIs:

- Standard, well defined interfaces into the API such as XML, SOAP, or REST
- Sample code, documentation, and software development kits (SDKs) to make the process of integration quick and easy
- Security controls including:
    - Access control into the API
    - Encrypted interface into the API such as HTTPS, SSL, or TLS to encrypt the sharing of data and services between the applications
    - Ability to completely lock down the API if it is not being used

REST Definitions

Representational State Transfer (REST) is one of the most popular types of API. It was designed to take advantage of existing protocols, such as HTTP, so that developers do not need to install additional software or libraries when creating a REST API. Some of the characteristics required of a REST service include simplicity of interfaces, identification of resources within the request, and the ability to manipulate the resources via the interface. A REST API is defined by a collection of XML documents that represent the objects on which the API operates.

One of the key advantages of REST APIs is that they provide a great deal of flexibility. Data is not tied to resources or methods, so REST can handle multiple types of calls, return different data formats, and even change structurally with the correct implementation of hypermedia. This flexibility allows developers to build an API that meets organization needs.

Unlike SOAP, REST is not constrained to XML; it can return XML, JSON, YAML, or any other format depending on what the client requests. Unlike RPC, users are not required to know procedure names or specific parameters in a specific order.

REST uses HTTP verbs (i.e., methods) to create, read, update, and delete – tasks referred to as CRUD operations. Other HTTP verbs exist, but they are not used as significantly as these four. The table in Figure 3.4 describes each verb and its operations within an NSX system.

HTTP "Verbs"	Use	CRUD
POST	Create an NSX object (e.g. logical switch)	Create
GET	Retrieve data about a single NSX object or multiple objects	Read
PUT	Modify the properties of an already existing NSX object	Update
DELETE	Remove an NSX object	Delete

Figure 3.4 HTTP verbs/methods and CRUD Operations

In the NSX world, the NSX Manager is responsible for the web service API over HTTPS (i.e., TCP port 443) where all API requests and responses are formatted in XML. Any REST API tools (e.g., cURL, RESTClient, Postman) that support HTTP operations can be used to interact with VMware NSX.

Figure 3.5, taken from the Automation Leveraging NSX REST API document, shows the typical structure of a REST API call for NSX using an example that shows scopes (i.e., transport zones).

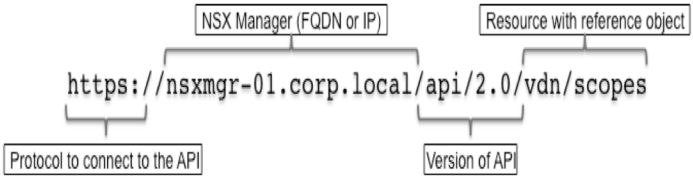


Figure 3.5 VMware NSX® API™ Structure (example)

All NSX REST API interactions are performed after authentication to enforce the security of operations.

Consuming NSX REST API thought different methods

There are different ways to consume the NSX REST API. This can be a simple tool used to transfer data from server to client like cURL or a REST client tool from the Chrome web browser like Postman. Going one step further, simple API calls via REST clients can leverage the NSX REST API via different programming languages to further automate tasks and create workflows/programs.

Many programming languages can be utilized, including Python, PowerShell, Java, Perl, and Go; some of these choices will be discussed in Chapter 7. Using these tools, organizations can automate NSX and watch it become helpful not only for management, but also for rapid development, troubleshooting, and code reuse.

Another option to leverage the NSX REST API is through the use of orchestrators and configuration management tools. With these solutions, users can define workflows or playbooks that includes a list

of tasks that will be executed in an automated way. Those tools are usually flexible and highly configurable, allowing organizations to define what needs to be automated. Some examples are VMware vRealize Orchestrator, a solution adopted by many VMware customers, and Ansible, an open source platform written in Python.

The final option is consumption of the NSX REST API through a cloud management platform (CMP). This approach is used by many organizations to build their private clouds. In this case, these businesses have realized that simply creating and deleting VMs is now insufficient; they need to rely on a solution that can also automate the networking and security services of an application. For such a use case, the NSX REST API can provide out-of-the-box integration. The most widely used CMPs are VMware vRealize Automation and OpenStack, both of which will be discussed in Chapters 4 and 5.

The VMware NSX RESTful API service can be consumed in several ways. This means that simple or elaborate workflows and complete systems/portals can be created to provide custom automation, management, and monitoring capabilities.

Both experienced developers and novice API users can refer to the NSX for vSphere API Guide for further insights. This guide describes how to install, configure, monitor, and maintain the VMware NSX system by using REST API requests. As always when working with APIs, it is important to check which NSX is deployed to ensure use of the correct API documentation.

XML Definitions

XML (eXtensible Markup Language) is a flexible way to exchange structured data between web services. It defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

XML code is a formal recommendation from the World Wide Web Consortium (W3C), similar to Hypertext Markup Language (HTML). XML data is known as self-describing or self-defining, meaning that the structure of the data is embedded with the data. When the data arrives, there is no need to pre-build the structure to store the data; it is dynamically understood within the XML.

The design goals of XML emphasize simplicity, generality, and usability. XML's power resides in its simplicity. It can take large chunks of information and consolidate them into an XML document. It is a representation of arbitrary data structures, so code should be human-legible and reasonably clear.

XML documents must contain one **root** element that is the **parent** of all other elements:

```
<root>
  <child>
    <subchild>...</subchild>
  </child>
</root>
```

In this example **<controllerSyslogServer>** is the root element:

```
<controllerSyslogServer>
<syslogServer>10.135.14.236</syslogServer>
<port>514</port>
<protocol>UDP</protocol>
<level>INFO</level>
</controllerSyslogServer>
```

This example informs the NSX Manager in a structured way (i.e., body using XML format) to configure a syslog exporter on the specified controller node. To accomplish that, an HTTP request (POST) is made, targeting the NSX Manager URL.

POST https://<nsxmgr-ip-address>/api/2.0/vdn/controller/{controller-id}/syslog

The XML data structure within the body field shows all the available fields with their respective values for the desired NSX object. When consuming NSX REST APIs through any REST client, it is important to set the header field of "content-type" as "application-xml", as shown in Figure 3.6.

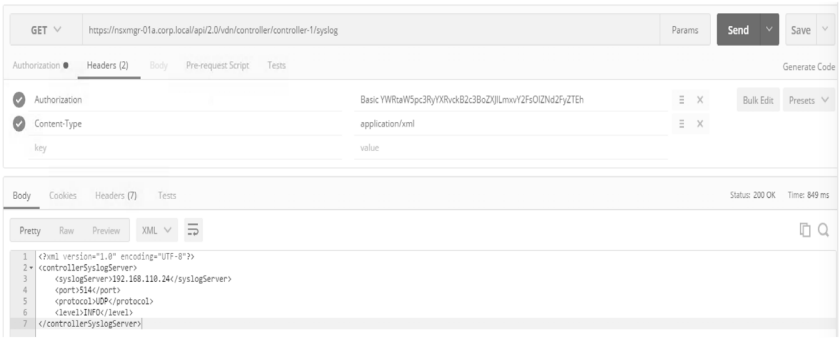


Figure 3.6 Postman within Chrome to GET Syslog Information

The API document NSX for vSphere API Guide provides a complete, detailed description of the XML structure that must be included in the HTTP request body to create NSX services.

Myth Buster – Automation is for Cloud Only

This section looks at what organizations have done so far and explores the myths behind a simple question – is automation for cloud only?

The previous few sections have examined how automation can be used in different ways for different use cases – on-demand IT infrastructure, service-like experience, or developer-centric IT. Beyond these concepts, VMware NSX can address automation use cases that are not necessarily tied to cloud projects.

Physical and Virtual Workloads Paradigm

When enterprises consider automation, they are often eager to start automating everything – all physical and virtual environments. A central question to ask when getting started – is it necessary to automate everything? Consider a subset of the environment that is not yet virtualized; is this environment so dynamic that there are benefits to automating it? A tangible example would be physical database servers that are deployed once and never get touched again in their lifetime from a network policy point of view.

Perhaps it is a static environment that does not requires a lot of changes? Is the automation effort worthwhile for a handful of changes over the year?

On the other side, there may be an extremely dynamic virtualized environment with a lot of changes, all this tied to different VMs sitting in separate data centers. It makes clear sense to automate some activities in this environment to deliver real benefits with quick wins.

The important thing is to see this as step-by-step adoption. Organizations can get benefits of automation for each project that can delivery agility, speed, flexibility, or time to market.

What are the other use cases that NSX can help to automate?

- Network services: Create on-demand logical networks, logical routers, logical switches, NAT and, DHCP services. Those services can be consumed without hours of manual configuration and extensive processes, avoiding extra costs and repetitive tasks.
- Security services: NSX Service Composer and grouping mechanisms can easily automate DMZ security projects or micro-segmented applications thru API calls to the distributed firewall (DFW).
- Load balancer services: NSX load balancers offer a fully featured RESTFull API. This API can be used to create load balancer services including profiles, redirects, VIPs, and pools. It provides the flexibility to create on-demand load balancer services, even enabling elastic load balancer functionality – the capability to add or remove resources to the load balancer and server farm when it reaches pre-configured thresholds.
- VPN services: For an enterprise that has remote users or business partner connectivity, it is possible to automate the provisioning of those users, their profiles, and all associated security postures. A simple collection of REST API calls can help to achieve this goal.

Another common myth revolves around the requirement to re-architect the physical network to solve for these use cases. This is not true; NSX runs over any physical network – L2 or L3 – as long as it can carry IP packets.

Historically, abstractions and overlays have resulted in simplification and efficiencies. NSX will facilitate future transitions to different underlay architectures – from core-aggregation-access to spine-leaf topologies – as it creates an overlay network, allowing an abstraction from the physical environment resulting in a simplified and more efficient design. When using VMware NSX, organizations can minimize the number of changes to the physical underlay, reducing time spent on manual configuration and number of errors when performing those changes.

Adopting new forms of automation for network and security environments requires a cultural change. How can organizations best get ready?

The network is the final obstacle to delivering dynamic IT systems that can more easily adapt to changing business requirements. Change is needed to increase networking and security efficiency, similar to what

has been done for server automation. Processes and procedures that have been developed over the past 15 to 20 years need to change when automation is used.

Organizations have a common ask – to help them on this transition to automation. Individual requirements differ, and the challenges are not only technical. Companies can move forward very smoothly in this transition with a focus on business priorities to determine how automation can help.

Organizations use many different sets of tools like PowerNSX, Ansible, Power CLI, or REST API client to automate the data center environment running NSX. The use cases may vary by industry or vertical, but the organization must learn how to best apply NSX technology and to develop processes to implement and maintain it. Several things must happen to make a project successful:

- **People** involved must be willing to learn new technologies to unlock organizational strengths to deliver on the major benefits of transforming how work gets done across the technology organization. The business executives must accept that the change is necessary to stay competitive.
- **Processes** must be reviewed, allowing employees to move away from unnecessary tasks and practices.
- **Technology** must be deployed to enable abstraction of networking and security functions from the underlying physical network. This will enable employees to better architect and manage their infrastructure moving forward.

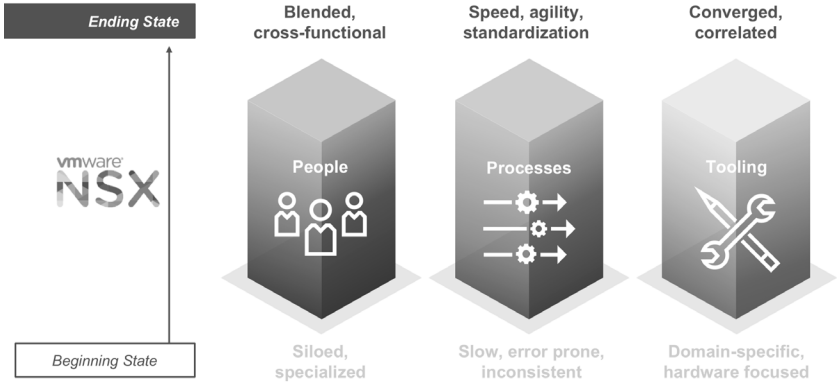


Figure 3.7 People, Process and Technology

The goal is to continually improve an enterprise's operations across each of these three dimensions.

A great first step towards an automation project is providing learning opportunities for organizations and individuals. To gain familiarity with NSX and learn how it can be automated, start a production pilot with a single use case and a few workloads. The VMware NSX Hands on Labs repository provides a sample use case – https://labs.hol.vmware.com/HOL/catalogs/.

Start simple and make incremental changes with a small, self-sufficient, and cross-functional team. Re-engineer a single process to start, then address others over time. Adjust it to a size that can be representative while still carrying lower risk, then expand the footprint of the initial use case before adopting new use cases. Train others and increase knowledge of NSX across more teams. The sooner this process is begun, the sooner the real benefits of automation can be realized.

NSX and vRealize Automation

In this new world where there is app for everything, where the information is available everywhere, and is consumed anytime from anywhere, organizations struggle with demands for new competencies. Those requirements call for being more agile, people-oriented, innovative, customer-centric, efficient, and adept at leveraging opportunities to change the status quo and tap into this new direction.

To achieve those requirements, IT teams are exploring new approaches to accelerate application development and delivery so they can respond better to business needs, speed time to market, and build competitive advantage. To gain agility, they need to expedite the delivery of infrastructure services and applications.

Many IT organizations have already achieved some efficiencies by virtualizing and automating compute resources with vRealize Automation. This has solved many problems in terms of provisioning virtual machines, creating profiles/templates, and deploying applications. Even with these capabilities, delivery of infrastructure – including networking and security – to development and production environments remains challenging to IT organizations; it continues to require a disproportionate amount of resources to properly execute these tasks. The last major pillar of automation remaining in the enterprise data center is networking and security.

By implementing vRealize Automation together with NSX, IT teams can extend capabilities to address slow networking and security provisioning without the need to change configuration on the physical layer every time a new service is provisioned. This integration allows automation of a complete application stack while maintaining compliance with the required security and connectivity policies. The result is improved flexibility and speed in responding to the needs of the business.

NSX and vRealize Automation have been tightly integrated since version 6.0 of vRealize Automation. From January 2018, the supported versions of vRealize Automation 7.x and NSX 6.3.x enable implementation of an IaaS platform where end users can deploy a complete application topology, in a secure and compliant manner, with a rich set of networking and security services through blueprint-based deployment. The end user does not need to understand networking or security as the configuration is pre-defined.

Current Product Interoperability (January 2018)

Figure 4.1 shows an example of interoperability confirmation for vRealize Automation version 7.3.0. Always refer to the VMware Official Product Interoperability Matrix for the latest information.

VMware NSX for vSphere	6.4.1	6.4.0	6.3.6	6.3.5	6.3.4	6.3.3	6.3.2	6.3.1	6.3.0
VMware vRealize Automation									
7.4.0	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figure 4.1 Product Interoperability Matrix

This integration highlights the power and flexibility of networking and security virtualization through automation. It demonstrates how the combination of NSX and vRealize Automation enables use cases like self-service IT, on-demand security, NAT, load balancing, and application isolation.

The power of both platforms combined delivers networking and security services through NSX with the complete service catalog and resource management capabilities of vRealize Automation – all through a standardized, repeatable process and on-demand delivery.

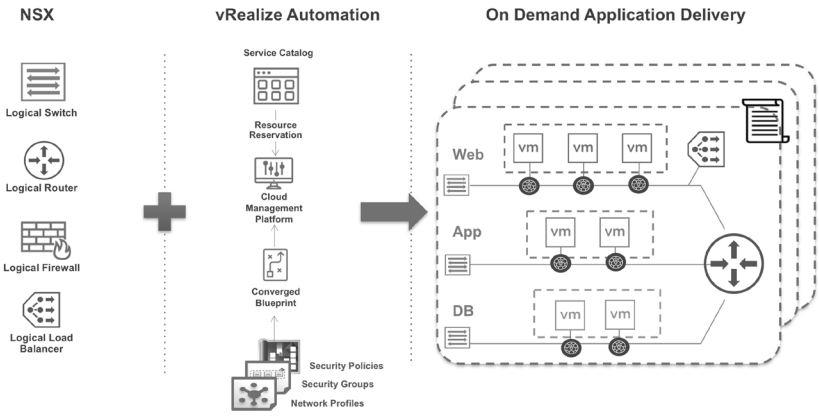


Figure 4.2 VMware NSX and vRealize Automation

The Converged Blueprint (CBP) Designer introduced in vRealize Automation 7.x allows practitioners to graphically design end-to-end blueprints. These blueprint support VMs, NSX security groups, security policies, security tags, on-demand load balancing, pre-created/on-demand networks, middleware, and application components. All this functionality can be consumed through a drag and drop canvas, allowing administrators to visualize the topology as it is being created.

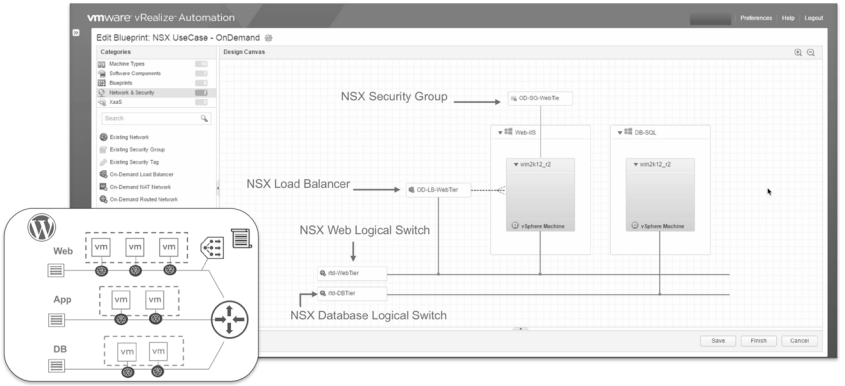


Figure 4.3 Converged Blueprint Designer with NSX

vRealize Automation Definitions

Every company in the world is looking to become successful; agility is key to making that happen.

In the cloud era, where everything needs to be ready quickly, processes that are very time-consuming are no longer permitted. IT departments aim to remove process inefficiencies using end-to-end automation tools that help IT better serve the business needs.

Inside enterprises, line-of-business (LoB) teams are comparing the services provided by IT to the simplified consumption model offered by public clouds (e.g., simply swipe a credit card). The bar has been set quite high; IT teams are struggling to provide the same agility, speed, and flexibility.

To address this demand, VMware has a powerful solution. vRealize Automation is well known in the industry as a cloud management platform that supports a multi-vendor, multi-cloud infrastructure. It allows IT services to be delivered across a wide range of vendors' products as well as virtual, physical, and cloud platforms. vRealize Automation defines, delivers, and governs the SDDC.

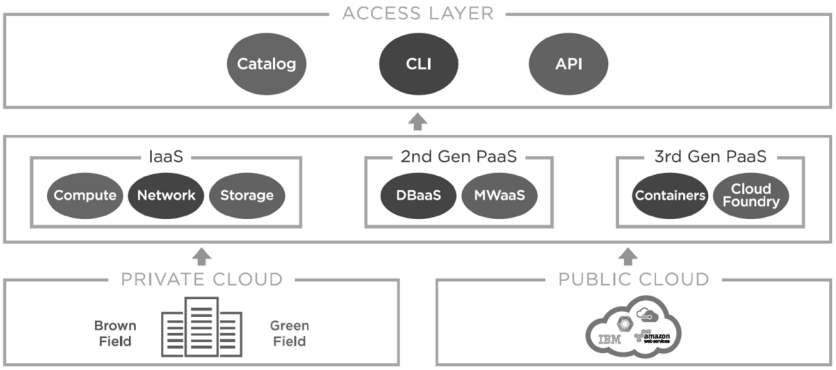


Figure 4.4 VMware vRealize Automation

vRealize Automation not only provisions the entire infrastructure (i.e., day one provisioning), but also automates the lifecycle management of IT resources and helps with day two operations (e.g., adds, updates, customizations).

As companies have more conversations around DevOps practices to accelerate application development, vRealize Automation gives IT teams the ability to provide easy access to traditional and cloud native application resources. This duality in the infrastructure is also known as bimodal IT – where a rich set of self-service capabilities is available for consumption, while at the same time there is the need to support developers through tools they already know (e.g., out-of-the-box integration with Puppet, containers, or other tools).

vRealize Automation Main Components

vRealize Automation comprises three main components:

- **vRealize Automation Appliance:** This is the main component – a preconfigured Linux virtual appliance delivered in the Open Virtualization Format (OVF). It hosts the web console, which comprises the self-service portal. The appliance includes a preconfigured instance of vRealize Orchestrator. vRealize Automation uses vRealize Orchestrator workflows and actions to extend its capabilities.
- **vRealize Automation Infrastructure as a Service (IaaS):** The installation files for the IaaS components are stored on the vRealize Automation appliance. The IaaS components themselves must be installed on a Windows machine and are responsible for the provisioning of IaaS resources. This Windows machine must be able to communicate with hypervisors, cloud environments, and physical hosts. IaaS uses a Microsoft SQL Server database to maintain information about the machines it manages, plus its own elements and policies.
- **Authentication Services:** The authentication services ship as part of the vRealize Automation appliance.

The requirements to deploy vRealize Automation will vary based on each scenario since enterprise deployments can be of varying sizes. A basic distributed deployment might improve vRealize Automation by hosting IaaS components on separate Windows servers, as shown in Figure 4.5.

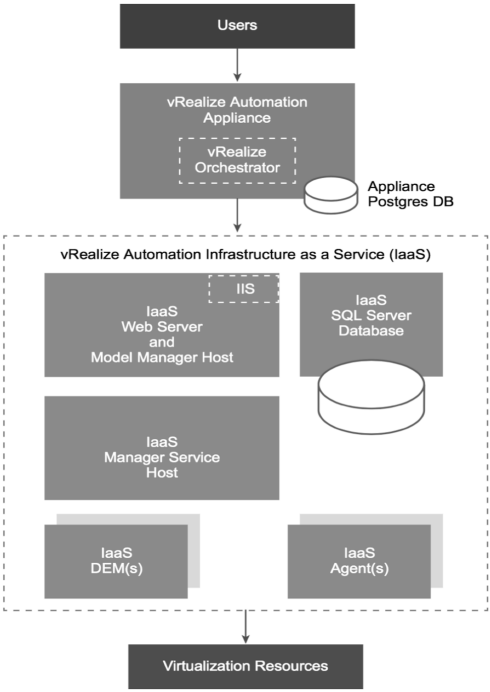


Figure 4.5 vRealize Automation Example Deployment

Many production deployments go even further with redundant appliances, redundant servers, and load balancing for even greater capacity. For more information about scalability and recommended design, please refer to the vRealize Automation installation guide at https://docs.vmware.com/en/vRealize-Automation/7.3/vrealize-automation-73-installation-and-configuration.pdf

Life Cycle Extensibility

vRealize Automation enables integration of solutions into existing IT environments using vRealize Orchestrator. Common examples include updating a CMDB when a machine is provisioned, calling an IPAM system for an IP address when provisioning a machine, or calling a ticketing system like Remedy when a machine operation is performed.

vRealize Orchestrator delivers out-of-the-box plug-ins and workflows – more than 30+ plug-ins and thousands of workflows – to make integration with existing third-party systems easy, achievable without professional services or customization services.

This extensibility allows organizations to create custom services – sometimes referred to as Anything as a Service (XaaS) – where vRealize Automation XaaS Designer can be leveraged to quickly deliver standalone workflows, day two operations, and other complex NSX services as-a-service. XaaS components can be dragged and dropped directly into a blueprint canvas.

Key Features

vRealize Automation has many different features and capabilities; highlights include:

- Support a self-service experience through a unified IT services catalog
- Policy-based governance ensures the right service level to meet specific business needs
- Unified blueprint model via Design Canvas drag and drop, command line, or API
- Enable the entire design and management process via API calls
- Support rapid iterations and DevOps principles by automating the release pipeline for infrastructure and application software
- Extensible platform that enables customization and extensibility at multiple levels across the IT ecosystem

- Design and automate the delivery of any IT services through service orchestration
- Out-of-the-box integration with VMware NSX to automate networking and security services

Common Use Cases for vRealize Automation

Some use cases are more common than others; this section presents a few popular examples.

Organizations are building private clouds through the adoption of VMware's SDDC, automating their entire infrastructure to be more agile with a fast go-to-market strategy. There are instances where these organizations already have vRealize Automation for compute; this streamlines creation of VMs and applications, however the networking and security pieces are still manually performed by IT admins. This results in disconnected processes and delays to application delivery.

There are other IT departments that want to create a catalog of standard operating systems or applications available on request through a self-service portal. Sometimes advanced governance is required, where approvals, standards compliance, and IT policy are mandatory on every automated blueprint.

Enterprises can also leverage a provisioning workflow that integrates with other systems like CMDB, ticketing system, backup, and monitoring. Another common IT request is a single pane of glass to integrate cloud environments, including Amazon Web Services (AWS) and VMware Cloud Providers.

A final example is the ability to offer other services via vRealize Automation extensibility, where tools like Puppet or Chef help with the lifecycle management of applications. This makes it possible to deliver infrastructure on demand for compute, storage, and networking.

NSX and vRealize Automation Benefits

IT Challenges to Automate Network and Security

IT teams have evolved when it comes to virtual and automated compute resources. Organizations can reduce the creation time of VMs to minutes through CMPs like vRealize Automation. Delivery of infrastructure is not only applicable to compute; it should always include networking and security. This aspect remains challenging for IT organizations and continues to require a disproportionate amount of resources for proper execution.

Organizations often spend a significant amount of time deploying the infrastructure to support applications. Businesses that are running hardware-based data center architectures face many challenges, including lengthy network provisioning, manual steps that can lead to configuration errors, and other costly processes. At the same time, as IT environments grow and diversify, technology teams are becoming more shackled by complexity. This complexity can distract IT teams from strategic priorities and projects, adding weeks or even months to project lead times.

When vRealize Automation is adopted without NSX, some of the challenges mentioned above may appear. Situations vary depending on size, use cases, and business needs; nonetheless, it is important to consider virtualizing both networking and security to achieve the best benefits from data center automation.

Organizations who achieved a good level of automation started virtualizing the infrastructure – not only compute, but also networking, security, and storage.

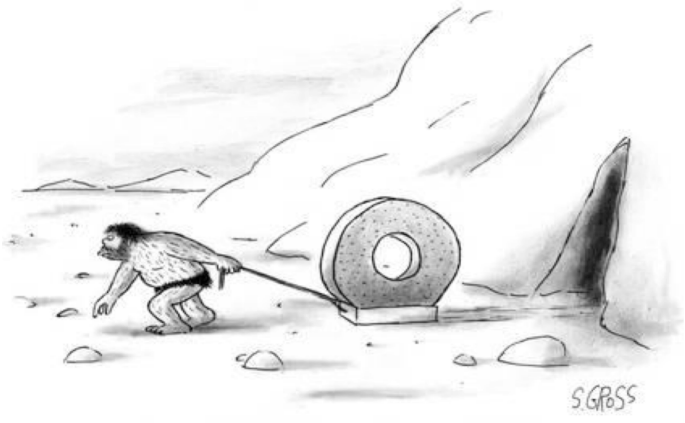


Figure 4.6 Cave man without the right tool

Virtualize and Automating Compute, Network and Security

vRealize Automation and NSX enable IT teams to drastically reduce manual efforts and eliminate bottlenecks by automating the provisioning of all infrastructure and application components. To achieve those results, the adoption of a software-based approach becomes necessary – one that does not rely on physical hardware to provide such capabilities. IT organizations can replicate all networking and security functions through software embedded in the hypervisor, regardless of the topology of their physical infrastructures.

Benefits of a Software Based Approach

Organizations that are using a software-based approach to automate IT data centers are seeing great results. Examples include:

- **Multi-machine topology:** Provides advanced networking topologies to cloud users as part of the vRealize Automation Catalog
- **Full automation:** Automated deployment of NAT topology with connected VMs
- **Consistent policy:** Repeatable deployment of customer environments to help diagnose technical issues

- **Isolation between environments:** Each deployment is completely self-contained
- **Full VMware SDDC:** Complete VMware stack with vRealize Automation, NSX, and vSphere

In this integration, the networking and security configuration can be done directly within the vRealize Automation User Interface. It supports flexible network topologies with application context for

Figure 4.7 vRealize Automation and NSX - Benefits

micro-segmentation along with extensibility for deploying specific configurations. This extensibility can be performed by the consumption of NSX APIs, as described in Chapter 3.

vRealize Automation and VMware NSX can help customers to:

- Reduce provisioning time, improve resource utilization, and eliminate error-prone processes by automating the delivery and management of production-ready infrastructure and application components
- Reduce the operational bottlenecks caused by manually configuring and provisioning security policies, networking, and infrastructure resources for applications, allowing IT staff to deliver applications and services faster
- Meet security and compliance requirements by templating networking and security services and policies to deliver standard, consistent applications

Business benefits

Enterprises that have already deployed both solutions have identified some business advantages when comparing to traditional approach like all hardware based.

- **The combination of VMware vRealize Automation and VMware NSX has allowed organizations to double or even triple in size without increasing central IT staff**

This is huge for some enterprises, as they can address business demands without increases to OPEX by hiring more technical people to get the job done. For big organizations, it represents efficiency as automation takes care of routine tasks, allowing IT teams to spend less time in the data center and more time focusing on critical issues for the business.

- **Developers can now provision a full environment faster, with zero reliance on IT**

Inside organizations it is common to see shadow IT, where developers consume their needed infrastructure away from their own IT company environment. This usually happens because IT admins are busy keeping the business running and cannot deliver what developers need. The ability to directly service developer requirements allows organizations to improve productivity and accelerate service time to market.

- **Costs reductions through automation and virtualization**

Virtual infrastructure dramatically reduces costs and simplifies both management and day-to-day operations. Enterprises are focused on minimizing hardware costs, increasing the lifecycle of their existing hardware equipment, building the platform within the hypervisor, and virtualizing not just compute, but also networking and security. This means buying more capacity as the environment grows, not buying hardware in an upfront model for the following years.

- **Simplified operations**

A unified management interface provides a single pane of glass, making it easier to operate and troubleshoot the whole environment. Organizations have improved the overall infrastructure visibility in ways that cannot be achieved using a traditional centralized hardware approach.

- **Better security means less breaches**

vRealize Automation and NSX can help applications be deployed already hardened, meaning all security parameters are in place and all unnecessary ports closed. This is hard to achieve without the micro-segmentation concept. This integration can bring such a concept to a higher level, not just securing but also automating the security of all applications. With this level of control, organizations can reduce the overall attack surface footprint along with associated expenses related to security breaches and stolen data.

NSX and vRealize Automation Integration

The first step in data center automation must be to virtualize the entire infrastructure – compute, storage, and networking – creating an abstraction model from the physical layer to obtain agility, rather than relying hardware models, types, or software features. Second, native integration between platforms like vRealize Automation and NSX can bring added value. Automation can fast-track NSX adoption by simplifying the deployment, operations, and management.

NSX and vRealize Automation are two major products from VMware heavily adopted by many organizations. Businesses are using vRealize Automation to build cloud environments and NSX for the networking and security services. Individually they are remarkable products; when integrated, an organization can unlock the real benefits of automation.

Both products have seamless out-of-the-box integration to deliver a variety number of use cases. This native integration highlights the value of NSX when combined with automation and self-service. It embeds the NSX constructs directly into the infrastructure and application level blueprints, eliminating the need for networking to be provisioned separately outside of vRealize Automation. This demonstrates how VMware brings together compute, storage, networking, and security virtualization to provide a comprehensive software based solution.

As of version 7.3, vRealize Automation provides deeper integration with VMware NSX. It supports native API-based integration between vRealize Automation and NSX to expose more capabilities and improve overall performance. NSX is now a dedicated endpoint, providing logical separation from other vSphere endpoints. This does not eliminate vRealize Orchestrator from the picture; it is still an area of investment for vRealize Automation/NSX extensibility as discussed in later examples.

Why this integration is helping organizations?

- **Network connectivity**

As part of the application provisioning process, vRealize Automation blueprint policies specify the configuration of new logical switches and routers, describing how they will be connected to the perimeter gateway. Organizations can automate connectivity to existing or on-demand networks. Each business group can be provided with reserved network connectivity between the virtual and physical world, or specific mission critical applications can be configured with dedicated virtual switches and routers depending on their performance and reliability needs.

- **Application Security**

As part of day one provisioning, vRealize Automation places each application in the appropriate NSX security group protected by firewall rules, intrusion detection integration, and agentless anti-virus. This granular level of isolation keeps traffic to specific group environments (e.g., development, test, production), isolated to the individual application, or between application tier levels.

- **Better availability and performance to any application**

The integrated solution improves application availability through the dynamic configuration of network load balancers in the context of deploying or updating application configurations. The NSX load balancer can be used in all phases of the application lifecycle – development, staging, or production – without requiring expensive

physical hardware or manual configuration of legacy load balancing components. All traffic between virtual machines on the same host will remain in the host; there is no requirement to exit to consume load balancing or firewall services.

- **Automated Application Network Deployment**

As part of the application provisioning process, network profiles in vRealize Automation define the type of virtual network that will be configured – NAT, routed, or external. The infrastructure service blueprint specifies the network profile for each type of infrastructure machine being provisioned.

What enterprise are looking for out of this integration?

There are three major areas – IT automation, developer cloud, and multi-tenant infrastructure – where this integration can bring value. Each one offers different reasons to use both products. In some instances, all three may interact as part of a transformational project, helping provide the IT at the speed of business.

IT Automation

Automation allows for a quick, secure, and automatic deployment of applications and services when and where they are needed. It works across organizational and geographic boundaries, delivering business value in minutes rather than days, weeks, or months.

Developer Cloud

Automation enables developers to use a unified API for on-demand networking and security services. It allows IT to deliver more value to the business without the constant need for hands-on involvement. Organizations can maintain parallel and secure development, test, and production environments on a common infrastructure.

Multi-tenant Infrastructure

Automation empowers organizations to provision and deliver networking and security services to multiple tenants on a shared infrastructure, increasing asset utilization and lowering overall infrastructure costs

Figure 4.8 Automation Pillars for vRealize and NSX Integration

vRealize Automation Network Profiles

Network profiles define how new VMs are connected to the network. They allow consumption of existing networks or on-demand logical switches.

Multiple types of network profiles are available in vRealize Automation – external, routed and NAT. These network profiles can be combined – with the exception of routed and NAT together – in the same blueprint to provide different types of topologies.

External Network Profile

The external network profile allows efficient management of IP allocation by sharing a common network across deployments. It can be used by pre-created networks, either VLANs or logical switches, and is used for uplinks on routed and NAT network profiles.

NSX load balancer is supported in one-arm mode. The IP addressing provided by vRealize Automation can use an existing DHCP or IPAM service.

When multiple deployments share the same network, overlapping IPs cannot be used; however, security can be provided by micro-segmentation.

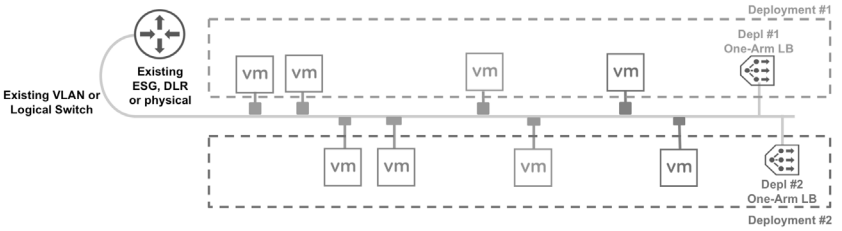


Figure 4.9 External Network Profile

Routed Network Profile

Routed networks contain a routable IP space divided across subnets that are linked together using a distributed logical router (DLR). Logical switches are created during blueprint deployment and have a unique subnet range, carved out from a pool.

The VMs provisioned with routed networks which share the same routed network profile can communicate with each other and the external network. All security features are supported (i.e., pre-created and on-demand security groups, security tags, and app isolation) while micro-segmentation allows for isolation across different tiers of a blueprint.

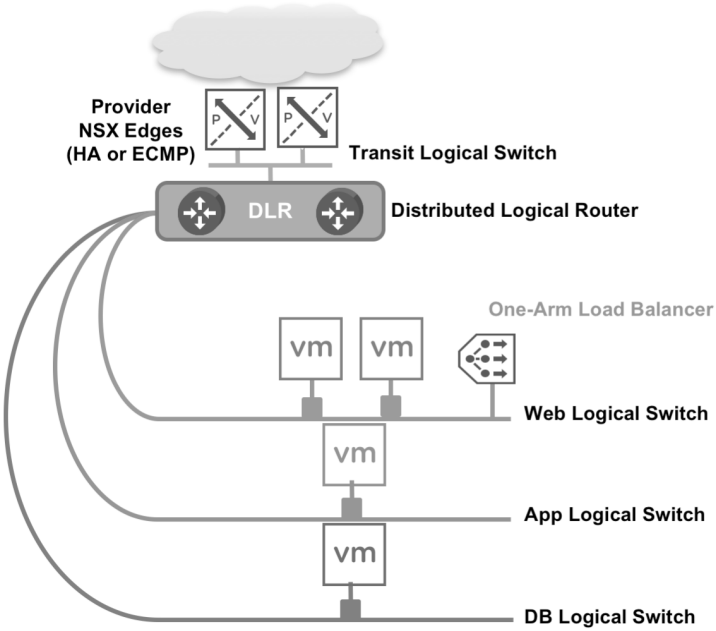


Figure 4.10 Routed Network Profile

NAT Network Profile

NAT networks use one set of IP addresses for external communication and another set for internal communication. With one-to-one NAT networks, every virtual machine is assigned an external IP address from the external network profile and an internal IP address from the NAT network profile. With one-to-many NAT networks, all machines share a single IP address from the external network profile for external communication.

A NAT network profile defines local and external networks that use a translation table for mutual communication.

The NAT rules are applied only on the ESG (Edge Services Gateway) uplink interface. NSX load balancer is supported only in inline mode. If used, additional VIPs are carved out from the network profiles and assigned as secondary IP addresses, different than the primary ones used in the network topology.

All security features are supported: pre-created and on-demand security groups, security tags, and app isolation.

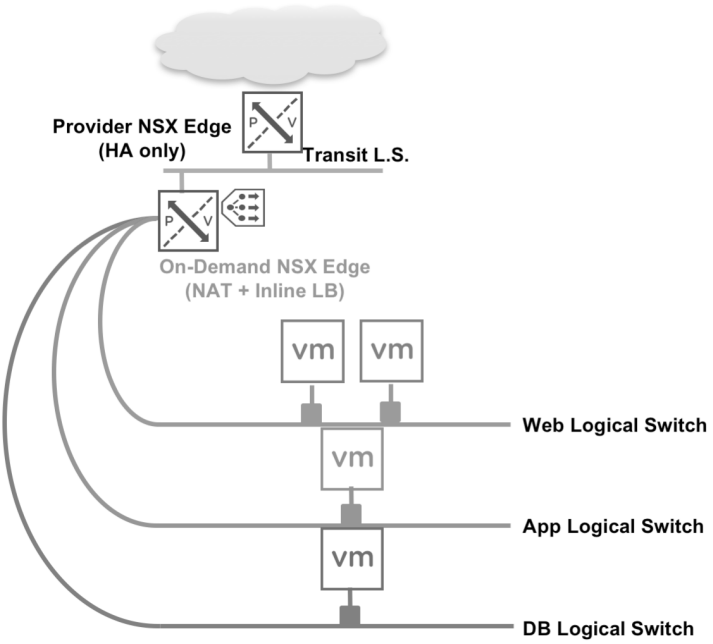


Figure 4.11 NAT Network Profile

Two types of NAT network profiles are supported – one-to-one and one-to-many.

One-to-One NAT Network Profiles

For each VM created on a one-to-one NAT network, one external IP is configured on the ESG uplink. Both SNAT and DNAT rules are created on the ESG uplink.

NSX ESG firewall rules provide granular filter access for both intra-app traffic as well as external communication to and from the VMs and distributed firewall. DHCP is not supported using 1:1 NAT network profile.

One-to-Many NAT Network Profiles

For each one-to-many NAT subnet created, one external IP is configured on the ESG uplink. An SNAT rule is configured on the ESG for outgoing traffic; no corresponding DNAT rule is configured as with one-to-one NAT.

NSX ESG FW rules allow intra-app and outgoing traffic access, while VMs can only be reached from the outside via a load balancer. DHCP with NSX ESG is supported on one-to-many NAT network profiles.

On Demand Load Balancer

vRealize Automation deploys an NSX ESG and auto-configures the load balancing policy. Multiple virtual servers can be configured per blueprint on the same load balancer, with a dedicated VIP used for each virtual server.

There are two types of supported topologies with NSX load balancers – one-arm and in-line.

One-arm load balancers are deployed with external and routed network profiles. All member VMs of each load balancer pool as well as load balancer VIPs must be on the same network. Every time a new one-arm load balancer is created, a new NSX Edge is deployed.
In-line load balancing is configured when using NAT network profiles.

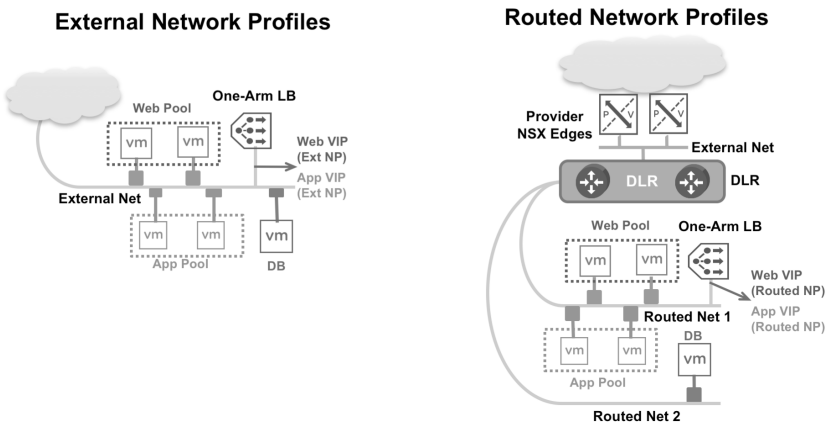


Figure 4.12 One-Arm Load Balancer with External and Router Network Profile

Load balancer VIPs can be attached to either internal or external interfaces of an ESG. Mixing VIP placement is supported (e.g., web VIP on external, app VIP internal network)

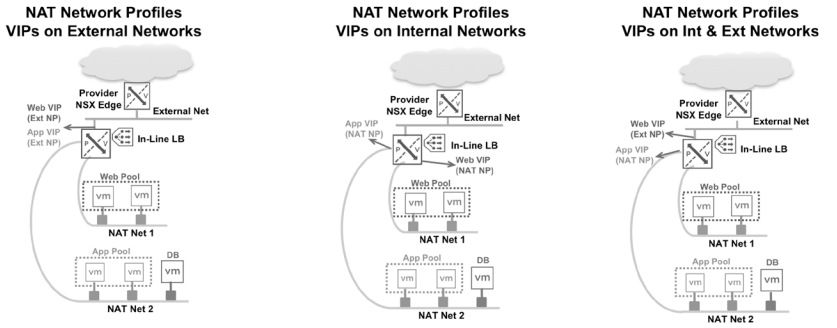


Figure 4.13 Inline Load Balancer with NAT Network Profile

Use Cases for vRealize Automation with NSX

Examples are provided below of the most common use cases for an automated environment, but organizations continue to find new and interesting use cases for this integration between VMware NSX and vRealize Automation.

Use Case: Automating Production

Delivery of productions workloads is challenging. There are many steps involved in a process that can take days, weeks, and sometimes months to release a new application. In addition to service tickets and manual configuration, lack of control is one of the top concerns observed in the organizations due to its potential for security issues.

A typical process is detailed in Figure 4.14:

- App owner requests a service
- Infrastructure admin requests IP addresses from the network team

- Infrastructure admin deploys VMs
- Security team configures firewalls
- Network team configures VLANs and load balancers
- If something goes wrong, the loop restarts

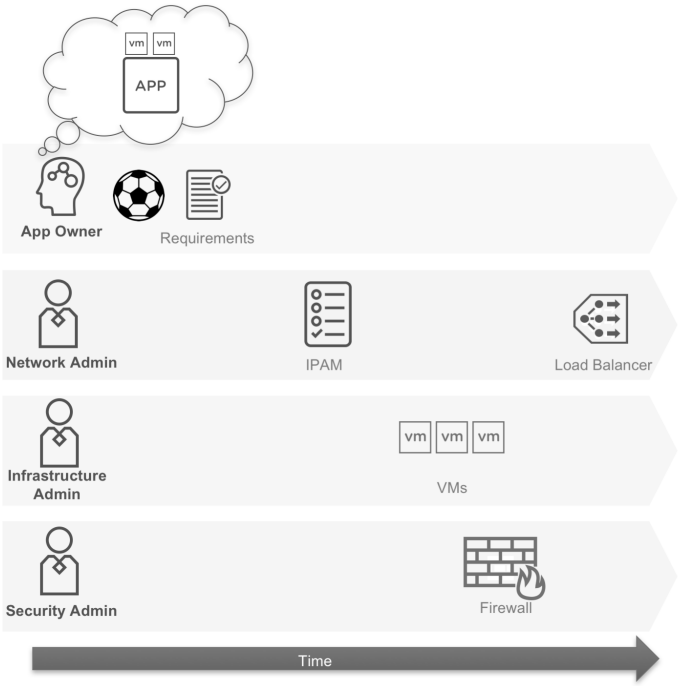


Figure 4.14 Application Delivery

In such an example, many steps and interactions are necessary to deploy an application. Additionally, these processes rarely consider the idea of cleanup during decommissioning of the same application.

With NSX and vRealize Automation, enterprises are creating multi-layer applications with a single click. They can deploy applications with network, firewall rules, and load balancers auto-configured at runtime through a repeatable process applied to each such instance. At the time of deletion, all the network, firewall, and load balancer

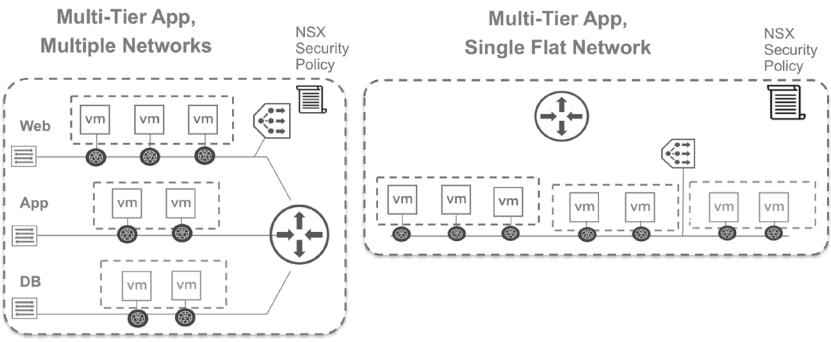


Figure 4.15 Application Deployment Topologies

configurations are deleted automatically; the IT admins – network, virtualization, and security – do not need to spend time cleaning up. For some scenarios, it is possible to build a catalog item with built-in approval policies and advanced services like NSX guest introspection for anti-virus/anti-malware protection. When a request is made for that item in the catalog, all features and configurations are provided, eliminating extra steps like post-deployment installation and tuning.

Use Case: Automating Development & Test Environments

Development and test environments are often manually provisioned, requiring significant amounts of labor and time to accomplish simple tasks. Some organizations have already eliminated scripts and are using vRealize Automation to create VMs in the compute infrastructure. This is still not sufficient, as networking and security components (e.g., VLANs, firewalls, load balancers) still require manual configuration.

Enterprises are also looking to automate dev/test environments to be used like a conveyor belt. IT admins could place pre-provisioned VMs in the test environment, then roll them into the development environment and further into production using the same parameters (e.g., IP addresses, firewall rules). This is possible using vRealize Automation and NSX to create NAT network profiles to move in and out of a fenced network environment for dev/test.

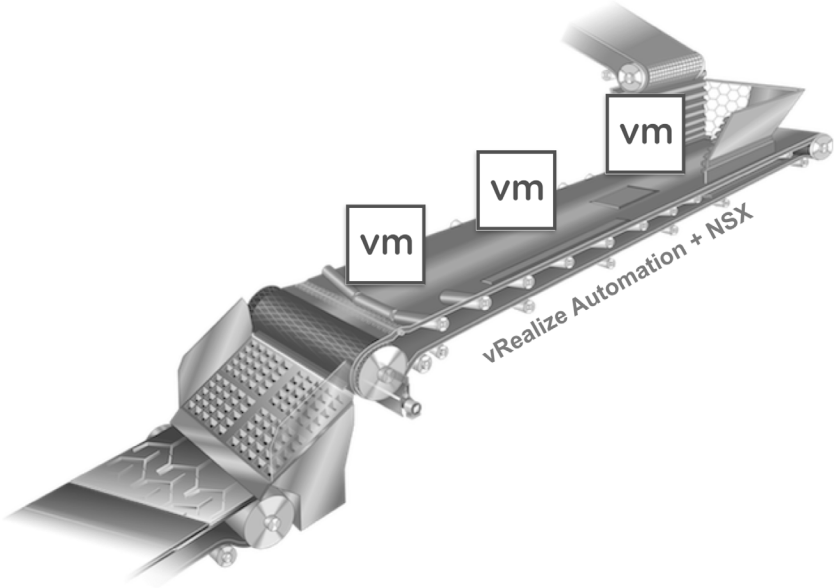


Figure 4.16 Conveyor Belt for Dev/Test/Production

Use Case – Multi-Tenant Infrastructure

Organizations are also using integration to deliver multi-tenant private cloud. Network separation is a key requirement for full multi-tenancy. Solutions based on physical hardware models cost a lot of money due to scalability and segmentation factors. The multi-tenant concept is associated with cloud providers, but can also be used by enterprises looking to build different environments (e.g., development, test, departmental) using the same shared infrastructure.

NSX and vRealize Automation can perform separation and isolation of different tenants using the network profiles embedded in the solution. It is possible to use NAT for overlapping IP addresses across networks that need external connectivity. An NSX ESG can be deployed for each tenant, and stateful services like NAT, firewall, and load balancer can be provided per each tenant. NSX firewall policy and isolation can also be delivered per tenant, where organizations can automate and scale multi-tenant environments to support their business.

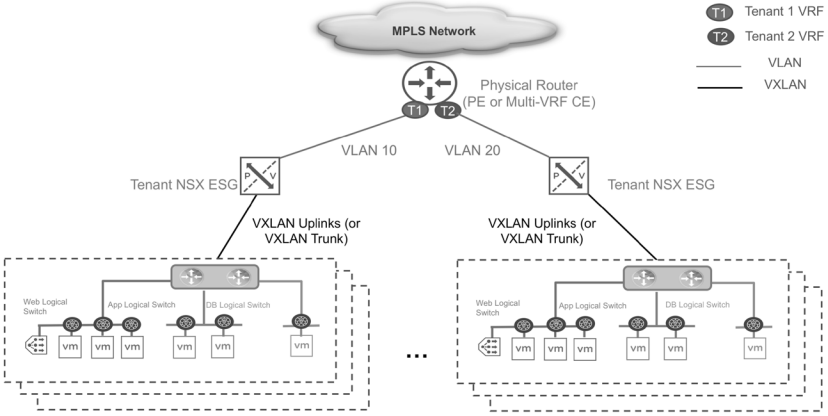


Figure 4.17 Multi-Tenant Topology

Use Case – On Demand Micro-Segmentation

Security is mandatory in all modern data centers. Perimeter security is no longer sufficient (e.g., internal threats, multiple tenants, contractors) while VM sprawl requires more granular security controls than those provided at the perimeter. Security teams struggle to manually configure firewalls at the pace of cloud initiatives, and it is hard to maintain consistency across the data center. Additionally, auditing and control are even more critical in today's dynamic IT environment.

The integrated solution provided by vRealize Automation and NSX delivers an automated way to create and consume pre-created security groups to isolate tenants or applications through micro-segmentation.

In vRealize Automation, security group membership can be defined both at:

- Reservation level by cloud admin for all VMs deployed in a reservation
- Blueprint level for specific VMs within an application

When VMs are created, vRealize Automation will add them to the selected security groups. NSX will also automatically remove VMs from the security group at deletion time.

It is possible to attach existing security tags to provisioned VMs. Existing NSX security groups can have a dynamic membership based on security tags. Third party solutions can also leverage the tag information.

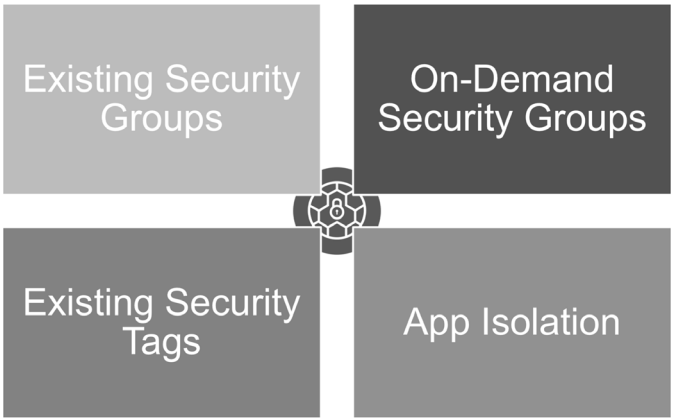


Figure 4.18 vRealize Automation & NSX: Security Options

App isolation is another powerful feature that can be leveraged in this integration. It provides an optional first level of security with the following functionality:

- All inbound and outbound application access is blocked
- Intra-application traffic is permitted
- Other policies are applied at a higher precedence to permit/deny selected traffic

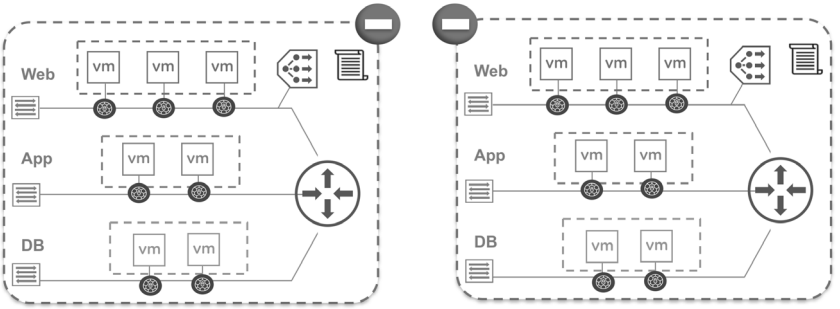


Figure 4.19 App Isolation Topology

App isolation configuration is simple. During the blueprint configuration, there is a checkbox to enable automatic configuration.

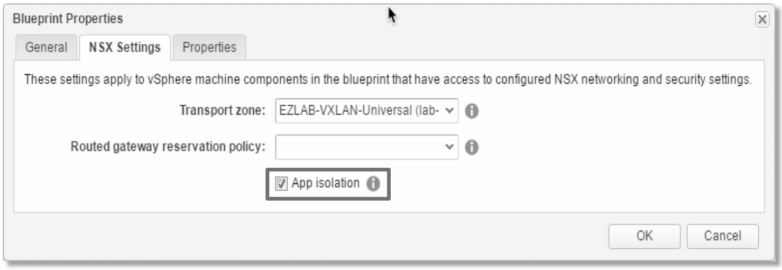


Figure 4.20 App Isolation configuration on vRealize Automation

vRealize Automation is an excellent fit for automating micro-segmentation and provides application context to enable a policy-based approach to security.

Day Two Operations with vRealize Automation and NSX

Day two operations functionality enables the cloud/IT admin, whenever they need to make a change to a device on pre-designed and implemented blueprint, to perform those changes without redeploying or rebuilding that specific blueprint.

vRealize Automation integration with NSX enables organizations to realize operational benefits, simplifying tasks and reducing maintenance windows to implement those changes.

Load Balancer

IT admins have an enhanced control of NSX-provisioned load balancers. They have control for customization per blueprint (e.g., algorithms, persistence, ports, health monitors) or creation/change of virtual servers – and their associated configuration – on existing NSX ESGs.

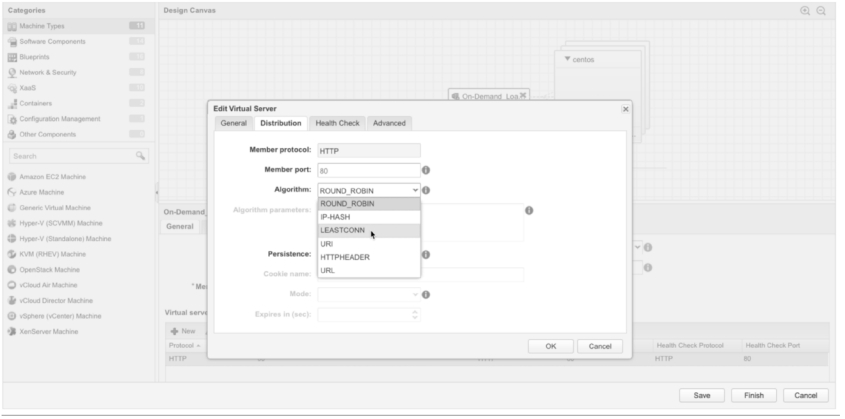


Figure 4.21 NSX Load Balancer Configuration

NAT

On-demand NAT port forwarding rules can be configured during application design, providing greater flexibility and feature parity with NSX management. Rules can be ordered, added, and removed – even after the creation time.

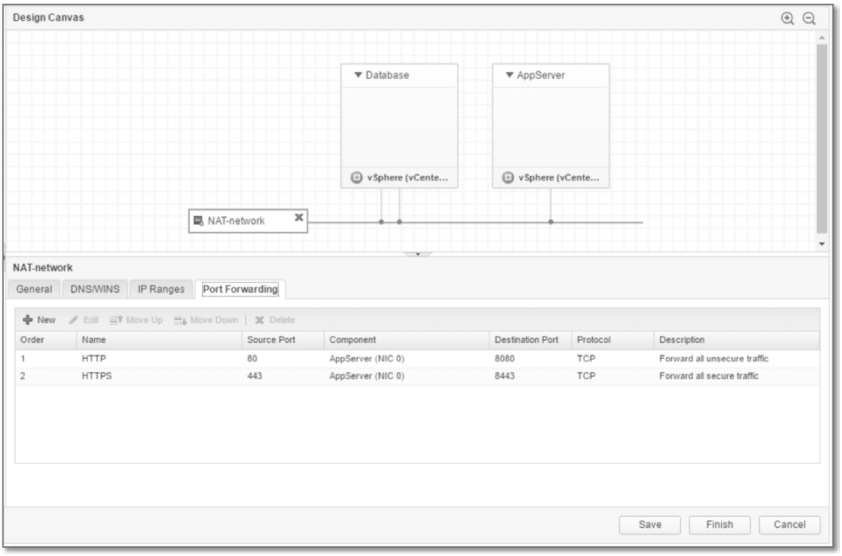


Figure 4.22 NSX NAT Configuration

Security

Security is never static; it always requires changes. Within vRealize Automation, it is possible to view and change active NSX security groups and tags. With this capability, cloud/IT admins have the power to add or remove existing NSX security groups or tags from running applications, or to edit NSX security policies for existing applications.

NSX and OpenStack

Consider the following scenario – a customer decides it is time to implement OpenStack to build their cloud, tests in the lab, evaluates the various distributions available, and hires specialized OpenStack staff. When the environment goes into production, they realize that Neutron is not integrating with their physical network.

Figure 5.1 Networking challenges in OpenStack Clouds

If this story is of concern, this chapter is critical to understand the challenges of networking with any OpenStack distribution and how VMware NSX is the missing piece for cloud infrastructure.

The biggest challenges for OpenStack cloud implementations are automation, integration, and orchestration of the required networking and security components at the physical infrastructure layer. The main difficulty is that these environments are extremely heterogeneous and most of the devices do not have an open and programmable interface for configuration. These challenges defined the initial method of running OpenStack; manually pre-provisioning the network and using only basic functionalities when implementing security services (e.g., iptables for L2-L4 security).

With the rise of network virtualization solutions and the evolution of Open vSwitch (OVS), some of these challenges have been solved. It is possible to create an abstraction layer from the physical elements of infrastructure and automate the virtual network through the programmable interface of network virtualization solutions. Commonly such solutions use an overlay mechanism to create virtual networks without the need for change in the physical (i.e., underlay) network.

The Neutron project, responsible for managing all OpenStack cloud security and networking services, has been undergoing constant modifications, especially around the need for more advanced functionality such as dynamic routing, VPN, and firewall. These constant changes impacted maturity, consistency, and resiliency.

Growth without planning has brought major challenges to the Neutron project. The most debated topic in the community today is whether the architecture of this project needs to be reworked in order to simplify its use and improve its integration with network virtualization solutions.

Few enterprises today are using OpenStack in production environments without a network virtualization platform. Those that have no platform in place often face major challenges similar to those previous discussed.

The benefits that NSX brings to Neutron are:

- Agility
- Mobility
- Security
- Multi-tenancy
- Simplified operations

Each of those topics will be discussed at a deeper level through the rest of this chapter.

To integrate with Neutron, VMware NSX has an open plugin available at Github (https://github.com/openstack/vmware-nsx) that can be used by any OpenStack distribution or implementation. This plugin translates the Neutron APIs calls into NSX REST APIs calls to build the networking and security services.

VMware NSX supports a variety of OpenStack distributions regardless of the underlying hypervisors.

OpenStack Definitions

OpenStack is an open source software platform usually deployed in an IaaS model for the creation of public or private clouds. Its purpose is to control compute, storage, networking, and security resource pools with management performed by an integrated panel or consumed through APIs.

OpenStack turns the sets of hypervisors within or across multiple data centers into pools of resources. These pools can be managed and consumed from a single dashboard or through APIs. Both administrators and users can perform their tasks in an easy and fast manner.

OpenStack is the control layer that sits above all the virtualized layers, providing a consistent way to access everything regardless of the hypervisor technology used underneath.

All tasks – from managing networks to handling storage – becomes easier as OpenStack abstracts the underlying layers, allowing users and administrators to consume resources with a consistent set of APIs from a common dashboard.

OpenStack was created in mid-2010 by Rackspace and NASA, and has been managed by the OpenStack Foundation since 2016. Today it has more than 60,000 registered members across a diverse set of verticals. The purpose of the foundation is to protect, develop, and promote OpenStack for the community using the solution.

Every six months, a new release of the OpenStack software is provided by the foundation, bearing the name of a city chosen by the community. The August 2017 OpenStack release was identified as Pike, with the following version designated to be named Queens.

OpenStack is a combination of several open source software projects that work together to bring simplicity and agility to the whole. Amongst the main projects, key components include:

- **Nova**: Responsible for management of the compute infrastructure and for creation and lifecycle of all instances in the OpenStack cloud
- **Glance**: Responsible for searching and storing images of the instances in the OpenStack cloud
- **Cinder**: Responsible for storing data in the OpenStack cloud; interacts with storage devices to provide an abstraction layer for end users

- **Swift**: Responsible for storing objects in the OpenStack cloud with high capacity
- **Neutron**: Responsible for all network components and security services of the OpenStack cloud
- **Keystone**: Provides authentication and authorization for users of the OpenStack cloud multi-tenant environment
- **Heat**: Template-based orchestration service that provides greater agility in creating and provisioning other OpenStack cloud projects
- **Ceilometer**: Responsible for collecting, normalizing, and transforming data produced by other OpenStack cloud services
- **Horizon**: Dashboard that provides a web based user interface to OpenStack services

A useful tool to understand the maturity and definition of each project is the OpenStack Foundation project navigator (https://www.openstack.org/software/project-navigator/). It is aimed at helping users make informed decisions about how to consume the software. Data used to power the project navigator website is provided by the OpenStack technical and user committees. With project navigator, enterprises can understand Neutron usage, project maturity, adoption rates, and problems/challenges specific to every project of interest.

The OpenStack architecture was created with the idea to make each project as independent as possible. This gives users the option to deploy only a subset of the functionality and integrate it with other systems and technologies that offer similar or complementary functions. This independence should not mask the fact that a fully functional private cloud is likely to require virtually all this functionality to operate smoothly or the importance of tightly integrating the elements.

A typical OpenStack implementation will integrate most projects. The most popular core projects interact with all the components in the system. Horizon is the graphical UI that administrators can use to manage all the projects. Keystone handles the management of authorized users. Neutron defines the networks that provide connectivity between the components and security policies between the instances.

Nova could be considered the main component of OpenStack, handling the orchestration of workloads. Its compute instances require some form of persistent storage, either block-based (e.g., Cinder) or object-based (e.g., Swift). Using this example, Nova will interface with

Glance to launch an image retrieved from Swift; this integration is built in and can be achieved without extensive customization.

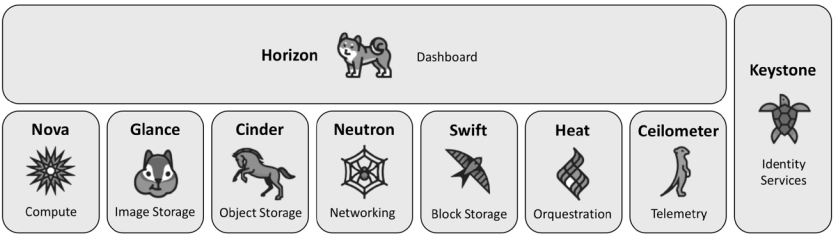


Figure 5.2 OpenStack Main Projects

The OpenStack Foundation created a committee named OpenStack DefCor, now called the Interop User Group, that establishes the required minimum feature set of projects and common APIs for OpenStack products. Their guidelines apply only to commercial uses of the OpenStack name.

There are several resources available to provide deeper understand about OpenStack and its projects. The primary resource should be the official site of the OpenStack Foundation – (www.openstack.org). Additionally, every six months the OpenStack Foundation hosts the OpenStack Summit, a global gathering of presentations and discussions focused on the OpenStack ecosystem.

Neutron Concepts and NSX Integration

Neutron was designed by OpenStack to provide networking-as-a-service between interface devices (e.g., virtual NICs) managed by other OpenStack services (e.g., Nova). Starting in the Folsom release, Neutron became a core project of OpenStack.

Neutron Project Concepts

Networking in OpenStack was designed to create and manage all networking and security components for the OpenStack cloud including networks, ports, subnets, security rules, and other items of interest to OpenStack projects. Neutron is the project responsible for providing an API that enables OpenStack administrators/users to define the connectivity and build the multi-tenant cloud. This framework allows each user to determine the unique characteristics for their specific environment.

The Neutron API also offers ways to configure a variety of services including routing, NAT, load balancing, firewalls, and VPN. Plug-ins are incorporated into Neutron to accommodate different networking solutions, offering simplicity and flexibility to OpenStack networking and the cloud.

For advanced networking topologies, users can create, configure, and design their networks and subnets as desired. Other OpenStack services like Nova can attach virtual instances to ports on the networks and provide connectivity to the instances. OpenStack Neutron supports multiple private networks and enables each user to choose their own IP addressing scheme.

Neutron Architecture

Since its creation, the Neutron architecture has been difficult to understand and manage. Starting from the Kilo release, the Neutron community decided to decompose Neutron for simpler development and maintenance. Neutron will remain focused in core services at L2 and L3, while layer 4-7 services will be pluggable, creating an extensible data model. This change generated a broader path for more companies and individuals to contribute for specific components on the Neutron project. NSX leverages this architecture model to offer its unique capabilities and features.

Components

Provider Networks

Provider networks are L2 segments that offer connectivity to the instances with optional DHCP support and metadata services. Those segments are mapped into existing L2 networks in the data center. The most common approach is using 802.1q tagging, separating each provider network into a separate VLAN.

The general idea of L2 provider networks is to offer simplicity and reliability for the cloud. Generally, only OpenStack administrators can create, delete, and update provider networks. This restriction is in place as configuration of the physical network infrastructure is usually required as well.

In the most common use case, each tenant will be connected to one or more provider networks via a physical external router or a Neutron-owned router. This connection offers external connectivity to reach another tenant or the Internet.

Tenant Networks

Tenant networks enable users to create, delete, and modify their networks whenever they want, without administrator intervention. Tenant networks are virtual and normally use an overlay protocols like VXLAN, GENEVE, STT, or GRE. An overlay protocol is preferred to a traditional VLAN to avoid physical network configuration and scalability limitations of the number of available VLANs.

By default, tenant networks are isolated and not shared with other tenants.

Subnets and Subnet Pools

Subnets and subnet pools are used to assign IP addresses for ports of the instances. Each user can create subnets and assign IP address without restriction. Some technologies and vendors support no-NAT implementations. For these situations, IP addresses from subnets should be different from each other, and subnet pools can be used to avoid conflicts.

Routers

Neutron routers offer tenants L3 services like routing and NAT. Users can create, delete, or update Neutron routers and they are not shared between tenants.

Tenant networks requires Neutron routers to connect to external networks. DHCP and metadata services are commonly associated with tenant networks. Those networks use private IP ranges (RFC1918) and connect to provider networks with a source NAT rule created in the Neutron router.

Floating IPs are also commonly used to enable the access to the instances from provider networks via a destination NAT configuration on the Neutron routers.

Neutron routers can be considered the edge between the tenant space to the external world including the Internet or other tenants.

Neutron implements routers using L3 agents that reside on the network nodes. Depending on the size of environment, oversubscription and redundancy should be considered when designing the network.

Figure 5.3 exemplifies how Neutron routers interact with provider and tenant networks.

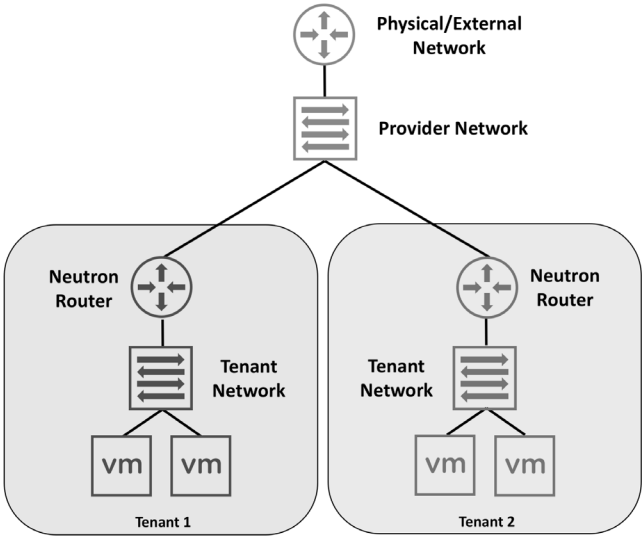


Figure 5.3 Neutron Components Interaction

Security Groups

Security groups offers firewall rules at port level for the instances. Rules can be configured for inbound and/or outbound traffic. By default, all security groups use an implicit deny rule; therefore, it is necessary to specify the rules for all the instances in the tenant.

The firewall driver translates security group rules into a configuration that the infrastructure uses technologies like iptables. Upcoming examples will discuss the efficiencies and benefits of NSX security along with the process of translating security group rules into NSX components.

DHCP

DHCP is an optional but commonly used service that manages IP addressing for instances that connect to provider or tenants networks.

Metadata

Metadata is an optional service that offers an API containing information of the networking and security components to other projects.

Services

VPNaaS
VPNs offer secure connectivity access to the external world. VPNaaS is a Neutron extension that provides an API to configure, delete, or update tenant VPN services.

LBaaS
Load balancers automate creation, deletion, and updating of resources based on demand. LBaaS offers an API to provision and configure load balancers for tenants.

FWaaS
Neutron's firewall-as-a-service (FWaaS) offers an API to create, delete, or update firewalls on demand for tenants as a first layer of protection compared to security groups.

What are Neutron Plugins?

The Neutron plugin architecture is divided in two categories:

- Core API: implements the core L2 abstracts like ports and networks
- API extensions: implement other services, including routing and LBaaS

Neutron allows the use of a set of different backends called plugins that work with a variety of networking technologies. These plugins may be available separately or distributed as part of the main Neutron release.

Neutron uses plugins to deal with hardware and software diversity at different layers of the OSI model.

Two main approaches are used for Neutron plugins:

- Monolithic plugins: The first plug-in implementation, but still commonly used. Monolithic plugins offer a complete set of features in a single package.
- Modular Layer 2 (ML2) plugins: A new modular Neutron core plug-in, introduced in the Havana release. The ML2 plugin architecture supports type drivers to support multiple networking technologies and mechanism drivers to facilitate access to the networking configuration in a transactional model.

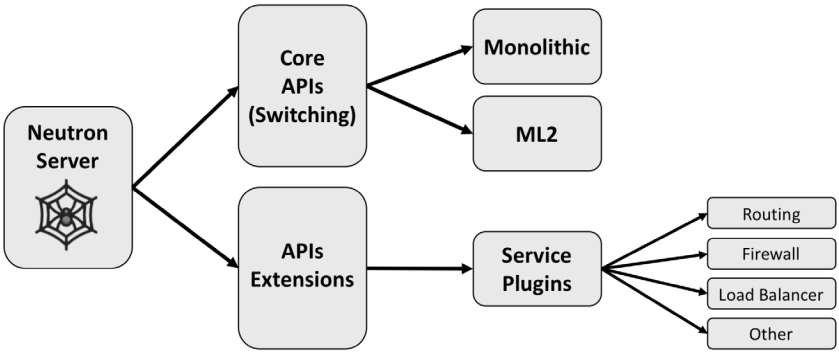


Figure 5.4 OpenStack Neutron Architecture

NSX Neutron Plugin

NSX integration with OpenStack is implemented as a Neutron plugin.

The NSX plugin is open source and can be used by any OpenStack distribution. More information and source code download is available on the GitHub page - https://github.com/openstack/vmware-nsx.

Neutron API calls are translated into NSX REST API calls. All calls are received by the NSX Manager, which is the API entry point for the entire NSX solution. The NSX Manager is then responsible for mapping and creating the components in the virtual networking and security infrastructure.

Since NSX can deliver all networking and security controls, the NSX plugin uses the monolithic approach instead of ML2. OpenStack users and administrators benefit from using a single monolithic plugin when using NSX for all Neutron services. If an ML2 plugin were used, the addition of service plugins could create challenges for operation and integration.

NSX and OpenStack Benefits

Networking Challenges in OpenStack

The OpenStack community considers Neutron one of the more complicated and complex project in the main core OpenStack projects. Several enterprises struggle to integrate their physical networking and security infrastructure with Neutron.

Every year, the OpenStack Foundation releases a survey detailing user adoption experiences and their opinions using OpenStack. Two quotes extracted from recent reports represent some of the challenges with Neutron:

> *"For Neutron, they wanted the networking service to be less complicated to use, with more substantial documentation and better integration with compute functions and PaaS layer integration."*
> *OpenStack User Survey 2016*

> *"Neutron needs to be reworked and simpler – we don't need to include every use case under the sun. Kick out the obscure architectures."*
> *OpenStack User Survey 2017*

Most OpenStack clouds leverage Open vSwitch as the reference implementation together with Neutron for the networking piece of OpenStack.

Figures 5.5 and 5.6 diagram some of the complexity of Open vSwitch, detailing the switching and routing architecture of Open vSwitch integration in OpenStack.

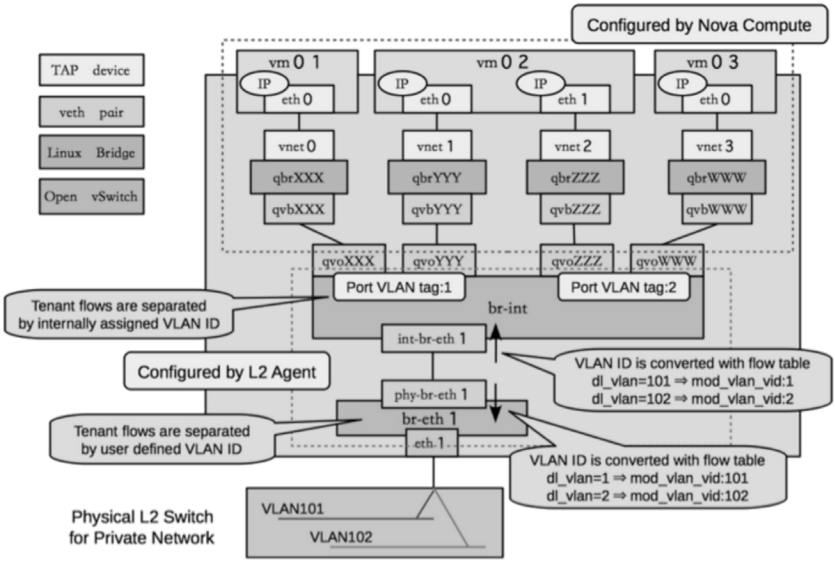


Figure 5.5 Neutron and Open vSwitch - Switching

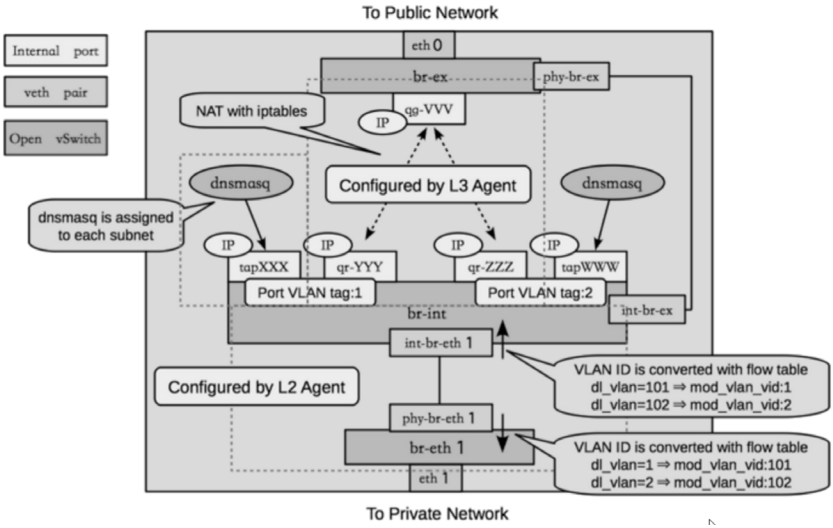


Figure 5.6 Neutron and Open vSwitch - Routing

These images only illustrate switching and routing functionality; the complexity grows when adding other services such as security groups, LBaaS, DHCP, VPNaaS, etc.

The most common challenges that enterprises face with Neutron and common Open vSwitch implementation can be summarized in five areas:

- Complexity
- Scalability
- Performance issues
- Lack of support
- Operations and monitoring tools for day two operations

Off-Topic: Open vSwitch (OVS)

Open vSwitch (OVS) was created by Nicira, then incorporated into VMware. OVS is one of many open sources projects where VMware contributes. In 2016, OVS became part of the Linux Foundation to enable further collaboration from the community.

Two of the biggest contributors of Open vSwitch – Ben Pfaff and Justin Pettit – are part of the VMware team, and both continue to contribute to the initiative. As a company, VMware is the top contributor of the Neutron project and continues to improve and create reliable code to maintain the use of Neutron for the entire community. Key areas of involvement include:

- Author and biggest contributor of OpenStack Neutron project
- Founder of OVN – Open Virtual Networking
- Author of OVSDB (Open Virtual Switch Database) – OVSDB is the most common management protocol used by virtual networks to communicate with physical networking hardware.
- Four out of the ten most active contributors to the OpenStack Neutron project work at VMware, including Gary Kotton – the top individual contributor and one of the most recognized members of the community.

Benefits of NSX

Automation is becoming a necessity for every enterprise in the world; they are looking for infrastructure that can deliver fast and agile environments, helping to avoid bottlenecks and enabling continued growing.

NSX abstracts the networking and security physical infrastructure, providing a reliable and complete virtual infrastructure with all components exposed via REST APIs.

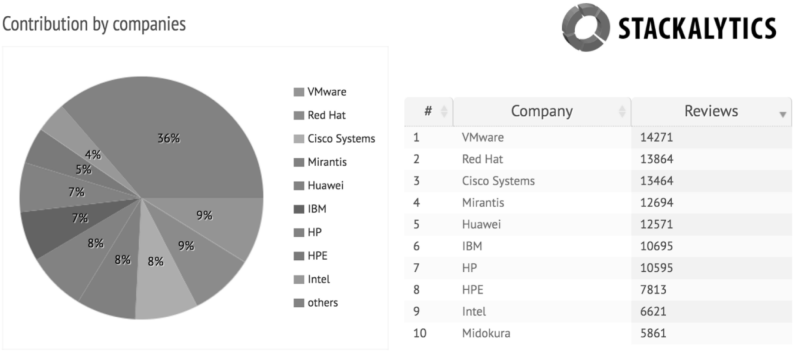
The benefits that VMware NSX brings to Neutron include:

- **Agility**: create networks at the same speed as the applications
- **Mobility**: provisioning and mobility of instances
- **Security**: micro-segmentation and chaining of partner services for advanced features
- **Multi-tenant**: possibility of using shared infrastructure among multiple tenants
- **Simplified operations**: centralized control and single monitoring

The challenges with Neutron can be addressed with NSX as follows:

- Simplified implementation of Neutron services
- Stability, scalability, and high availability
- Continuous development of new functionality
- Higher performance due to distributed NSX architecture
- Management, day two operations, and native troubleshooting tools in NSX

Figure 5.7 was taken from Stackalytics (www.stackalytics.com), whose mission is to provide transparent and meaningful statistics regarding contributions to both OpenStack and related projects.



#	Company	Reviews
1	VMware	14271
2	Red Hat	13864
3	Cisco Systems	13464
4	Mirantis	12694
5	Huawei	12571
6	IBM	10695
7	HP	10595
8	HPE	7813
9	Intel	6621
10	Midokura	5861

Figure 5.7 OpenStack Neutron Vendors Overall Contributions (August'2017)

NSX completely removes the hardware-centric barrier to the automation of networking operations. By moving networking and security services into the data center virtualization layer, NSX delivers the same automated operational model of a virtual machine for the entire network.

Working in conjunction with OpenStack, NSX can automate a variety of processes, significantly the accelerating service delivery cycle and reducing provision times from weeks to minutes. The business impact of this includes dramatically reduced operational complexity and cost as well as improved governance, compliance, and consistency.

Off-Topic: OpenStack Neutron History

OpenStack was created by Rackspace and NASA in 2010; however, Neutron was not initially one of the core projects. The first project related to networking and security was called Quantum; its first release was in the Diablo version of OpenStack in September of 2011.

Quantum was completely developed by Nicira, the SDN startup acquired by VMware 2012, helping transforming the NVP product into VMware NSX.

NSX and OpenStack Integration

VMware NSX is designed to address application frameworks and architectures that have heterogeneous physical endpoints and technology stacks. IT and development teams can use NSX to choose the technologies best suited for their particular applications. NSX is also designed for management, operation, and consumption by development organizations in addition to IT.

By enabling developers to consume networking resources either through APIs or natively with the OpenStack NSX Neutron plug-in, the NSX platform creates an abstraction layer to provide faster, secure, and more agile access to networking and security functionality rather than requiring access to physical components.

NSX offers to OpenStack:

- L2 services with VLAN or VXLAN, DHCP, support for overlapping IP addresses, and an L2 gateway for VLAN bridging.
- L3 services with distributed routing, external network, floating IP, no-NAT support, and integrated dynamic routing (i.e., floating IP and/or tenant subnets are automatically advertised to physical world without changes on the physical routers).
- Security groups leveraging stateful distributed firewall on NSX as well as 3rd party integration.
- XaaS model for network functionality, including LBaaS and FWaaS
- Instance-specific metadata access services

NSX Neutron Components and Equivalents

Figure 5.8 illustrates how Neutron components are mapped into NSX components.

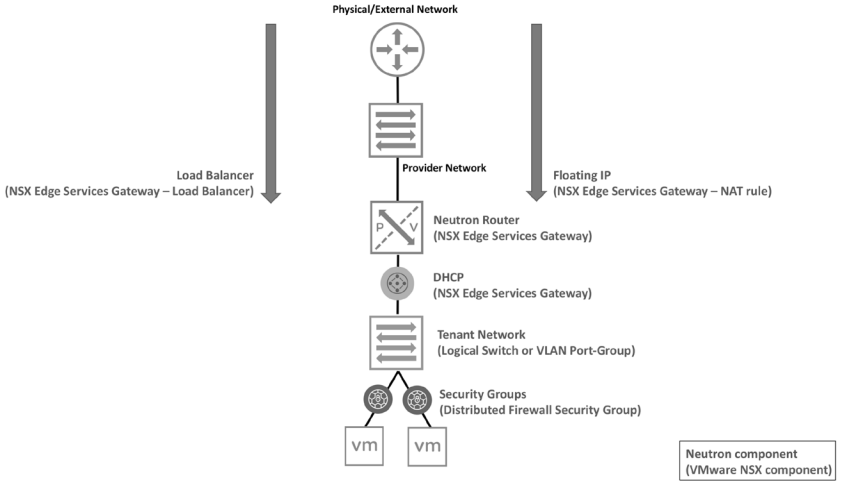


Figure 5.8 Neutron components equivalence in NSX

Tenant or provider networks can be mapped into traditional VLANs, leveraging vSphere VDS port-groups, or NSX logical switches, leveraging VXLAN or GENEVE overlay protocols, depending on the NSX edition.

NSX Edge Services Gateways can be used to replace Neutron routers. Depending on the topology, one or more NSX ESGs may be required. The NSX ESG will be responsible for routing and for DHCP services for tenant networks. When leveraging the floating IP feature, NAT entries are added into the NSX ESG. LBaaS and FWaaS are also features that are implemented using the NSX ESG.

Security services like security group rules are translated into the NSX distributed firewall. These stateful rules can prove a more granular, flexible, and advanced protection compared to other implementations.

Supported topologies will be discussed in more details later in this chapter.

NSX Supported Topologies and Integration

The NSX Neutron plug-in supports the following topologies:

	Topology	Use Case	Comments
1	VLAN for L2 services; no L3 services	Micro-segmentation only	No overlay networks; security groups leverage DFW policies.
2	VLAN for L2 services; L3 services LBaaS and FWaaS optional	Leverage VLANs for L2; NSX Edge for L3 services	No overlay networks; L3 provided by NSX ESG; no distributed routing supported.
3	L2 services; L3 services LBaaS and FWaaS optional	Enterprise customers that do not need overlapping IP addresses	Can use distributed router, NSX ESG, or both; no overlapping IPs allowed; very efficient; preferred enterprise model
4	L2 Services; L3 services with NAT; LBaaS and FWaaS optional	Enterprise customers that need overlapping IPs	Can use distributed router, NSX ESG, or both; very efficient; preferred cloud provider and service provider model

Topology 1: VLAN for Layer 2 Services. No Layer 3 Services

This topology, diagrammed in Figure 5.9, is only recommended for laboratory environments and small deployments. In this example, NSX only performs security tasks. VLANs leverage vSphere distributed switch port groups. All other features (e.g., switching, routing, security, load balancing) need to be pre-provisioned or created manually in the physical infrastructure.

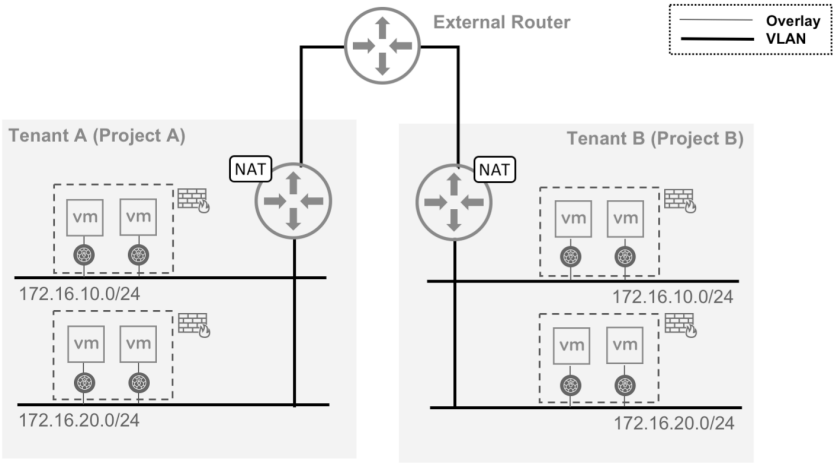


Figure 5.9 Use Case 1 – VLAN for Layer 2 Services | No Layer 3 Services

Topology 2: VLAN for Layer 2 Services. Layer 3 Services

The topology shown in Figure 5.10 is used by enterprises as a starting point to discover and gain familiarity with NSX and OpenStack integration.

NSX performs L2 services (i.e., connecting VLANs to NSX ESGs by leveraging vSphere distributed switch port groups), security, and routing. No pre-provisioning or manual configuration is required in the physical infrastructure outside of VLANs used as tenant networks. Overlapping IP addresses can be used with NAT in place. LBaaS and FWaaS are optional, delivered by the same NSX Edge Services Gateway when desired.

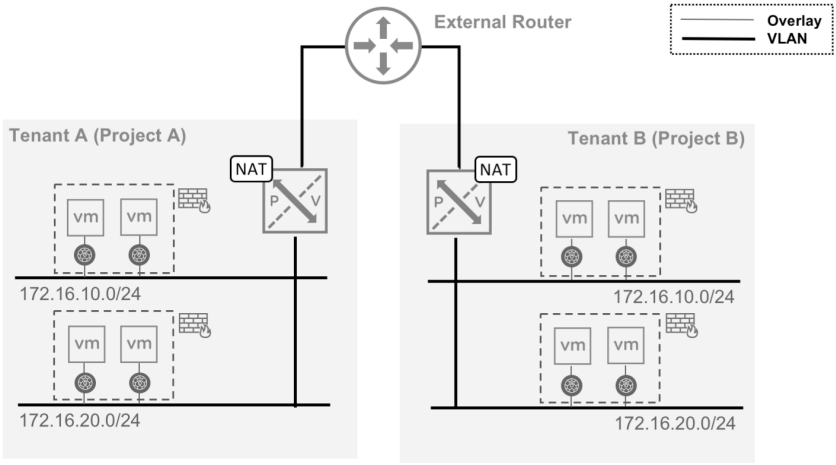


Figure 5.10 Use Case 2 – VLAN for Layer 2 Services | Layer 3 Services

Topology 3: Layer 2 Services and Layer 3 Services

The topology in Figure 5.11 is broadly recommended for enterprise deployments, especially where there are no overlapping IPs.

NSX performs L2 services, security, and routing. No pre-provisioning or manual creation is required in the physical infrastructure. Overlapping IP addresses are not allowed for tenant networks. LBaaS and FWaaS are optional, delivered by the NSX ESG when desired.

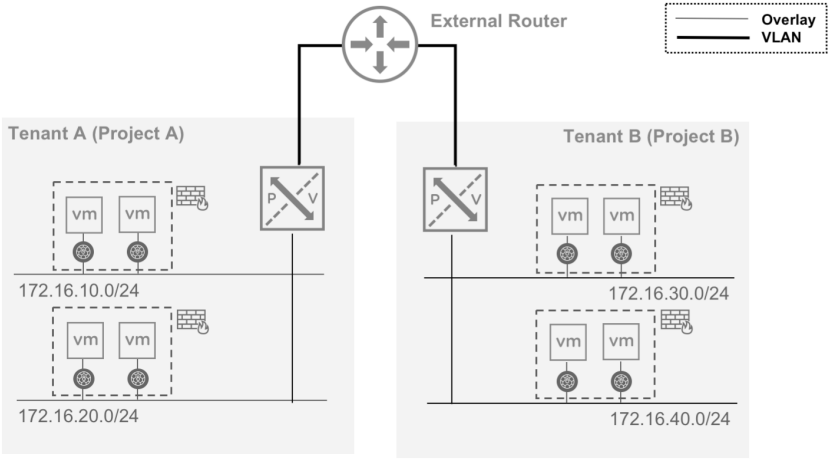


Figure 5.11 Use Case 3 – Layer 2 Services | Layer 3 Services

Topology 4: Layer 2 Services and Layer 3 Services with NAT

The final topology, shown in Figure 5.12, is also broadly recommended for enterprise deployments, especially service providers and cloud providers.

NSX performs L2 services and L3 services. No pre-provisioning or manual creation is required in the physical infrastructure. If overlapping IP addresses are used, each tenant will have its own NSX Edge Services Gateway. LBaaS and FWaaS are optional and delivered by an NSX ESG as desired.

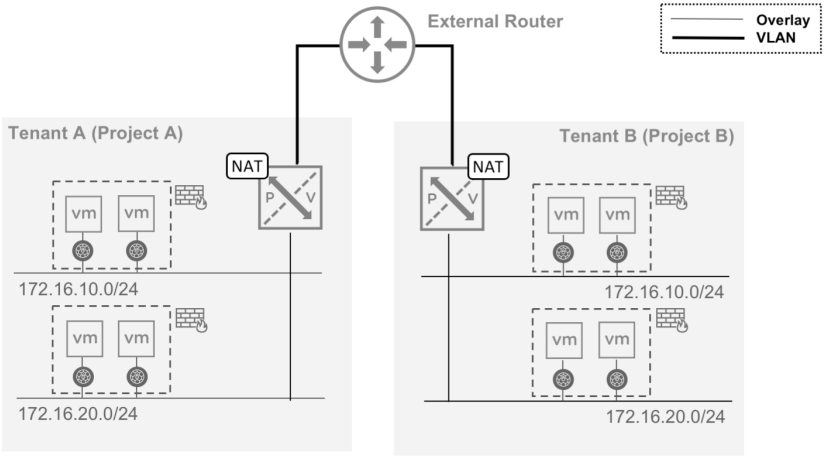


Figure 5.12 Use Case 4 – Layer 2 Services | Layer 3 Services with NAT

NSX Security and Micro-Segmentation

NSX introduces support for distributed firewall functionality for workloads running on hypervisors. The NSX DFW provides the capability to enforce firewalling functionality directly at the workload vNIC layer, providing an optimal micro-segmented environment. Stateful firewall rules are supported.

NSX implements an NSX security group for every Neutron security group. Membership criteria for the security group is based on the virtual machine vNIC.

Under the NSX firewall rules, a section is created per tenant (i.e., project) with security groups used as sources/destinations. Those rules are applied only for the security groups related to the rules, creating a true micro-segmented architecture per tenant.

There is a default section for NSX firewall rules which assigns security groups an implicit drop at the end.

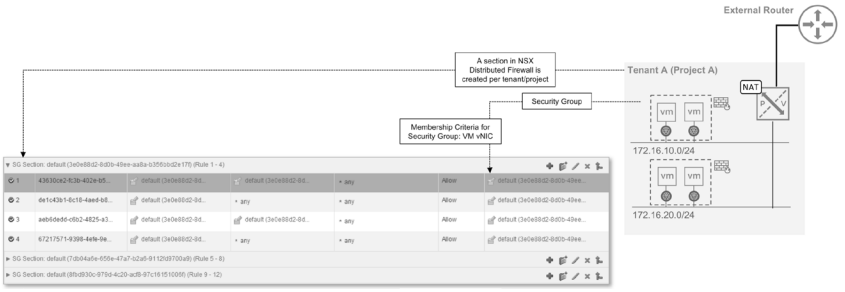


Figure 5.13 Security Group Rules Mapping into NSX Distributed Firewall

Neutron security groups and NSX micro-segmentation can be used standalone, without adopting L3 overlays and L3 routing virtualization. While not as flexible as full network virtualization, the micro-segmentation-only use case could be an entry point for enterprise adoption. It provides a great insertion vector for OpenStack and NSX without disrupting any operational model that is already in place.

NSX Policy Redirection

Security in Neutron relies on tenant security groups. In OpenStack, tenants are allowed control of their own security firewall rules. Native Neutron security groups were insufficient to address requirements of regulatory and enterprise security standards when implemented in OpenStack. Neutron security groups rely on firewall rules from layers 2 through 4; they do not provide more advanced/granular protection in other layers.

VMware NSX brings an additional layer of protection through policy redirection, also known as network extensibility. This functionality provides an open platform for security partners to integrate solutions into NSX.

Relying on NSX DFW security group rules, administrators can add another layer of security (e.g., IPS, antivirus, anti-malware) for specific traffic with additional needs. This is achieved through integrating solutions into NSX Manager and configuring redirection policies in the NSX unified dashboard to determine which type of traffic should receive advanced treatment. There is no requirement to change or reconfigure any additional networks for traffic redirection; this is done before the packet even arrives in the virtual network. All processing is performed at the vNIC level of the VMs.

The advanced features rely on VM appliances on each hypervisor where the performance and throughput need to be correctly designed.

Best practices recommend identifying the traffic that requires advanced features; this traffic can be redirected to those appliances while the standard L2-L4 firewall rules are handled by NSX DFW.

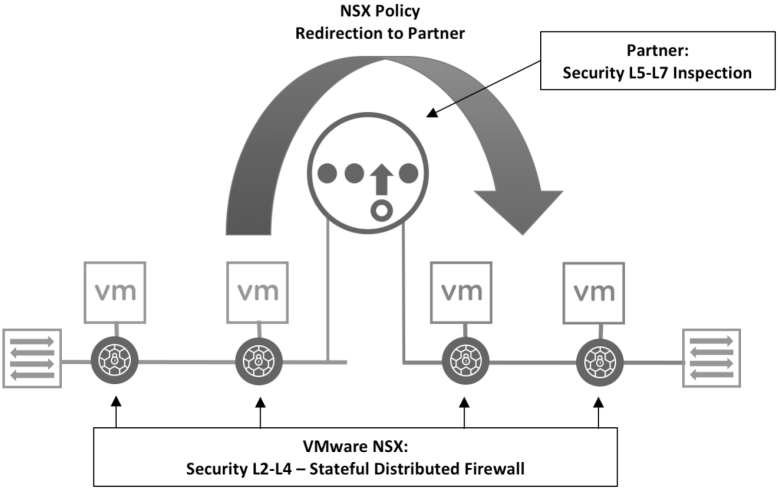


Figure 5.14 NSX Policy Redirection

NSX Edge Services Gateway Integration

Tenants in OpenStack are allowed to create networks, configure subnets, and determine routing between those networks and external world. When users create a Neutron router in OpenStack, the NSX Neutron plug-in translates that action into a API call for NSX Manager to create an NSX Edge Services Gateway.

When networks are connected to a Neutron router, NSX creates a connection between the network (e.g., VXLAN or VLAN) to the correspondent NSX ESG, establishing routing between those networks.

When Neutron routers are connected to an external network, the ESG is connected to a vSphere port group mapped to an external VLAN; this provides connectivity to the external world. If tenants have overlapping networks, the Neutron router must perform NAT to enable connectivity between tenants and the external world. NAT is not required if tenants maintain distinct public address ranges.

If users do not want to use VXLAN to back tenant networks, NSX administrators must pre-create VLANs in the physical infrastructure. Due to the complexity this involves, best practices recommend using VXLAN to back tenant networks.

Neutron routers can be exclusive or shared between tenants depending on the implementation (i.e., NAT or no-NAT topologies). Factoring this with performance and scalability requirements, users and administrators can choose the best design for their OpenStack network.

Another benefit of using NSX is the optimization of east-west routing traffic between tenant networks using a DLR. An NSX DLR is connected to an ESG to manage the routing between tenant networks while the ESG continues to be responsible for tenant north-south traffic and networking services. With an NSX DLR, performance increases as routing is handled inside each hypervisor. If two instances (i.e., VMs) of a specific tenant reside on the same hypervisor, traffic will not need to travel to the correspondent NSX ESG.

Figure 5.15 illustrates topologies with and without an NSX DLR.

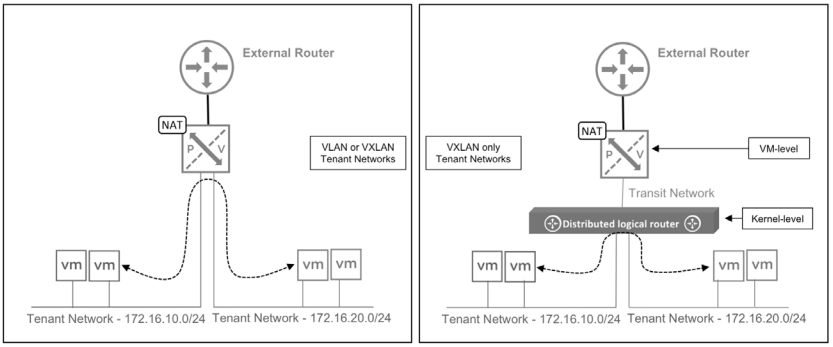


Figure 5.15 Comparison between using ESG or DLR for east-west routing

DHCP

NSX offers DHCP services to Neutron. NSX DHCP is implemented with static DHCP bindings; this provides greater reliability at scale for OpenStack clouds with a significant number of instances.

The NSX Neutron plug-in will automatically determine DHCP implementation depending on the topology. When using topologies with overlapping IPs, DHCP is implemented per NSX Edge. When using topologies without overlapping IPs (i.e., no-NAT), DHCP is implemented in a shared NSX ESG.

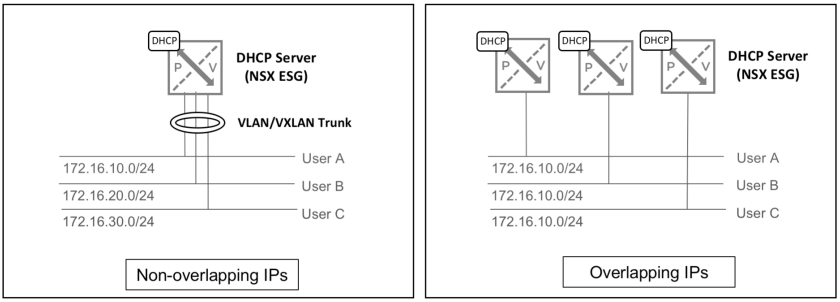


Figure 5.16 DHCP implementation for non-overlapping and overlapping IPs topologies

NSX Load Balancing Integration

Beginning with the OpenStack Mitaka version of the NSX plug-in, Neutron load balancing-as-a-service (LBaaS) version 2.0 is incorporated and supported.

Rather than relying on third party integration, NSX has embedded load balancing capabilities provided by the ESG.NSX provides an L4-L7 load balancer that offers OpenStack users and administrators the capability to balance one or more applications on-demand. When integrating with Neutron LBaaS, an ESG acts as one-arm load balancer; this is a different ESG than one acting as the tenant's Neutron router. Tenants use the following process to establish NSX LBaaS for supported protocols (i.e., TCP, HTTP, HTTPS):

- Create application pools
- Add members (i.e., instance IP addresses) to the pool
- Create one or more health monitors
- Associate the health monitors with the pool
- Create a virtual IP with the pool

Figure 5.17 illustrates use of an NSX load balancer integration with Neutron.

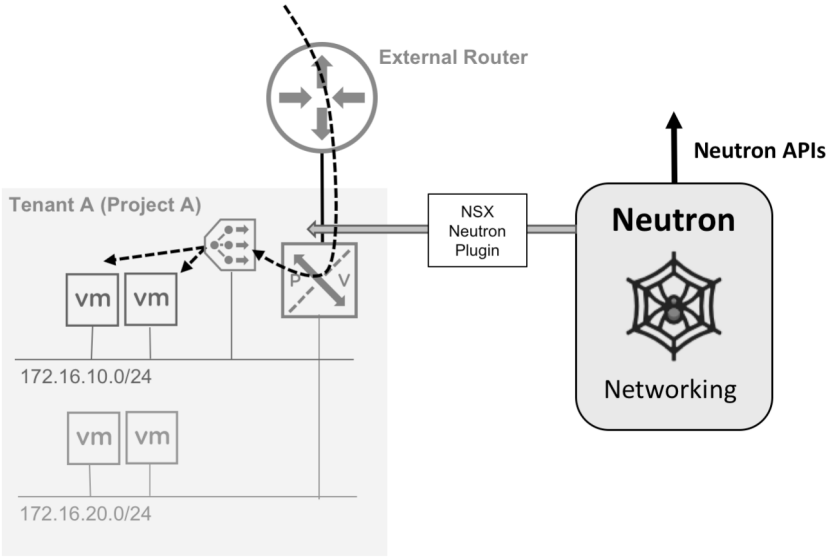


Figure 5.17 NSX ESG with Load Balancer integration with Neutron LBaaS

NSX Operations Tools

> *Be more user- and ops-oriented. It's been an issue since the beginning of OpenStack and it still is, even though we've seen huge improvements.*
> *OpenStack User Survey 2017*

Managing, monitoring, and troubleshooting OpenStack is a top concern of administrators and users who have implemented OpenStack for public or private clouds. Processes to optimize the installation of OpenStack are getting better, but day two operations need further improvement, especially with troubleshooting Neutron networking and security activities.

VMware NSX addresses these issues perfectly. NSX offers users and administrators a set of built-in tools that can be leveraged independently on a cloud management platform or used with other automation tools integrated with NSX. Even with a complex and multi-tenant environment like OpenStack, native operations tools are extremely useful for troubleshooting and monitoring.

NSX provides a robust logging framework with consistent log formats, trackable event identifiers, error codes, and tags across the distributed components. This helps build a correlated, log-based dashboard through centralized log collectors such as VMware vRealize® Log Insight™.

Aggregation services in NSX offer a centralized view of information such as statistics, routes, and MAC table information from distributed components, presenting a unified picture in a single pane of glass and eliminating the need log into individual components.

Another benefit is the granular flow and packet-level visibility through standard tools such as IPFIX and port mirroring. This enables customers to use their existing monitoring and troubleshooting tools for network visibility when troubleshooting.

One of the most efficient features that NSX provides is Traceflow. With Traceflow, users and administrators can simulate the path between two instances to help identify potential problems. Traceflow is a powerful tool that provides insights of both virtual networks and security rules in the path. In OpenStack deployments, Traceflow can be used to identify and determine the traffic between and inside tenants as well as connectivity to the external world.

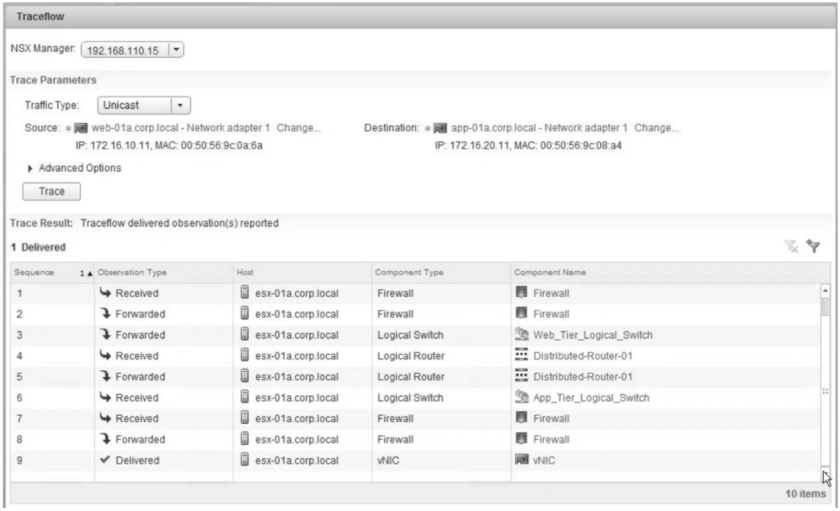


Figure 5.18 NSX TraceFlow tool

The following section of this chapter discusses additional benefits of NSX integration with OpenStack as well as how organizations are gaining competitive advantages by running VMware NSX for their production OpenStack environments. The final section offers details about VMware® Integrated OpenStack™ along with its connection to and interaction with NSX.

NSX and VMware Integrated OpenStack

What is VMware Integrated OpenStack?

Like many other software companies, VMware has its own OpenStack distribution. It is called VIO – VMware Integrated OpenStack.

There is a growing trend of using commercial OpenStack distributions by enterprises adopting OpenStack for production environments, despite of the availability of open source software offered by the OpenStack Foundation. The main motivation for such a choice is that the support provided by these companies is critical to any enterprise wanting to maintain a reliable production environment with OpenStack.

The goal of VMware Integrated OpenStack is not and will never be to contain the largest number of projects available in the community. It will focus on offering the most stable projects through a robust, high-performance, resilient, and secure distribution for environments that use VMware's own computing infrastructure. One of the major challenges of most OpenStack cloud implementations today is upgrading between versions. This update process is extremely challenging to perform in the vast majority of OpenStack distributions and is one of the main differentials available since version 2.0 of VIO.

VMware Integrated OpenStack is based on standard upstream OpenStack. Core projects that makes up VIO releases are:

- Neutron - Network Connectivity
- Cinder - Block Storage for Volumes
- Nova - Compute Services
- Glance - Image repository
- Horizon - UI Portal
- Ceilometer - Telemetry
- Heat - Orchestration
- Keystone - Identity Management

Swift, the object store module, is not included in VIO by default. Third-party integration is required, typically with SwiftStack (https://www.swiftstack.com/).

Why VMware Integrated OpenStack?

VMware Integrated OpenStack's main goal is offer simple installation, upgrade, and operations using VMware infrastructure to provide the most stable release of OpenStack in the market.

Enterprise deployments of OpenStack lack features like monitoring, management, and logging. VIO addresses these deficiencies with other VMware solutions like VMware vRealize® Operations Manager™ and VMware vRealize® Log Insight™, integrating them without the need for additional customization.

VMware Integrated OpenStack is a standard DefCore compliant OpenStack distribution. Its implementation is fully supported by VMware, providing an IaaS solution using VMware vSphere for compute and storage and NSX for network.

Many customers are adopting VMware Integrated OpenStack to leverage their existing investment in people, skills, and infrastructure based on their current VMware footprint.

The word integrated is key; deep customization is limited when running VIO. Customers need to deploy and use the entire set of projects to leverage the benefits. In that sense, only vSphere, NSX, and vSAN can be deployed with VIO.

This integrated approach delivers the following benefits:

- Single vendor support
- Stable and scalable solution leveraging existing vSphere expertise and management, monitoring, and operations tools
- Best in class security solution delivering real micro-segmentation
- Fully validated end-to-end solution considering computing, networking, and storage

VMware Integrated OpenStack Architecture

The 3 major pillars in VMware Integrated OpenStack are:

- vSphere Compute for Nova Compute API
- vSphere Networking (VDS) or NSX for Neutron Network API
- vSphere Storage or VSAN for Cinder and Glance API

vRealize solutions like VMware vRealize® Operations™, vRealize Log Insight, VMware vRealize® Network Insight™, and VMware vRealize® Business™ are critical to day two operations, management, billing, and monitoring. These could also be integrated with VIO.

Figure 5.19 illustrates a high level overview of the VIO architecture.

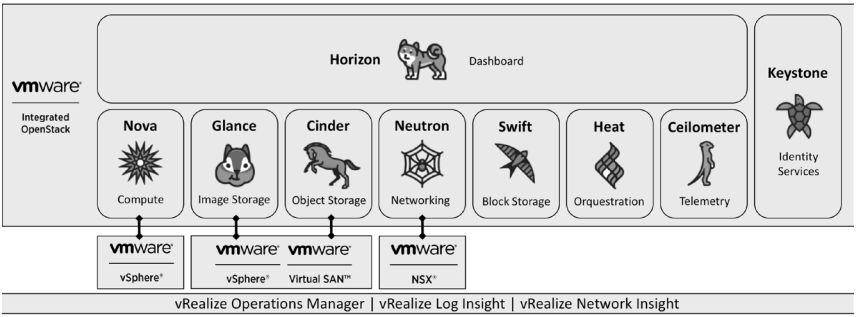


Figure 5.19 VMware Integrated OpenStack Architecture

Compute

The only option available in VIO for compute is vSphere. Nova connects to vCenter using available drivers and translates Nova API calls into vSphere API calls.

The reason for supporting only VMware solutions underneath VIO is the core principle to maintain a simple, stable, and repeatable solution. This approach provides a production-grade distribution to enterprises, allowing users to truly experience the power and benefits of OpenStack.

In standard OpenStack deployments, each Nova Compute is a hypervisor node. When integrating with vSphere, each Nova Compute is a cluster with one or more vSphere hypervisors. This architecture allows users and administrators to leverage some of the top class features like vSphere HA and vSphere DRS (Distributed Resource Scheduler) to make optimal usage of the compute infrastructure.

It is important to understand that vSphere is not part of VMware Integrated OpenStack solution; separate licenses are required.

Figure 5.20 illustrates the computing architecture of VIO.

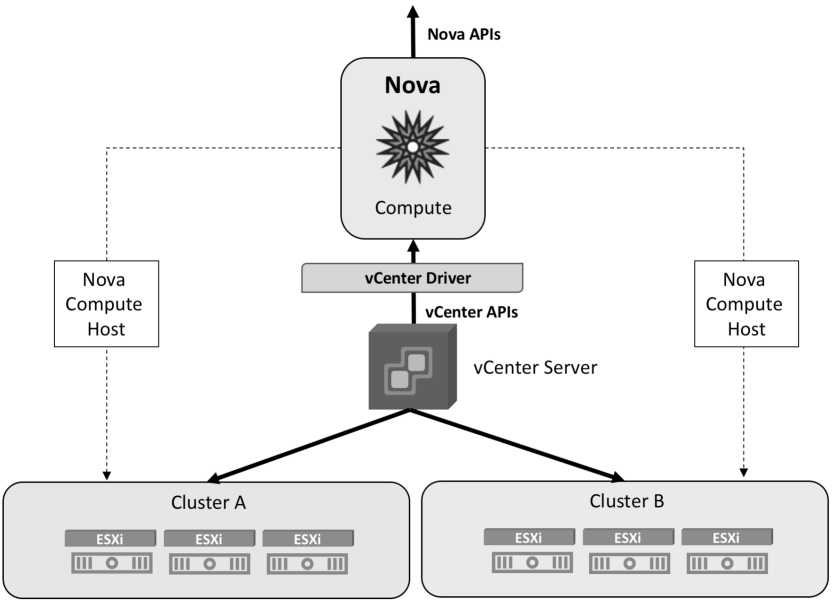


Figure 5.20 OpenStack Nova and vCenter Integration

Storage

Another differentiator of VIO is its ability to leverage vSphere Storage or vSAN features for both Cinder and Glance. Those projects can potentially use any datastore available, independent of the underlay physical storage providing the capacity to read/write data.

Cinder integration is simple. vCenter first creates a volume and attaches it to a shadow VM. When an instance is created in Nova and a volume is attached into this instance, vCenter changes the link between the shadow VM to the newly created VM.

Figure 5.21 illustrates the storage architecture of VIO related to Cinder.

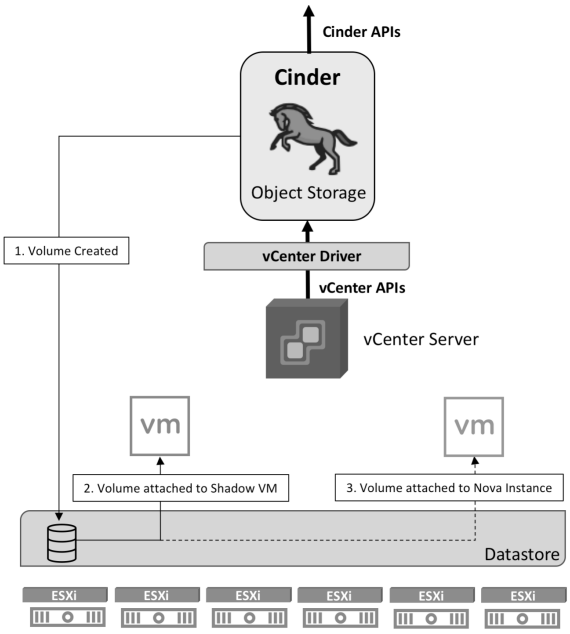


Figure 5.21 OpenStack Cinder and vCenter Integration

Glance is responsible for the image repository. VIO supports a different number of formats including ISO, OVA, and VMDK. When an image is imported, VIO converts it to VMDK or ISO behind the scenes. When a VM is booted, this VMDK or ISO image is copied from the image service to the vSphere datastore.

Networking

VMware Integrated OpenStack allows two options for networking integration with Neutron project: vSphere VDS or NSX.

vSphere VDS is purely a layer 2 switch embedded into the vSphere hypervisor. NSX is a more complete networking virtualization solution, containing not only switching but also routing, security features, and other advanced functionality.

When using vSphere VDS, Neutron API calls are translated using the vCenter API. When using VMware NSX, Neutron API calls are translated by the NSX plugin.

When deploying a vSphere VDS-only model, Neutron features such as security groups, L3 agent, and tenant networking are not available. For a production grade implementation of OpenStack, VMware NSX is highly recommended.

It is important to understand that NSX is not part of VMware Integrated OpenStack solution; separate licenses are required.

Figure 5.22 illustrates the networking architecture of VIO related to Neutron.

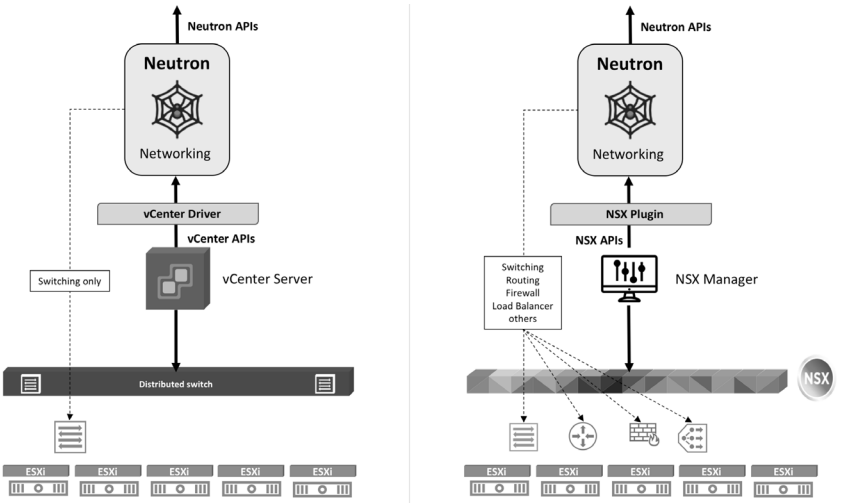


Figure 5.22 Comparison between Neutron integration with vSphere Distributed Switch and NSX

Operations

vRealize Log Insight (vRLI) analyses structured and unstructured data. With VMware Integrated OpenStack, VMware vSphere, and VMware NSX, vRLI can automatically identify structures of data and create an index for performing analytics. This offers OpenStack users and administrators a complete suite with dashboards to display time series events to simplify troubleshooting. vRealize Log Insight also allows extensibility from different vendors through content packs which allow integration of commonly used applications into a single pane of glass.

vRealize Operations Manager provides identification, remediation, and capacity planning for VMware Integrated OpenStack and vSphere. It offers a single pane of glass for visibility and monitoring across applications and infrastructure.

vRealize Network Insight retrieves networking and security analytics information from vCenter and NSX. vRNI is able to help with security planning and networking visibility for both underlay and overlay networks.

It is important to understand that vRealize Log Insight, vRealize Operations Manager, and vRealize Network Insight are not mandatory; they are not part of VMware Integrated OpenStack but are recommended when running VIO in production. Separate licenses are required for all solutions.

VMware Integrated OpenStack and VMware NSX

OpenStack Server uses the Neutron plugin to communicate with NSX Manager. All the benefits of NSX for OpenStack can be also achieved with VIO integration, including L2 and L3, security with micro-segmentation, and LBaaS with native load balancer.

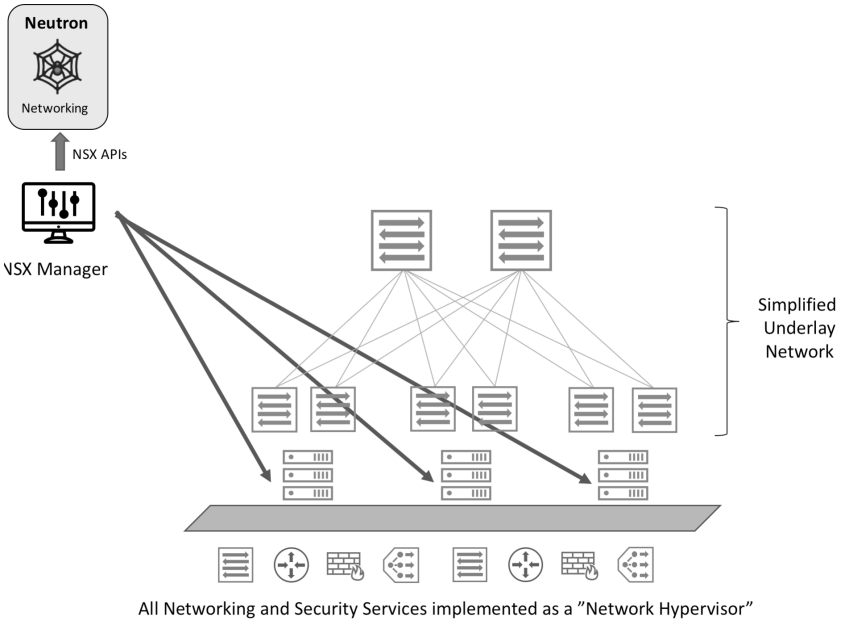


Figure 5.23 NSX Integration with OpenStack

The Neutron plugin is open source, and all OpenStack Neutron operations map directly to NSX Manager. NSX ESGs function as OpenStack L3 agent, metadata server, DHCP L2 agent, tenant networking, and security group policy enforcement. As a direct result of leveraging enterprise-grade virtualization with vSphere and enterprise grade networking with NSX, customers will have an enterprise-grade OpenStack layer. This will mitigate the risks and shortcomings of the reference implementation.

NSX is strictly integrated with vSphere and VMware Integrated OpenStack. Users relying on VMware infrastructure for their complete data center automation solution have the benefit of a centralized support from VMware.

Tales from the Field – Marcos Hernandez

How VMware Integrated OpenStack and the SDDC Power one of the Largest E-commerce Websites in the World

After trying to become more agile by implementing a number of COTS automation solutions, a customer of VMware's with a market focus on direct-to-consumer sales decided to leverage VMware's OpenStack distribution, VIO. This particular customer had already built a custom, homegrown, declarative automation tool, and now they just needed to make it OpenStack-aware. Because of the fact that the OpenStack API is public, documented, and universal, this task, while not insignificant, was easier to attain than previous attempts to do the same on proprietary technologies.

By implementing OpenStack as the abstraction layer for IaaS, the DevOps team responsible for the custom tool was able to port some of the primitives they were leveraging in public cloud with only minor changes. Not being tied to a VMware specific API also gave this customer the option to use someone else's OpenStack in the future, if they choose to do so, without discarding the huge investments in intellectual property made over the years while building their consumption strategy.

When done right, OpenStack-based clouds have a tendency to grow very quickly. This customer was no exception. In a matter of months, their development pipeline and corresponding compute footprint grew 1000%, while time-to-market of production quality software was reduced to minutes. This customer is now pushing updates to their production, user-facing platform, several times a day.

The network was NOT an afterthought. Recognizing the many challenges of running the reference implementation of Neutron that leverages open source components, this customer settled on NSX as the alternative. NSX replaces flaky or older open source network services that have not been tested at scale in OpenStack, for example, DHCP. NSX has brought stability to their Neutron layer and after several years of being in production, the number of severity 1 issues and outages are a thing of the past.

VIO and the SDDC are central to the business in this particular case, and other customers could benefit from a prescriptive way to deliver private cloud in this manner, while appealing to a user population that wants the tooling freedom unleashed by an open API. There is no reason why organizations like yours, with similar ambitions, can't bake the proverbial cake and eat it too.

VMware vRealize Automation, OpenStack, or Both?

VMware vRealize Automation or OpenStack? This is a reasonable question to ask, and is the focus of this chapter.

Before jumping into this topic, note the clear distinction between the roles of infrastructure and developer teams. Infrastructure teams are responsible for creating VM images, defining security and network parameters, and operating/maintaining the whole infrastructure. Developer teams consume the VM images along with the predefined security and networking components for their application needs.

Developer teams need a basic infrastructure consumption model to simplify the deployment offered through APIs. This can be generic definition of infrastructure as a service, but most enterprises have other needs including lifecycle management of workloads, governance, approval workflow, or service catalogs – tasks usually handled by the infrastructure team. External cloud connectivity for burst, migration, or creation of a hybrid environment is yet another critical requirement for enterprises.

In short, enterprises will likely need both models and both solutions. Both infrastructure and developer teams may want to deploy vRealize Automation and OpenStack approaches, and an integrated design could be the end goal for any enterprise.

Figure 6.1 exemplifies how VMware vRealize Automation and OpenStack work together when creating the building blocks of a private cloud.

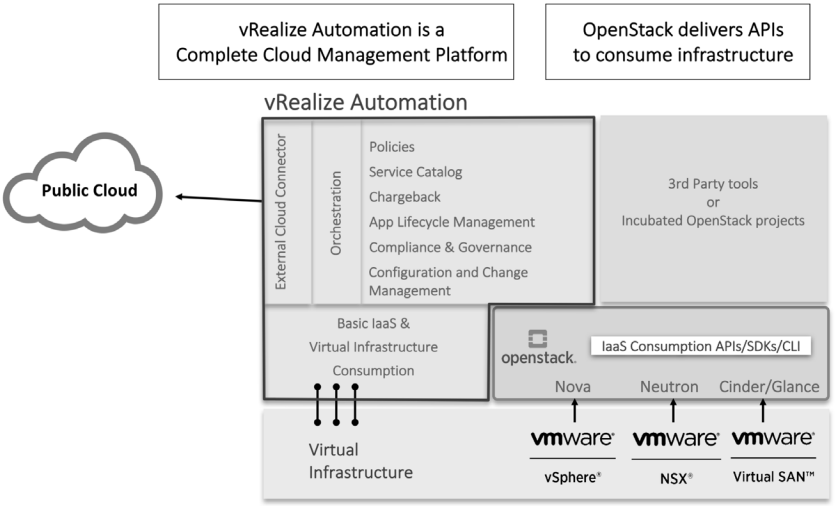


Figure 6.1 vRealize Automation and OpenStack Building Blocks

An OpenStack deployment could be a vRealize Automation end-point, centrally managed and orchestrated by the infrastructure team, offering the OpenStack APIs to developer teams to consume and create the infrastructure primitives as needed. Instead of requesting networks and VMs for an application, developer teams can request that as part of the code, gaining agility without the need to wait for an infrastructure team provision the desired environment.

The table in Figure 6.2 explains the requirements needed to define the best solution depending on the use case.

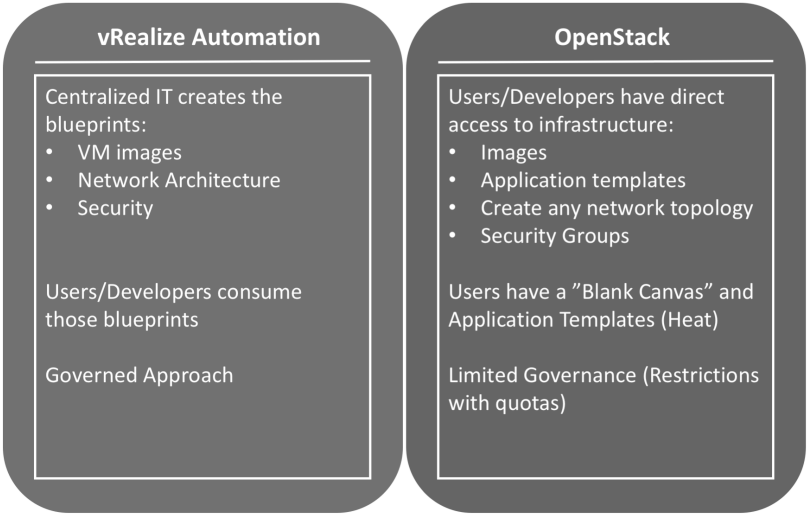


Figure 6.2 vRealize Automation and OpenStack consumption

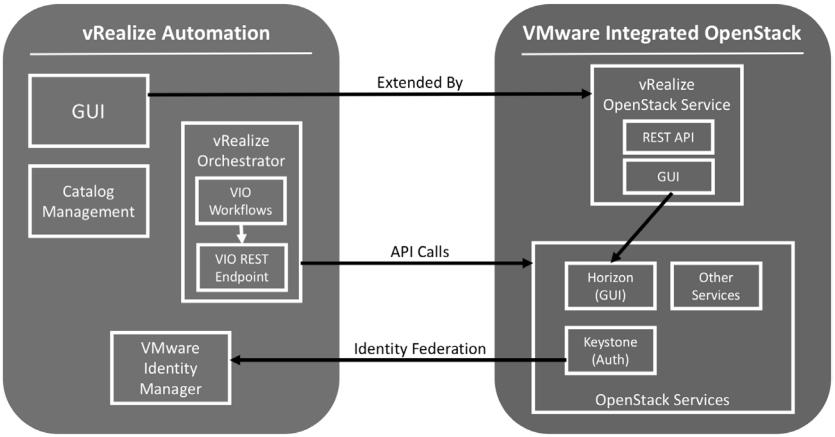


Figure 6.3 vRealize Automation and VIO integration

Starting in VIO 4.0, vRealize Automation administrators can embed VIO components in blueprints. They can also manage their VMware Integrated OpenStack deployments through the Horizon UI as a tab in vRealize Automation. This integration provides additional governance as well as single sign-on for users.
The following four directives offer guidance during the project definition for automation solution:

- OpenStack is primarily attractive for developers that want an AWS-like API, do not care about a GUI or policy and governance, and are developing next generation resilient applications that follow an agile development flow (i.e., spin up, test, tear down).
- Both OpenStack and vRealize Automation feature a RESTful API. In many cases, vRealize Automation will meet their requirements and is much easier to deploy and manage.
- OpenStack is targeted at developers that only require an API. OpenStack is not a good fit for an infrastructure automation solution that presents a customized portal or a third-party system like ServiceNow. In this case, vRealize Automation is a much better fit and more widely adopted for that use case.
- OpenStack requires a different skillset, often requiring organizations to hire specialists for the role. vRealize Automation was built by vSphere administrators for vSphere administrators.

Using VMware infrastructure with vSphere, NSX, and vSAN is the ideal scenario. Enterprises can create an abstraction layer that will be consumed as virtual components by both automation solutions without the requirement of having different or separate physical infrastructure for each solution.

VMware offers a choice of cloud management platforms that enterprises choose based on their requirements:

- VMware Integrated OpenStack provides the ability to quickly deploy and maintain a robust OpenStack IaaS using existing infrastructure and investments – both technology and people.
- vRealize Automation provides a fully-featured cloud management platform – beyond simply IaaS – to also offer hybrid cloud, extensibility and end-to-end lifecycle management.

Tales From The Field – Angel Villar Garea

Before looking for a product, look at your needs

The large presence of VMware in the market allows us to find all kind of needs and requirements among our customers. Infrastructure automation has been one the most popular requests recently, with almost any organization looking at building some sort of private cloud. While a number of them started investigating OpenStack with non-VMware hypervisors due to the hype at the moment, most of them finally turned to familiar VMware technologies, either combined with OpenStack or not. And why is that? Why did they finally choose VMware technologies? Let's find out some of the reasons by looking at a VMware financial customer experience.

This specific customer started looking at OpenStack as a solution for infrastructure automation. While it provided some benefits like vendor-neutral APIs, during the testing phase the customer found OpenStack limited in aspects including:

- Out of the box integration with existing enterprise systems like IPAMs, load balancers, CMDBs, or Active Directory, to name a few.
- Extensibility – as time goes by, the customer will need to integrate the cloud platform with new systems they cannot foresee at the moment. Will that be feasible? How difficult will it be?
- Governance – how to control who can modify CPU and memory of deployed VMs, or who can deploy a specific app? How to integrate with existing enterprise approval systems?
- Learning curve – how much time will it take infrastructure admins to learn to operate and troubleshoot the new cloud environment? Will they have the same visibility and control they enjoy in their VMware infrastructure?

Overall the main concern was, will they really save time and be infrastructure independent or will they have to develop in-house components that will require a huge investment on time and new skills, while at the same time could be tying themselves to a specific kind of infrastructure?

So they kept investigating, looking at alternative ways to take most out of their existing infrastructure without having to rebuild everything from scratch, in terms of solutions and skills.

They found out it was possible to add automation on top of the technologies they already knew, allowing them to protect their existing investments and leveraging their years of experience and knowledge, as well as their well-known monitoring and troubleshooting tools. They also discovered they could build a private cloud with granular controls on who does what and when, and with easy integration with software defined networking and security, so they could automate deployments and reduce time to market while guaranteeing the new platform met their corporate security standards.

Additionally, they realized that integrating the new cloud with their existing enterprise systems was fairly simple, and that there were next generation monitoring tools that greatly simplified day two operations.

But the final decision point was discovering that the moment they required it, they could also add OpenStack on top of their existing infrastructure, providing their developers the environment they needed with almost no learning curve for the infrastructure administrators.

In summary, this customer felt reinforced after learning that with the right combination of technologies, they could protect their investment and satisfy all their business needs, without having to rebuild their infrastructure from scratch.

VMware NSX and Other Automations Tools

As discussed in the previous chapters, NSX gives IT organizations speed and security through standardization and consistency; not just leveraging CMPs like vRealize Automation and OpenStack, but a wide variety of automation tools. This helps administrators accomplish tasks in a shorter amount of time and can turn routine chores, such as creating virtual subnets or security policies, into something that requires less intervention and effort from IT.

There are some IT professionals that are hesitant of diving too deeply into a new automation tool, worried that they could actually automate themselves out of a job. Some fear that if they can do their job in half the time, either they or one of their co-workers may no longer be necessary. This is far from the truth; there are lots of tasks left undone because there is not enough time to do everything with a minimum quality.

In many cases an automation tool may provide a hidden value, such as the task of creating a new VLAN. First it must be configured into different switches. If running spanning tree protocol, it is necessary to check the root instance, bridges, and ports. Additionally, it is important to make sure security policies are in place. These steps can become so detailed and mundane that even seasoned administrators will find their skills challenged by those processes. Using an automation tool the administrator now shifts from managing each individual deployment to managing the single automation task, bringing increased value and process improvement to the organization.

There are a variety of different tools available; this section selects a few to highlight central NSX capabilities for automation. Some organizations decide to go full open source, using tools like Python or Chef without formal enterprise support. Other organizations prefer enterprise versions of those automation tools which come with support and implementation services.

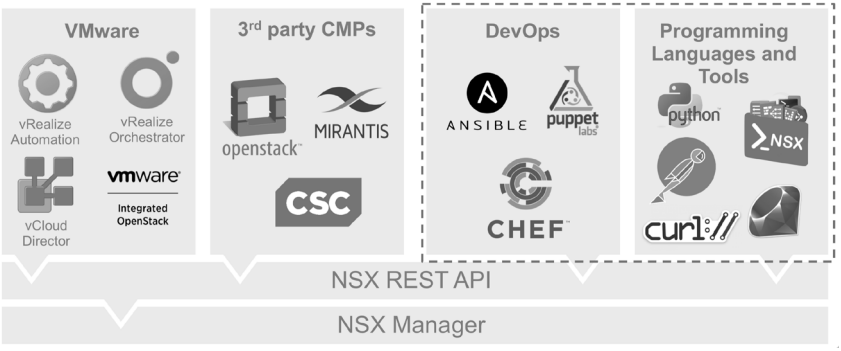


Figure 7.1 Different automation tools that interacts with NSX REST API

DevOps

DevOps has been a hot topic for the last several years. The modern DevOps world is full of outstanding and unique open source tools. Because of the intensive pace of work, DevOps engineers are always looking for tools to help them improve efficiency and productivity. They need tools to ship code faster and remove the barriers that slow down the development process.

Some of the most popular DevOps tools include:

- Chef
- Puppet
- Ansible
- SaltStack
- CloudFormation
- Terraform

These tools enable DevOps engineers to work with infrastructure as code – a type of IT infrastructure that DevOps teams can automatically manage and provision through code – rather than using a manual process. This is also referred to as programmable infrastructure.

Chef

Chef is an open source automation tool available in cloud-hosted and on-premises varieties. It turns IT configuration and infrastructure management into code. Chef is a thin domain-specific language (DSL) built on top of the Ruby on Rails programming language.

Chef is used to streamline the task of configuring and maintaining a company's servers. It can be integrated with cloud-based platforms such as vRealize Automation, OpenStack, and Amazon Web Services EC2.

Chef uses the core components of a workstation and a server to create and run sets of instructions, called recipes and cookbooks, on target nodes. A recipe is a single file that contains one or more resources along with the instructions for that resource. A resource can be practically anything: a Windows service, a file, or a PowerShell script. The instructions available for any particular resource are dependent on the resource type.

A cookbook is a collection of recipes. It addition to the recipes, it includes a method of setting some code (e.g., HTML, PowerShell script) aside as a template resource. Once included into a cookbook, these scripts and code can be referenced by multiple recipes within the cookbook.

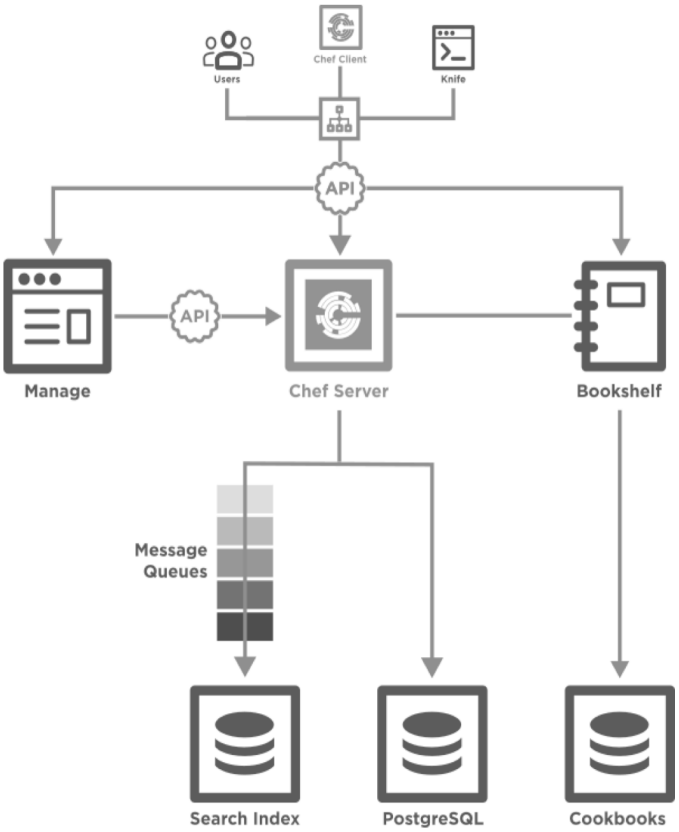


Figure 7.2 Chef Architecture

Development teams can use Chef to describe and automate networking and security for their application workloads within an OpenStack environment. vRealize Orchestrator has a plug-in that enables organizations to use Chef to bring DevOps efficiency to deal with OS configuration, lifecycle management, and application deployment. This plug-in offers a collection of workflows that can be used to automate these tasks. More information about VMware and

Chef can be found at https://www.chef.io/implementations/vmware/ and https://marketplace.vmware.com/vsx/solutions/chef-plugin-for-vrealize-orchestrator.

Puppet

Puppet is written in the Ruby on Rails programming language and uses a declarative Ruby-like domain specific language in its command line environment. It runs on Unix systems as well as on Microsoft Windows. The Puppet DSL is declarative: only a description of the desired end state(s) of the system must be provided; Puppet sorts out the steps needed to get there. Unlike an imperative language, there is no need to figure out all the steps required to achieve the desired end state.

Puppet comes in two flavors: a commercial enterprise product – Puppet Enterprise – and a free open source version. Many enterprises start with the open source version, only to realize they do not have the time or resources to maintain the deployment or that they need additional capabilities which they have to build and maintain themselves.

Open source Puppet is free for use and open to modification and customization. Organizations can get a comprehensive tool with core capabilities and functionality out of the box. For those with the skillset, it is possible to modify the source code directly. With access to a huge collection of modules – reusable, sharable units of code that can be used to extend Puppet across the infrastructure – it is possible to automate tasks such as setting up a database, adding users, installing packages, and updating server configurations.

Puppet Enterprise includes over 40+ open source projects (e.g., MCollective, Facter, Hiera, etc.) with a straightforward installer, better out-of-the-box scalability, and role-based access control (RBAC) that allows delegation of tasks to individuals and groups. Enterprise packaging includes Puppet server reporting for collection of a wide variety of metrics about Puppet server health. Access to professional services is also available.

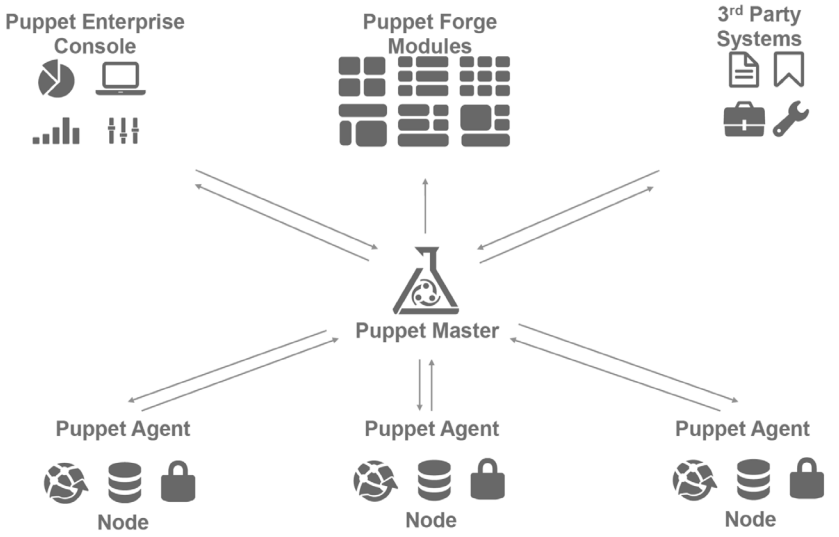


Figure 7.3 Puppet Platform

Examples of simple automation configurations include: enabling NSX ESG syslog, VXLAN segment ID, and NSX ESG load balancing configurations.

Additional useful information about Puppet and NSX integration is available at GitHub (https://github.com/vmware/vmware-nsx).

	Puppet	**Chef**
Language	Puppet's custom JSON-like language; Ruby option available beginning in version 2.6	Subset of Ruby
Approach	Declarative	Procedural/Imperative
License	Apache; earlier versions are GPL	Apache
Approach	List dependencies and Puppet figures out how to order the install.	Write an install script in Ruby using all the extra helper functions from Chef.
Deployment	Puppet Enterprise runs on a machine.	Private Chef runs on a machine; hosted Chef (same price) runs in Opscode's cloud.

Figure 7.4 Comparing Puppet vs Chef

Ansible

Ansible is an open source automation platform. It is an IT automation engine that can be used to automate cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs. It is very simple to set up, easy to learn, and has good documentation.

Designed for multi-tier deployments, Ansible can be used to describe the state of a system and manage its entire lifecycle, rather than just manage one system at a time. Unlike Puppet or Chef, it does not use an agent on the remote host; instead, Ansible accesses all systems under management through SSH. It is written in Python, which must be installed on the remote host. It uses a very simple language – YAML, in the form of Ansible playbooks – that allows users to describe their automation jobs in an easy way.

Ansible also ships with several modules, called the module library, that can be executed directly on remote hosts or through playbooks. Users can also write their own modules. These modules can control system resources like services, packages, or files. They can also execute system commands. VMware NSX Ansible modules are available for teams looking to automate network virtualization. More information about those modules and their use can be found at GitHub (https://github.com/vmware/nsxansible). This repository contains several Ansible modules, written in Python, that can be used to create, read, update, and delete objects in NSX for vSphere.

The other key characteristic of Ansible that makes it a great tool for networking and security is the principle of idempotency. This means it is safe to run the same playbook multiple times against the network, as no change will be performed if the current state or configuration is already the same as defined in the playbook.

Ansible is available for free and runs on Linux, Mac, or BSD. Ansible also has an enterprise product, called Ansible Tower by Red Hat. Using Ansible in NSX allows for automation of end-to-end provisioning of logical networks, logical routers, or logical load balancing without need to interact with anything on the physical (i.e., underlay) network. These virtual networks created by Ansible can be consumed by virtual machines, as they have L2 and L3 connectivity to both virtual and physical networks. In this example, the environment could be part of a continuous integration flow (CI/CD) allowing the networks to be stood up and torn down as required. This allows DevOps engineers to interact with the environment without manual intervention or waiting for physical network changes.

Programming Languages and Tools

There are a variety of programming languages that can be used with NSX REST API, including Python, PowerShell, PowerNSX, Ruby on Rails, Perl, and Go. Using programming languages goes a step beyond simply calling NSX REST API calls via REST client tools; programming languages such as Python or Perl allow for the introduction of custom advanced logic and workflows.

Python

Python is a widely used open-source tool. It supports object-oriented programming with classes and multiple inheritance, where objects can inherit attributes from other parent objects. The simple syntax allows programmers to build the same task using fewer lines of code when compared to other languages. Those characteristics make it the perfect language for someone trying to learn computer programming for the first time. It is easy to find a way to run Python on a computer – Windows, Mac OS X, or Linux – as it is a cross-platform programming language. As Python is very readable, easy to learn, and troubleshoot, it has become one of the most popular tools and is preferred among network admins and DevOps engineers.

Another great feature is the ability to group the code into modules (i.e., a file containing Python definitions and statements) and packages (i.e., several modules), allowing designated blocks of code to run as scripts. There are different kind of modules:

- Those written in Python; they have the suffix: .py
- Dynamically linked C modules; suffixes include .dll, .pyd, .so, and .sl
- C modules linked with the interpreter

Python also comes with a large library. This is often cited as one of Python's greatest strengths. It enables programmers to reuse or customize available code to rapidly deploy new projects within their own environments.

There is a high-level python based library, named PyNSXv, located at GitHub that exposes workflows and a CLI tool to control NSX for vSphere.

PyNSXv can be used in two different ways: as a library by importing the files in the library subdirectory into code; or as a CLI tool by executing "pynsxv" on the command line after installation. More information on how to install and run PyNSXv can be found at https://github.com/vmware/pynsxv.

Power NSX

PowerNSX is a PowerShell module that abstracts the NSX API to a set of easily used PowerShell functions. It is based on VMware Power CLI, a command line and scripting tool built on Windows PowerShell. It provides more than 500 cmdlets for managing and automating components of vSphere, VMware Site Recovery Manager™, vSphere Automation SDK, VMware vCloud Director®, VMware vCloud® Air™, VMware vSphere® Update Manager™, VMware vRealize® Operations Manager™, and VMware Horizon®. The main objective of VMware Power CLI is to provide a PowerShell interface to the VMware product APIs.

PowerNSX also provides a series of PowerShell cmdlets to manage and automate NSX for vSphere. It works seamlessly with VMware Power CLI to bring unprecedented power and flexibility to NSX administrators. For organizations that already know how to use a Power CLI tool, they will quickly feel at home using PowerNSX.

PowerNSX is an open and free software offering. More information is available at GitHub (https://github.com/vmware/powernsx). PowerNSX focuses on exposing new, update, remove, and get operations for all key NSX functions as well as adding additional functionality to extend the capabilities of NSX management beyond the native UI or API.

This module is not supported by VMware, and comes with no warranties express or implied. Best practices recommend testing and validating all functionality before using this product in a production environment.

PowerNSX is a work in progress; it is unlikely that it will ever expose 100% of the NSX API. Feature requests are welcome via the issues tracker on the project's GitHub page – (https://github.com/vmware/powernsx/issues).

PowerNSX can be leveraged to create some interesting tools. An example of Visio integration is shown in Figure 7.5. The NSX Visio Diagramming Tool provides users the chance to diagram networks programmatically. This removes the human time and error elements from documentation. Within a minute of tool execution, a user will have the data needed to visualize. More information about PowerNSX along with numerous use cases can be found in the VMware Press book *Automating NSX for vSphere with PowerNSX.*

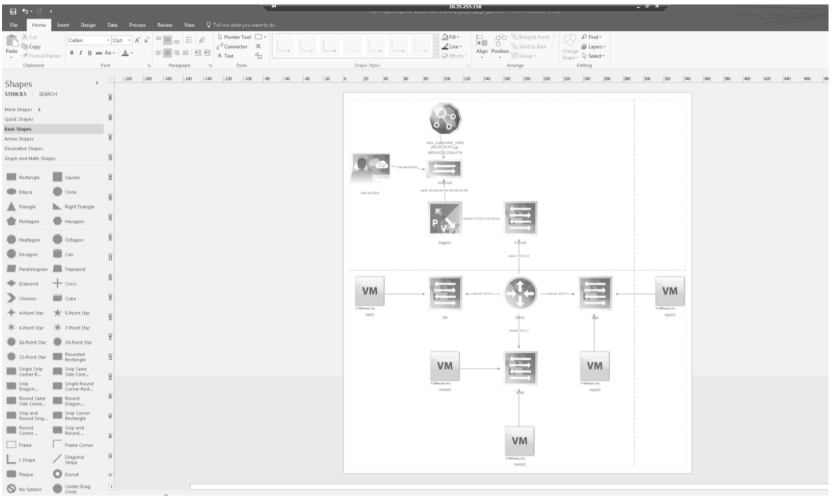


Figure 7.5 NSX Visio Diagramming Tool Example

Conclusion

VMware NSX has many use cases, with automation, security and application continuity the most frequently adopted. As part of VMware's software-defined data center strategy, NSX has been an open platform where anybody can automate and leverage the built-in APIs. While NSX does not require automation tools, most customers use a combination of tools and CMPs. NSX is helping different companies to implement the modern data center, creating on-demand infrastructure where ongoing and frequent reconfiguration is no longer required.

The NSX community continues to provide several tools that can be used to serve specific purposes through the platform's open APIs.

NSX treats the physical network as a pool of transport capacity with networking and security services attached to workloads using a policy-driven approach. This automates networking operations and eliminates bottlenecks associated with the traditional hardware-based approach.

Together with any automation platform, NSX helps enterprises that are embracing cloud computing to speed application development lifecycles, enabling them to rapidly realize and respond to new business opportunities.

Bibliography

Books

VMware NSX – Micro-segmentation – Day 1 – Wade Holmes – VMware Press

VMware NSX – Micro-segmentation – Day 2 – Geoff Wilmington – VMware Press

Operationalizing VMware NSX – Kevin Lees – VMware Press

Automating NSX® for vSphere with PowerNSX – Anthony Burke – VMware Press

API for Dummies Book

Videos

Openstack and VMware Getting the Best of Both – Andrew Pearce - OpenStack Summit 2017 Boston - https://youtu.be/EVM498h4eWE

VMware - OpenStack and VMware: Enterprise-Grade IaaS Built on Proven Foundation – Xiao Gao – OpenStack Summit 2017 Boston - https://youtu.be/QH90GjcSeOs

VMware - Is Neutron Challenging to You? Learn How VMware NSX is the Solution for Regular OpenStack Network & Security Services and Kubernetes – Dimitri Desmidt and Yves Fauser – OpenStack Summit 2017 Boston - https://youtu.be/DHmCga2tAV8

VMware: Production-Ready Clouds with VMware NSX Networking and OpenStack – Dimitri Desmidt – OpenStack Summit 2016 Barcelona - https://youtu.be/r3cT-3qpT60

VMware: Integrated OpenStack with NSX Policy Redirection for NFV: Technical Deep Dive and Demo – Marcos Hernandez and Vanessa Little – OpenStack Summit 2016 Barcelona - https://youtu.be/GwdhITcxp9Q

OpenStack Architecture and Stability – Steve Tegeler - https://youtu.be/fYL4W3aQZMM

Why OpenStack – Steve Tegeler - https://youtu.be/Bk4NoUsikVA

VMware vRealize and OpenStack: A Side By Side Look – Steve Tegeler - https://youtu.be/eNv1iQf52ZE

Service Chaining in OpenStack with NSX – Marcos Hernandez - https://youtu.be/xY1uz6PjWlo

Neutron in the Enterprise – Marcos Hernandez - https://youtu.be/q-O6Noip0Hg

OpenStack and VMware NSX – Marcos Hernandez - https://youtu.be/L3Q7jtQyTdA

NSX for Compliance – Hadar Freehling - https://youtu.be/sFUvglk8Ni4

Documents

Automation Leveraging NSX REST API Guide

https://communities.vmware.com/servlet/JiveServlet/downloadBody/31921-102-3-43550/Automation%20Leveraging%20NSX%20REST%20API.pdf

NSX and vRealize Automation Micro-Segmentation Tech Guide

https://communities.vmware.com/servlet/JiveServlet/download/32774-1-161592/NSX-vRA-Microsegmentation-v1.0.pdf

VMware NSX API Guide

https://docs.vmware.com/en/VMware-NSX-for-vSphere/6.3/nsx_63_api.pdf

https://github.com/vmware/nsxraml/blob/master/md-version/nsxvapi.md

VMware Validated Design Converged Blueprints Implementation Guide

https://www.vmware.com/pdf/vmware-validated-design-30-sddc-converged-blueprints-implementation.pdf

VMware Integrated OpenStack – Design Guide

VMworld Sessions

NET7956 - vRealize Automation and NSX Design Best Practices

NET10011-EDU - Automating Your Network Services Deployments with VMware NSX and Realize Automation

NET8731 - IT Automation with NSX Network Virtualization and Security

NET1535BU - NSX Design—Reference Design for SDDC with NSX and vSphere: Part 1 (2017 – Nimish Desai)

NET1536BU - NSX Design—Reference Design for SDDC with NSX and vSphere: Part 2 (2017 – Nimish Desai)

NET1416BU - NSX Logical Routing (2017 – Jerome Catrouillet and Pooja Patel)

MTE4864U - NSX Load-balancing with Dimitri Desmidt (2017 – Dimitri Desmidt)

NET3282BU - The NSX Practical Path (2017 - Nikhil Kelshikar and Ron Fuller)

NET1338BU - VMware Integrated OpenStack and NSX Integration Deep Dive (2017 – Marcos Hernandez and Russ Starr)

NET3235SU - Why Networking Is at the Heart of Digital Transformation (2017 – Bruce Davie)

Links

www.openstack.org

https://docs.openstack.org/ocata/networking-guide/intro-os-networking.html

https://docs.openstack.org/ocata/install/

www.vmware.com

www.vmware.com/go/nsx

https://www.ibm.com/developerworks/cloud/library/cl-openstack-overview/index.html

https://github.com/vmware/nsxansible

https://github.com/vmware/nsxraml

https://github.com/vmware/nsxramlclient

Index

V

X

This book explores how VMware NSX® delivers the power of automation. *VMware NSX Automation Fundamentals* brings guidance and knowledge on designing the automation for the software defined data center (SDDC), unlocking NSX's full potential to provide the flexibility and agility needed by enterprises today.

VMware NSX improves the network and security posture of the SDDC by fundamentally changing the approach for networking and security. Through NSX's open API model, organizations can select the automation solution best aligned to their operational practices. VMware NSX has already helped over a thousand organizations design, deploy, and manage their SDDCs.

VMware NSX Automation Fundamentals delivers the roadmap to understanding networking and security automation challenges in today's data centers. It demonstrates the fundamental nature of NSX in the data center architecture while detailing integrated solutions for both VMware and third party offerings (e.g., VMware vRealize® Automation™, OpenStack, Puppet, Chef, PowerNSX) that assist in creating networking and security components on-demand.

vmware® PRESS

ISBN-13: 978-0-9986104-5-0
ISBN-10: 0-9986104-5-3

Cover design: VMware
Cover photo: iStock / PhonlamaiPhoto

www.vmware.com/go/run-nsx

$12.99