CCIE and CCDE Evolving Technologies Study Guide

Brad Edgeworth, CCIE No. 31547
Jason Gooley, CCIE No. 38759
Ramiro Garza Rios, CCIE No. 15469

Cisco Press

CCIE and CCDE Evolving Technologies Study Guide

Brad Edgeworth, CCIE No. 31547
Jason Gooley, CCIE No. 38759
Ramiro Garza Rios, CCIE No. 15469

Cisco Press

CCIE and CCDE Evolving Technologies Study Guide

Brad Edgeworth, Jason Gooley, Ramiro Garza Rios

Published by:
Cisco Press

01 18

ISBN-13: 978-0-7897-5972-6
ISBN-10: 0-7897-5972-1

Warning and Disclaimer

This book is designed to provide information about Evolving Technologies in the CCIE and CCDE written certification exam. 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 is provided on an “as is” basis. The authors, Cisco Press, and Cisco Systems, Inc. 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 or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the authors and are not necessarily those of Cisco Systems, Inc.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks

have been appropriately capitalized. Cisco Press or Cisco Systems, Inc., cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through email at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Editor-in-Chief: Mark Taub

Alliances Manager, Cisco Press: Arezou Gol

Product Line Manager: Brett Bartow

Managing Editor: Sandra Schroeder

Development Editor: Kiran Panigrahi

Project Editor: Mandie Frank

Copy Editor: Bart Reed

Technical Editor: David Hanes

Designer: Chuti Prasertsith

Composition: codemantra

Proofreader: Christopher Morris

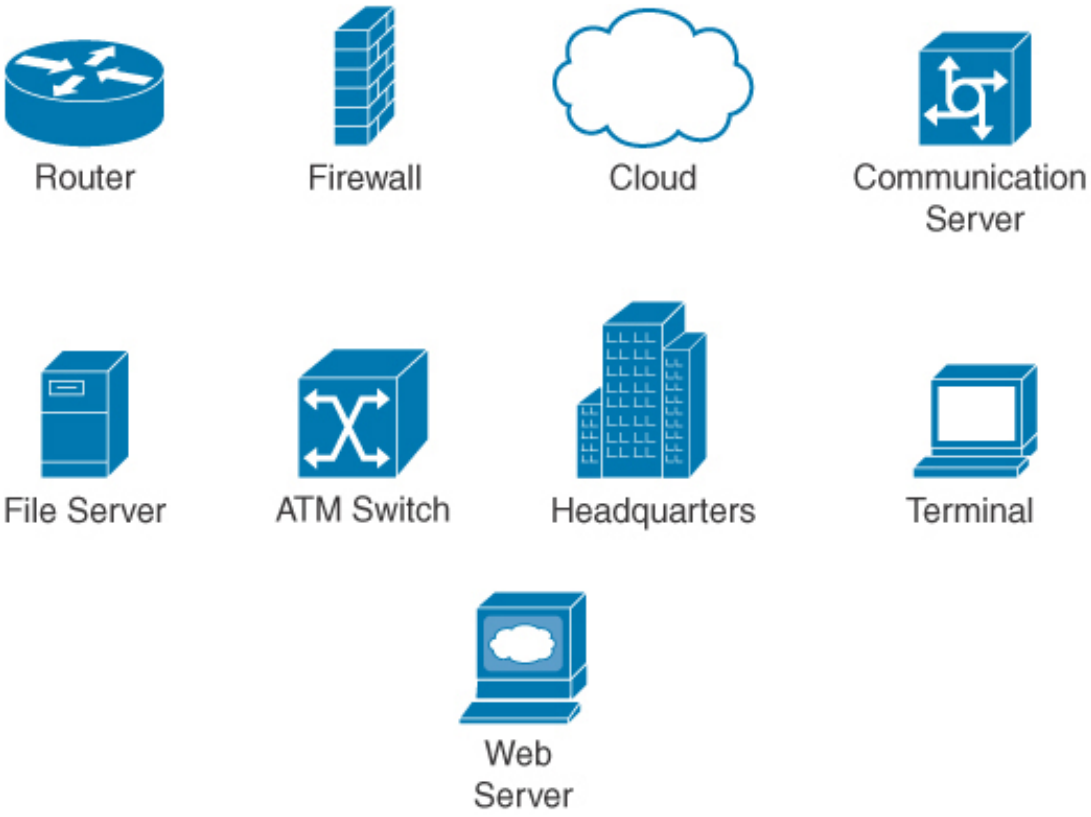
Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices**.

Icons Used in This Book



Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in the IOS Command Reference. The Command Reference describes these conventions as follows:

- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a **show** command).
- *Italic* indicates arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets ([ ]) indicate an optional element.
- Braces ({ }) indicate a required choice.
- Braces within brackets ([{ }]) indicate a required choice within an optional element.

Introduction

Cisco is once more leading the way in building a workforce capable of moving with technological changes through the evolution of its certification programs. Changes to the Expert-Level (CCIE/CCDE) programs will enable candidates to bridge their core technology expertise with knowledge of the evolving technologies that organizations are adopting at an accelerated pace, such as cloud, IoT, and network programmability.

Combining this book with the other Cisco Press certification books that are written for a specific track will provide a complete source of knowledge to help CCIE and CCDE candidates succeed on their written exams.

GOALS AND METHODS

The most important and somewhat obvious goal of this book is to help you pass the written CCIE and CCDE exams. One key methodology used in this book is to help you discover the exam topics that you need to review in more depth, to help you fully understand and remember those details, and to help you prove to yourself that you have retained your knowledge of those topics. This book does not try to help you pass by memorization, but helps you truly learn and understand the topics.

WHO SHOULD READ THIS BOOK?

This book is not designed to be a general networking topics book, although it can be used for that purpose. This book is intended to tremendously increase your chances of passing the evolving technologies components of the CCIE and CCDP written exams. Although other objectives can be achieved from using this book, the book is written with one goal in mind: to help you pass the exam.

HOW THIS BOOK IS ORGANIZED

Although this book could be read cover to cover, it is designed to be flexible and allow you to easily move between chapters and sections of chapters to cover just the material that you need more work with.

The chapters, Chapters 1 through 3 , cover the following topics:

- **Chapter 1, "Internet of Things"**—This chapter discusses the Internet of Things (IoT) from a perspective of business transformations, connectivity, and methods of securing it. Most IoT networks share similarities with Enterprise Networks, but are different in behavior and operational aspects.

- **Chapter 2, "Cloud Fundamentals"**—This chapter provides a holistic overview of cloud environments using virtual machines (VMs) or containers in a public, private, or hybrid model. Topics include cloud service models, connectivity, security, scalability, and high availability designs.

- **Chapter 3, "Foundational Network Programmability Methods"**—This chapter covers modern programmability and automation methods that can be used to interact with different applications and devices through the use of APIs. This chapter also focuses on the Cisco DevNet developer community as well as other important tools to help readers on their programmatic journey.

CERTIFICATION EXAM TOPICS AND THIS BOOK

The questions for each certification exam are a closely guarded secret. However, we do know which topics you must know to *successfully* complete the evolving technologies portion of all CCIE/CCDE-level written exams. Cisco publishes them as an exam blueprint for CCIE/CCDE Evolving Technologies. Table I-1 lists each exam topic listed in the blueprint along with a reference to the book chapter that covers the topic. These are the same topics you should be proficient in when working with Cisco wireless LANs in the real world.

Table I-1 *CCIE/CCDE Evolving Technologies Topics and Chapter References*

CCIE/CCDE Evolving Technology Exam Topic		Chapter(s) in Which Topic Is Covered
1.0	**Cloud**	
1.1	*Compare and contrast public, private, hybrid,*	

	and multicloud design considerations	2
1.1.a	Infrastructure, Platform, and Software as a Service (XaaS)	2
1.1.b	Performance, scalability, and high availability	2
1.1.c	Security implications, compliance, and policy	2
1.1.d	Workload migration	2
1.2	*Describe cloud infrastructure and operations*	2
1.2.a	Compute virtualization (containers and virtual machines)	2
1.2.b	Connectivity (virtual switches, SD-WAN, and SD-Access)	2
1.2.c	Virtualization functions (NFVi, VNF, and L4/L1)	2
1.2.d	Automation and orchestration tools (CloudCenter, DNA Center, and Kubernetes)	2
2.0	**Network Programmability**	

2.1	*Describe architectural and operational considerations for a programmable network*	3
2.1.a	Data models and structures (YANG, JSON, and XML)	3
2.1.b	Device programmability (gRPC, NETCONF, and RESTCONF)	3
2.1.c	Controller-based network design (policy-driven configuration and northbound/southbound APIs)	3
2.1.d	Configuration management tools (agent and agent-less) and version control systems (Git and SVN)	3
3.0	**Internet of Things**	
3.1	*Describe architectural framework and deployment considerations for Internet of Things (IoT)*	1
3.1.a	IoT technology stack (IoT network hierarchy, data acquisition, and flow)	1
3.1.b	IoT standards and protocols (characteristics within the IT and OT environment)	1

3.1.c	IoT security (network segmentation, device profiling, and secure remote access)	1
3.1.d	IoT edge and fog computing (data aggregation and edge intelligence)	1

Each version of the exam can have topics that emphasize different functions or features, and some topics can be rather broad and generalized. The goal of this book is to provide the most comprehensive coverage to ensure that you are well prepared for the exam. Although some chapters might not address specific exam topics, they provide a foundation that is necessary for a clear understanding of important topics.

It is also important to understand that this book is a "static" reference, whereas the exam topics are dynamic. Cisco can and does change the topics covered on certification exams often.

This exam guide should not be your only reference when preparing for the certification exam. You can find a wealth of information available at Cisco.com that covers each topic in great detail. If you think that you need more detailed information on a specific topic, read the Cisco documentation that focuses on that topic.

Note that as technologies continue to evolve, Cisco reserves the right to change the exam topics without notice. Although you can refer to the list of exam topics in Table I-1, always check Cisco.com to verify the actual list of topics to ensure that you are prepared before taking the exam. You can view the current exam topics on any current Cisco certification exam by visiting the Cisco.com website, hovering over Training & Events, and selecting from the Certifications list. Note also that, if needed, Cisco Press might post additional preparatory content on the web page associated with this book at **http://www.ciscopress.com/title/9780789759726**. It's a good idea to check the website a couple of weeks before taking your exam to be sure you have up-to-date content.

TAKING THE CCIE/CCDE CERTIFICATION EXAM

As with any Cisco certification exam, you should strive to be thoroughly prepared before taking the exam. There is no way to determine exactly what questions are on the exam, so the best way to prepare is to have a good working knowledge of all subjects covered on the exam. Schedule yourself for the exam and be sure to be rested and ready to focus when taking the exam.

The best place to find out the latest available Cisco training and certifications is under the Training & Events section at Cisco.com.

TRACKING YOUR STATUS

You can track your certification progress by checking http://www.cisco.com/go/certifications/login. You must create an account the first time you log in to the site.

HOW TO PREPARE FOR AN EXAM

The best way to prepare for any certification exam is to use a combination of the preparation resources, labs, and practice tests. This guide has integrated some practice questions and example scenarios to help you better prepare.

ASSESSING EXAM READINESS

Exam candidates never really know whether they are adequately prepared for the exam until they have completed about 30 percent of the questions. At that point, if you are not prepared, it is too late. The best way to determine your readiness is to work through the "Review Questions" at the end of each chapter and review the corresponding section for any questions you answered incorrectly. It is best to work your way through the entire book unless you can complete each subject without having to do any research or look up any answers.

BOOK CONTENT UPDATES

Because Cisco occasionally updates exam topics without notice, Cisco Press might post additional preparatory content on the web page associated with this book at **http://www.ciscopress.com/title/9780789759726**. It is a good idea to check the website a couple of weeks before taking your exam, to review any updated content that might be posted online. We also recommend that you periodically check back to this page on the Cisco Press website to view any errata or supporting book files that may be available.

Chapter 1

Internet of Things

This chapter covers the following topics:

- Business Transformation and Digitization
- IoT Fundamentals
- Securing IoT

The *Internet of Things,* also known as *IoT*, has become the latest industry buzzword. So what exactly does it mean? At its simplest, IoT is a network of *things* (or devices) that traditionally are not a part of a computer network (printers, laptops, servers, or cell phones). With all these things connected to the network, data can be collected off these *things* and extrapolated in ways that were almost impossible to imagine before. This information can be used to change the ways people have done things in the past and improve the way we live, work, play, and learn.

BUSINESS TRANSFORMATION AND DIGITIZATION

Over the past years, specific companies have dominated their respective markets. The barrier to entry has been high to enter a market, thus making it difficult for new companies.

Entrepreneurs realized that starting a new business in these markets required a new business model to overcome these barriers. By incorporating digitization—the act of transforming information into a digital format—new business models can be developed and thus change the paradigm. Digitization allows new companies to provide new benefits to customers while allowing them to compete with large, well-established businesses.

For example, ride-share companies like **Uber** and **Lyft** have revolutionized and transformed the taxi industry with the use of smartphones. Customers request a ride from their phone, are picked up, and are then transported to their destination seamlessly. The process of hailing a taxicab or trying to find a taxicab company (which varies from city to city) is simplified. Drivers can choose to work a number of hours based on their availability, making it easy for them to maintain work-life balance—and making it more desirable employment than working for a cab company. Digitization offers all these benefits to customers and employees while providing an experience that is better, faster, and cheaper than taking a traditional taxi.

Other companies like **Airbnb** and **VRBO** have disrupted the hotel markets. In fact, most companies have realized that they need to embrace the digitization process and look at other business models so that they can maintain their existing customer base while acquiring new customers through different business use cases.

Digitization is occurring in all market verticals, including banking, healthcare, manufacturing, utilities, real estate, mining, and even government municipalities. In all these markets, organizations are collecting more data, analyzing the data in real time, and taking action based on the real-time analysis of that data.

IoT technologies are a component of a company's digitization architecture. The following market segments are already deploying IoT technologies:

- Manufacturing
- Mining
- Oil and gas
- Utilities
- Smart buildings
- Health and medical
- Retail
- Hospitality
- Transportation
- Connected cities and emergency services

The first computer network derives from the mainframe architecture. Mainframe computers contained massive amounts of processing power, storage, and memory. Mainframes were very expensive to acquire and operate, but they could run multiple programs simultaneously, making them cheaper than other computer systems at the time. Direct console access to the mainframe was limited, and users were force to connect to the mainframe from “thin” (dumb) terminal clients via the first computer network.

As processors became more powerful and easier to manufacture, personal computers provided a lot of functionality at a smaller cost, allowing smaller companies to have access to the benefits of a computer without the costs associating with owning and maintaining a mainframe. Computer networks evolved as they no longer connected thin terminal clients in a small network, instead connecting multiple devices spread across large geographical boundaries.

The protocols to communicate between devices evolved and became more complicated, and they could no longer be managed by the mainframe team. Today, most corporate computer networks are managed by the *information technology (IT)* departments.

Just as the design of these IT networks evolved as more and more computers were connected to them, the network architecture for IoT networks must evolve too. Traditional IT networks follow the common Access-Distribution-Core model that can connect thousands of devices.

An IoT network is often associated with *operational technology (OT)*, which is defined as the hardware/software that influences physical processes through direct monitoring and/or control. OT networks exist in manufacturing facilities. For example, a programmable logic controller (PLC) is responsible for controlling the movement of motors in robots or conveyors based on feedback from sensors. The OT network connects the PLCs, robots, and sensors together.

An IoT network must be able to scale to handle hundreds of thousands of devices or possibly more. IoT networks become more complicated because they can be large scale (hundreds of thousands) and consist of different devices that communicate with different protocols.

Note

> OT and IoT networks are often maintained by engineers who are not from IT departments and hence might not be familiar with traditional IT best practices.

IoT Architecture Models

IoT networks are similar to the first computer networks, which ran multiple network protocols like IPX/SPX, NetBEUI, and TCP/IP. IoT networks must support a range of devices that use a variety of protocols. This is because the hardware refresh cycle is longer than corporate IT systems. As new systems are installed in some parts of a facility, the older systems using different protocols are not updated. Connecting devices that use different protocols requires some skill, and the architecture must also be able to scale to thousands of things while providing security.

Some common IoT architecture models are discussed next.

Machine-to-Machine (M2M) IoT Architecture

In 2013, the European Telecommunications Standards Institute (ETSI), 13 of the founding members, launched an initiative to generate a reference M2M IoT architecture. A key concept was the use of stacks, similar to that of the Open Systems Interconnection (OSI) model used in networking. A key benefit to using stacks is that a component in a stack can be swapped out without impacting the components of the stack next to it.

This provides benefits in the IoT space, considering that multiple things made by different manufacturers often must communicate using different protocols. The M2M IoT architecture focuses on IoT services, applications, and networks by providing interoperability through a variety of application programming interfaces (APIs):

- The **IoT M2M application layer** is focused on providing connectivity of things to the applications. A key component is that the application layer is responsible for tying into other business intelligence (BI) systems.
- The **services layer** refers to the logical components of the network and the management protocol they use.
- The **network layer** refers to the realm in which the things and other devices communicate. It includes the physical network that links them together.

Figure 1-1 demonstrates the machine-to-machine IoT reference architecture.

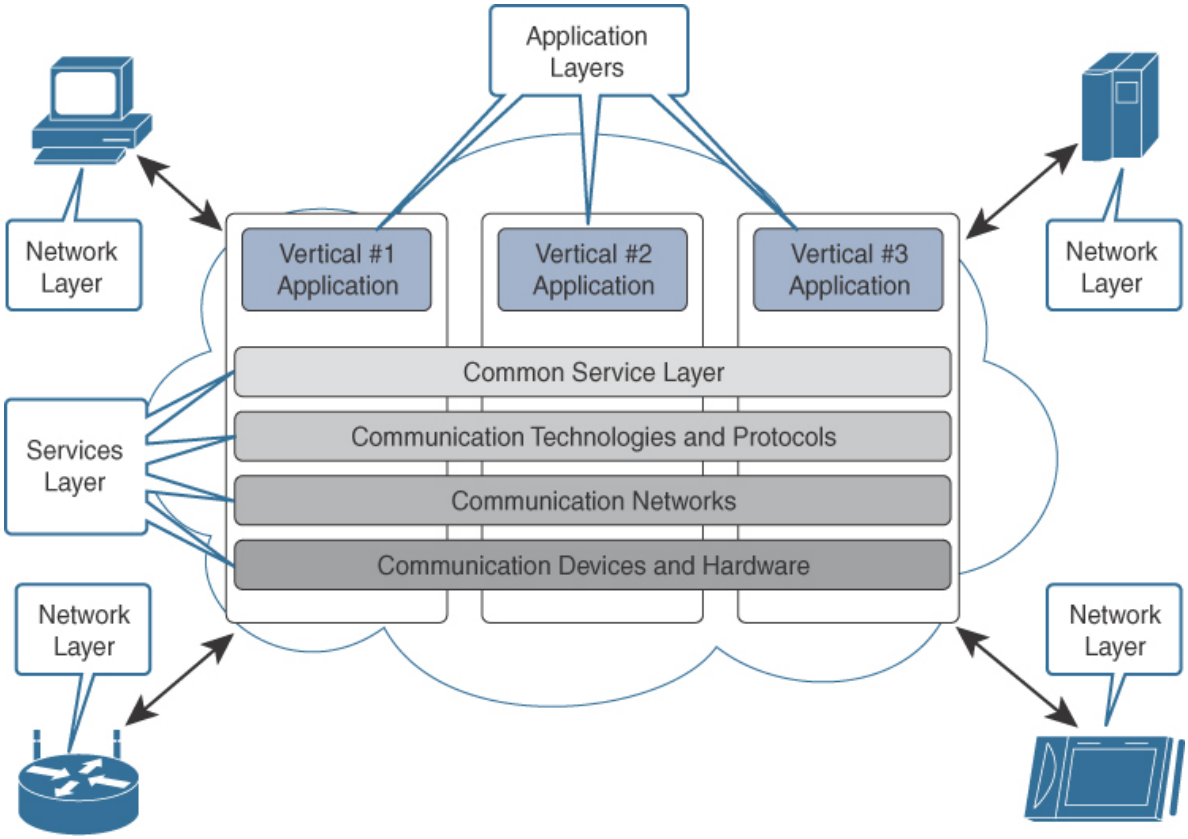


Figure 1.1 *Machine-to-Machine IoT Architecture*

The IoT World Forum (IoTWF) Architecture

In 2014, while in Chicago, the IoTWF created a seven-layer architectural reference model. This model provides clarity and simplicity through visualization of all the IoT components: **sensors**, **network**, **computing**, and **storage**. The model consists of the following seven layers:

- **Physical Devices (Layer 1)**: The bottom layer, which contains things (devices, sensors, and so on).
- **Connectivity (Layer 2)**: This layer provides connectivity among things in Layer 1, Layer 2, and Layer 3. Communications and connectivity are concentrated in this one level.
- **Edge Computing (Layer 3)**: The functions in this level are determined by the need to convert data into information that is ready for processing at a higher level.
- **Data Accumulation (Layer 4)**: This layer is responsible for storing data that was traditionally transmitted live across the wire. The storage of the data allows for analysis or computation at a later time.
- **Data Abstraction (Layer 5)**: This layer is responsible for rendering data and its storage in ways that enable the development of faster or simpler applications. It is responsible for reconciling multiple data formats from different sources, assuring

consistent semantics, and confirming that the data is complete.

- **Application (Layer 6)**: This layer is where information is analyzed and interpreted.

- **Collaboration and Processes (Layer 7)**: This layer encompasses people and processes. In essence, this layer is responsible for providing people the right data, with the right analysis, at the right time so that they can engage the correct process.

Figure 1-2 demonstrates the IoT World Forum Reference Architecture.

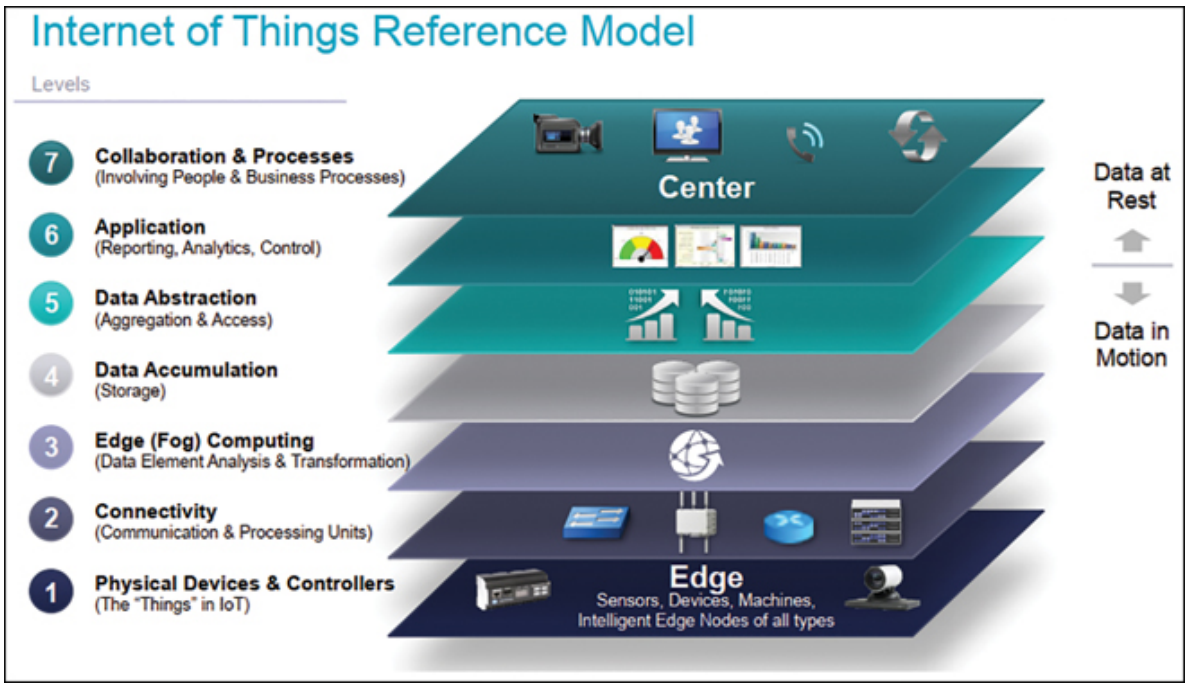


Figure 1-2 *IoT World Forum Reference Model*

Common IoT model

Multiple other IoT reference models have continued to evolve over the years. Some of the models have minor differences between them, whereas others have major differences in how they interact. A simplified IoT Architecture consists of the following:

- Things

- Communications

- IoT applications

Figure 1-3 demonstrates the components of the Common IoT model that will be discussed in the following sections.

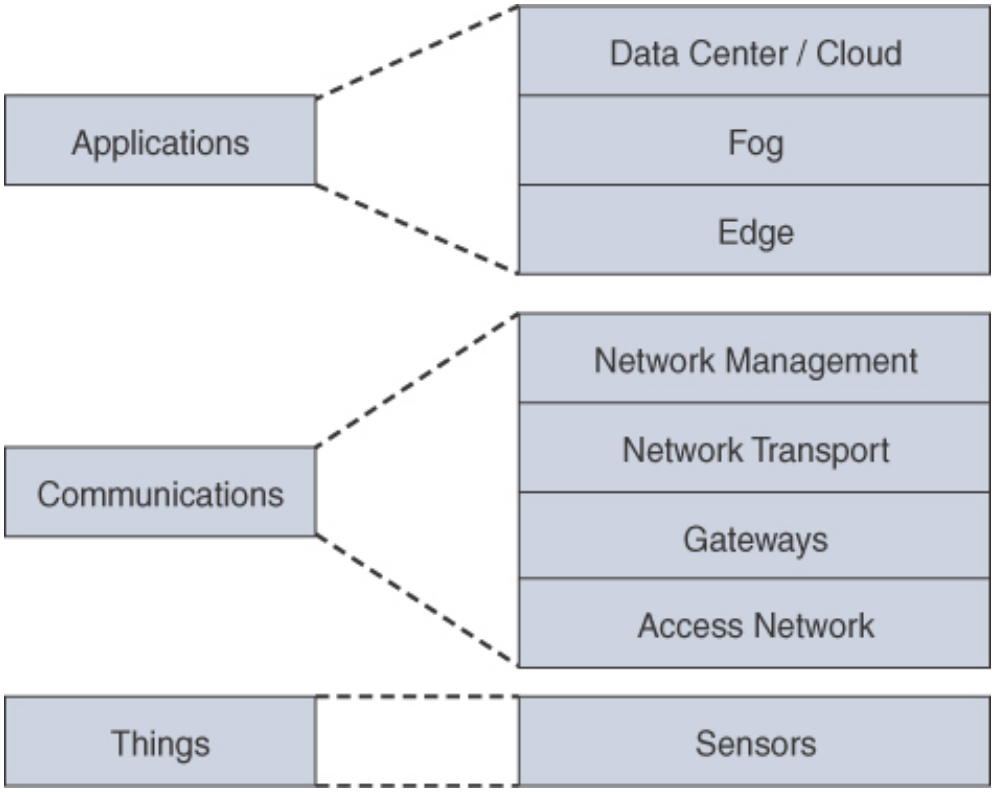


Figure 1-3 *Common IoT Model*

Things

As a simple definition, a thing is a device that provides relevant information. The intelligence on a thing can vary from device to device. Communication between things can occur in the following manners:

- **Unidirectional**: A temperature sensor collects the current temperature from its local thermometer and sends the temperature to a management device for analysis.
- **Local**: A manual thermostat can collect the current temperature for its internal thermometer and then turn on or off an HVAC (heating, ventilation, and air conditioning) unit.
- **Bidirectional**: An electronic thermostat can collect the current temperature from its internal thermometer and then send the data to a management device for analysis. The management device could send a signal back to the thermostat to turn on or off the HVAC unit.

A **smart thing** (also known as a smart object) can take relevant information from the data it receives and take appropriate action after analysis of that information. Some of the newer wireless electronic thermostats are examples of a smart things because they have the capability to be programmed remotely.

Some electronic thermostats can poll a remote temperature sensor and average the temperature of its local sensor to identify the average temperature before turning the

HVAC unit on or off. To some people that might not seem very “smart.” However, what if the electronic thermostat could check two different types of sensors? The electronic thermostat could check a temperature sensor and a sensor attached to a door to see if the door is open. If the door is open, turning on the HVAC system would result in a waste of electricity, and now the thermostat has relevant information about whether the cooled air would just flow out through the open door.

Here are some of the common sensors used in an IoT infrastructure:

- **Temperature**: These sensors collect temperature and report temperature at set intervals.
- **Vibration**: These sensors collect movement across one, two, or three axes. Vibration sensor is a generic term for proximity sensor, displacement sensor, and accelerometer. The frequency of the vibration and application depends on which sensor is used.
- **Pressure**: These sensors can report back liquid or air pressure.
- **Air quality**: These sensors can report air quality, or the chemical breakdown in a specific area. They can be used to detect chemicals that are harmful to humans without protective gear, an improper balance of chemicals for a specific manufacturing process, or the release of toxins into the air that do not meet governmental standards so that the general public can be informed.
- **Water quality**: These sensors report water quality, or the chemical breakdown in the water. Water is a vital component for manufacturing. It can be used for cooling of equipment, for washing of material as a phase of manufacturing, and for the cleanup process. Generally, manufacturers try to reuse their water supply when they can, but certain processes require the water to have an acceptable level of nonstandard components, which could result in the water needing further treatment before reuse.
- **Energy consumption**: Energy consumption is observed and reported as electricity is consumed from the perspective of a wire, electrical component (such as a circuit breaker), or the motor itself.
- **Location**: Location sensors could be GPS sensors that provide specific latitude and longitude, or they could be enabled with a unique wireless signature (Wi-Fi, Bluetooth, and so on) where they can be detected via antennas by an existing wireless infrastructure, and the location can then be calculated using triangulation

based on signal strength at multiple radio towers.

- **Occupancy**: Occupancy sensors can track if an area is being occupied or the number of occupants in an area. This information can be correlated with occupancy in an area at a specific time, which is helpful for identifying resource utilization. Some occupancy sensors have mechanisms to identify one person from another to track repetitive occupancy/utilization.

- **Video**: Video can be used as a livestream camera or can be stored for analysis or replay at a specific point-in-time. Video sensors can be used for a variety of use cases, from physical security to occupancy sensors, for assisting with the visual inspection of manufacturing processes, and for locating a specific person for law enforcement reasons.

> Note
>
> Vibration, energy consumption, and location sensors are used in some of the most common business use cases.
>
> **Vibration**: Placing vibration and torque values on motors and machines allows computers to detect when a failure will occur on a device (motor, engine, truck, robot, and so on). This allows maintenance to be scheduled in advance and repairs to be made without affecting other processes. For example, if a rock truck breaks down in a mine, it blocks the tunnel, preventing other rock trucks from passing in the tunnel. The broken-down rock truck must be towed out of the tunnel to allow operations to be restored. During this time, the mine cannot haul raw product.
>
> **Energy consumption**: Users and businesses are now capable of tracking the amount of energy being consumed in real time. This may result to changes in how a business uses a building or machine in an attempt to lower its cost. However, by connecting these sensors with the utility companies, businesses can now create smart grids. This allows companies to detect power outages and pinpoint failure locations quicker so that service can be restored. Another benefit of using a smart grid is that it allows consumers to use green technologies like solar power to run their homes, or return power to the grid.
>
> **Location**: Location sensors allow for real-time location services (RTLS) such as the following:
>
> - Enabling the transportation industry to track its vehicles and make that

information available to its customers.

- Locating specific resources (forklifts, torque tools, laptops, and so on) quickly. These sensors can also be used for inventory management for manufacturers.
- Locating employees. Generally, this is done to locate an employee with a specific skillset, but could also be used for man-down safety reasons in chemical/refinery environments.
- Restricting or prohibiting certain things or devices from entering a specific area. When an area is defined in the RTLS system, when a tag crosses the barrier, it creates an alarm. This is known as **geo-fencing**.

Table 1-1 provides a high-level overview of typical IoT sensors used in various market verticals.

Table 1-1 *Things and Market Segment Consumption*

	Manufacturing	Mining	Oil and Gas	Utilities	Smart Buildings	I a M
Temperature	✓	✓	✓	✓	✓	✓
Humidity	✓	✓	✓		✓	
Vibrations	✓	✓	✓			
Pressure	✓	✓	✓	✓		

Occupancy					✓	
Air Quality	✓		✓	✓	✓	
Water Quality	✓		✓	✓	✓	
Energy Consumption	✓			✓	✓	
Location	✓		✓	✓	✓	✓

A variety of variables and factors need to be taken into account concerning the type of smart thing that should be used. Specifically, the type of data and business use cases should always dictate which sensors are to be used. The selection of things used should also take into account the following items:

- **Power source**: Is the power source wired or battery driven? If battery driven, how long can the battery last, and how often does it need to be replaced or recharged?
- **Ability to move**: Is the thing physically attached to an immovable object (steel beam, pole, and so on) or is it mobile via a wearable, attached to an animal or a cart?
- **Frequency that data is transmitted**: How often is data expected to be transmitted from the thing? The amount of data and frequency can directly correlate to the life of the thing before a temporary power source must be recharged

or replaced.

- **Connectivity method (wired or wireless)**: How does the thing communicate with other devices? Does it use a wired technology (serial, Ethernet, and so on) or does it use a wireless technology? The transmission rate for that media and distance to the next upstream network device should be accounted for, too.

- **Density of things**: How many things are needed in an area to effectively capture data? Can data be transmitted through other things like a mesh network?

- **Processing capability**: Not all IoT devices have the same amount of processing power or storage. These components vary based on the function and size of the device and directly correlate to the cost.

Communications

This layer provides a mechanism to communicate with other things or external sensors. This layer contains the following constructs:

- Access network (medium)

- Gateways

- Network protocols

- Data exchange protocols

Access Network (Medium): This sublayer refers to the physical medium or wireless technology that is used to provide connectivity between things. Specifically, it is connectivity for thing-to-thing communication, or it provides a centralized aggregation on the network. Physical cables provide consistent availability and bandwidth but are not always available because of cost (initial procurement and/or installation) or not feasible for things that are mobile.

In these circumstances, wireless technologies are commonly deployed. As a general rule of thumb, bandwidth decreases as the wireless range increases. Figure 1-4 demonstrates some common wireless technologies that correlate with a technology's ability to cover bandwidth against the range available. It is important to take into consideration power consumption and cost when selecting a technology. In general, power consumption and cost increase with high throughput and higher distances.

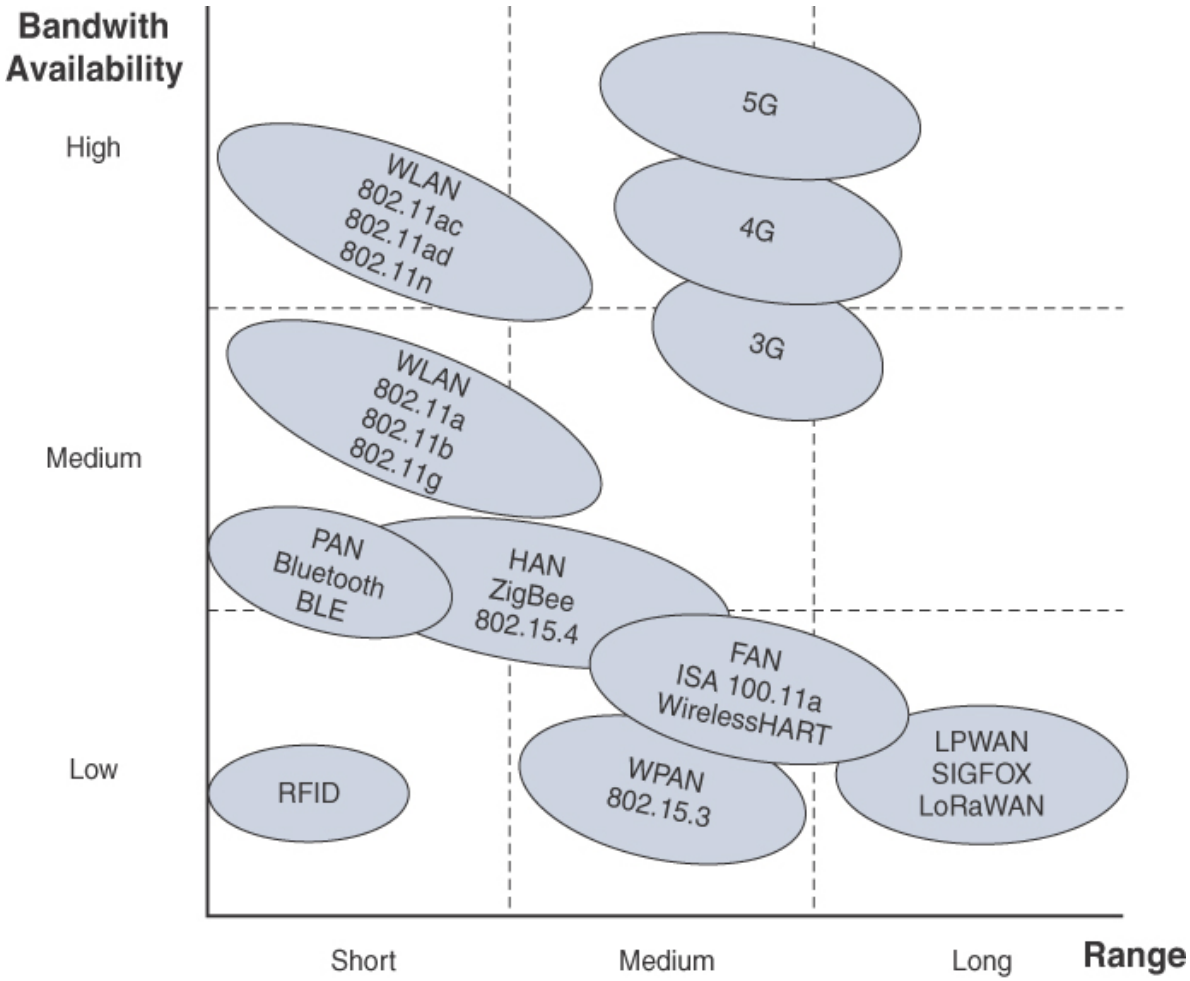


Figure 1-4 *The Effect of Wireless Range on Bandwidth*

This sublayer consists of networks that group things as follows:

- **Personal area network (PAN)**: The network around a person. Bluetooth is a common technology used in PANs.
- **Home area network (HAN)**: The network around a home. This includes Bluetooth Low Energy (BLE), ZigBee, and 802.11 Wireless.
- **Local area network (LAN)**: Network around local devices. Most IoT environments prefer to use Category 5/6 cabling because most electricians can repair replace this cabling. Category 5/6 cables are restricted to 100m for signal integrity. Fibre optic cable can be used to exceed this restriction but is not as commonly used in a LAN environment because of the limited workforce that can service this medium.
- **Field area network (FAN)**: Network around a large field area. In general, the FAN is a wide-open area that covers less than 4km.
- **Wide area network (WAN)**: Network that provides connectivity between sites kilometers apart.

- **Low-power wide area network (LPWAN)**: Wireless network that is designed for long-range communications using low power because devices are powered by battery. LoRaWAN and SIGFOX are common LPWAN technologies.

Gateways: As a simple definition, gateways provide a method of connecting different technologies together through translation at the highest portion of the networking stack. Gateways map semantics between two different things and perform translation between devices. It is important to note that gateways do not just provide protocol translation, but that they can merge different architectures.

From the connectivity context, the gateway connects PANs, HANs, LANs, FANs, LANs, LPWANs to the backbone network. Gateways connect to a backbone network to provide a high-speed and hierarchical architecture.

Gateways can provide a method of aggregation of things and for processing data at the edge of the network. This concept is explained further in the "Data Transportation and Computation" section.

> Note
>
> There is a direct correlation between the increase in cost and operational complexity and the number of technologies and protocols that are merged into a design.

Network Protocols: The network protocol is an essential component for all network transmissions. It is responsible for locating things on the network and controlling the flow of data. The transport layer collects all the data and prepares it into a format that can be presented to an application.

The network protocol should be an open standard considering that most IoT networks come from devices and things from different manufacturers. Proprietary protocols would require translation and introduce complexity when things communicate with proprietary protocols with other manufacturers' things that do not have access to them. The Internet Protocol (IP) has been well established and has become one of the de facto network protocols for IoT.

> Note

Network protocols include the IPv6 protocol too. The use of IPv6 is encouraged because the address space does not have the limitation of the IPv4 protocol. In fact, multiple IoT specifications such as 6LoWPAN and RPL do not include IPv4.

Data Exchange Protocols: The flow of data between things and systems must either be initiated or requested. Some sensors will push data at regular intervals, whereas other sensors require polling of information (also known as a pull). Table 1-2 displays multiple data exchange protocols and some key factors used for selecting a data exchange protocol.

Table 1-2 *Well-Known Data Exchange Protocols Consumption*

Protocol	Transport	Format	Discovery Mechanism	Function
Constrained Application Protocol (CoAP)	UDP	Binary	Negotiation and well-known URI.	REST resource manipulation. Resource tagging with attributes. Resource discovery.
Extensible Message and Presence Protocol (XMPP)	TCP/HTTP	XML	Server with Presence announcement.	Manage presence. Establish sessions. Data transfer.
Message Queuing Telemetry Transport	TCP	Binary	Things connect to a message broker.	Publisher/subscriber.

(MQTT)

IoT Applications

Now that all of these smart things can communicate with each other, the true power of IoT can be realized with the usage of applications. IoT applications are classified as analytic and control applications.

IoT Analytic Application: This type of application collects information from multiple things and organizes it in a manner such that calculations can be made easily. Analytic applications can provide historical data, trends, and suggestions based on the results calculated.

An example of an IoT analytic application could be one that collects the amount of energy consumed in a manufacturing plant on a thing-by-thing basis. The analytic application could provide reports that contain power consumption on a daily basis that could be correlated with the types of machines and product yield for that day. The plant planner could make changes to his schedule or process based on this information to improve efficiency.

IoT Control Application: This application controls smart things. A key concept is that the IoT control application contains the ability to process logic that the smart thing does not contain, or to provide a level of orchestration across multiple things.

An example of an IoT control application comes from the oil and gas segment, where oil needs to be moved from one location to another. The application would open the necessary valves in the pipeline and then start the pumps so that oil can flow from one location to another. In this example, the valves would not have any knowledge of the pumps, or vice versa, and the control application would orchestrate turning the pumps on/off or making the valves opened/closed.

It is important to note that some IoT applications can perform both control and analytic functions.

> Note
>
> Smart services are IoT applications that increase overall efficiency. For example, a program on a thermostat that can detect if a door is open and

overrides the thermostat from turning on the HVAC system can be considered a smart service.

The design of an IoT application must take into account the data structure, frequency, size, and volume of the data transmitted. All of this information directly correlates to the volume of the data transmitted from things to the IoT applications:

- **Structure**: Data exists in either a structured or unstructured format when it is transmitted and stored. Structured data is organized in a logical manner and provides a consistent form of data entry. Unstructured data comes from human language. The same data that is sent in structured format could be sent in unstructured format, but would require context to understand where that data maps.
- **Frequency**: The timeframe in which a thing will transmit data to another thing.
- **Size**: The size of the data correlates to whether the data is structured or unstructured. If the data is structured, then every field should be examined to identify whether or not it truly is necessary, because every field will contribute to the size of the update sent from thing to thing.
- **Volume**: The volume directly correlates with the type of data, frequency of transmission, size of the data, and the number of devices. If a thing transmits 10KB of data every 5 seconds, this might not seem like a large amount of data. However, if there are 1,000 such things, then the total amount of daily data is 172.8GB.

On the other hand, if a thing transmits 100KB of data every minute and there are 1,000 such things sending data; then the daily data amounts to only 144GB. So even if a thing sends more data, if the frequency is lower, then the overall amount of data is lower.

Data Transportation and Computation

Normal corporate networks use a hierarchical network model that divides the network into three layers: **core**, **distribution**, and **access**. Most end-user devices connect to the access layer, and the data center (DC) connects to the core devices (or a server switch block that connects to the core). Most of the data is generated, stored, and processed in the data center.

However, IoT networks generate almost all of their data at the access layer. IoT networks are generally very large, and they must scale appropriately to handle the data

from all of the smart things. The data structure and number of smart things directly correlate to the volume of traffic that is analyzed.

A large IoT network with a million sensors that transmit 15KB of data every 30 seconds can result in 1.8 terabytes (1,800,000MB) of data on an hourly basis (~16 petabytes of data annually). IoT networks of this size must take into account the amount of available bandwidth the access layer has. Depending on the physical access medium, an update could saturate the link.

Figure 1-5 demonstrates four things that transmit via radio towers that can only support 150KB of network traffic. The size of an update is 45KB, which can easily pass across a 150KB link. However, when Thing 1's packet is combined with the updates for Things 2, 3, and 4, the total data exceeds the supported speed of the radio towers. There is not enough bandwidth for all four things' updates.

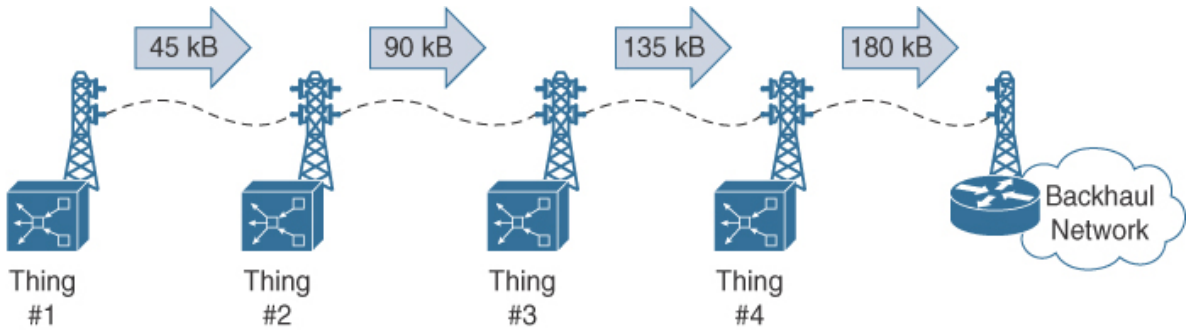


Figure 1-5 *Link Saturation*

Connected vehicles can generate large amounts of data that exceed 20GB per day. The transfer of this data to the manufacturer on a daily basis is not necessary because most of the relevant processing should be performed within the vehicle. Only a small subset of the data generated is needed by the manufacturer to understand how the vehicle can be improved based on its usage.

DATA CENTER AND CLOUD

Data centers are the heart of delivering IT services by providing storage, compute resources, communications, and networking to the devices, users, and business processes and applications. Data centers are designed with proper power and cooling requirements to sustain large amounts of servers for the processing of data. Recently, data analytics have added value and growth in data centers by touching on aspects of an enterprise and its processes.

The growth of cloud-based services is due to an increased focus on business agility and cost optimization. The adoption of cloud services and applications enables faster delivery of services with improved operational efficiencies. Connecting to public-based

cloud services can provide additional cost reductions with improved accessibility, with the assumption that a mechanism exists to encrypt/protect the data in transit.

Latency can become an issue with sending all the data from sensors to the cloud or DC. Depending on the amount of time for a packet to transmit from a thing to a DC, analysis of the data at the DC with the proper response sent back to the thing could exceed an acceptable amount of time. For example, suppose a camera takes a photo of a manufacturing line as parts leave a stamping press for visual inspection as part of the quality control process. Inferior parts should be discarded and removed from the conveyor belt as they are moved to the next station. If there is too much latency, a failed part could move down the conveyor belt. One solution could be to lower the speed of the conveyor belt, but that might reduce the performance of that manufacturing facility.

Placing compute resources in a hierarchical fashion can result in reducing latency, conserving network bandwidth, and increasing local efficiency.

Fog Computing

Fog computing provides a mechanism for reducing data crossing the network by converting flows into information that is ready to be stored or processed at higher levels. The processing occurs in the gateways and network devices.

> Note
>
> The term “fog computing” comes from the fact that the processing is occurring close to the data source, which is similar to how traditional fog is close to the ground. This is in direct contrast to “cloud computing,” which is far away from the source and often outsourced to a third party.

Fog computing can reduce the amount of northbound network traffic through analysis and reduction of data transmitted. The reduction of data occurs outside of typical network data compression techniques. Most network data compression is a data de-duplication algorithm that examines crossing between two devices for redundant patterns. This means that a device examines packets looking for patterns in incremental sizes (for example, 4KB, 16KB, and so on) and creates a signature for each of those patterns. If the pattern is sent a second time, the data is replaced with a signature by the first network device. The signature is sent across the WAN link and replaced with the data by the second network device. This drastically reduces the size of the packet as

it crosses the WAN, but still keeps the original payload between the devices communicating.

Analysis and reduction of data could result in changing the way the data flows. For example, if a sensor is sending temperature data from a sensor in 5-second intervals that includes just the sensor's IP address, temperature, and timestamp, this could result in a 5KB packet being sent every 5 seconds (60KB a minute). However, if the temperature is constant, the fog device could send the first update, which includes the sensor's IP address, temperature, and start time, but not provide an update if the temperature stays the same. When the temperature does change, the fog device would send a 6KB update that includes the sensor's IP address, temperature, and the start and stop times for that temperature cycle. It would then send a second update with the sensor's IP address, the new temperature, and start time.

Just within the timespan of a minute, the fog device reduces 60KB of traffic to 11KB if the temperature does not change. Now if the temperature remains consistent for 1 hour, that would result in a difference of 3.6MB of data, compared to 11KB, which is a drastic reduction in data transmitted.

It is possible for fog computing to be placed into a hierarchical format. For example, a large refinery can contain 100,000 temperature sensors. A fog device could be responsible for receiving and reducing the data that is sent for 1,000 sensors. The fog device could then report to another fog device that receives information from the 100 lower-level fog devices, thus providing a holistic view of the entire refinery with a minimal amount of relevant information to compute in the data center.

Note

Another common use case for fog computing is the translation of information between different network vendors. For example, one sensor may report temperature in Fahrenheit and a different sensor may report temperature in Celsius. It is important for the data analysis to be consistent, so the fog device could translate both sensors' temperature to Kelvin before transmitting for computation.

More information can be found at http://www.openfogconsortium.org.

Edge Computing

Fog computing presents multiple benefits to IoT applications at a very quick pace because of its location to the devices and sensors. Those same advancements could further be refined by increasing the amount of compute resources on the actual smart things. This allows for basic low-level analytics to be performed, and for even faster decisions to be made. In addition, edge computing can still occur and provide feedback to the local operations in the event of a network failure.

> Note
>
> The term “mist computing” has been assigned to edge computing because mist is even lower to the ground when compared to fog.

HIERARCHICAL COMPUTATION STRUCTURE

The use of edge, fog, and centralized computing (data center and cloud) should be structured in a hierarchical manner to provide the most efficient form of processing. The following constructs should be used:

- Time-sensitive data should be analyzed as close to the data source as possible. Processing should occur at the edge or fog node.
- Fog nodes should be used as aggregation points of data to reduce the load on the network. This is especially critical in places that have low-bandwidth medium access to reach the gateway/backhaul network.
- Data can be stored at smaller on-site data centers for local analysis so that actions can be taken by local resources if necessary. However, in general, all of the data for all sites and facilities is housed in a centralized data center or cloud environment to provide historical analysis and long-term storage.

The IoT network architecture needs to take into account multiple factors, such as the amount of data that the things generate, the density of things (from a cell, area, or region), the network access medium, application latency requirements, and the duration for which data is retained.

IOT SECURITY

A key component of any organization is the data it has. It provides a mechanism to differentiate the organization from its competitors or as an incentive to form partnerships with other organizations. Most users assume that steps have been taken to protect their sensitive data. A properly designed network provides data confidentiality, integrity, and availability. Without these components, a business might lose potential customers if they think their information isn't secure.

The following list details the terms and functions associated with data confidentiality, data integrity, and data availability:

- **Data confidentiality**: Ensuring that data is viewable only by authorized users. Data confidentiality is maintained through encryption.

- **Data integrity**: Ensuring that data is modified only by authorized users. Information is valuable only if it is accurate. Inaccurate data can result in an unanticipated cost (for example, if a product design is modified and the product then does not work). When a product breaks, the time it takes to identify and correct the issue has a cost associated with it. Data integrity is maintained via an encrypted digital signature, which is typically a **checksum**.

- **Data availability**: Ensuring that the network is always available allows for the secure transport of the data. Redundancy and proper design ensure data availability.

Other factors must be taken into consideration when we are securing an OT network versus an IT network. With an OT network, the following factors must be considered:

- **Greater scale**: The effects of an outage caused by a security incident could result in a power grid that covers multiple states being shut down.

- **Physical safety**: The compromise of a system could result in the injury of a worker, negative impact to surrounding the environment, or death. If an IoT system for a refinery is compromised, it could open valves that release dangerous or explosive chemicals into an area.

- **Product quality**: The shutdown of the manufacturing processes for some products, or if the processes are modified maliciously, could result in the manufacturing of inferior products that could fail later. Imagine a scenario involving the structural integrity of a product being modified. When the product is later installed and fails, it could result in injury to people years after the security

breach.

- **Software updates**: Most of the devices in an IoT environment are older and typically do not receive software updates. Some devices might not receive an update because they are in use 24×7 for manufacturing processes. This results in things running software with potential defects that can be exploited.

- **Data leaks**: Some environments are closed systems (like utilities), so ensuring data confidentiality is critical.

- **Conflicts between OT and IT teams**: It is very common that IT and OT teams do not trust each other. Ensuring a consistent security model between two different teams that do not trust each other can prove to be a difficult task. Another factor is that IT teams will typically attempt to block traffic when an attack is detected, whereas OT teams will allow manufacturing cycles to continue and then attempt to remediate the systems during the next break in the manufacturing cycle. The OT teams use this logic because of safety concerns for shutting down a system for the aforementioned reasons without following preventative processes for the problems identified.

Threat Vectors

A threat is a possible danger to an environment; it could be natural, accidental, environmental, or malicious. Information security (InfoSec) engineers are primarily concerned with threat vectors (methods for maliciously attacking an organization) and putting mechanisms in place to reduce them.

Most enterprise environments use a **castle security theory** and rely heavily on applying security technologies only at the perimeter of a network that includes the following:

- **Ecommerce connectivity**: These are servers exposed on the Internet that provide additional information to customers or business partners. Generally, hackers are attacking the servers using known software vulnerabilities to gain administrative privileges. Generally, this segment is protected with a firewall, intrusion protection system (IPS), and by ensuring that operating systems and applications are patched for security exploits on a timely basis.

- **Business-to-business links**: These are direct WAN circuits or point-to-point VPN links between business partners to exchange data. Generally, these are safe networks, but a company must protect itself from partner companies in case of

malicious activity or if the partner company were to be compromised. This segment is protected with a firewall and IPS system.

- **Email**: Email is a critical component to any business these days. It provides communication both internally and externally, and normally there is no form of authentication for receiving email from an external sender. This makes it a great technology for sending phishing attacks or embedding malware in attachments to users. This segment is protected with a combination of techniques that use DNS, and/or appliances/cloud-based services that specialize in the scanning of content, URLs, and attachments for known attacks and malware.

- **Internet connectivity**: The Internet is one of the largest threat vectors for enterprise networks. It is extremely easy for a user to be directed to a malicious site that infects their machine with a virus or malware that could then spread to other computers on the network. This segment can be protected with web proxy servers that use the categorization of websites to ensure users can access known-safe sites and that scan content. However, these services are not available when the user takes their work laptop home and tries to access the network. Deploying an antivirus or antimalware client to all of the local computers can provide a second layer of defense for this attack vector.

It is important to realize that the threat vectors for an OT network expand drastically when compared to an IT network. The first major component that must be taken into consideration is the scale of the devices in use. An enterprise with 1,000 employees might have a couple hundred servers in a data center and therefore is looking at about 1,300 to 1,500 potential points of compromise. On the other hand, a small refinery with 1,000 employees might have a couple hundred servers as well as 20,000 sensors spread across the plant. Therefore, that refinery could have about 22,000 to 23,000 potential points of compromise.

Starting at the bottom of the Common IoT model, things become the first point of compromise. Some of these things have very small and limited processing capability. Most of the smaller devices probably will not have a method to secure access to their operating system. Even if they did, how can 20,000 things be patched upon discovery of a security exploit? This comes with the assumption that the device manufacturer is aware of the exploit and produces a patch.

The next threat vector comes from the fact that a majority of the manufacturing protocols are insecure. When Profinet, CIP, EthernetIP, and Modbus were built, the programmable logic controllers (PLCs), robots, and other devices all had minimal

compute and memory resources, so the protocol was built with efficiency in mind. Security was never taken into consideration because the devices operated in a closed-loop environment.

The physical access can present challenges to securing the network, too. Most OT engineers are concerned with basic connectivity when connecting devices with Ethernet. OT engineers would deploy unmanaged network switches that could not support 802.1x network access control (NAC). The more advanced OT engineers have learned the benefits of a managed switch, but still have yet to deploy 802.1x NAC. In either scenario, anyone can walk up to a switch and have direct access to PLCs, robots, or other components of a network.

> Note
>
> Many companies have had their OT networks impacted or shut down because a vendor attached to their network and did not have InfoSec protections that matched the company standard and accidentally released viruses into their company's network.

> Note
>
> Just like some manufacturing protocols do not have security mechanisms included, some of the wireless mediums like LPWAN are built for small packets using low bandwidth, so the use of encryption technologies could have a significant impact on the amount of traffic that can be transmitted across them.

SECURING IOT NETWORKS

Developing a security strategy for an IoT network must take into account the OT network priorities:

- Availability
- Integrity
- Confidentiality

Availability takes priority because an IoT network directly correlates with the capability to generate revenue. If an IoT network is down, it could result in hundreds of dollars to millions of dollars of lost revenue per hour. These values are calculated by loss of goods manufactured, wasted man-hours (assembly workers are still paid even if a robot fails on the line), wasted raw product, or financial penalties for failing to meet *service level agreements (SLAs)* for delivery of utilities (that is, power 99.999% of the time).

Integrity is vital to ensure that data is not manipulated in transit. Manipulated data could result in analytics that result in a different change to a process (that is, the addition of an extra chemical that makes a chemical compound unstable) or that ensure that traffic between a PLC and robot is not modified maliciously (thus causing a motor to break). Integrity is maintained via an encrypted digital signature, which is typically a checksum.

Confidentiality ensures that data is only viewable by authorized users. Data confidentiality is maintained through encryption. The type of data being transmitted between IoT devices typically dictates a company's requirement for confidentiality of that data. For example, the torque being reported by a manufacturing robot might not require confidentiality, but information flowing across an electrical power grid might require confidentiality for safety reasons.

> Note
>
> Some industries contain InfoSec engineers for IT systems, but they do not have InfoSec engineers for OT systems. This results in different levels of security at the OT level, depending on the facility.

IOT SECURITY MODEL

The **Purdue Model for Control Hierarchy** is a well-known model among manufacturing, oil and gas, and utility markets. The Purdue model places devices and equipment into seven hierarchical functions, as illustrated in Figure 1-6.

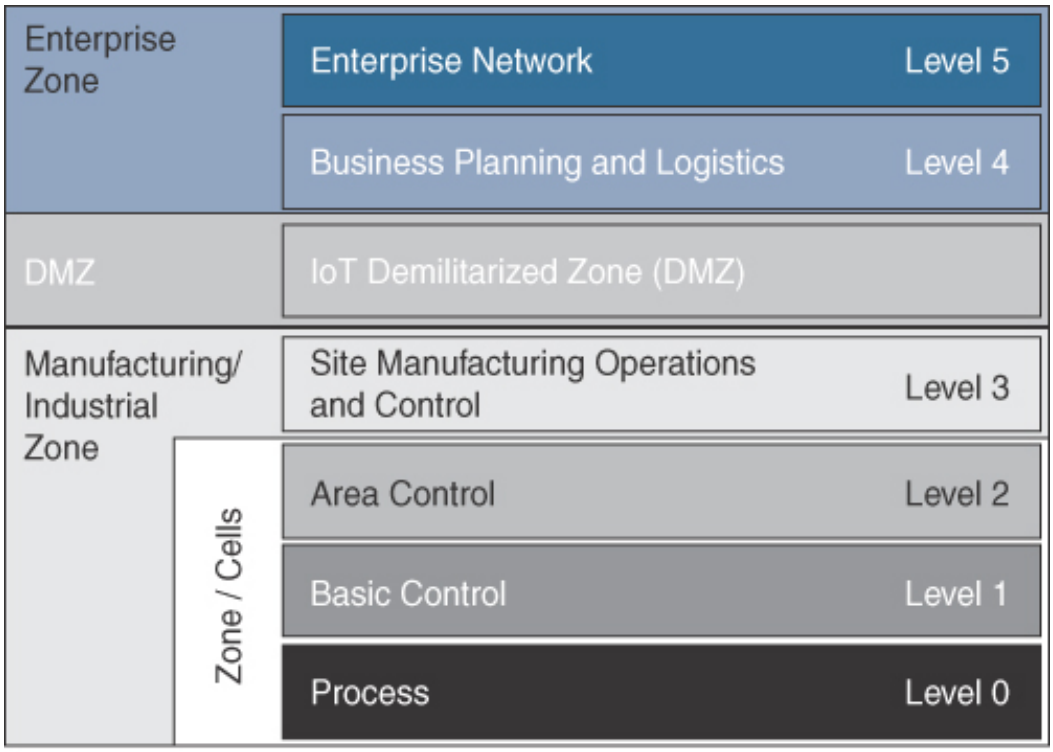


Figure 1-6 *The Purdue Model*

The model consists of the following zones:

- **Enterprise Network**: This is composed of enterprise-level applications that consist of a document repository, sales and billing databases, along with Internet and remote VPN access.

- **Business Planning and Logistics**: This level contains IT-based applications that include scheduling and planning systems such as manufacturing resource planning (MRP). This can also include other services like email, voice, and instant messaging.

- **IoT Demilitarized Zone**: This level is not assigned a number, but it provides a barrier between the manufacturing/industrial and enterprise zones, while allowing users to securely share data and services. All traffic from either side of the DMZ stops in the DMZ. It is possible for a system from the enterprise zone to communicate with a system in the demilitarized zone that triggers a job to send information into the manufacturing/industrial zone. The DMZ should be the only path for traffic in to and out of the manufacturing/industrial zone.

- **Site Manufacturing Operations and Control**: This level includes systems that manage and orchestrate an entire system to produce an entire product. This can include OT network management tools, production scheduling, security management, and control-based applications.

- **Area Control**: This level includes application programming for an entire area or line. It includes access to tools that control a line (that is, human-machine interface,

or HMI) and historians that capture data.

- **Basic Control**: This level contains information from within a smaller area or zone of a manufacturing process. There may be HMI interfaces, but it does not allow for control of devices outside of that zone.

- **Process**: This level controls motors, robots, and sensors. In general, these devices will communicate in this area with PLCs that control them.

Deploying proper security mechanisms in an IoT network should use the following technologies and techniques.

Network Access Control

Most IT-based systems provide authentication within the application itself. For example, a login is required to access a file server or an email account. So even if a user was able to gain access to the network, they would still have to bypass the security mechanisms for that application.

However, OT systems do not have these security components integrated into them because they were developed in a closed loop system. This means that access to the network needs to be restricted to only approved users and devices. When NAC is deployed, users can plug in their computers to a network port, but they will not be able to send or receive network traffic until the NAC system grants them access.

Authentication

The central component to NAC is the authentication layer used to provide and verify the identify information of an endpoint. In IT environments, the identity is associated to a user ID; however, in OT environments, the identity must be linked to a device. The identity could be a MAC address, static user credentials, certificates, or some other form of hardware identification.

Authorization

The second component is authorization, which controls the amount of access a device has when connected to the network. This layer builds upon the core authentication layer by leveraging the identity information. For example, one user can authenticate and have access to the entire network, whereas a different user can authenticate and have access to only the email servers.

Vendors are further expanding the capabilities of NAC by adding context. For example,

a U.S.-based employee may be able to authenticate to the network while visiting his company's office in Australia, but the NAC system can detect that he is out of the country and restrict his access. Context within NAC can now check the following items:

- Platform (Windows, Macintosh, Android, Linux, and so on)
- Operating system version
- Patches installed
- Antivirus installed
- Device that is authenticating the user (abstracting location of the user, too)
- Network traffic patterns

Most IoT devices and applications do not have a built-in authentication mechanism to prevent unauthorized access. This limitation can be overcome by deploying NAC globally in an environment to restrict access to devices by user ID and dynamic ACLs, TrustSec, or similar technologies.

Network Segmentation

The role of the network engineer is to ensure that network traffic can flow from one device to the other, and the role of an InfoSec engineer is to put security measures and controls in place to keep the network safe. The Purdue model does a great job of forcing segmentation between the IT and OT networks, but IT and OT networks are still vulnerable to east/west traffic. For example, assume that a user/device becomes infected with a computer virus in one portion of the network. Because access is not restricted in an area, the virus can spread freely to all the devices within the same network segment.

Outbreaks like this have forced network engineers to become more conscious of security. They are being forced to understand the traffic patterns between devices and insert security mechanisms to permit only business-relevant traffic between areas of the network. This same concept needs to be applied to the OT space because most systems only need to communicate within their area. It is the fog or edge devices that need to communicate northbound, and that traffic could then be inspected further by firewalls or intrusion protection systems.

Figure 1-7 demonstrates a set of four manufacturing cells. Each cell contains its own HMI stations, PLCs, input/output blocks, and robots. There is no interaction between

these cells. Proper network segmentation would block communication between cells at the area switch. This can be accomplished with ACLs, private VLANs, or by forcing traffic through a firewall at a higher level.

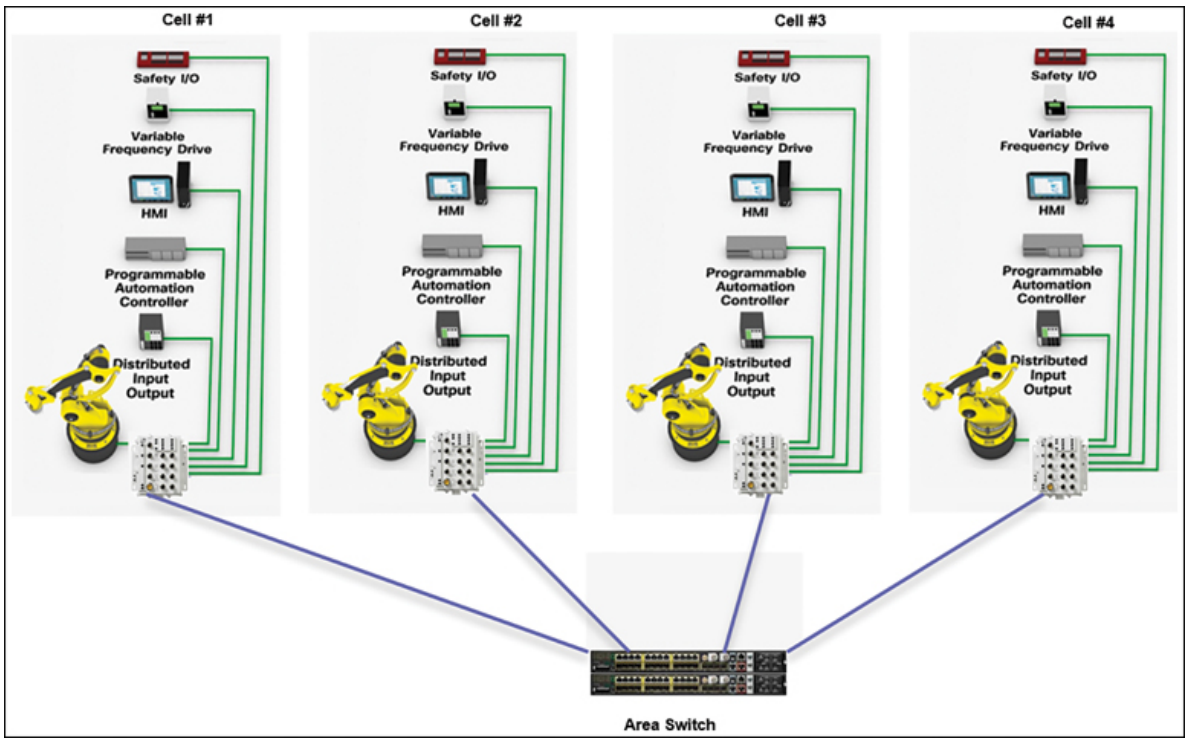


Figure 1-7 *Manufacturing Cells*

> Note
>
> Network segmentation should occur at every level possible: the cell, area, and zone. Conduits are then created to allow proper communication between devices that need to talk to each other.

Network Visibility

Historically, InfoSec engineers would review the logs and information provided to them from their firewalls and intrusion protection systems to detect security incidents. This information would be fed into a security information and event management (SIEM) system that would cross-correlate events to detect an attack. However, the ratio of security devices to hosts or network devices is very small, and these security devices might not detect every security incident.

More advanced security departments have realized that they can collect information from network devices (routers and switches) that can provide additional contextual information for traffic patterns. Specifically, collecting information about network traffic (source/destination IP address, protocol, and port) through NetFlow can provide

relevant data. This data can then be processed with analytics to create a baseline of normal network traffic. If a host starts to transmit data that is irregular from a new application or volume, it could alert InfoSec engineers of a potential security incident so that it can be investigated. This same type of analytics can also be used by NAC to help provide context to the device and identify the type or function of the device as part of the authorization process.

> Note
>
> There are significant advances in network visibility systems that provide direct interaction with NAC systems. For example, a threat could be detected on the network visibility system that forwards commands to the NAC system. The NAC system notifies the switch to shut down the switch port that the attacker/infected PC is connected to.

Secure Remote Access

Supporting devices in the lower levels of the OT network can provide complications of their own. If a device fails and requires troubleshooting, it could take hours for an engineer to travel for on-site diagnostics. Often, organizations will ask for remote connectivity to their robots and sensors to improve responsiveness to customer demands and reduce travel costs.

In IT environments, a vendor is provided with VPN credentials to log in to the organization's network, which then provides the vendor with connectivity to their systems to provide support. While this works well, as those systems are in the enterprise portion of the Purdue model, access to OT devices in the manufacturing/industrial zone need more scrutiny for safety reasons.

How can an organization be assured that the vendor has verified that it is safe to operate a device (like a robot) without causing injury to someone nearby? This question alone reiterates the fact that no traffic from the enterprise level should be allowed direct access to the manufacturing/industrial zone.

The best way is to provide a jump box system in the industrial DMZ. A jump box is a dedicated computer with all of the appropriate tools to manage devices in the OT portion of the network. The vendor then establishes connectivity to the OT network through the jump box in the DMZ. For example, an on-site technician can implement all the proper physical safety controls and then launch a screen share session on the

jump box.

> Note
>
> Remote access via dial-up modem, LTE router, or other VPN device that is placed in the OT network could represent a threat. Some vendors will place these devices as part of their design, unbeknown to the customer, as a backdoor. In some countries, this is an illegal activity for certain industries like utility companies.

SUMMARY

The Internet of Things has grown drastically over the past years. Businesses are now realizing the value that can be extracted from sensors when combined with real-time analytics. These technologies are allowing them to digitize their business model and reinvent themselves by inserting technology into markets that reside outside of the tech industry. These technologies provide them with a differentiator, thus allowing them to be more efficient and profitable.

Enterprise networks are managed by IT network engineers, whereas OT networks have been managed by non-IT engineers. A company is more successful with its IoT deployment when it can have the IT and OT networks communicate and exchange information.

The design of an IoT network must accommodate the following items:

- **Things**: What type of thing is being used, how much data is being sent, how frequently is the data being sent, and how many things are deployed in an area?
- **Communications**: What network medium is being selected, is a gateway being used, what does the backhaul network look like, what network protocol is being used, and which protocol is being used to exchange information between devices?
- **Computation models**: Are the things capable of edge processing, or can we use fog-based processing to reduce latency and bandwidth requirements on the network?
- **Security**: How can we ensure secure communication between the devices, and how can the number of threat vectors be reduced?

The widespread adoption of IoT devices has begun. Some companies have migrated their product portfolio to IoT devices because of the future growth. Analysts predict that there will be 50 billion devices that produce over 3.3 zettabytes (that is, 3,300,000,000,000,000,000,000 bytes) of traffic in the near future (2021).

REVIEW QUESTIONS

1. Where is the preferred location for the processing of IoT data?

1. Edge nodes
2. Fog nodes
3. Data centers
4. Mainframe computers
5. Cloud service providers

2. A fog node can process data and send the data to another fog node. True or False?

1. True
2. False

3. IoT networks are easier to secure than traditional IT networks. True or False?

1. True
2. False

4. When defining an IoT security model, which factor is the most important to consider?

1. Cost of solution
2. Availability
3. Complexity
4. Integrity

5. Confidentiality
6. None of the above

5. What is one significant factor that must be taken into consideration with an OT network as compared to an IT network?

1. Data confidentiality
2. Safety
3. Data integrity
4. OT protocol capability for encryption
5. Remote access for vendors

REFERENCES

Hanes, David, Gonzalo Salgueiro, Patrick Grossetete, Rob Barton, Jerome Henry. *IoT Fundamentals*. Indianapolis: Cisco Press, 2017. Print

Philippe Roggeband. "Architecting Security for the Internet of Things" (presented at Cisco Live, Barcelona, 2018).

JP Vasseur. "The Internet of Things: An Architectural Foundation and Its Protocols" (presented at Cisco Live, Milan, 2014).

Robert Albach, Sr. "Industrial Security: IT vs. OT Deployment Practices" (presented at Cisco Live, Las Vegas, 2017).

Marc Breissinger. "Understanding and Troubleshooting Edge and Fog Fabric Solution" (presented at Cisco Live, Barcelona, 2018).

Samer Salam. "The Evolution from Machine-Machine (M2M) to the Internet of Everything: Technologies and Standards" (presented at Cisco Live, San Diego, 2015).

The Zettabyte Era: Trends and Analysis. 2017. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html. June 7, 2017.

Chapter 2

Cloud Fundamentals

In this chapter, you will learn the following:

- Cloud Fundamentals
- Performance, Scalability, and High Availability
- Security Implications, Compliance, and Policy
- Workload Migration
- Compute Virtualization
- Virtualization Functions
- Cloud Connectivity
- Automation and Orchestration Tools

It's a multicloud world, where organizations are leveraging cloud computing services from multiple cloud providers. Cloud services are available on demand, are scalable, and can be accessed from anywhere. They allow organizations to focus their IT resources on strategic innovations such as high-end business application development, business analytics, new business models, and applications to enhance the productivity of employees as well as the customer experience. With the use of cloud, expenses move from data center infrastructure costs in hardware and software to costs based on the usage of compute resources, and this shift dramatically reduces costs.

CLOUD FUNDAMENTALS

Before we explain the fundamental concepts of cloud computing, you need to understand the following terminology to be able to quickly grasp the concepts explained in this chapter:

- **Abstraction**: To reduce or hide details to make something easier to understand. For example, the cloud hides or abstracts all the underlying hardware and software components that comprise it.

- **Server**: A server is a specialized computer, either physical or virtual, that is used to host server programs that can accept and process requests from clients. Some examples of the most typical servers include web servers, application servers, database servers, email servers, and file servers.

- **Tenant**: A logical application container that requires isolation from other tenants or the outside world. A tenant can be an application, an organization, a business unit, a single department, and so on.

- **Bare-metal server**: This simply refers to a physical server that is dedicated to a single tenant (in other words, not shared with other tenants).

- **Virtualization**: Virtualization is the process of using software on a computer or physical server that allows multiple operating systems to run on virtual machines (VMs) on a single computer. A bare-metal server, for example, can only operate with one operating system (OS) at a time, whereas a server running virtualization software (VMware, Hyper-V, and so on) can be used to run multiple VM instances at the same time, each with independent OS configurations.

- **Multitenant**: A software architecture in which a single server or computing environment is shared but logically isolated to serve multiple tenants.

- **Databases**: An organized collection of data that that can be efficiently queried, managed, and updated by other applications (for example, SQL).

- **Middleware**: Anything that is not part of an OS and not part of a software application, but rather lies in between. Therefore, a web server, for example, qualifies as middleware because it is not part of the OS and it's not the actual application.

- **Workload**: A workload is an abstraction of the amount of work or processing required by a software program, and if applicable, the number of incoming

connections from clients interacting with the software program. In cloud terms, a workload usually refers to one of the following components:

  - Virtual machine
  - Container
  - Container pod
  - Application
  - Middleware (that is, a web server)
  - Database
  - Service

- **On-premises (on-prem)**: This simply refers to a location that belongs to the customer. For example, a traditional IT data center would be an on-prem data center.
- **As a Service (aaS)**: This means services are delivered through software from the cloud.

The National Institute of Standards and Technology (NIST) defines cloud as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

In addition, it mentions that a cloud environment is composed of five essential characteristics, three service models, and four deployment models, as shown in Figure 2-1.

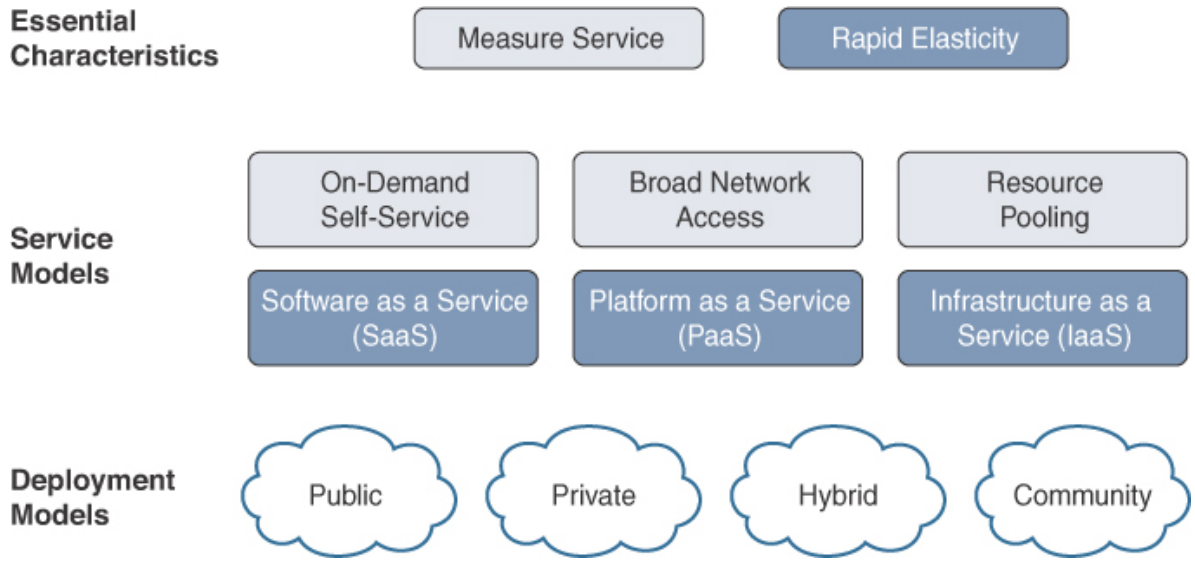


Figure 2-1 *NIST Cloud Model*

Essential Characteristics

The following are five essential characteristics as defined by NIST that all cloud services should offer. If any of them is lacking, it is likely not a cloud service.

- **On-demand self-service**: This means end users have access to a portal where they can directly request computing resources from the cloud. Deployment will be completely automatic without the need for any human interaction with the cloud service provider (CSP).

- **Measured service**: Just like with a water or power bill, you may be charged for cloud resources such as CPU, storage, and bandwidth, depending on how much you use them. There are some CSPs that offer free basic services, and for additional premium capabilities a paid subscription is required (for example, Dropbox, Box, and Lucidchart).

- **Rapid elasticity**: This means the cloud should be able to automatically scale up/out and scale down/in based on service utilization. When a consumer requests changes through a portal to increase or decrease the amount of computing resources needed, the changes should happen automatically in the cloud.

- **Resource pooling**: The CSP should have a pool of resources that can be shared among different consumers or tenants. This means that the CSP should support multitenancy and that the resources used by the tenants are shared. When a tenant stops using a resource, it can be decommissioned, put back into the pool of resources, and become available for other tenants to consume. This can be achieved by using orchestration and automation tools.

- **Broad network access**: This is the ability to reach the cloud from anywhere using

mobile phones, tablets, laptops, workstations, and so on.

> Note
>
> One big difference between cloud computing and traditional virtualization is that although virtualization can abstract resources from the underlying physical infrastructure, it lacks the orchestration to pool them together and deliver them to consumers on demand and without manual intervention.

Service Models

The following sections describe the different cloud service models available today.

Infrastructure as a Service (IaaS)

Figure 2-2 illustrates that with IaaS, the service provided to the customer is virtualization, servers (compute capacity), storage, networking, and other fundamental computing resources where the customer is able to deploy and run any OS, middleware, and applications. The customer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications. The cloud service providers may also include limited control of networking components (such as a virtual switch).

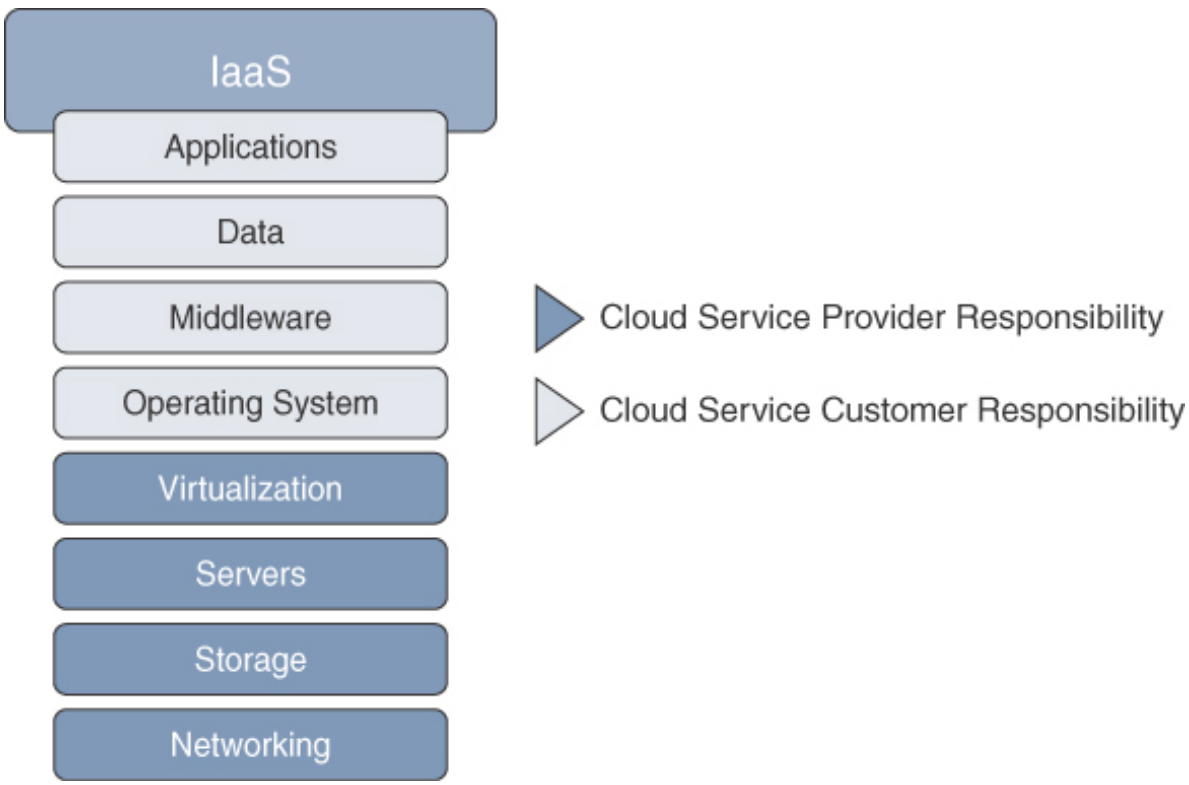


Figure 2-2 *IaaS Stack*

This type of service model is great for IT departments because it allows them to migrate their on-prem applications easily into the cloud.

Some examples of IaaS services include the following:

- Amazon Web Services (AWS) Elastic Compute Service (EC2)
- Google Compute Engine
- Microsoft Azure IaaS
- IBM Bluemix

Platform as a Service (PaaS)

This is very similar to IaaS; the only difference, as illustrated by Figure 2-3, is that PaaS also provides the OS as well as development application platforms (that is, the ability to run PHP, Python, or other programming languages), databases, file storage, collaboration, machine learning, big data processing, and so on. The consumer can deploy third-party applications supported by the CSP or they can develop their own application in the cloud without the worrying about the complexities of managing the underlying infrastructure services.

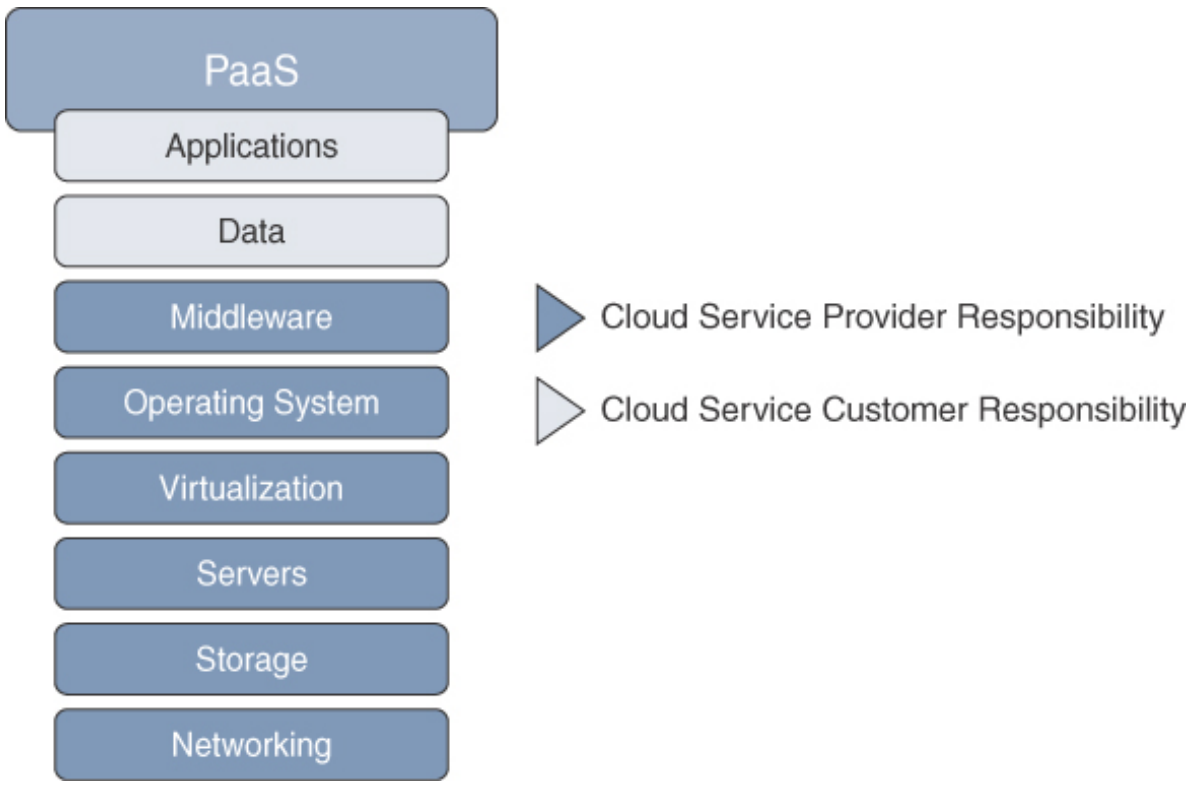


Figure 2-3 *PaaS Stack*

This type of service model is ideal for developers because they don't have to worry about having to install a database or any middleware. With a PaaS service model, all of

this is included and there is no need to manage the underlying servers, networks, or other infrastructure. All that needs to be provided is the application and the application data.

Some examples of PaaS services include the following:

- AWS Elastic BeanStalk
- Google App Engine
- Microsoft Azure
- IBM Bluemix

Software as a Service (SaaS)

In this type of service model, everything is provided by the cloud service provider except the application data, as illustrated in Figure 2-4. This is the type of service that everyone uses on a daily basis; for example, web-based email (such as Gmail), Dropbox, Facebook, and so on, are all SaaS services.

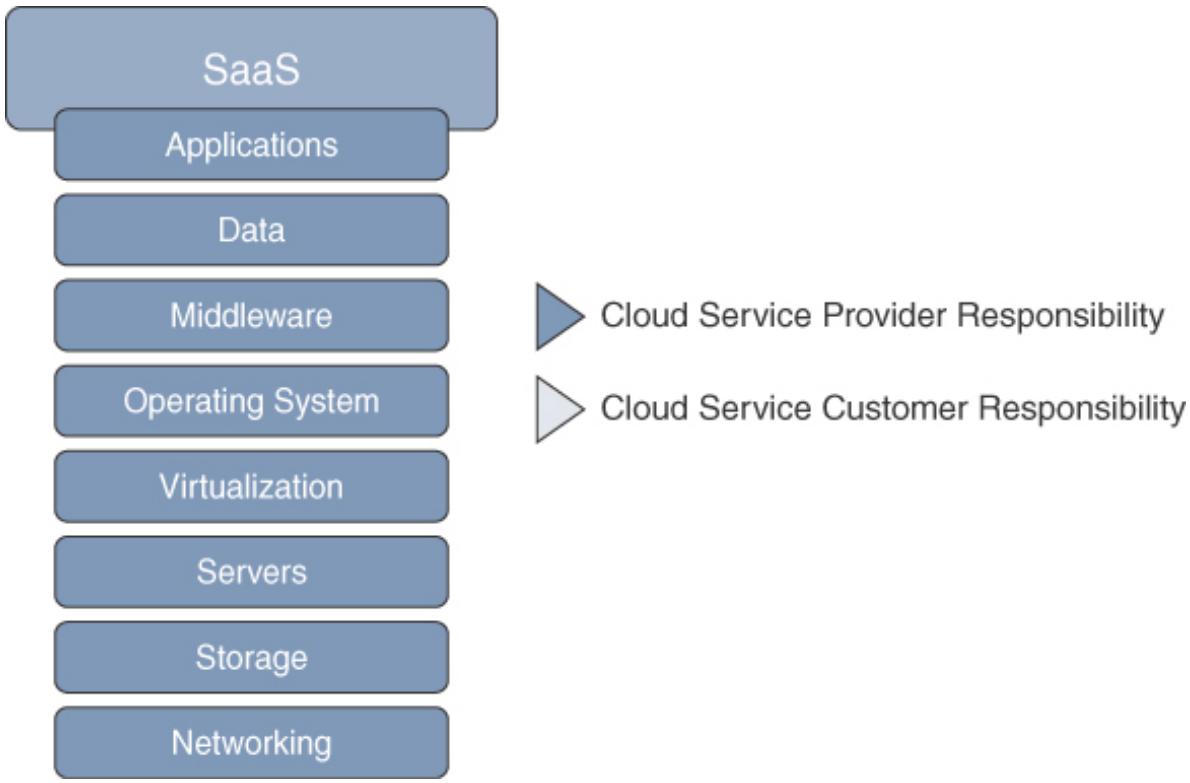


Figure 2-4 *SaaS Stack*

Some examples of SaaS services include the following:

- Salesforce.com
- SAP Business by Design

- Oracle on Demand
- Office365
- Cisco Webex
- Box

XaaS (Everything as a Service)

This can be seen as a combination of IaaS, PaaS, and SaaS where everything can be a service. Here are some examples:

- Desktop as a Service
- Backup as a Service
- Database as a Service
- Security as a Service
- IP Telephony as a Service
- Disaster Recovery as a Service
- Monitoring as a Service
- VPN as a Service
- Analytics as a Service

Cloud Deployment Models

The following sections describe the different cloud deployment models.

Public Cloud

For this type of cloud deployment model, the cloud infrastructure is available to the general public over the Internet. This type of cloud is owned, managed, and operated by the CSP. Some examples include the following:

- Amazon Web Services (AWS)
- AWS GovCloud

- Oracle Cloud
- Google Cloud Platform (GCP)
- Microsoft Azure
- Microsoft Azure Government
- IBM Cloud
- Alibaba Cloud

Private Cloud

The cloud infrastructure is provisioned for exclusive use by a single organization. It may be owned, managed, and operated by the organization, a third party, or a combination of the two, and it may exist on or off premises.

Here are some examples of on-prem private clouds:

- OpenStack
- Microsoft Azure Stack
- VMware vCloud Suite Private Cloud

Cloud providers can also emulate a private cloud within a public cloud environment (think of it as a cloud within a cloud). Amazon Web Services and Google Cloud Platform call this type of private cloud a Virtual Private Cloud (VPC), whereas Microsoft Azure calls it a Virtual Network (VNet).

A VPC or VNet isolates resources for a cloud tenant from other tenants through a private IP subnet and a virtual network segment defined by the user.

Community Cloud

A community cloud is a private cloud that serves a group of organizations that have shared common interests such as mission objectives, security, privacy, and compliance policies and standards. Similar to a private cloud, a community cloud may be owned, managed, and operated by the organizations or by a third party, and it may exist on premises (that is, an on-site community cloud) or off premises (that is, an outsourced community cloud).

Community clouds are very common for the public sector or the government because they require regulatory standards that will be described later in this chapter in the "Security Implications, Compliance, and Policy" section.

Examples of community clouds include the following:

- AWS GovCloud
- Google Apps for Government
- Microsoft Government Community Cloud
- Salesforce Community Cloud
- Capital Markets Community Platform (NYSE)
- Healthcare Community Cloud (Carpathia)

Hybrid Cloud

Figure 2-5 illustrates a hybrid cloud, which is when a private cloud and a public cloud combine and are bound together by standardized or proprietary technology that enables data and application portability (that is, cloud bursting for load balancing between clouds).

Figure 2-5 *Hybrid Cloud Deployment Model*

Examples of hybrid cloud solutions include the following:

- Cisco and Google Cloud Hybrid Cloud Solution
- Microsoft's Hybrid Cloud Platform with Microsoft Azure Stack
- Terremark CloudSwitch
- IBM Hybrid Cloud Solution
- Rackspace Hybrid Cloud
- Fujitsu Hybrid Cloud Services

Multicloud

A multicloud (not defined by NIST) enables customers to consume applications and services from two or more clouds, where at least one cloud is public. To understand the difference between multicloud and hybrid cloud, let's compare them from the networking standpoint:

- **Multicloud networking**: Network transport is from private cloud to multiple public cloud providers and/or between multiple public cloud providers (where supported) and/or between multiple private clouds (including vNETs/VPCs). This is illustrated in Figure 2-6.

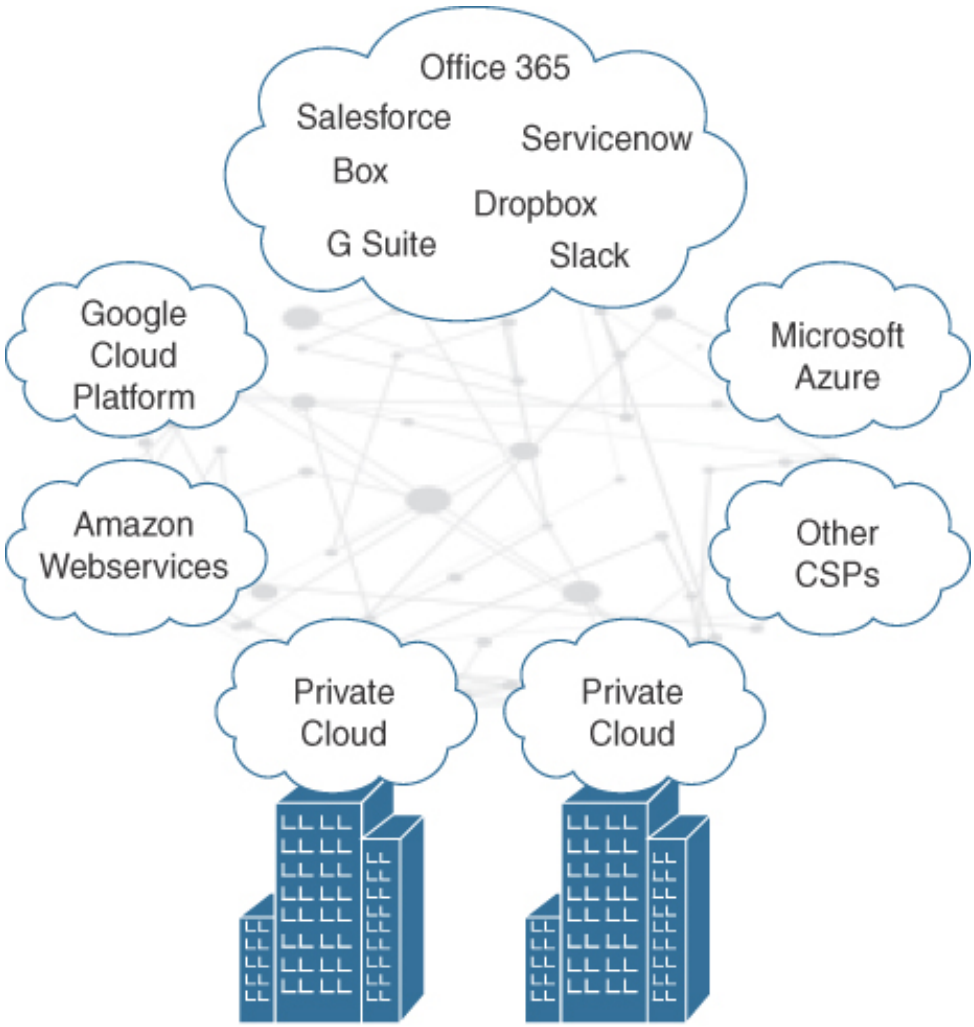


Figure 2-6 *Multicloud Deployment Model*

- **Hybrid cloud networking**: Network transport is from private cloud to a single public cloud provider. In other words, hybrid clouds are a subset of multicloud.

The following are some of the benefits from adopting a multicloud solution:

- CSP high availability
- More options for regional cloud provider access
- The ability to overcome feature disparity between CSPs, regions, and/or services (machine learning, tooling, security regulations, and so on)
- No CSP lock-in

PERFORMANCE, SCALABILITY, AND HIGH AVAILABILITY

Performance, scalability, and high availability are commonly used interchangeably, because to a certain degree, they can affect each other, but they are not exactly the same thing.

For example, application performance indicates how fast an application can execute an action within a time interval, whereas scalability would be a system's ability to handle an increase in demand without impacting the application's performance. If the increase in demand is too large to handle and resources are not available to keep up with the demand, this would have an impact in application performance that would cause the application to become sluggish, unresponsive, or to crash. This is where high availability comes in, which can be achieved by scaling (up or out) the resources or seamlessly failing over to another system while the application remains available and accessible without any noticeable interruption. For a system to have true high availability, it not only needs to be always on but must also provide a service level agreement (SLA) of at least 99% of uptime/availability.

Application Scalability and Elasticity

Scalability can either be vertical (scale-up), by adding more resources to a system (memory, CPU, storage, bandwidth, and so on), or horizontal (scale-out), by adding more systems (that is, by means of clustering or load balancing). This way, applications can either scale up or scale out to prevent a lack of resources from affecting performance or availability. But there is more to just scaling up or scaling out; what if the demand decreases and you end up with an overprovisioned system? That's where cloud elasticity comes in, which is the ability to increase (scale out/up) or decrease (scale in/down) resources to meet changing demand without overprovisioning capacity.

Elasticity is typically referred to as auto-scaling by CSPs. Some examples of these services include the following:

- AWS Auto Scaling
- Microsoft Azure Autoscale
- Google Cloud Platform Autoscaling

Elasticity can give the illusion of infinite cloud resources while addressing performance, scalability, and high availability concerns.

Application Performance with WAN Optimization

One key factor that can have an impact in application performance is that most traditional applications do not take the network characteristics into account and typically rely on protocols like TCP for communicating between different systems. Traditional IT applications tend to be chatty and are usually designed for LAN environments that can provide high-speed bandwidth and are not easily congested. When an application resides across a WAN, most users assume the application responsiveness or performance is directly related to the available bandwidth on the WAN link. While bandwidth is one of the factors that can affect application performance, there are other factors that can also affect performance, such as path latency, congestion, and application behavior.

When those applications are migrated to the cloud, they are typically accessed across the WAN from branch locations. To make sure the application performance is not affected, consistent, high-quality performance with maximum reliability and minimum latency is required on the WAN link, and this can be achieved with WAN optimization.

Cisco Wide Area Application Service (WAAS) and Akamai Connect technologies provide a complete WAN optimization and application acceleration solution for overcoming WAN performance issues that can have a direct negative impact on application performance, which consequently affects end-user productivity. They are transparent to the endpoints (clients and servers) as well as devices between the WAAS/Akamai Connect devices.

Cisco WAAS incorporates data redundancy elimination (DRE) technology to identify repeating data elements or patterns in network traffic. When repeating data patterns are detected, they are simply replaced by simple pointers or references so that the long data patterns do not have to repeated multiple times in their entirety. This drastically reduces the size of the packet as it crosses the WAN, and at the same time keeps the original payload between the communicating devices.

Cisco WAAS and Akamai Connect also provide a method of caching objects locally. Caching repeat content locally shrinks the path between two devices and can reduce latency on chatty applications. For example, suppose the latency between a branch desktop and the local object cache (WAAS/Akamai Connect) is 10ms, and it takes 100ms to retrieve a file from the cloud. Only the initial file transfer would take the 100ms delay, and subsequent requests for the same file are provided locally from the cache with a 5ms response time.

Application Performance with Quality of Service

Quality of Service (QoS) should also be implemented in conjunction with WAN optimization to improve application performance. Business-critical or latency-sensitive applications residing in the cloud should be classified as high priority and given enough WAN link queue bandwidth so they can continue to work optimally in case of WAN link congestion.

A detailed discussion of QoS is outside the scope of this book. Further information on QoS deployment can be found in the Cisco Press publication *End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks, Second Edition*, by Tim Szigeti, Christina Hattingh, Robert Barton, and Kenneth Briley.

Performance Routing

Performance Routing (PfR) is another feature that can be used to optimize application performance by providing dynamic selection of the best WAN link for applications based on a traffic class policy (delay, loss, and jitter measurements). It can also perform application-based load balancing across multiple WAN links, allowing for maximum utilization of bandwidth and high availability.

Application Performance Monitoring and Management

Application performance management (APM) is critical in dynamic cloud environments where components and services are continuously being added and removed. The purpose of APM is to predict, detect, and diagnose complex application performance problems to ensure optimal cloud computing services. Following are some of the key benefits developers can obtain from using APM tools or suites to improve the quality and performance of their software and to ensure an optimal end-user digital experience:

- Visualize application performance in real time.
- Capability to troubleshoot problems such as slow response times and application errors.
- Automatically discover application topology and how the different components in the application environment work together to fulfill key business transactions for its users.
- Measure end-to-end business transaction performance (that is, capability to measure end-to-end performance of paying with a chip card), along with the health

of individual application and infrastructure nodes.

- Receive alerts based on custom or built-in health rules, including rules against dynamic performance baselines for issues in the context of business transactions.
- Provides end-to-end visibility of web and mobile application performance.
- Helps developers troubleshoot problems such as slow web applications, mobile network requests, and IoT application errors.
- Automatically captures errors, crashes, network requests, page-loading details, and other metrics covering an entire user session.
- Identify front-end issues faster and understand how third-party APIs and content services affect the app and web performance.
- Get traceability from front-end to back-end application dependencies.

Gartner's 2018 "Magic Quadrant for Application Performance Monitoring Suites" publication positions New Relic, Dynatrace, and Cisco's AppDynamics as the three leading APM suites.

Application Performance with DNA Center

Another option to aid in application performance is to use Cisco's Digital Network Architecture (DNA) Center. Cisco's DNA Center is a controller and analytic platform that includes service assurance capabilities and has rich contextual visibility into the user-to-application experience; in other words, it includes historical, real-time, and predictive insights across users, devices, applications, and the network. It can also send notifications in case of any service-impacting network issues that require proactive attention and suggests remediation before end users notice them themselves.

Application Scalability with Cloud Bursting

Cloud bursting is a way to scale out a private cloud into a public cloud whenever there is an increase in demand that goes above a specified threshold. When the demand decreases, the public cloud resources are released. Another way to look at cloud bursting is as elasticity in a multicloud environment that allows scaling out and in between cloud providers.

A good use case for cloud bursting would be retail; for example, if a good volume of sales is expected during the holiday season, more resources from the private cloud

could be required during that time. If the private cloud does not have enough resources to cope with the increase in service demands, it can then burst into the public cloud, leveraging extra capacity as required, and when the resources are not needed, they can be released and the consumer can just pay for the resources that were used.

For this type of cloud, an orchestrator/management solution is required to know when to leverage the private cloud and when to use resources from the public cloud and how to merge and connect the control processes between the two. Security and compliance requirements, latency, load balancing, and platform compatibility should also be considered.

Application High Availability

Cloud providers offer *service level agreements* (SLAs) in which they indicate the monthly uptime percentage of availability/uptime for their services. For example, at the time of writing, AWS EC2, Google Cloud Platform, and Microsoft Azure all offer 99.99% availability/uptime, and if they do not meet the agreement, they all offer financial credit that varies, depending on the severity of the downtime and other stipulations in their SLAs. In other words, 99.99% availability/uptime essentially implies that AWS, Google, and Microsoft Azure can go down every day for 8.6 seconds (or no more than once a year for 52 minutes and 37.7 seconds). If the downtime goes beyond this, then the cloud consumer is eligible for financial credit.

If an application requires more than 99.99% availability (for example, 99.999%), then a multicloud solution is what's needed. Applications can then be hosted and replicated to multiple providers to greatly reduce the chances of failure.

SECURITY IMPLICATIONS, COMPLIANCE, AND POLICY

Cloud security is more than just securing the cloud; it is also about the new security implications that come with the shift in how the networks and endpoints access data and applications. Organizations accessing data and applications in the cloud have shared risks and responsibilities with the CSPs that vary depending on which type of cloud service model is in use.

The cloud service models (IaaS, PaaS, and SaaS) can be thought of as stacks, where IaaS is at the bottom of the stack, with PaaS in the middle and SaaS at the top. In this stack, the security responsibilities are shared between the cloud customer and the CSP, where at the bottom of the stack the consumer has the most responsibility and, as you move further up the stack, the CSP has the most responsibility. In other words, security responsibilities in the cloud map to the degree of control that the cloud customer or

CSP has over the service model; this is known as the shared responsibility model.

Figure 2-7 shows an example of the shared responsibility model where on the left side, the customer is mostly responsible, and higher up the stack, the CSP is mostly responsible.

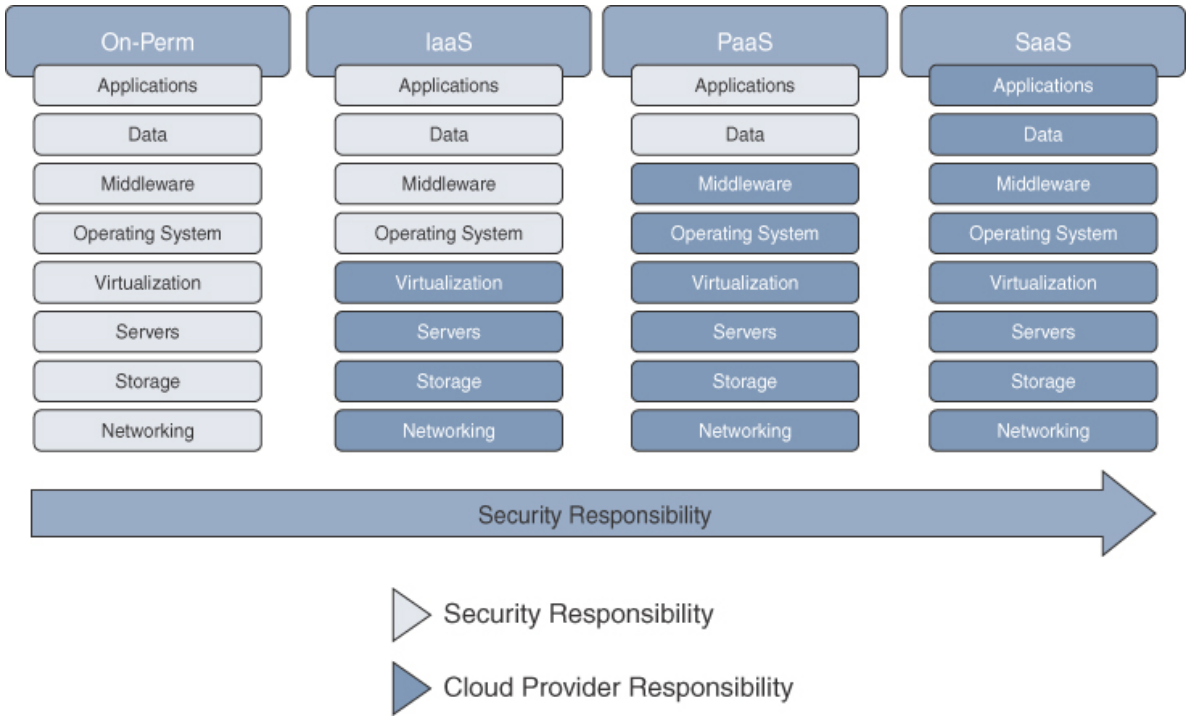


Figure 2-7 *Standard Responsibility Model*

Cloud Security Alliance's "Security Guidance for Critical Areas of Focus in Cloud Computing v4.0" includes two recommendations that directly correlate with the shared responsibility model:

- CSPs should properly design, implement, and document their security controls and customer features.
- Cloud consumers should develop a responsibilities matrix to document how and who is implementing which security controls as well as any needed compliance standards.

The Cloud Security Alliance (CSA) provides two tools to help meet the recommended requirements:

- The Consensus Assessments Initiative Questionnaire (CAIQ)
- The Cloud Controls Matrix (CCM)

Industry Regulatory Compliance Guidance

With cloud computing environments, data can reside anywhere and services are delivered on demand to any endpoint. Public clouds allow organizations to reduce their IT infrastructure costs, as well as management costs, by storing data assets in a multitenant, but secured, cloud-hosted environment. Organizations can also build their own private clouds that can deliver cloud-based services to their own organization.

Regulatory compliance is an organization's adherence to regulations, laws, guidelines, and specifications that are relevant to their business. Violations of compliance regulations often result in regulatory actions, such as federal fines or litigation.

There are different regulatory compliance laws for different verticals, such as the following:

- **Payment Card Industry Data Security Standard (PCI DSS)**: This is for companies that handle credit card information, and it serves to protect customer data in an attempt to reduce credit card fraud.
- **Federal Risk and Authorization Management Program (FedRAMP) and the Federal Information Security Management Act (FISMA) / NIST 800-53**: These are for government agencies and their service providers. They assist in assessing and meeting FISMA requirements to attract government agency business moving to the cloud as part of FedRAMP.
- **Health Insurance Portability and Accountability Act (HIPAA) and Health Information Technology for Economic and Clinical Health (HITECH)**: These are for the healthcare segment, bringing multilocation medical centers and healthcare organizations into compliance.
- **International Organization for Standardization (ISO) 27001 (2013):** Provides requirements for establishing, implementing, maintaining, and continuously improving an information security management system (ISMS).
- **ISO 27018 (2014)**: Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors.
- **ISO 27017 (2015):** Code of practice for information security controls based on ISO / International Electrotechnical Commission (IEC) 27002 for cloud services.
- **Service Organization Controls (SOC), SOC1 / SOC2:** SOC 1 is the reporting option for which the Statement on Standards for Attestation Engagements (SSAE)

16 professional standard is used, resulting in a SOC 1 SSAE 16 Type 1 and/or a SOC 1 SSAE 16 Type 2 report. SOC 2 is the reporting option specifically designed for many of today's cloud computing services, SaaS, and technology-related service organizations.

Independent parties need to perform periodic audits for most of these standards to validate an organization's continuous compliance. Cloud environments and their security risks are taken into consideration in such audits.

Top Cloud Threats

The CSA Top Threats working group conducted a survey to compile industry expert professional opinions on the greatest security issues within cloud computing. With the survey results and their expertise, they crafted a report titled "The Treacherous 12—Top Threats to Cloud Computing in 2016." Table 2-1 displays the 12 top threats included in the report. They are arranged based on order of severity, as per the survey results, where 1 is the highest severity.

Table 2-1 *Cloud Computing Top Threats*

Sev.	Critical Issues	Description	Security Measures
1	Data breaches	A data breach is an incident in which protected, sensitive, or confidential information such as health records and financial records are released, viewed, stolen, or used by someone who is not authorized to do so.	Multifactor authentication. Data encryption. Implement a data loss prevention (DLP) solution. Implement a cloud access security broker (CASB) solution.

2	Insufficient identity, credential, and access management	Data breaches typically occur because of a lack of proper identity access management systems and practices (for example, not using multifactor authentication or allowing users to set weak passwords).	Multifactor authentication. Strong passwords.
3	Insecure APIs	Cloud computing providers expose application programming interfaces (APIs) that developers use to manage and interact with cloud services, and they are typically a target of heavy attacks.	Use API keys instead of usernames and passwords and store the key in a safe place. Perform penetration testing.
4	System and application vulnerabilities	System vulnerabilities are exploitable weaknesses that are typically caused by bugs that attackers can use to infiltrate a computer system to steal data, take control of the system, or cause a denial of service.	Regular vulnerability scanning. Quickly taking action on reported system threats. Installation of security patches or upgrades.
5	Account hijacking	This type of attack is typically achieved by using attack methods such as phishing and exploitation of system vulnerabilities.	Prohibit sharing of credentials. Enforce strong two-factor

			authentication. Monitor accounts and activities. Install security patches or upgrades. Educate end users.
6	Malicious insiders	A malicious insider is an employee, contractor, partner, and so on, who has authorized access to internal confidential resources and then misuses those resources, negatively affecting the organization.	Educate end users. Role-based access control (RBAC). Logging, monitoring, and auditing of administrator's activities.
7	Advanced persistent threats (APTs)	APTs (aka malware) are a parasitical form of cyberattack that infiltrates the computing systems of target companies from which they smuggle data and intellectual property.	Be up-to-date on latest advanced cybersecurity attacks. Educate users to recognize phishing and spear phishing, which are commonly used to introduce

			APTs. Use antimalware controls.
8	Data loss	An attacker gaining access to cloud resources could cause a massive permanent loss of data, which can put a company out of business (for example, by encrypting it with ransomware or deleting it).	Implement a data loss prevention (DLP) solution. Implement mobile device management (MDM) techniques. Employ RBAC. Implement encrypted backups.
9	Insufficient due diligence	This is when an organization opens itself up for a large number of threats by adopting a cloud service from a CSP directly or via a merger without evaluating the CSP properly.	Adhere to the guidelines described in CSA's "Security Guidance for Critical Areas of Focus in Cloud Computing." Use CSA's CCM and CAIQ tools.
10	Abuse and nefarious use of cloud	This includes using cloud services to launch DDoS attacks, email spam and phishing, digital	A CSP must have an incident response

	services	currency mining, and hosting of malicious or pirated content.	framework to address misuse of resources.
11	Denial of service	This is an attack that consumes an inordinate amount of cloud resources or targets a specific vulnerability to make the cloud service slow or completely unavailable to other cloud users.	Employ distributed denial of service (DDoS) attack detection. Put in place a mitigation plan. Use multicloud.
12	Shared technology vulnerabilities	CSPs rely on multitenant environments. This can lead to shared technology vulnerabilities that can potentially be exploited in all delivery models.	Multifactor authentication. Intrusion prevention system (IPS).

Note

Keep in mind that these security issues are not unique to cloud computing and could happen in a traditional IT environment.

To make appropriate cloud security strategies, the CSA group recommends to use its threat research report in conjunction with the latest available versions of the following guides:

- CSA Security Guidance for Critical Areas in Cloud Computing
- CSA Security as a Service Implementation Guidance

- NIST Risk Management Framework

Cloud Security

Cloud services provide reduced cost and efficiency gains for businesses as long as security policies, processes, and best practices are taken into account. If not, businesses are vulnerable to security data breaches or other threats, as mentioned earlier in this chapter, which can eliminate any benefits gained from switching to cloud technology.

A cloud security architecture should extend protection to all aspects of security across the endpoint, network, and cloud ecosystem, as illustrated in Figure 2-8, to stop threats before, during, and after attacks.

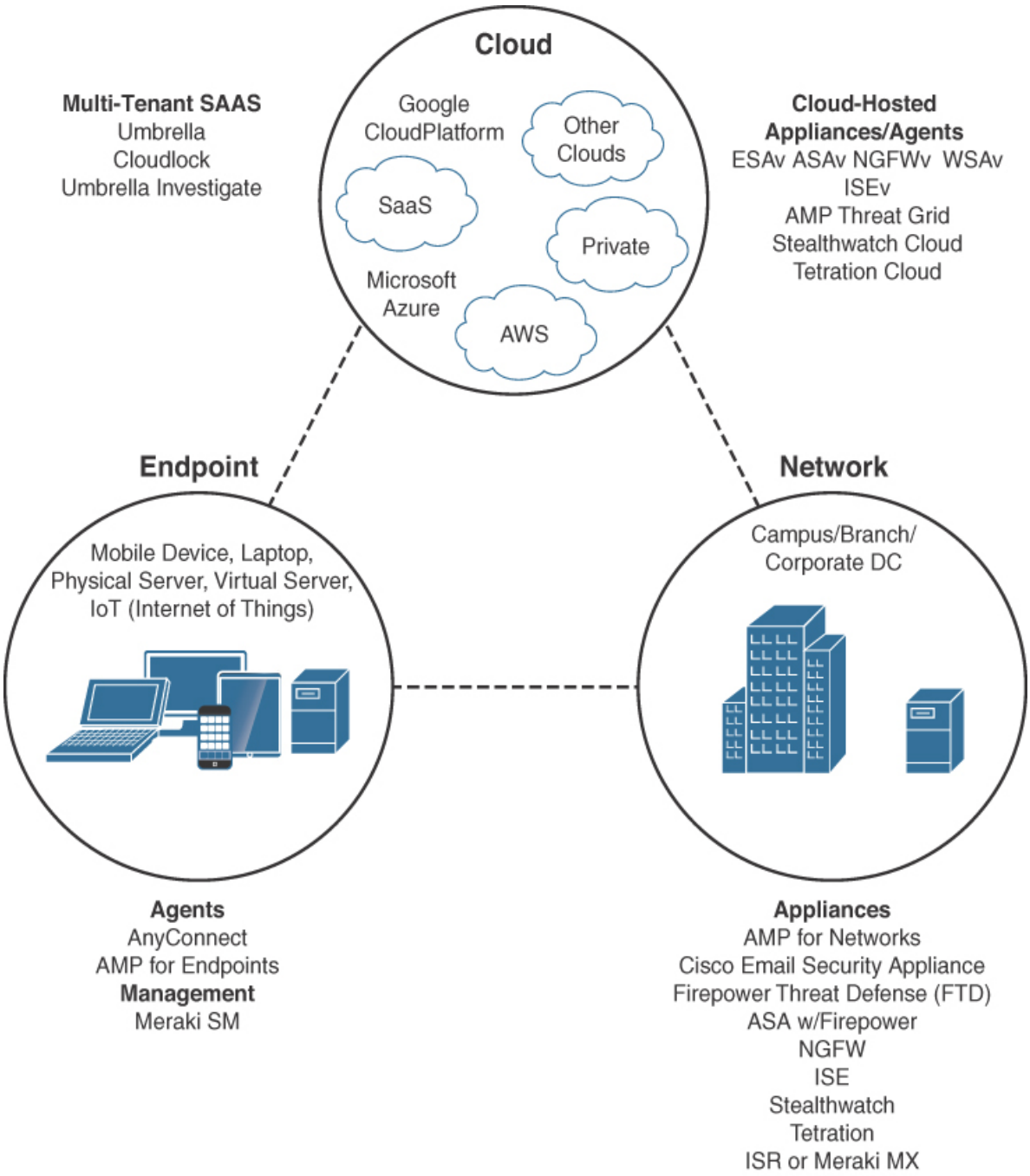


Figure 2-8 *Security Ecosystem*

Table 2-2 includes only the most important Cisco security products shown in Figure 2-8. It covers their capabilities, the form factor, as well as which part of the security ecosystem they play a security role in.

Table 2-2 *Cisco Security Products*

Product	Capability	Operates In	Form Factors
Advanced Malware Protection (AMP)	Antimalware, malware sandboxing	Endpoint, network, cloud	Endpoint Agent, NGIPS (AMP for Networks), Next-Generation Firewall, Meraki MX UTM platform, Branch router (ISR), cloud service
AnyConnect	Network Admission Control (NAC), virtual private network (VPN)	Endpoint	Endpoint agent
Cloudlock	User behavior analytics (UEBA), data security, DLP, cloud access security broker (CASB)	Cloud	Cloud service
Cisco Email Security Appliance (ESA)	Email security	Network, cloud	Physical or virtual appliance

Firepower (NGFW, NGFWv)	Firewall	Network	Physical or virtual appliance
Firepower Threat Defense (FTD)	IPS	Network	Physical or virtual appliance
Cisco Identity Services Engine (ISE)	NAC, context	Network	Physical or virtual appliance
Stealthwatch	Flow analytics	Network	Physical or virtual appliance
Stealthwatch Cloud	UEBA, flow analytics	Cloud	Cloud service
Tetration	Flow analytics	Network, cloud	Endpoint agent, cloud service
Umbrella	Domain name system (DNS) security, web security	Cloud	Cloud service
Cisco Web Security Appliance (WSA,	Web security	Network, cloud	Physical or virtual appliance

WSAv)

WORKLOAD MIGRATION

Migrating applications into the cloud takes planning. Organizations need to do a discovery and an analysis of the applications in their environment to determine which applications are suitable for migration and which are not; for example, many applications were not developed to be portable or to work in a virtualization environment, and can only work in very specific environments that cannot be re-created in the cloud. They may have specialized hardware requirements, use legacy databases, or lack proper security.

There are also performance-intensive applications that are better suited to run on bare-metal servers (for example, trading applications where fast transactions dictate revenue). Furthermore, security compliance and regulations may keep certain applications from running in public clouds.

Other considerations need to be taken into account to determine whether or not an application or workload is suitable for a cloud environment. For example, applications or workloads suitable for cloud may include the following:

- Applications that need significant compute resources (IoT, big data, analytics)
- Applications that require bursting (rapid scaling of resources)
- Collaboration applications (email, voice, messaging)
- DevTest and development applications
- Storage (file backups)

Examples of applications or workloads not suitable for cloud may include the following:

- Applications that use extremely sensitive data
- Applications that are sensitive to latency
- Applications that run on specialized hardware
- Applications that run on operating systems not supported by the cloud provider

- Applications running frequent large-volume transactions with on-prem databases
- Applications with too many dependencies on other applications that cannot be migrated

To migrate applications into the cloud, a migration methodology can be applied just as when migrating from an old infrastructure to a new one. The first step for the migration is to do an application discovery in the environment and categorize the applications in a matrix to determine whether or not they are suitable for migration. Here are some of the items to consider apart from the ones mentioned previously:

- Feature requirements
- Service model (IaaS, PaaS, or SaaS)
- Cloud deployment (public, private, community, or hybrid)
- Performance sensitivity
- Availability requirements
- Application migration priority
- Application complexity
- Type of application (development, test, and so on)
- Security considerations (using CSA's security tools)
- Regulatory compliance
- Dependencies to applications that cannot be migrated
- Business impact (critical or not)
- Benefit of migrating
- Networking requirements
- Hardware dependencies
- Software dependencies

- Migration complexity
- Applications that can be used as pilots

After identifying the applications that are suitable for migration, the organization should identify a migration strategy for each and add them to the matrix as well. Amazon AWS has identified six different migration strategies (called the 6 R's) that they see customers implementing when migrating applications to the cloud:

- **Rehosting (aka lift-and-shift)**: This involves redeploying applications to an IaaS or *Bare Metal as a Service* (BMaaS) environment without making any changes to the applications. This type of redeployment allows for migrating away from outdated physical hardware into an IaaS environment, or it could also allow non-cloud applications to be migrated into a BMaaS environment.
- **Replatforming (aka lift-tinker-and-shift)**: This is when a few cloud optimizations are done to the application to make it "cloud ready" in order to take advantage of the characteristics from the cloud provider without changing the core architecture of the application.
- **Repurchasing:** This means discarding an existing application (or set of applications) and moving to a different product. For example, migrating a CRM to Salesforce.com or moving from Exchange to Gmail for Business.
- **Refactoring or re-architecting**: This involves discarding existing code for an existing application and re-architecting the application, typically using cloud-native features. Although rebuilding causes loss of familiarity with existing code and frameworks, the advantage of rebuilding an application is access to innovative features such as better portability and improved performance.
- **Retiring**: This is when an application is decommissioned because it was found to no longer be in use during the application analysis and discovery process.
- **Retaining**: This applies to applications that cannot or should not be migrated or need to be revisited.

> Note
>
> AWS's 6 R's migration strategies build upon the 5 R's migration strategies originally outlined by Gartner.

After the migration strategy is completed, here are the next steps to follow:

- Perform a pre-migration performance baseline of each application.
- Evaluate different cloud providers and their SLAs.
- Evaluate different automation and orchestration tools.
- Evaluate monitoring tools.
- Perform migration testing.

At this point, the application migrations can be executed and managed. It is recommended to start with easy and non-business-critical applications (low-hanging fruit) to get familiar with the process and document lessons learned.

After migrating an application, perform a post-migration performance baseline and compare it to the pre-migration performance baseline to make sure the application is working as expected; this can be easily accomplished with tools such as Cisco's AppDynamics.

As can be seen, migrating applications can be quite involved; fortunately, Cisco has a solution that can simplify this whole process called Cisco CloudCenter (CCC).

The CCC solution, along with AppDynamics iQuate and CloudEndure, can help seamlessly migrate applications to and monitor them in the cloud. CloudCenter is discussed in the "Automation and Orchestration Tools" section of this chapter.

COMPUTE VIRTUALIZATION

VMs and containers are discussed in this section, and for you to be able to clearly understand their differences, it is necessary to understand all the following basic server components:

- **Central processing unit (CPU)**: The CPU (aka the processor) is without a doubt the most important component of a computer. It is responsible for the majority of processing jobs and calculations.
- **Internal storage**: The most usual internal storage devices are hard drives, which are a permanent or persistent form of storage. This is where the operating system,

applications, and application data are typically stored.

- **Main memory**: Main memory, also known as *random access memory* (RAM), is a fast, volatile (temporary) form of storage that is constantly being accessed by the CPU. If the CPU had to constantly access the hard drive to retrieve every piece of data it needs, it would operate very slowly.

When an application runs, its most important components that need to be accessed by the CPU are loaded into RAM, and other pieces of the application are loaded as required. Once the application is running in RAM, any file that is opened with the application is also loaded into RAM. When you save the file, it is saved to the specified storage device (for example, the hard drive), and when you close the application, the application as well as the file are purged from RAM to make room for other applications to run.

- **Network interface card (NIC)**: This is commonly known as a network adapter, and the vast majority of NICs are Ethernet network adapters.
- **Input/output (I/O) devices**: These are peripheral devices such as the mouse, keyboard, monitor, network adapters (NICs), and so on.
- **Operating system (OS)**: An OS is considered the most fundamental piece of software in a computer system and can be defined as the software that controls all of the computer resources and provides common services for computer applications that run on it. Examples of OSs include the following:
    - Microsoft Windows
    - Linux
    - FreeBSD
    - Chrome OS
    - Apple iOS
    - Apple macOS
    - Android
    - Cisco IOS

- **Kernel**: The kernel ("core" in German) is a program that is the central (core) part of an OS. It directly manages the computer hardware components, such as RAM and CPU, and provides system services to applications that need to access any hardware component, including NICs and internal storage.

- Because it is the core of an OS, the kernel is executed in a protected area of the main memory (kernel space) to prevent other processes from affecting it. Therefore, non-kernel processes are executed in a memory area called the user space. This is illustrated in Figure 2-9.

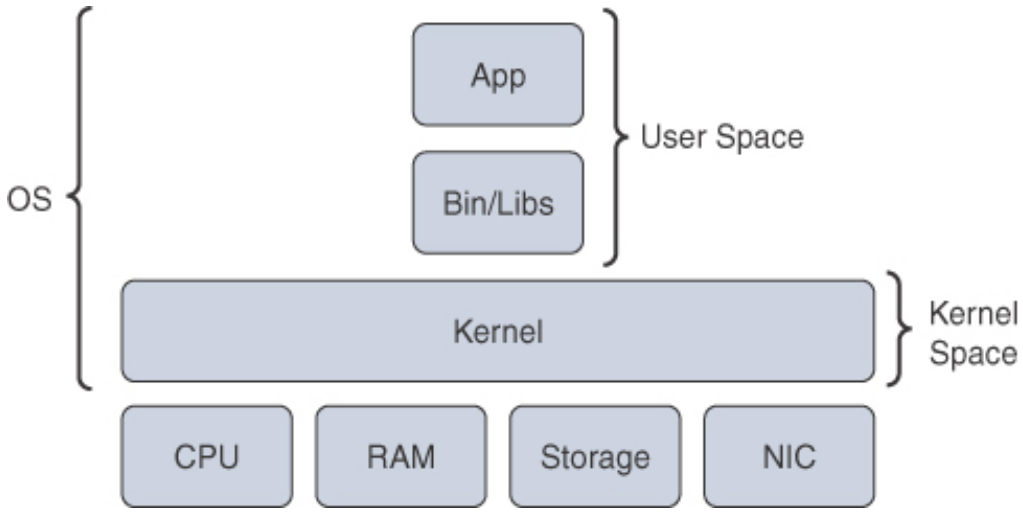


Figure 2-9 *Operating System Kernel*

There are two types of kernels: monolithic and microkernel. The difference between the two is that a monolithic kernel executes all its services and functions in the kernel space, which makes the kernel rigid and difficult to modify and enhance, whereas a microkernel only executes basic process communication and I/O control in the kernel space while the remaining system services such as the file system, device drivers, and so on, are executed in the user space. This makes microkernel OSs more flexible and allows for easy enhancements and modifications, which makes them ideal for cloud computing. Recent versions of Windows and macOS are examples of microkernel OSs.

Virtual Machines

As mentioned earlier in the "Cloud Fundamentals" section of this chapter, virtualization allows a physical server to run many independent operating systems and applications known as virtual machines (VMs). One of the main drivers behind virtualization was that server hardware resources were being underutilized, where servers were typically running a single application using only about 10 to 25% of the CPU utilization. VMs increase the overall efficiency and cost-effectiveness of a server by maximizing the use of the available resources.

Figure 2-10 illustrates a side-by-side comparison of a bare-metal server and a server running virtualization software. The virtualization software performing the hardware abstraction that allows multiple operating systems to run concurrently is known as a hypervisor. VMware vSphere, Microsoft Hyper-V, Citrix XenServer, and Red Hat KVM are the most popular hypervisors in the server virtualization market.

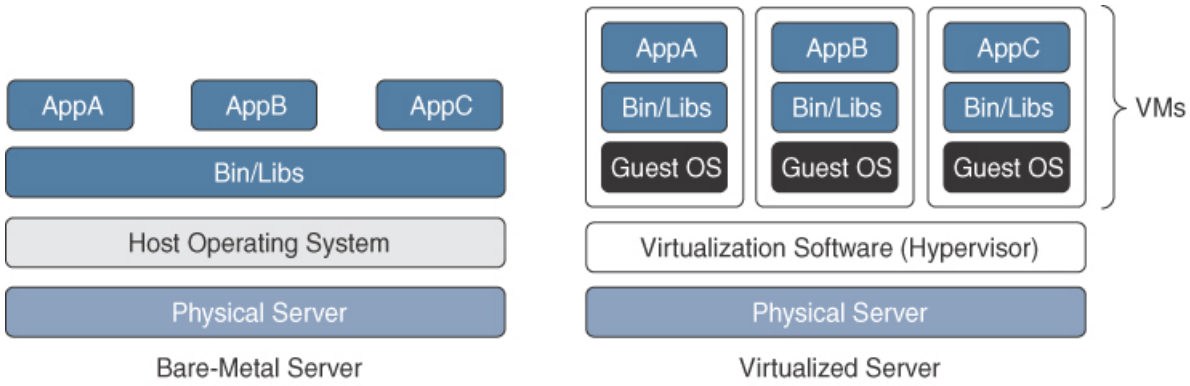


Figure 2-10 *Bare-Metal Server vs. Virtualized Server*

There are two types of hypervisors, as illustrated in Figure 2-11:

- **Type 1**: This type of hypervisor runs directly on the system hardware. It is commonly referred to as "bare metal" or "native."
- **Type 2**: This type of hypervisor requires a host OS to run. This is the type of hypervisor that is typically used by client devices (for example, VMware Fusion).

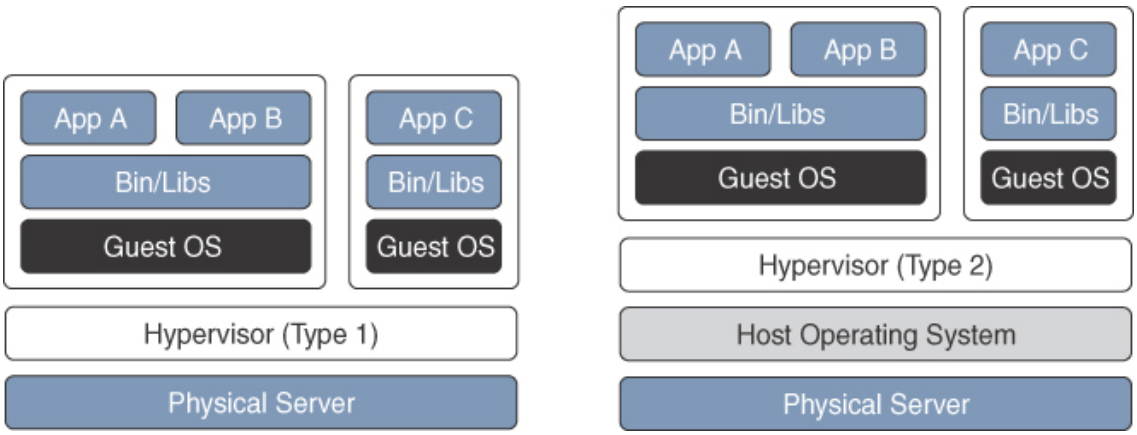


Figure 2-11 *Type 1 vs. Type 2 Hypervisors*

One key capability of VMs is that they can be migrated from one server to another while preserving transactional integrity during movement. This can enable many advantages; for example, if a physical server needs to be upgraded (for example, by adding more memory), the VMs can be migrated to other servers with no downtime. Another advantage is that it provides high availability; for example, if a server fails, the VMs can be spun up on other servers in the network, as illustrated in Figure 2-12.

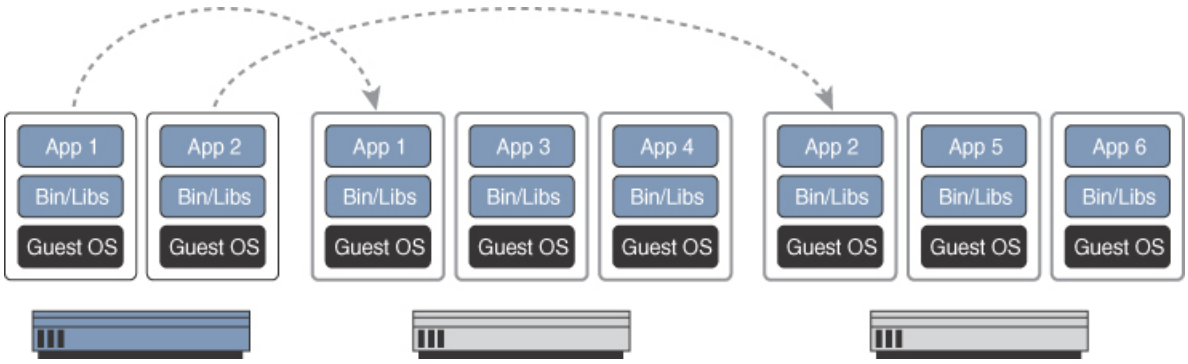


Figure 2-12 *VM Migration*

Containers

A container, much like a VM, provides an isolated environment to run applications, and it can be moved between servers (they are portable). Even though they have these and many other similarities to VMs, containers are not the same as VMs and they should not be referred to as "lightweight VMs."

Figure 2-13 shows a side-by-side comparison of VMs and containers. Notice how each VM requires an OS and that containers all share the same OS while remaining isolated from each other.

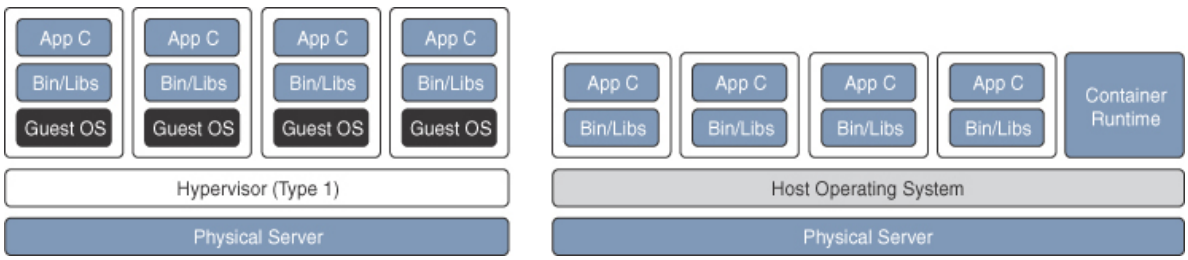


Figure 2-13 *Side-by-Side Comparison of VMs and Containers*

A VM includes a guest OS, which typically comes with a large number of components (including libraries and dependencies) that are really not required for the application to run, and it's up to the developer to strip any unwanted services or components from it to make it as lightweight as possible. Remember, a VM is just taking a physical server and virtualizing it, which means that it includes all of the components of a physical server, but in a virtual fashion.

Containers, on the other hand, share the underlying resources of the host operating system, and each application, along with the dependencies that it needs to run, is completely isolated, which makes the applications very lightweight (small size) and portable (easy to move/migrate). In other words, a container is typically just a tarball (that is, an archive file similar to a ZIP file) that packages the code, configuration, and dependencies of an application into a single file. This eliminates the typical problem of "It worked on my machine, why isn't working on yours?" or the typical issue when trying to install an application and the needed libraries to run it are not part of the

operating system and need to be downloaded (via apt-get, yum, and so on) to make it run.

A container does not try to virtualize a physical server like VMs do; instead, the abstraction is the application or the components that make up the application.

Here is one more example to help you understand the difference between VMs and containers: When a VM starts, the OS needs to load first, and once it's operational, the application in the VM can then start and run, which usually takes minutes. When a container starts, it leverages the kernel of the host OS, which is already running, and it typically takes a few seconds to start.

To develop, deploy, and run applications with containers, a container runtime is required. Many container runtimes are available, with the most popular being Docker. Here's a list of some of the other container runtime options available:

- rkt (pronounced "rocket")
- Open Container Initiative
- LXD (pronounced "lexdi"), from Canonical Ltd
- Linux-VServer
- Windows Containers

Cloud Native Applications and Services

As previously mentioned, one of the benefits of containers is that they make it easy for developers to know that their software will run, no matter where it is deployed, without having to worry about any dependencies their apps or components might have. Another benefit of containers is that they facilitate applications based on a microservices architecture. Instead of having one large, monolithic application, a microservices architecture breaks an application down into multiple, smaller components that can communicate with each other, and each of these components can be placed into a container. This facilitates a *continuous integration/continuous delivery* (CI/CD) approach, which increases feature velocity; in other words, different development teams can more easily work on different components of an application, and as long as they don't make any major changes to how those application components interact, they can work independently of each other. This allows for delivering small batches of software into production continuously.

Cloud-native applications and services are specifically built for the cloud to leverage all the advantages of cloud computing (availability, elasticity, and so on), and they typically use containerized microservices. The reason for using containers and not VMs is that VMs are too heavy (since they emulate a full physical server with an OS) and are very slow to start, whereas containers are very lightweight and start fast.

Cloud-native applications or services are often developed using the 12 Factor App (https://12factor.net/) design methodology as a baseline. This methodology dictates that applications employ a number of best practices to ensure portability, flexibility, and resiliency. The 12 Factor App methodology was created by developers at Heroku and was presented for the first time by Adam Wiggins circa 2011.

VIRTUALIZATION FUNCTIONS

Network Functions Virtualization (NFV) is an architectural framework created by the European Telecommunications Standards Institute (ETSI) that defines standards to decouple network functions from proprietary hardware-based appliances and have them run in software on standard x86 servers. It also defines how to manage and orchestrate the network functions. A *network function* (NF) refers to the function performed by a physical appliance (for example, a firewall or a load balancer function).

Some of the benefits of NFV include benefits similar to server virtualization and cloud environments:

- Reduced operator capital expenditure (CAPEX) and operational expenditure (OPEX) through reduced equipment costs and efficiencies in space, power, and cooling
- Faster time to market (TTM) because VMs or containers are easier to deploy than hardware
- Improved return on investment (ROI) from new services
- Scale up/out and down/in capacity on demand (elasticity)
- Openness to the virtual appliance market and pure software networking vendors
- Opportunities to test and deploy new innovative services virtually and with lower risk

Figure 2-14 illustrates the ETSI NFV architectural framework.

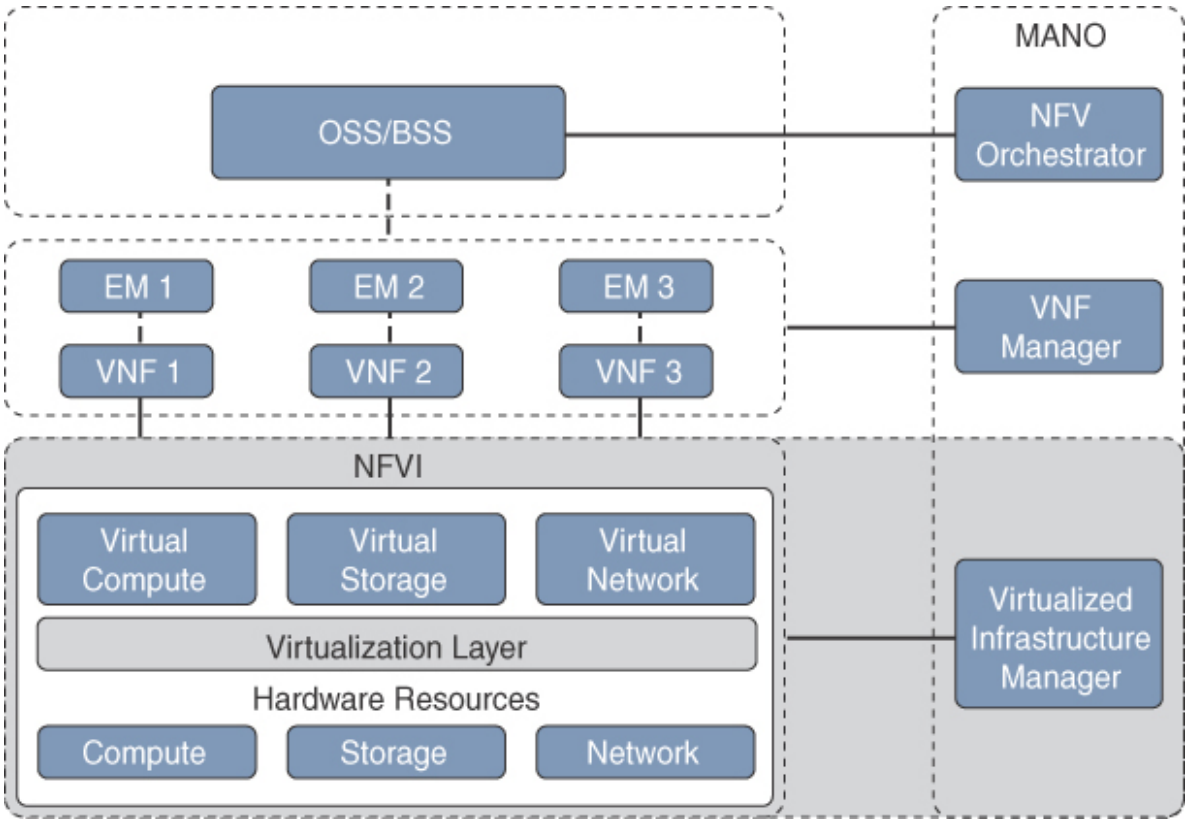


Figure 2-14 *ETSI NFV Architectural Framework*

NFV infrastructure (NFVI) is all the hardware and software components that comprise the platform environment in which virtual network functions are deployed. It is the section highlighted in green in Figure 2-14.

A VNF, as its name implies, is the virtual or software version of an NF, and it typically runs on a hypervisor as a VM. VNFs are commonly used for L4–L7 functions such as those provided by load balancers (LBs) and application delivery controllers (APCs), firewalls, intrusion detection systems (IDSs), WAN optimization appliances, and so on. However, they are not limited to L4–L7 functions; they can also perform lower-level L2–L3 functions, such as those provided by routers and switches.

Some examples of Cisco VNFs include the following:

- Cisco vEdge Cloud
- Cisco Cloud Services Router 1000V (CSR 1000V)
- Cisco Cloud Services Platform 2100 (CSP2100)
- Cisco IOS XRv 9000 (XRv9000)
- Cisco NextGen Firewall Virtual Appliance (NGFWv)
- Cisco Adaptive Security Virtual Appliance (ASAv)

- Cisco Web Security Virtual Appliance (WSAv)
- Cisco Email Security Virtual Appliance (ESAv)
- Cisco Advanced Malware Protection Private Cloud Virtual Appliance (AMPv)

The NFVI Virtualized Infrastructure Manager (VIM) is responsible for managing and controlling the NFVI hardware resources (compute, storage, and network) and the virtualized resources. It is also responsible for the collection of performance measurements and fault information. In addition, it performs lifecycle management (setup, maintenance, and teardown) of all NFVI resources as well as VNF service chaining.

Service chaining refers to chaining VNFs together to provide an NFV service or solution, as illustrated in Figure 2-15.

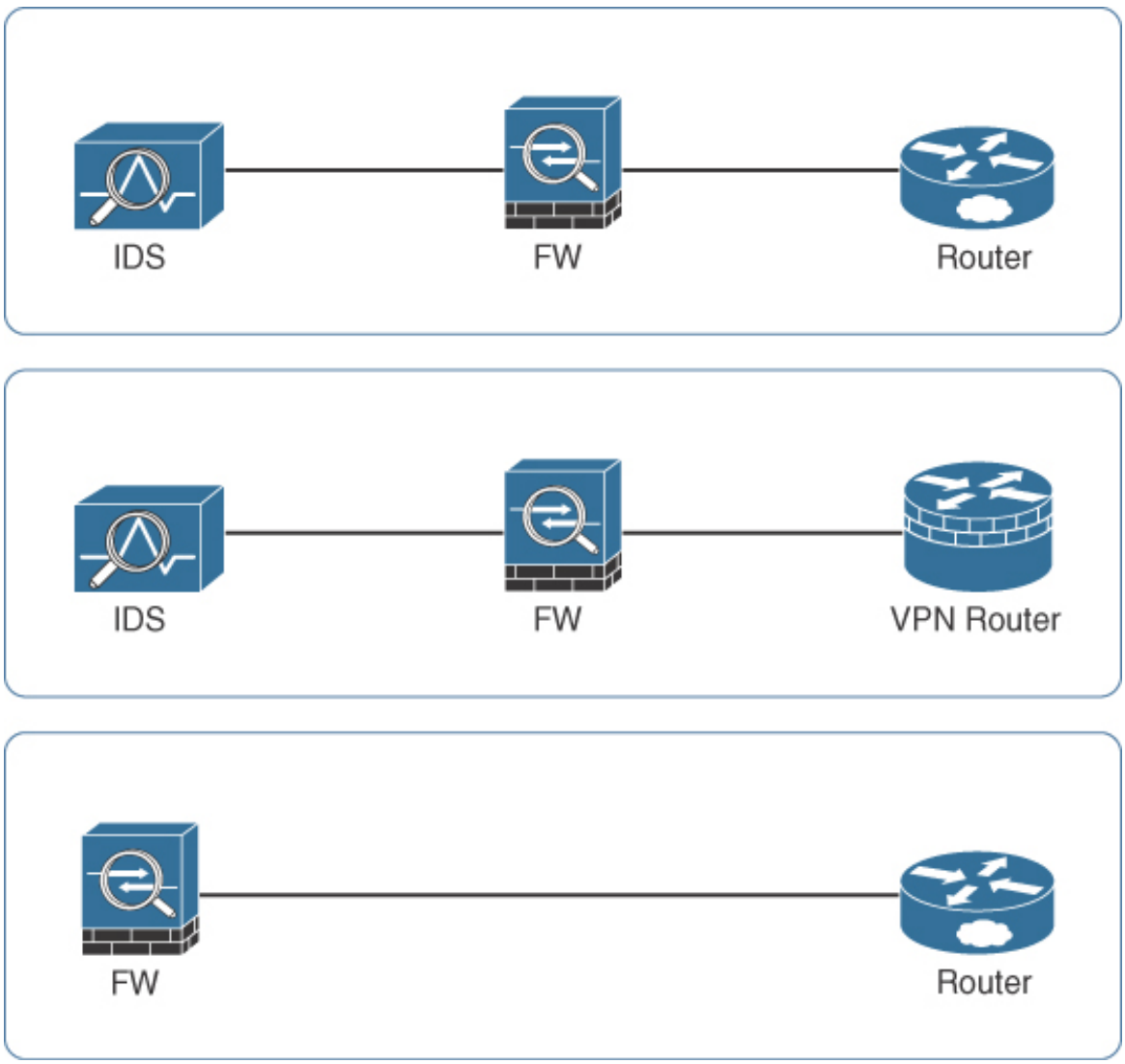


Figure 2-15 *Service Chaining*

The element managers (EMs), also known as element management systems (EMSs), are responsible for the functional management of VNFs; in other words, they perform

fault, *configuration*, *accounting*, *performance*, and *security* (FCAPS) functions for VNFs. A single EM can manage one or multiple VNFs, where an EM can also be a VNF.

The NFV Orchestrator is responsible for creating, maintaining, and tearing down VNF network services. If multiple VNFs are part of a network service, the NFV Orchestrator enables the creation of an end-to-end network service over multiple VNFs. The VNF Manager manages the lifecycle of one or multiple VNFs as well as FCAPS for the virtual components of a VNF. The NFV Orchestrator and VNF Manager together are known as NFV *Management and Orchestration* (MANO).

Operations support system (OSS) is a platform typically operated by service providers (SPs) to support all their network systems and services. The OSS can assist them in maintaining network inventory, provision new services, configure network devices, and resolve network issues. It commonly operates in tandem with the *business support system* (BSS) to improve the overall customer experience. BSS is a combination of product management, customer management, revenue management (billing), and order management systems that are used to run the SP's business operations.

Cisco's NFVI solution addresses the following key SP requirements:

- **Carrier-class performance**: In the same way physical devices provide carrier-class performance, the NFVI platform should provide carrier-class performance.
- **Use-case-agnostic infrastructure**: Capability to use NFVI as a platform on which to onboard multiple use cases.
- **Open standards based, modular, and elastic:** Being open and interoperable allows for onboarding any VNF from any third-party vendor in a modular and elastic manner.
- **Easy to use with unified management**: Single pane of glass to manage the lifecycle of the NFVI platform.
- **Integrated solution with single point of ownership**: Satisfies the demand for an NFVI platform from a single vendor. Having a single point of ownership makes it easier for the operation and support on a day-to-day basis.
- **Multilevel security**: Ensures all the components that comprise NFVI are secure and not easily susceptible to attacks (such as hacking, DDoS, and so on).

Figure 2-16 shows Cisco's NFV solution architecture that was designed to meet the

NFVI standards as well as the requirements just discussed. The NFVI components are shown in the green box, which should be looked at as a single integrated platform (composed of software and hardware) that can be used to deploy VNFs.

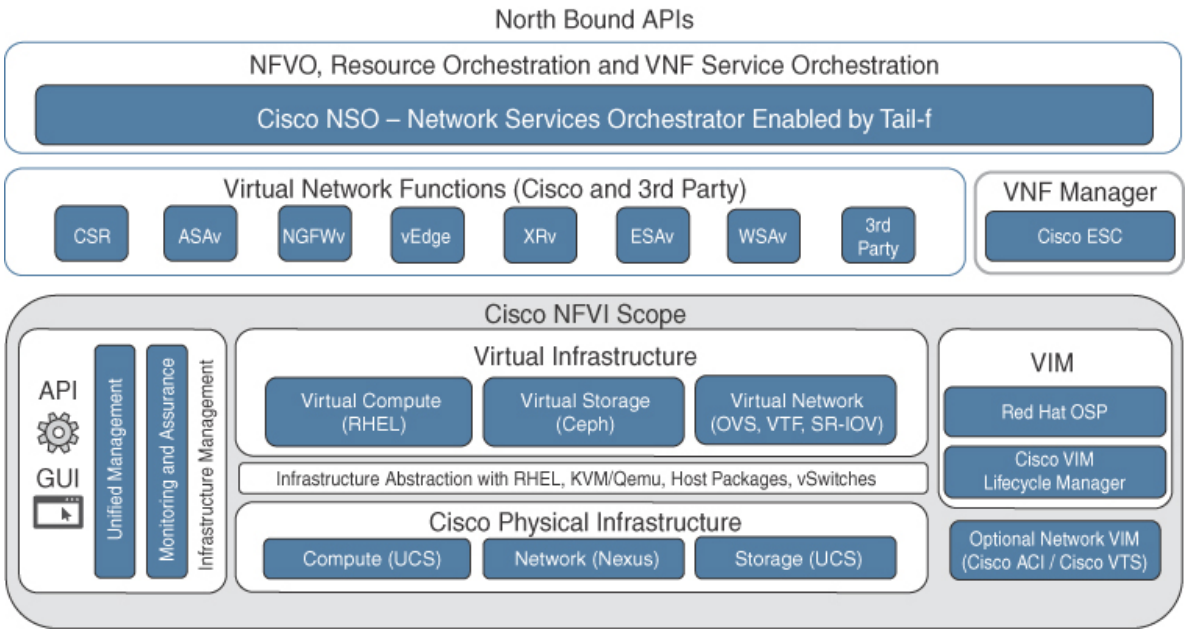


Figure 2-16 *Cisco NFVI Platform*

The solution has been tested to onboard simple and complex VNFs from multiple third-party vendors spanning different network services—for example, routing, firewalls, session border controllers, Virtualized Evolved Packet Core (vEPC), Virtualized Policy and Charging Rules Function (vPCRF), Virtualized IP Multimedia System (vIMS), and so on.

The components shown at the top of Figure 2-16 include Cisco's and third-party VNFs as well as Cisco's Network Services Orchestrator (NSO) as the NFV orchestration component and Cisco's Elastic Services Controller (ESC) as the VNF Manager. These components, although part of Cisco's NFV solution, are optional, and the customer may choose any third-party components that meet the NFV requirements.

There are currently three Cisco-integrated solutions available that rely on Cisco's NFV solution offering:

- **Cisco Managed Services Accelerator (MSX)**: Delivers *virtual customer premises equipment* (vCPE), cloud VPN, SD-WAN, Mobile Enterprise, Managed Security, and Collaboration cloud-based services

- **Cisco Ultra Services Platform**: With this solution, mobility services for machine-to-machine (M2M), IoT, machine-to-people (M2P), or people-to-people (P2P) applications can be turned up in days, for example, connected cars, homes and businesses. It lowers the total cost of ownership (TCO) by transitioning services

from a centralized to a distributed Mobile Virtual Network Operator (MVNO), and it can also be used to provide cloud-based services such as Mobility as a Service, Mobile Enterprise and Voice-over-LTE (Volte).

- **Cisco Virtualized Video Processing**: Orchestrates virtual video functions such as encoding, multiplexing, ad splicing, encrypting, transcoding, recording, packaging, playout, and delivery to provide the ability to operate a single production line for all video workflows.

> Note
>
> MSX was formerly Cisco *Virtual Managed Services* (VMS).

CLOUD CONNECTIVITY

Typically, an organization that decides to adopt a public cloud starts by connecting its on-prem networks to the cloud providers via an Internet VPN connection, simply because it is a great, simple, and affordable option to connect to them. If the organization has certain applications that require high throughput and low latency/jitter, then relying entirely on Internet VPN connections might not be enough. Fortunately, there are alternatives to connect to the "big three" CSPs (AWS, Microsoft Azure, and GCP), which are discussed in this section.

AWS

This service establishes a dedicated network connection between a customer's private network and the closest AWS geographic region with which the customer location is associated. A customer's WAN router connects to one of many AWS's DX locations, which are directly connected to AWS. This provides access to all of the services offered within that specific AWS geographic region. Using 802.1q virtual LANs (VLANs), the same dedicated connection can be used to access AWS's VPCs as well as all public AWS services in all other public regions (for example, AWS S3).

The DX connectivity options available are as follows:

- **Co-location**: The customer's network is co-located with an existing AWS DX location.

- **Partner**: The customer connects to an AWS DX partner who is a member of the

AWS Partner Network (APN).

- **Independent Service Provider**: The customer connects to an independent service provider to connect to AWS DX.

Speeds of 1Gbps to 10Gbps are available through the co-location option for a single link. Speeds of 50Mbps to 500Mbps can be ordered from any APN partner supporting AWS Direct Connect.

Microsoft Azure ExpressRoute (ER)

Microsoft's direct network connectivity option ExpressRoute (ER) allows for the extension of a private network to any of Microsoft's cloud services, including Microsoft Azure, Office 365, and Dynamics 365. A customer's WAN router connects to one of many peering locations and gains access to all regions within the geopolitical region. A premium add-on is available to extend connectivity across geopolitical regions around the world.

The ExpressRoute connectivity options available are as follows:

- **Co-location**: Co-location providers at a cloud exchange can offer either Layer 2 cross-connections or managed Layer 3 cross-connections between the customer's network and the Microsoft cloud.
- **Point-to-point Ethernet connections**: Point-to-point Ethernet providers can offer Layer 2 connections or managed Layer 3 connections between the customer's site and the Microsoft cloud.
- **Any-to-any (IPVPN) networks**: Customers can integrate their WAN with the Microsoft cloud via an MPLS L3VPN provider (IPVPN). The Microsoft cloud can be interconnected to the customer's WAN to make it look just like any other branch office.

Speeds supported range from 50Mbps to 10Gbps.

Google Cloud Dedicated Interconnect

Cloud Interconnect provides two options for extending the customer's private network into their GCP VPC networks.

The Google Cloud Dedicated Interconnect connectivity options available are as follows:

- **Dedicated Interconnect (co-location)**: Provides direct physical connections between the customer's private network and Google's network. The customer's network must physically meet Google's network in a Google-supported co-location facility.

- **Partner Interconnect**: Partner Interconnect provides Layer 2 or Layer 3 connectivity between the customer's private network and the VPC network through a supported service provider partner. A Partner Interconnect connection is useful when the customer's private network is in a physical location that can't reach a Dedicated Interconnect co-location facility or if the customer data needs don't warrant an entire 10Gbps connection.

The Dedicated Interconnect option supports speeds of 10Gbps per circuit with a maximum of eight circuits per Dedicated Interconnect connection. If less speed is desired, then the Partner Interconnect option offers speeds of 50Mbps and up to 10Gbps per VLAN attachment.

Region and Availability Zone Concepts

The "big three" CSPs (AWS, Microsoft Azure, and GCP) as well as other CSPs such as IBM have global infrastructures that are built around regions and zones. How each CSP defines regions and zones varies slightly, but in the end, they all serve the same purpose. In this section, AWS will be used as an example to define regions and zones.

AWS has multiple regions around the world. A *region* is an independent separate geographic area, and each region has multiple, isolated *availability zones* (AZs) that are connected to each other through low-latency links, as illustrated in Figure 2-17. Customers can deploy their applications and databases across multiple AZs within a region to make them highly available, fault tolerant, and scalable.

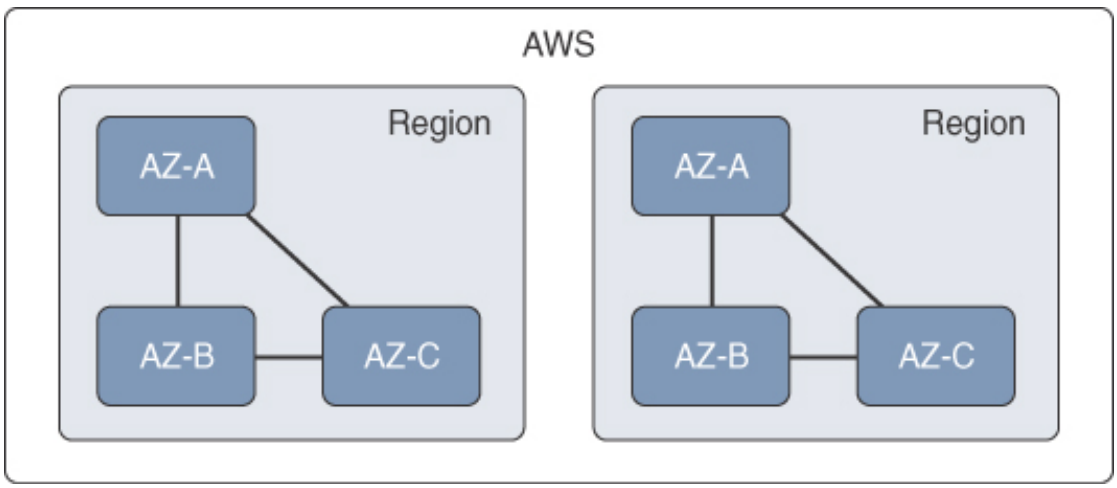


Figure 2-17 *Regions and Availability Zones*

In addition, to increase redundancy and fault tolerance even further, AWS allows for replicating applications and data across multiple regions. Resources aren't replicated across regions unless explicitly specified.

Multicloud Connectivity

Organizations all over the world are on a nonstop trend of using multiple CSPs, in part due to the different cloud services and functionalities each one of them offers. However, most organizations are not fully migrating their applications and data to the CSPs; instead, CSPs are becoming extensions to their on-prem or private-cloud environments where workloads and data are expected to move across their WAN to multiple CSPs, as well as between CSP VPCs/vNETs, while providing secure connectivity.

Using the basic networking services provided by CSPs such as DirectConnect, ExpressRoute, and so on, is not enough to provide the visibility and capabilities that are needed to achieve multicloud connectivity.

One of the ways to achieve multicloud connectivity is by using a virtual router such as the CSR 1000V router. The CSR 1000V ensures secure, scalable, and consistent connectivity for multicloud networking. It can run on VMware ESXi, Red Hat KVM, Citrix Xen, and Microsoft Hyper-V, as well as on Microsoft Azure and Amazon Web Services. Some of the security features it supports include IPSec VPNs and built-in zone-based firewalls. It also supports Cisco Digital Network Architecture (DNA) Encrypted Traffic Analytics (ETA), which has the ability to find threats in encrypted traffic.

Cisco has co-developed solutions using the CSR 1000V with cloud providers such as Transit VPC/VNET. The Transit VPC solution, illustrated in Figure 2-18, is a hub-and-spoke design built on AWS by deploying two CSRs in the transit VPC (the hub) for redundancy and using AWS VGWs (virtual private gateways) on the spoke VPCs, which host customer applications. The spoke VPCs join the transit VPC via automation, and this allows for spoke VPCs to be able to communicate with each other. The on-prem network can also be extended into the cloud securely using IPSec encryption through the Transit VPC. This is useful for enterprise customers using multiple VPCs for different departments or projects, such as Development VPC, Production VPC, and Test VPC, and they need connectivity between VPCs as well as to on-prem resources.

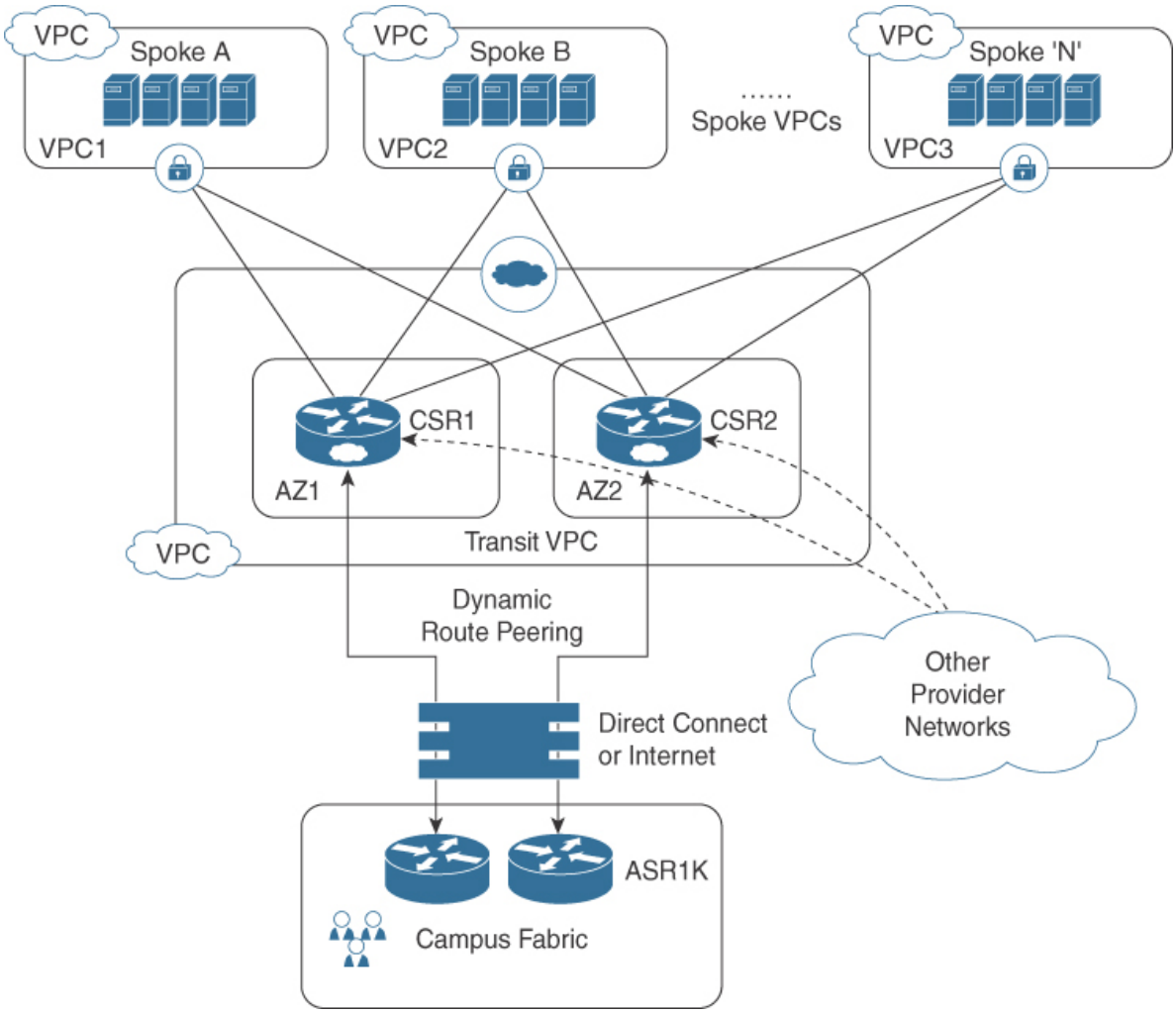


Figure 2-18 *Transit VPC*

Software-Defined Access (SD-Access) User-to-Cloud Access Control

SD-Access is the industry's first intent-based networking solution for the enterprise built on the principles of Cisco's Digital Network Architecture (DNA). It provides automated end-to-end segmentation to separate user, device, and application traffic without having to redesign the network. SD-Access is the combination of the following components:

- **DNA Center:** An enterprise SDN controller that provides a graphical user interface (GUI) that is the single pane of glass for management and provides all of the base and fabric automation services. It also provides analytics and assurance services with the integrated Cisco Network Data Platform (NDP) software.

- **Identity Service Engine (ISE):** Provides all of the identity and policy enforcement services.

- **Campus Fabric**: An evolved campus network that allows for host mobility without stretching VLANs, network segmentation without Multiprotocol Label Switching (MPLS), and RBAC without end-to-end support for TrustSec.

SD-Access can be used to extend TrustSec into AWS Transit VPC to control access to spoke VPCs. This can be achieved based on Security Group Tags (SGTs) and ISE policy enforcement on the Transit VPC Hub CSR 1000Vs, as illustrated in Figure 2-19. The table in the lower left of the figure shows the VPCs the end users would have access to according to their role.

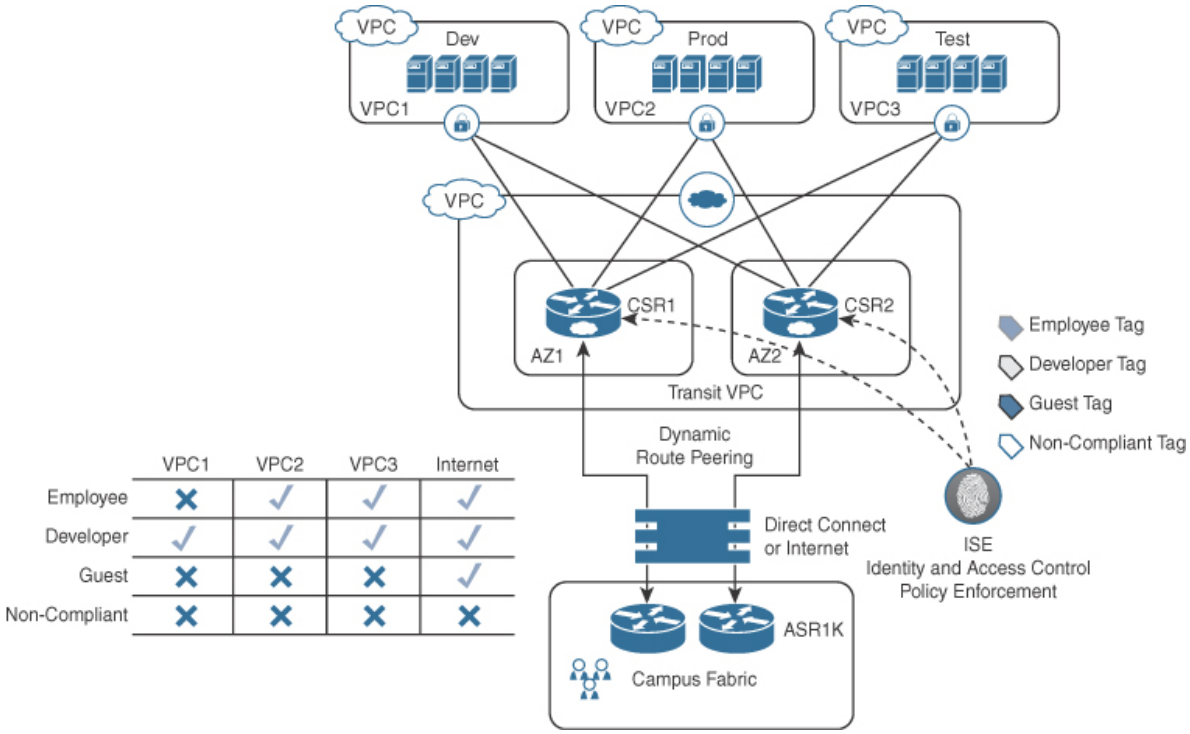


Figure 2-19 *Transit VPC User to Cloud Access Control*

This is all deployed from the DNA Center GUI, which works along with ISE to push the SGT Tag and Policy Enforcement configuration to the CSR 1000Vs. Extending TrustSec to the Transit VPC Hub has the following benefits:

- Control traffic between VPCs
- Simplify security configurations
- Scale security group control
- Single control point

Figure 2-20 illustrates another way to enforce user-to-cloud access control by extending TrustSec to a virtual firewall or SGACL-capable virtual router in the cloud environments, such as Cisco Firewpower Threat Defense Virtual (FTDv), ASAv, CSR 1000V, or Integrated Services Virtual Router (ISRv), to provide policy enforcement at the application level within a VPC rather than at the VPC level, as in the previous transit VPC case. The SGT policy is pushed by ISE to the virtual router or firewall, and it can then be mapped to the cloud provider's own Security Groups. The table in the

lower left of the figure shows the apps the end users would have access to according to their role.

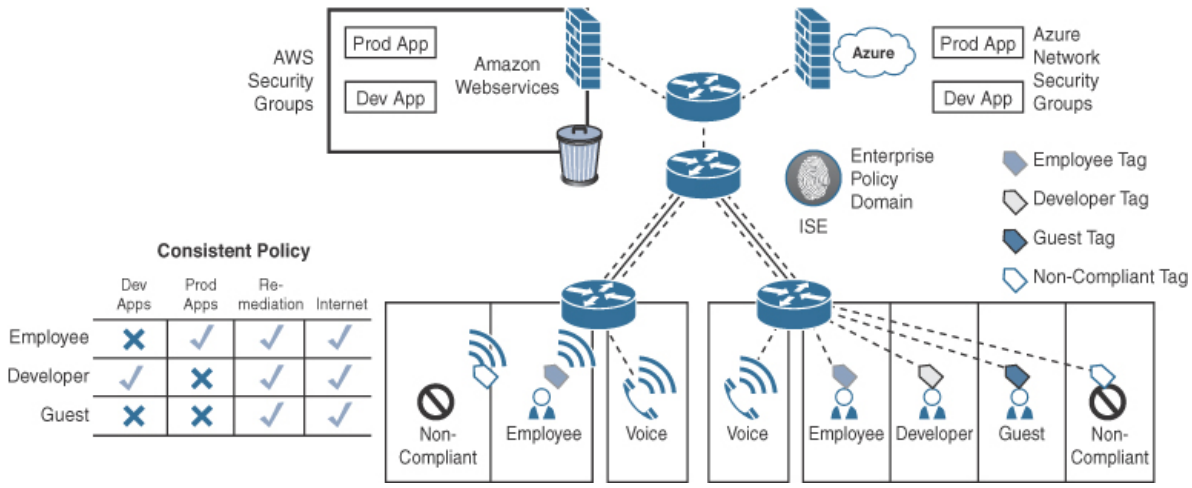


Figure 2-20 *User-to-Cloud Access Control*

Software-Defined WAN (SD-WAN)

Managing enterprise networks is becoming more complex, with customers embracing a multicloud approach, applications moving to the cloud, mobile and IoT devices growing exponentially in the network, and Internet edge moving to the branch. This digital transformation is powering the adoption of SD-WAN by customers looking to do the following:

- Lower costs and risks with simple WAN automation and orchestration.
- Extend their enterprise networks (such as branch or on-prem) seamlessly into the public cloud.
- Provide optimal user experience for SaaS applications.
- Be able to leverage a transport-independent WAN for lower cost and higher diversity.
- Enhance application visibility and use that visibility to improve performance with intelligent path control to meet SLAs for business-critical and real-time applications.
- Provide end-to-end WAN traffic segmentation and encryption for protecting critical enterprise compute resources.

Cisco currently offers three SD-WAN solutions:

- Cisco SD-WAN (Cisco + Viptela)

- Meraki SD-WAN
- Cisco Intelligent WAN (IWAN)

The following guidelines should be considered when deciding which SD-WAN solution is the right choice for the business:

- **Cisco SD-WAN (based on Viptela)**: This is the preferred solution for organizations that require an SD-WAN solution with cloud-based initiatives that provides granular segmentation, advanced routing, and complex topologies while connecting to cloud instances. This solution is based on Viptela's vEdge platform.
- **Meraki SD-WAN**: This is the recommended solution for organizations that require unified threat management (UTM) solutions with SD-WAN functionality or are existing Cisco Meraki customers looking to expand to SD-WAN. UTM is an all-in-one security solution delivered in a single appliance and typically includes the following security features: firewall, VPN, intrusion prevention, antivirus, antispam, and web content filtering.
- **Cisco IWAN**: Cisco's first SD-WAN solution on the ISR platform that provides advanced services such as voice, compute, WAN optimization, integrated threat control capabilities, switching, and so on.

> Note
>
> At the time of writing, Viptela functionality was actively being integrated into all IOS XE enterprise routing platforms, such as Integrated Services Router (ISR), CSR, ASR 1000 Series Router (ASR1K), CSR 1000V, and Enterprise Network Compute System 5K (ENCS 5K) platforms, along with the advanced services mentioned previously (voice, compute, and so on). This will bring Viptela/Cisco SD-WAN the advance service capabilities of IWAN through a simple software upgrade.

All three SD-WAN solutions work in a similar fashion, but in this chapter only Cisco SD-WAN based on Viptela will be covered.

Cisco SD-WAN

Cisco SD-WAN is a cloud-delivered overlay WAN architecture that facilitates digital and cloud transformation for enterprises, and it addresses all of the customer requirements mentioned earlier. Figure 2-21 illustrates Cisco's SD-WAN solution architecture.

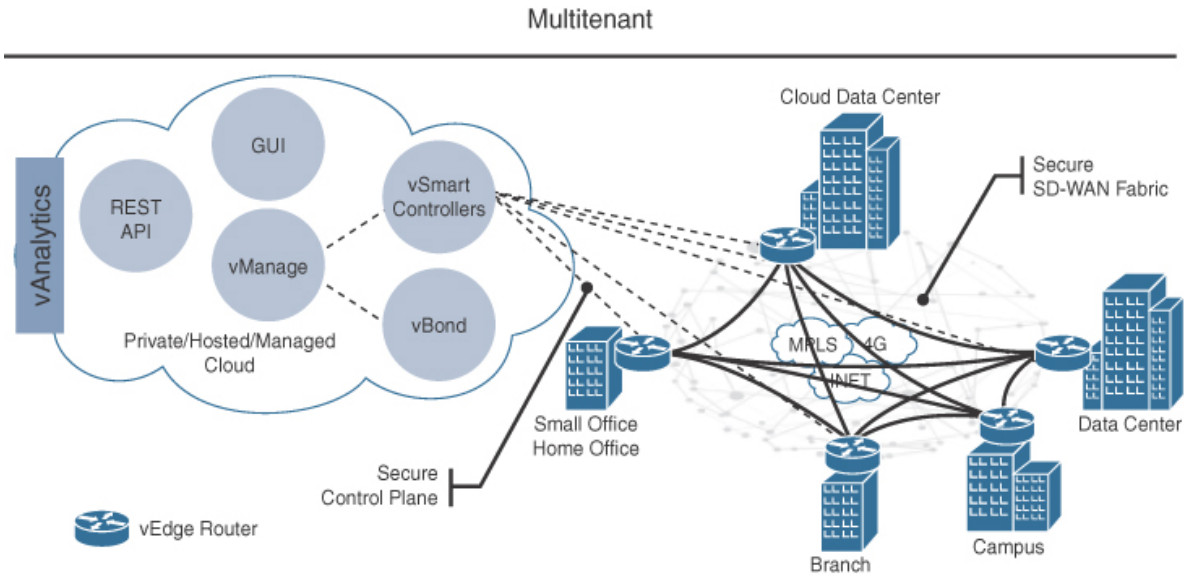


Figure 2-21 *Cisco SD-WAN Cloud-Delivered Architecture*

The Cisco SD-WAN solution is composed of four main components and an optional analytics service:

- **vManage**: This is a single pane of glass network management system (NMS) GUI used to configure and manage the full SD-WAN solution. It enables centralized provisioning and simplifies changes.

- **vSmart**: This is the brains of the solution. It establishes a secure Datagram Transport Layer Security (DTLS) connection to all vEdge routers in the SD-WAN fabric and runs a proprietary Overlay Management Protocol (OMP) to share routes, security, and policy information. It implements all of the control plane policies created on vManage, such as service chaining, traffic engineering, and segmentation per VPN topology.

  For example, when a policy is created on vManage for an application (such as YouTube) that requires no more than 1% of loss and 150ms of latency, that policy is downloaded to the vSmart controller. vSmart will take the policy and convert it into a format that all the vEdge routers in the fabric can understand, and it automatically implements the policy on all vEdge routers without the need to rely on a command-line interface (CLI). The vSmart controller also works in conjunction with the vBond orchestrator to authenticate the devices as they join the network and to orchestrate

connectivity between the vEdge routers

- **vEdge routers**: These are routers that support standard router features, such as OSPF, BGP, ACLs, QoS, and routing policies, in addition to the SD-WAN overlay control and data plane functions. Each vEdge router automatically establishes a secure DTLS connection with the vSmart controller and standard IPSec sessions with other vEdge routers in the fabric.

- **vBond orchestrator**: The vBond orchestrator authenticates the vSmart controllers and the vEdge routers orchestrate connectivity between them. It is the only device that must have a public IP address so that all Viptela devices in the network can connect to it.

- **vAnalytics**: This is an optional analytics and assurance service that has many advanced capabilities, including the following:

  - Visibility of applications and infrastructure across the WAN

  - Forecasting and "what if" analysis

  - Intelligent recommendations

These capabilities can bring many benefits that are not possible without vAnalytics; for example, if a branch office is experiencing latency or loss on its MPLS link, vAnalytics will detect this, and it will then compare that loss or latency with other organizations in the area that it is also monitoring to see if they are also having that same loss and latency in their circuits. If they are, they can then report this issue with confidence to their SPs. It can also help predict how much bandwidth is truly required for any location, and this is useful to decide if a circuit can be downgraded to a lower bandwidth, resulting in reduced costs.

Out of these components, the vEdge routers and the vBond orchestrator are available as physical appliances and VMs, whereas vManage and vSmart are only available as VMs.

All of the VMs, including the vEdge router, can be hosted on-prem using ESXi or KVM, or they can be hosted in AWS and Microsoft Azure.

> Note
>
> At the time of writing, vManage capabilities were being integrated into DNA

Center to bring full DNA Center capabilities to Cisco SD-WAN, such as assurance, analytics, and integrated workflows.

Cisco SD-WAN Cloud OnRamp

Traditional enterprise WAN architectures are not designed for the cloud. As organizations adopt more SaaS applications like Office 365 and public cloud infrastructures like AWS and Microsoft Azure, the current network infrastructure poses major problems related to the level of complexity and end-user experience.

Cisco's SD-WAN solution includes a set of functionalities addressing optimal cloud SaaS applications access and IaaS connectivity called Cloud OnRamp. Cloud OnRamp delivers the best application experience for SaaS applications by continuously monitoring SaaS performance across diverse paths and selecting the best-performing path based on performance metrics (jitter, loss, and delay), and it simplifies hybrid cloud and multicloud IaaS connectivity by extending the SD-WAN fabric to the public cloud while at the same time increasing high availability and scale.

Cloud OnRamp for SaaS

SaaS applications reside mainly on the Internet, and to be able to achieve optimal SaaS application performance, the best-performing Internet exit point needs to be selected.

Figure 2-22 illustrates a remote site with dual direct Internet access (DIA) circuits from two different Internet service providers (ISP1 and ISP2). When Cloud OnRamp for SaaS is configured for a SaaS application on vManage, the vEdge router at the remote site will start sending small HTTP probes to the SaaS application through both DIA circuits to measure latency and loss. Based on the results, the vEdge router will know which circuit is performing better and will send the SaaS application traffic out of that circuit (ISP2). The process of probing continues, and if a change in performance characteristics of the ISP2's DIA circuit occurs (for example, due to loss or latency), the remote site vEdge router will make an appropriate forwarding decision.

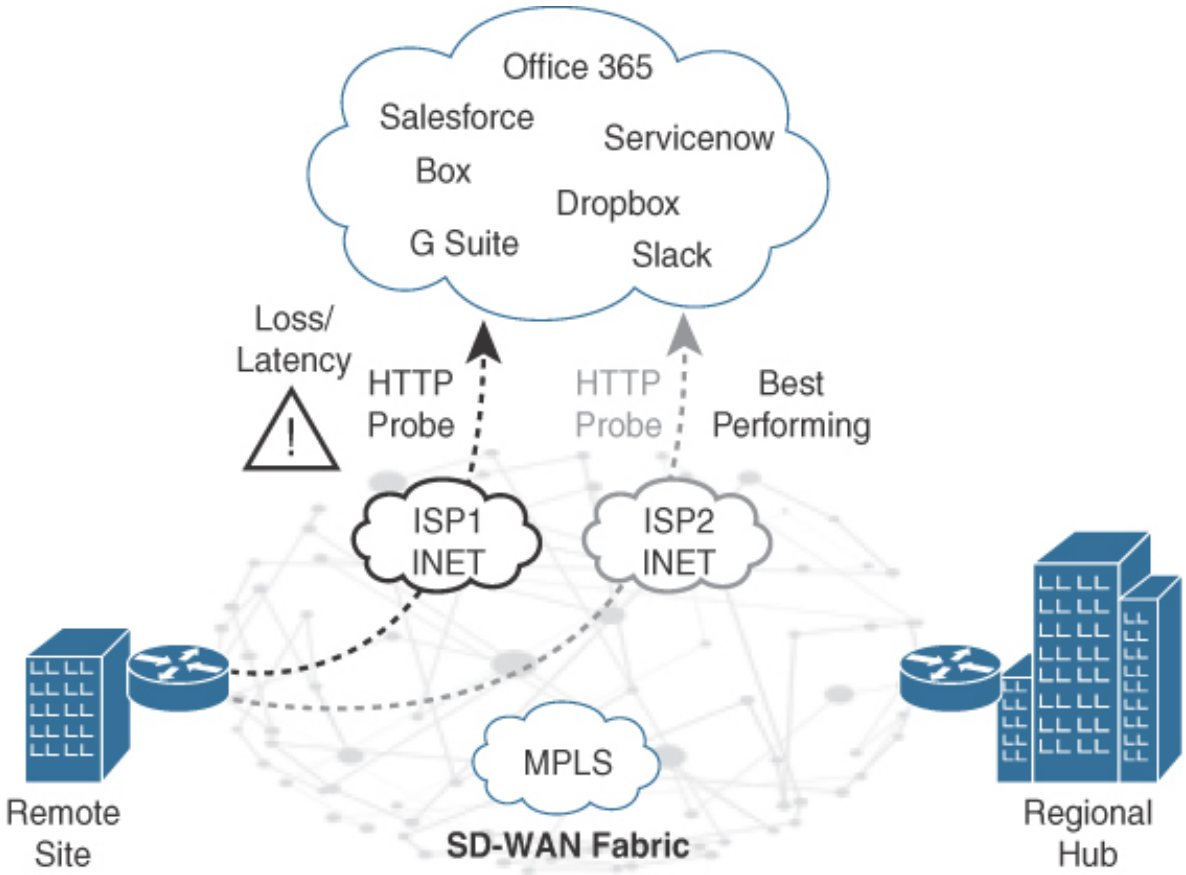


Figure 2-22 *Cloud OnRamp for SaaS with Dual DIA*

Figure 2-23 illustrates another example of Cloud OnRamp for SaaS. The remote site has a single DIA circuit to ISP1 and an SD-WAN fabric DTLS session to the regional hub.

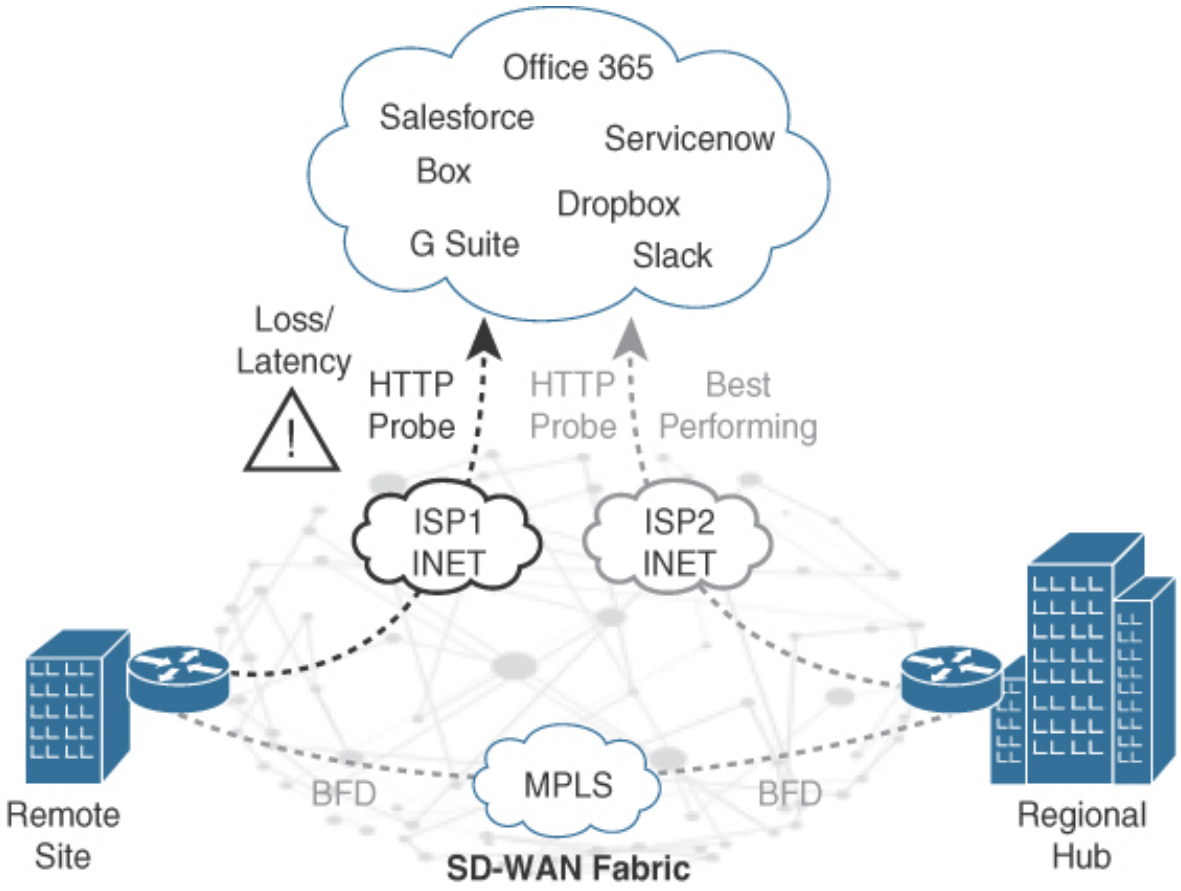


Figure 2-23 *Cloud OnRamp for SaaS DIA and Gateway*

Similar to the previous case, Cloud OnRamp for SaaS can be configured on the

vManage and become active on the remote site vEdge router. However, in this case, Cloud OnRamp for SaaS also gets enabled on the regional hub vEdge router and is designated as the gateway node. Quality probing service via HTTP toward the cloud SaaS application of interest starts on both the remote site vEdge and the regional hub vEdge.

Unlike the HTTP probe sent towards the SaaS application via the DIA link, Bidirectional Forwarding Detection (BFD) runs through the DTLS session between the remote site and the regional hub. BFD is a detection protocol originally designed to provide fast forwarding path failure detection times between two adjacent routers. For SD-WAN, it is leveraged to detect path liveliness (up/down) and measure quality (loss/latency/jitter and IPSec tunnel MTU).

For SaaS over DIA, BFD is not used because there is no vEdge router on the SaaS side to form a BFD session with. The Regional hub vEdge router reports its HTTP connection loss and latency characteristics to the remote site vEdge router via an Overlay Management Protocol (OMP) message exchange through the vSmart controllers. At this time, the remote site vEdge router can evaluate the performance characteristics of its local DIA circuit versus the performance characteristics as reported by the regional hub vEdge. It also takes into consideration the loss and latency incurred by traversing the SD-WAN fabric between the remote site and the hub site (calculated via BFD) and then makes an appropriate forwarding decision sending application traffic down the best performing path toward the cloud SaaS application of choice.

The quality of cloud SaaS application connection is quantified as a Viptela Quality of Experience (vQoE) score on a scale of 0 to 10, with 0 being the worst quality and 10 being the best. vQoE can be observed in the vManage GUI.

Cloud OnRamp for IaaS

Multicloud is now the new norm for enterprises. With multicloud, certain enterprise workloads remain within the boundaries of the private data centers, while others are hosted in the public cloud environments, such as Amazon Web Services (AWS) and Microsoft Azure. This approach provides enterprises the greatest flexibility in consuming compute infrastructure, as required.

With the Cisco Software-Defined WAN (SD-WAN) solution, you can extend ubiquitous connectivity, zero-trust security, end-to-end segmentation, and application-aware Quality of Service (QoS) policies of the organizational WAN into the IaaS environments, as illustrated in Figure 2-24. The transport-agnostic capability of the

Cisco SD-WAN solution allows the use of a variety of connectivity methods by securely extending the SD-WAN fabric into the public cloud environment across all underlying transport networks. These include the Internet, MPLS, 3G/4G LTE, satellite, and dedicated circuits such as AWS's DX and Microsoft Azure's ER.

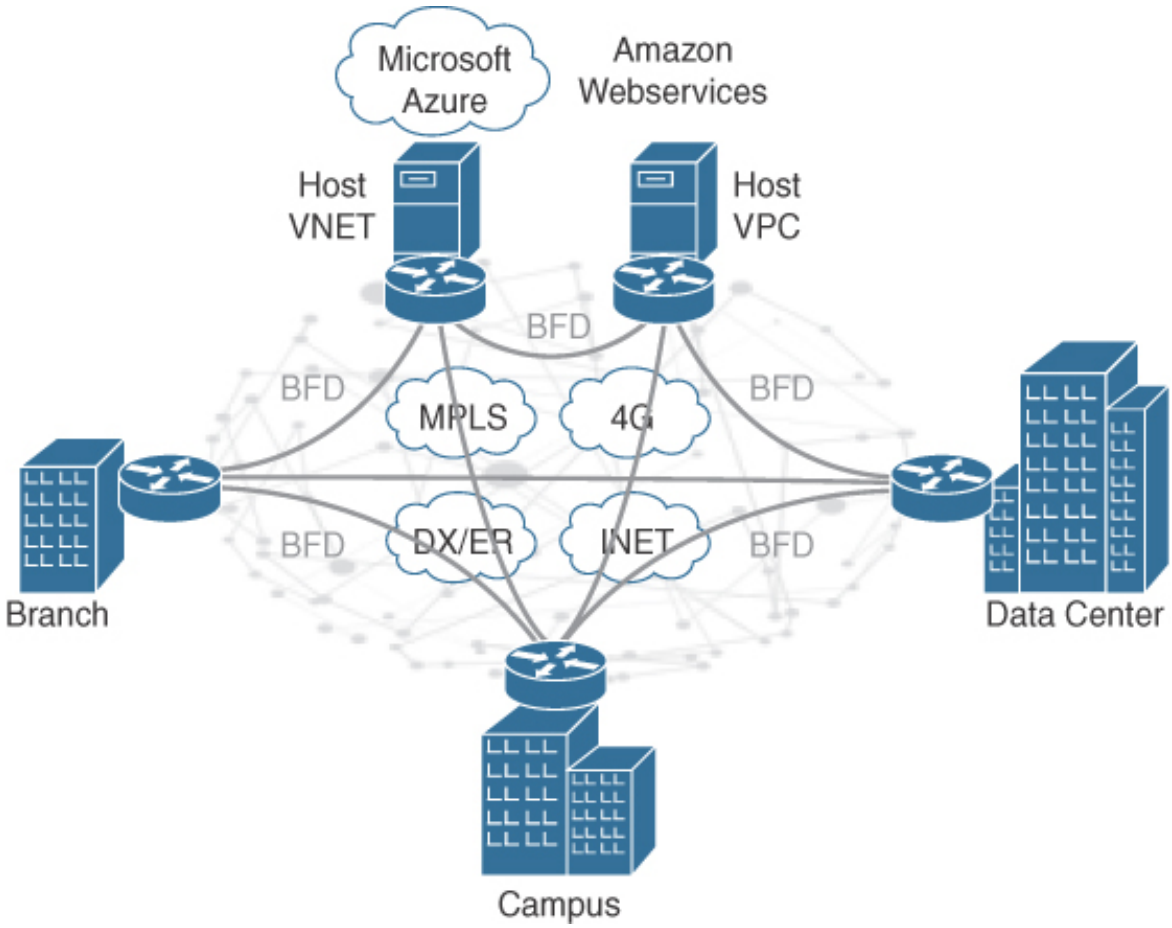


Figure 2-24 *Cloud OnRamp for IaaS*

Virtual Switching

A virtual switch (vSwitch) operates like a physical Layer 2 Ethernet switch, and it enables VMs to communicate with each other within a virtualized server and with external physical networks via the physical network interface cards (pNICs). Multiple vSwitches can be created under a virtualized server, but network traffic cannot flow directly from one vSwitch to another vSwitch within the same host, and they cannot share the same pNIC.

Figure 2-25 illustrates a virtualized server with three vSwitches connected to the virtual network interface cards (vNICs) of the VMs as well as the pNICs. vSwitch1 and vSwitch3 are uplinked to pNIC 1 and pNIC 3, respectively, to access the physical network, whereas vSwitch2 is not uplinked to any pNICs. Since network traffic cannot flow from one vSwitch to another, network traffic from VM1 destined to the external network, or VM0, will need to flow through the firewall (NGFWv).

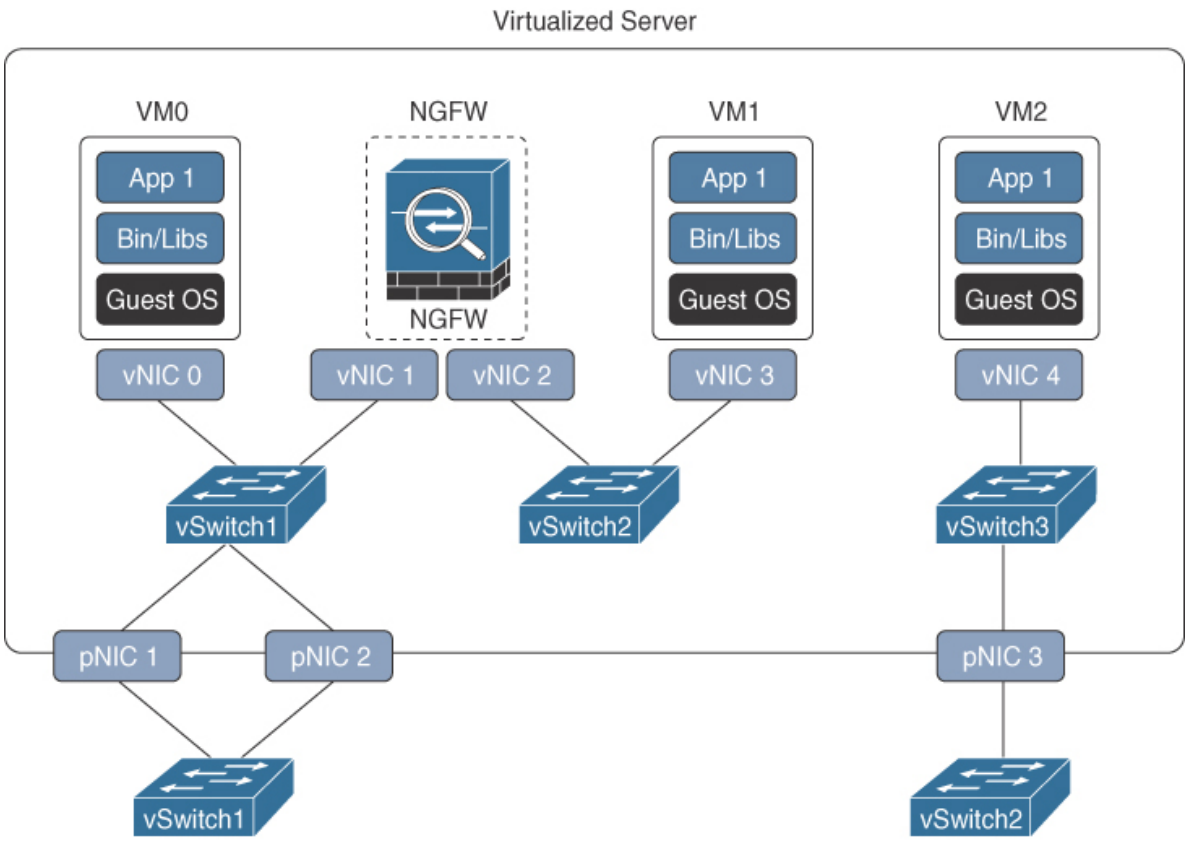


Figure 2-25 *Virtualized Server with vSwitches*

The downside to using standard vSwitches is that every vSwitch that is part of a cluster of virtualized servers needs to be configured individually in every virtual host, which is where distributed switching comes into the picture. Distributed virtual switching is a feature that aggregates vSwitches together from a cluster of virtualized servers and treats them as a single distributed virtual switch. These are some of the benefits of distributed switching:

- Centralized management of vSwitch configuration for multiple hosts in a cluster, which simplifies administration
- Allows for networking statistics and policies to migrate with virtual machines during a live VM migration
- Configuration consistency across all the hosts that are part of the distributed switch

Examples of virtual distributed switches include Cisco Nexus 1000V, Cisco VM-FEX, Cisco AVS, HPE 5900v, Open vSwitch (OVS), IBM DVS 5000v, and vSphere Distributed Switch.

Table 2-3 provides a quick glance at the vSwitch options available from multiple vendors.

Table 2-3 *The Most Popular vSwitch Options*

Hypervisor	Native vSwitch	Third Party
vSphere	Standard vSwitch Distributed Virtual Switch	Cisco Nexus 1000V * Cisco VM-FEX * IBM DVS 5000V * HPE 5900v *
Hyper-V	Native Hyper-V Virtual Switch	Cisco Nexus 1000V Cisco VM-FEX NEC Broadcom
KVM	Linux Bridge (some distributions include OVS natively)	Cisco Nexus 1000V OVS
XEN	OVS	OVS

* VMware ceased support for the third-party virtual switches starting with vSphere 6.5U2 and beyond.

Containers, just like VMs, also rely on vSwitches (aka virtual bridges) for communication within a node (server). Docker, for example, by default creates a virtual bridge called Docker0, and it is assigned the default subnet block 172.17.0.1/16. This default subnet can be customized, and user-defined custom bridges can also be used.

Figure 2-26 illustrates how every container created by Docker is assigned a virtual Ethernet interface (veth) on Docker0. The veth interface appears to the container as eth0. The eth0 interface is then assigned an IP address from the bridge's subnet block. As more containers are created by Docker within the node, they are each assigned an eth0 interface and an IP address from the same private address space. This results in all containers being able to communicate with each other only if they are within the same node. Containers in other nodes are not reachable by default, which is something that needs to be managed via routing at the OS level, or by using an overlay network.

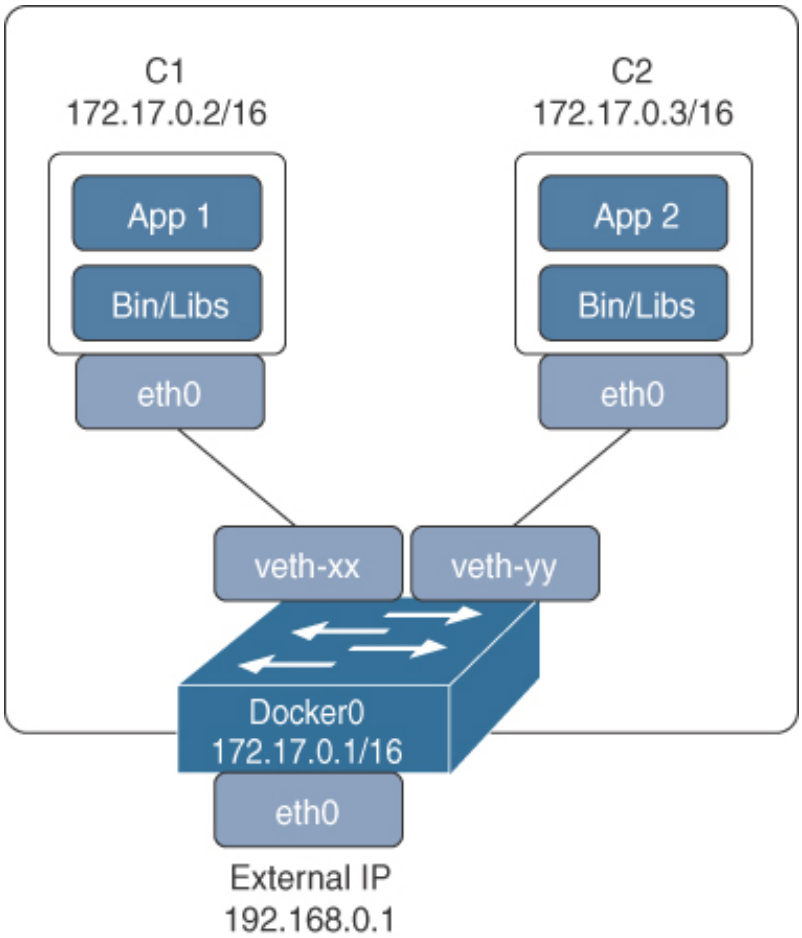


Figure 2-26 *Container Bridging*

If Docker is installed on another node using the default configs, then it will end up with the same IP addressing as the first node, and this needs to be resolved on a node-by-node basis.

A better way to manage and scale containers and the networking connectivity between them within and across nodes is to use a container orchestrator such as Kubernetes,

which is described in the next section, "Automation and Orchestration Tools."

AUTOMATION AND ORCHESTRATION TOOLS

To be able to achieve scalable multicloud environments, it is necessary to use automation and orchestration tools. Trying to manually create VMs, containers, applications, application migrations, and provision networks is not the way to go. With orchestrators and automation, all of these actions are abstracted and can be executed from a GUI.

Kubernetes

Google has been using containers for over a decade. Every application that runs at Google—for example, Gmail, Google apps, YouTube, or GCP—is a distributed application that run on containers, so needless to say, Google is an expert in managing distributed containerized applications at scale. In June 2015, Google published a research paper at Eurosys with details about Borg, Google's internal container-oriented large cluster-management system.

Kubernetes is a container orchestrator that supports cloud and bare-metal environments, and it is the open-source version of Google's Borg. Kubernetes is an entirely new codebase (meaning the code was entirely rewritten) that incorporates the best ideas from Borg into Kubernetes, and it addresses lessons learned and pain points that users identified with Borg over the years. Additionally, many of the developers at Google working on Kubernetes were formerly developers on the Borg project.

Other container orchestrations engines such as Docker Swarm exist, but Kubernetes is the most popular go-to container orchestration system in the industry.

Clusters, Nodes, and Pods

In Kubernetes, a cluster, as illustrated in Figure 2-27, is a collection of nodes. A node is a physical server or a virtual machine, and a pod is a collection of one or more containers that are all serving a specific purpose (usually they are all microservices that make up an application). A pod is the smallest entity that can be scheduled by Kubernetes; individual containers are not scheduled. Each pod gets assigned only one IP, and containers within the pod share the localhost address (that is, 127.0.0.1). A pod can be started, stopped, and replicated as a single unit, almost as if it was a VM.

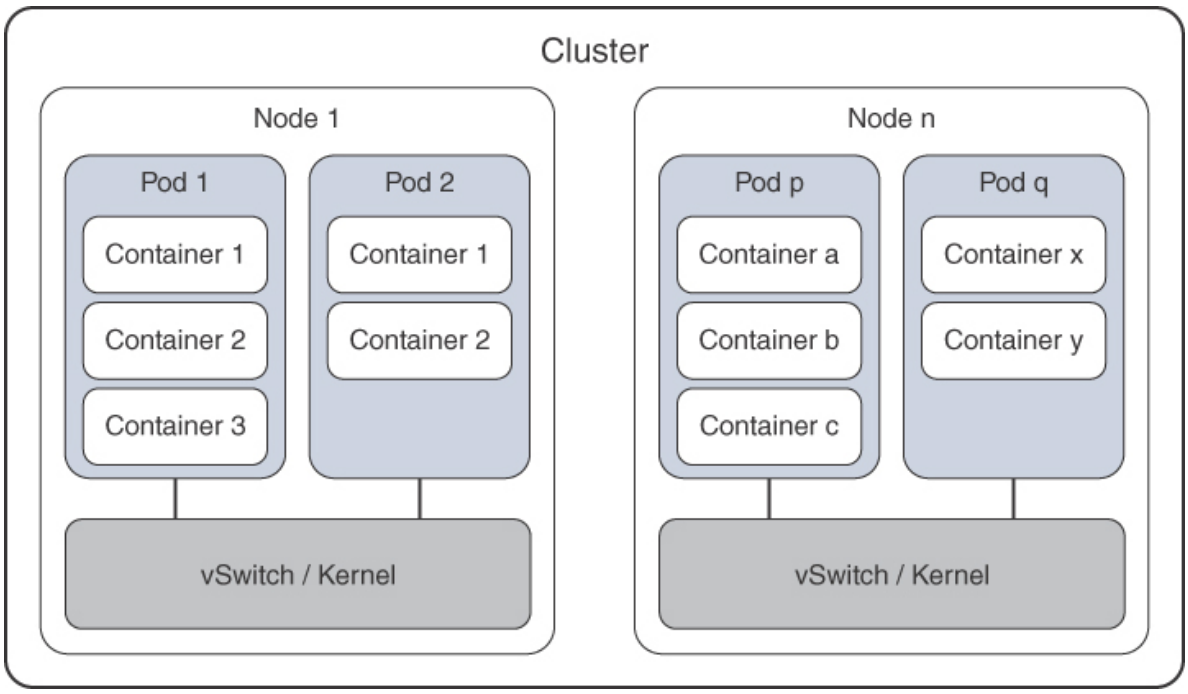


Figure 2-27 *Cluster, Nodes, and Pods*

Volumes

A pod volume (persistent storage) is required for containers that need to maintain the state of an application after reboots, relocations, and crashes. The reason for this is because a container file system only lives as long as the container does; if it restarts, any data and state will be lost.

Labels

In a Kubernetes environment many pods can be created, and to keep them organized, labels are necessary. Labels are nothing but a key-value pair, as shown in Figure 2-28. Label selectors can be used to request the apiserver (explained later) to retrieve a list of objects that match that label selector.

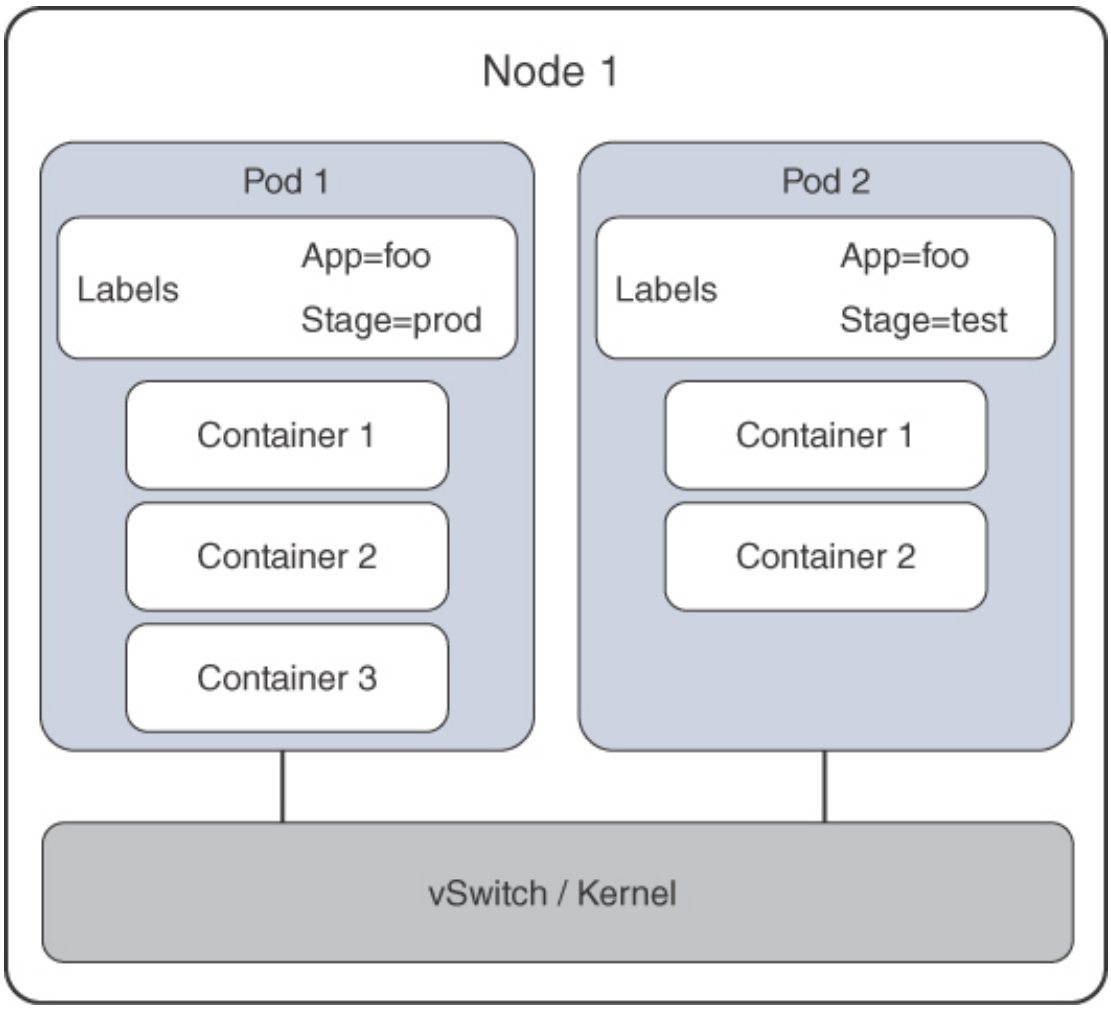


Figure 2-28 *Pod Label Assignment*

Kubernetes Cluster

Figure 2-29 illustrates a Kubernetes cluster that will be used as a reference to explain all of the basic components.

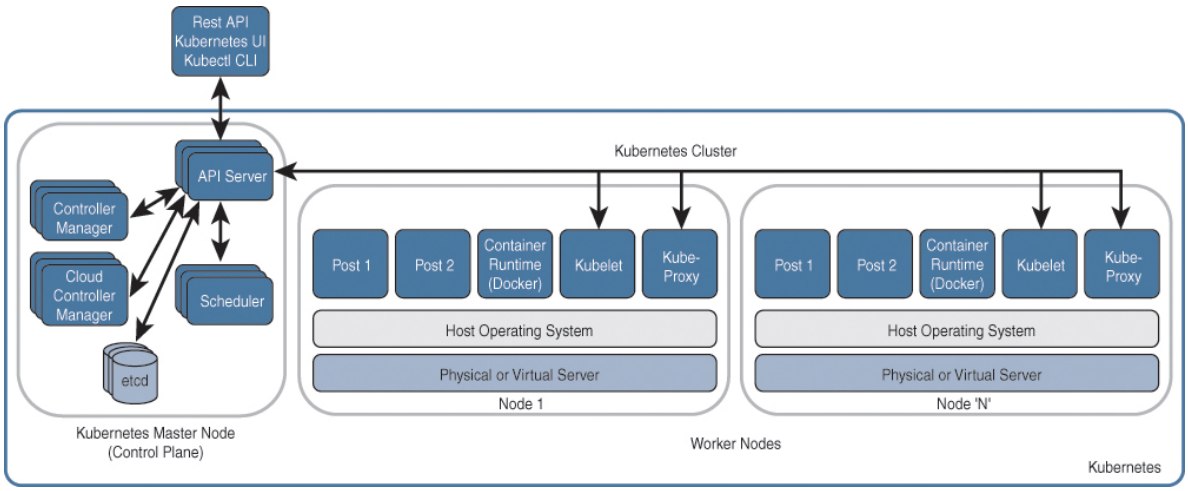


Figure 2-29 *Kubernetes Architecture*

The Kubernetes master node can be thought of as the control plane of Kubernetes, and it is recommended to have at least three for redundancy. The master node is composed of five main components:

> Note
>
> In some special cases, the master node can also act as a worker node, which means it would also include a container runtime, a kubelet, and a kube-proxy.

- **API server**: It services *Representational State Transfer* (REST) operations and provides the front end to the cluster's shared state through which all other components interact. Every operation that involves changing the state of an object, such as a pod, needs to go through the API server.
- **etcd**: It's a distributed key-value store. Examples of data stored are jobs being scheduled, created, and deployed, labels, pod or service details and state, and replication information.
- **scheduler**: It checks the environment to see where the workloads can run and create pods for them. Workloads come in all different shapes and sizes, and as they come in, the scheduler needs to figure out where to put them in a very similar way to a Tetris game.
- **controller-manager**: The controller manager is a daemon that runs all the controllers included with Kubernetes. A control loop is a nonterminating loop that regulates the state of a system. A controller is a control loop that watches the shared state of the cluster through the apiserver and makes changes, attempting to move the current state toward the desired state.

Logically, each controller is a separate process, but to abstract complexity, they all run as a single process.

The worker nodes are where the pods are run, and they include the following components:

- **Container runtime**: This is the software responsible for creating and managing the containers, where Docker is the default and most popular one. Other options include rkt, runc and any OCI runtime-spec implementation.
- **kubelet**: A kubelet is an agent that runs on each node that receives the configuration for a pod from the apiserver and ensures that the defined containers are up and running and in a healthy state. The kubelet doesn't manage containers

that were not created by Kubernetes.

- **kube-proxy**: kube-proxy acts as a network proxy for pods within a single worker node and for connections that originate from one pod to another pod and not connections coming from outside the cluster.

> Note
>
> In Kubernetes, the functionality of providing network proxy and load balancer capabilities is known as a **service**.

Kubernetes Networking

With default Kubernetes networking, as mentioned previously, each pod is assigned a single interface eth0 and a single IP address. All containers within a pod share a common networking space; in other words, if one container in the pod is running a web server listening on port 8080, another container within the same pod can connect to it using http://localhost:8080 or http://127.0.0.1:8080.

The default switch for a node is called cbr0 (instead of Docker0). This bridge is connected to the pods as well as to the external network. How it connects to the external network depends on the networking plug-in used.

Kubernetes allocates a user-configurable internal subnet range to each node, and from this range, it assigns an IP address to the bridge and to each pod within the node, as illustrated in Figure 2-30.

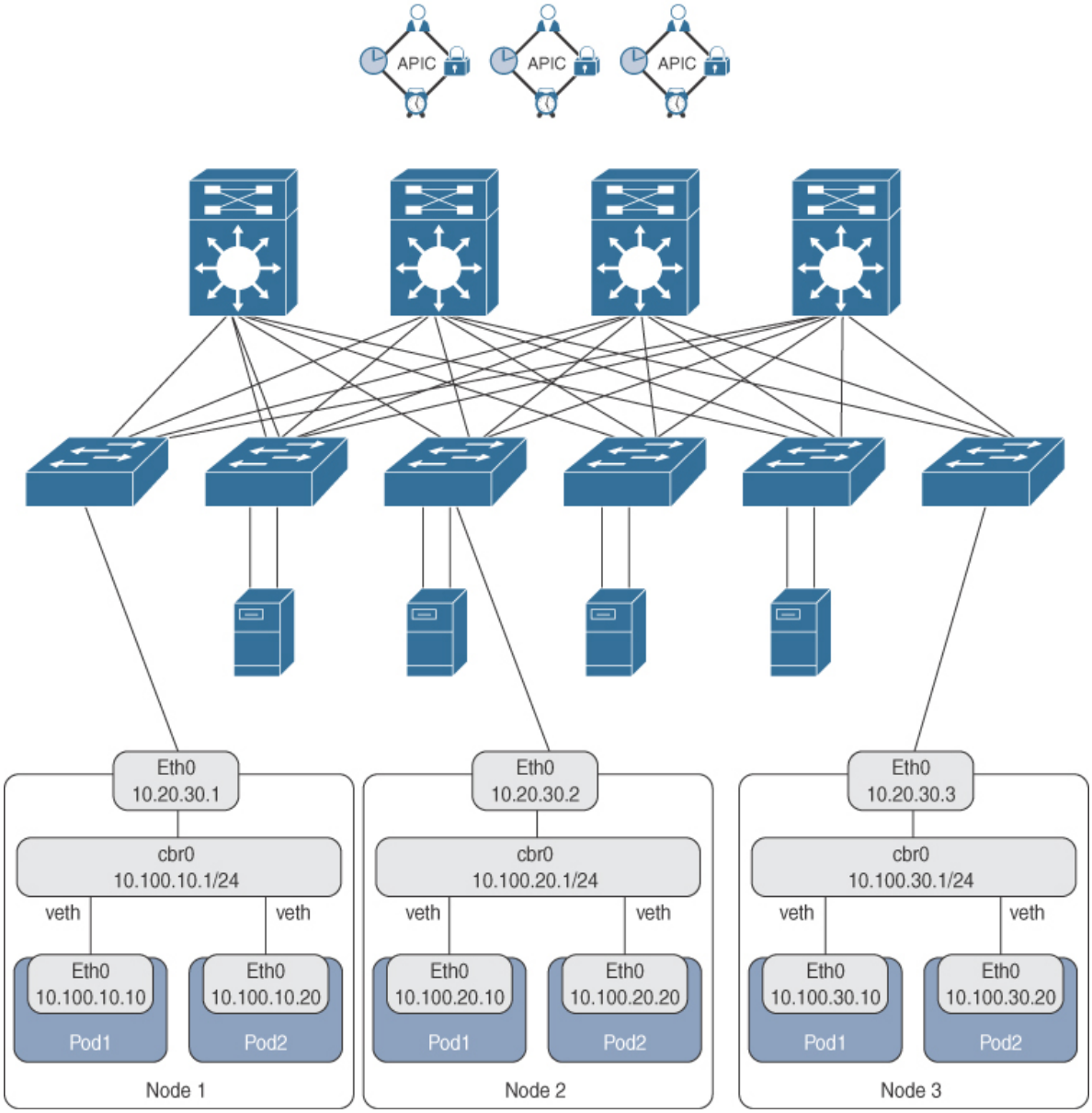


Figure 2-30 *Kubernetes Networking*

While there are many options on how to deploy an external network for inter-pod communication in Kubernetes, the Cisco recommended option is to use Cisco Application Centric Infrastructure (Cisco ACI), which is designed to offer policy-based automation, security, mobility, and visibility for application workloads, regardless of whether they run on bare-metal servers, hypervisors, or Linux containers. The Cisco ACI system-level approach extends the support for Linux containers by providing tight integration of Kubernetes.

Creating a Pod

A simple way to deploy a pod (or pods) is by creating a declarative YAML or JSON deployment configuration file. *Declarative* means the configuration file has instructions on what is desired (the intent), but it does not include details on how to do it; that is for Kubernetes to figure out.

The high-level steps that take place when a pod is created, as shown in Figure 2-31, are

as follows:

1. The file is posted to the API server via a REST API post through the Kubernetes GUI, the kubectl CLI, or from an application such as CloudCenter.

2. The information in the file is then sent to the scheduler, and it is also stored in etcd.

3. The scheduler then needs to figure out what the intent is of the configuration file; understand the constraints, policies, requirements, and how many CPU cores the pod needs; see what the required memory is, as well as any affinity to a specific node, replica sets (how many pods should be created), a specific persistence requirement (volume), and so on. The scheduler then selects a node that meets the intent of the configuration file.

4. The scheduler sends a message to the kubelet running on the selected node concerning what the requirements are for the pod.

5. The kubelet then contacts Docker and instructs it on what it needs to do (for example, number of pods, number of containers within the pod, IP address assignment for the pod, image download, and so on). At this point, the pod is created and ready for use.

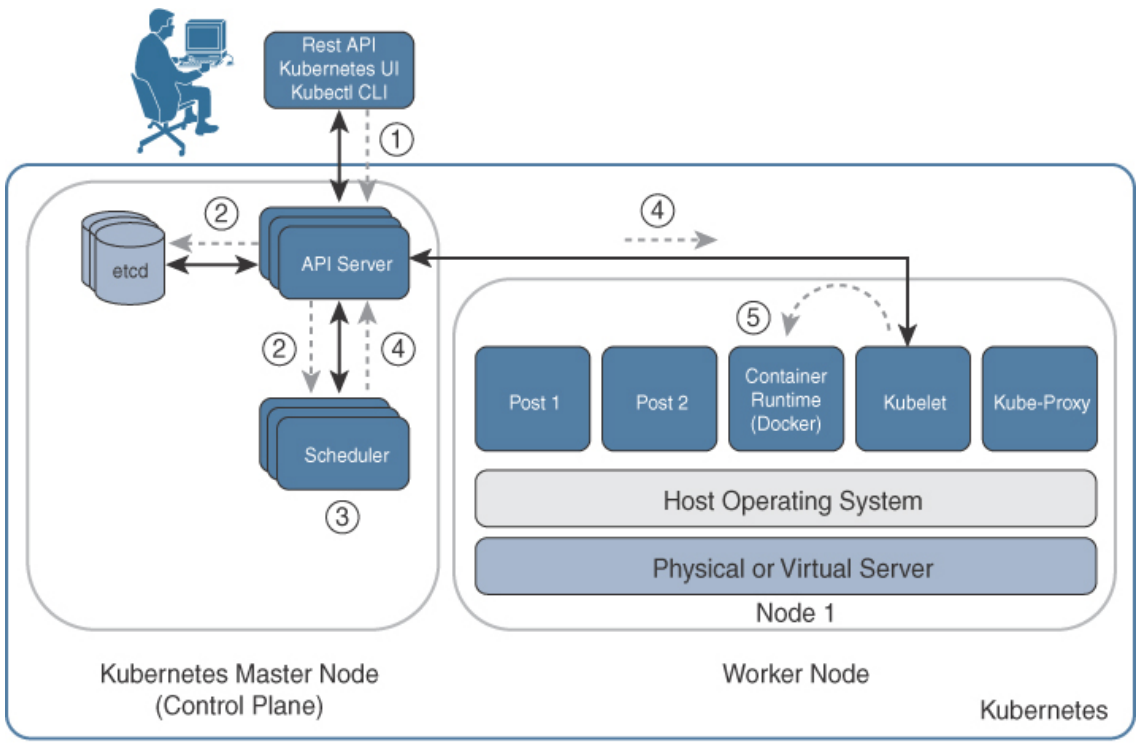


Figure 2-31 *Creating a Pod Walkthrough*

OpenStack

OpenStack is a cloud operating system that can be used to build private and public clouds; however, it is more prevalently used to build private clouds. It has a modular architecture, where each of its components (aka services or projects) is developed in individual development projects. Table 2-4 includes the OpenStack projects available in the Queens release. The services highlighted in bold are part of the core functionality of OpenStack.

Table 2-4 *OpenStack Projects (Queens Version)*

Project	Service	Description
Horizon	Web front end	Provides a web-based dashboard to manage OpenStack
Sahara	Workload provisioning	Provides simple means to provision a Hadoop cluster
Magnum	Workload provisioning	Provisions container orchestration engines such as Kubernetes, Docker Swarm, and Mesos
Trove	Workload provisioning	OpenStack Database as a Service (DBaaS)
Murano	Application lifecycle	Presents cloud-ready applications in a browsable categorized catalog
Freezer	Application lifecycle	Provides tools for backup, restore, and disaster recovery services

Solum	Application lifecycle	Provides automation to make cloud services easier to consume and integrate into an application development process
Aodh	Orchestration	OpenStack telemetry (ceilometer) alarming
Senlin	Orchestration	Clustering service for OpenStack clouds
Zaqar	Orchestration	Multitenant cloud messaging and notification service for web and mobile developers
Heat	Orchestration	Manages OpenStack's entire orchestration of infrastructure and applications
Mistral	Orchestration	Workflow service for OpenStack cloud
Blazar	Orchestration	A resource reservation service for virtual and physical resources
Ironic	Compute	Provisions bare-metal servers
Nova	Compute	Provisions virtual compute such as VMware, Xen, Hyper-V, KVM, and so on
Zun	Compute	OpenStack containers service

Designate	Networking	Provides DNS services to OpenStack tenants
Octavia	Networking	Load Balancing as a Service (LBaaS) for OpenStack
Neutron	Networking	Software-defined networking project that delivers Networking as a Service (NaaS) for OpenStack
Cinder	Storage	Provides access and manages block-based storage (Ceph, CloudByte, and so on) for compute instances
Swift	Storage	Distributed object storage system designed to scale
Manila	Storage	Manages a shared file systems in a multitenant cloud environment
Karbor	Shared services	Application Data Protection as a Service in OpenStack
Keystone	Shared services	Provides identity services for OpenStack
Glance	Shared services	Provides image management services for OpenStack and supports Raw, VirtualBox (VDI), VMware (VMDK, OVF), Hyper-V (VHD), and Qemu/KVM

		(qcow2) virtual machine images
Searchlight	Shared services	Provides indexing and search capabilities across OpenStack resources
Barbican	Shared services	REST API designed for key management
Panko	Monitoring tools	Event storage and REST API for ceilometer
Ceilometer	Monitoring tools	Provides telemetry services
Monasca	Monitoring tools	Provides monitoring services
Rally	Optimization/policy tools	Provides a framework for performance analysis and benchmarking for OpenStack components and full cloud deployments
Congress	Optimization/policy tools	Provides governance and compliance services
Watcher	Optimization/policy tools	Provides resource optimization services for OpenStack
Vitrage	Optimization/policy tools	OpenStack Root Cause Analysis (RCA) Engine

Cloudkitty	Billing/business logic	OpenStack Rating as a Service (RaaS)
Tricircle	Multi-region tools	Provides networking automation across Neutron in multiregion OpenStack clouds deployment
OpenStack-Ansible	Deployment/lifecycle tools	Ansible playbooks for deploying OpenStack
Kolla	Deployment/lifecycle tools	Deploys OpenStack services as Docker containers
Tripleo	Deployment/lifecycle tools	Heat templates for deploying OpenStack
Kuryr	Container infrastructure	Docker network plug-in that uses Neutron to provide networking services to Docker containers
Tacker	NFV	ETSI MANO NFV Orchestrator / VNF Manager
Blazar	Reservation	A resource reservation service for virtual and physical resources

Note

The codenames are usually related to the service function; for example, cinder provides block storage (to remember this, think of a cinderblock). As another example, ironic provisions bare-metal servers (think of iron as bare metal).

New versions of OpenStack are released every six months, which usually include new feature improvements and sometimes new projects. The OpenStack version-naming convention follows an alphabetical order—for example, Austin, Bexar, Cactus, Diablo, Essex, Folsom, Grizzly, Havana, Icehouse, Juno, Kilo, Liberty, Mitaka, Newton, Ocata, Pike, and Queens, with Queens being the latest release available at the time of writing.

Cisco CloudCenter (CCC)

CCC (formerly CliQr) is a multicloud platform for modeling, deploying, and managing applications with a simple architecture that includes two software components:

- **Cisco CloudCenter Manager (CCM)**: Single pane of glass for CCC
- **Cisco CloudCenter Orchestrator (CCO)**: Deployed locally on every cloud and on-prem environment.

CCM is the primary interface for users and administrators, and it can be accessed through a web browser user interface, CLI, or REST API. It includes all the management tools required for modeling, securely deploying, and managing applications in multiple on-prem, private cloud and public cloud environments. It also includes functions that can be used by administrators to have full visibility and control across all applications, users, governance rules, and clouds. For a traditional on-prem CCM deployment, the manager is delivered as a preinstalled virtual appliance, and only one CCM is required for each deployment, but additional managers can be added to meet disaster-recovery or high-availability requirements. CCM supports multitenancy, which means it can be used with multiple fully or partially isolated tenants, as needed.

A manager is linked to one or many orchestrators, and it can scale to simultaneously support thousands of applications.

Figure 2-32 illustrates the CloudCenter Multicloud Management Platform.

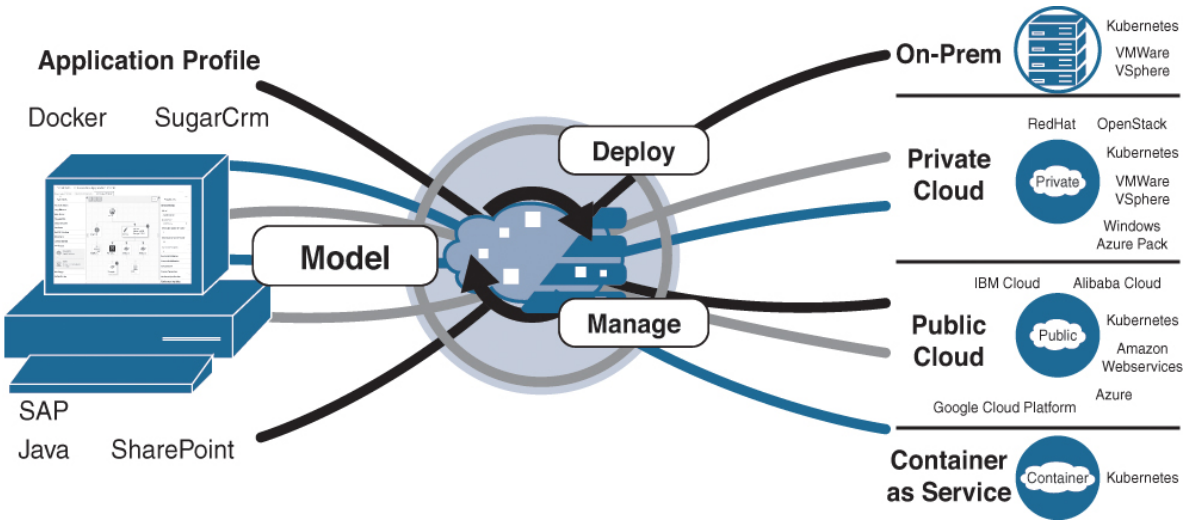


Figure 2-32 *CCC Multicloud Management Platform*

In Figure 2-32, *Model* refers to a CCC user-created application profile that describes how applications should be deployed, configured, and managed. The application profile model enables quickly and easily onboarding new or existing applications in any supported on-prem or cloud environment.

Figure 2-33 shows CCM's simple, visual topology modeler. Creating an application profile is as easy as dragging and dropping from the library of ready-to-use or customized services, images, and containers on the left pane onto the topology modeler and then just adding the connections between them. The application can be modeled with a mixture of containers, VMs, and services.

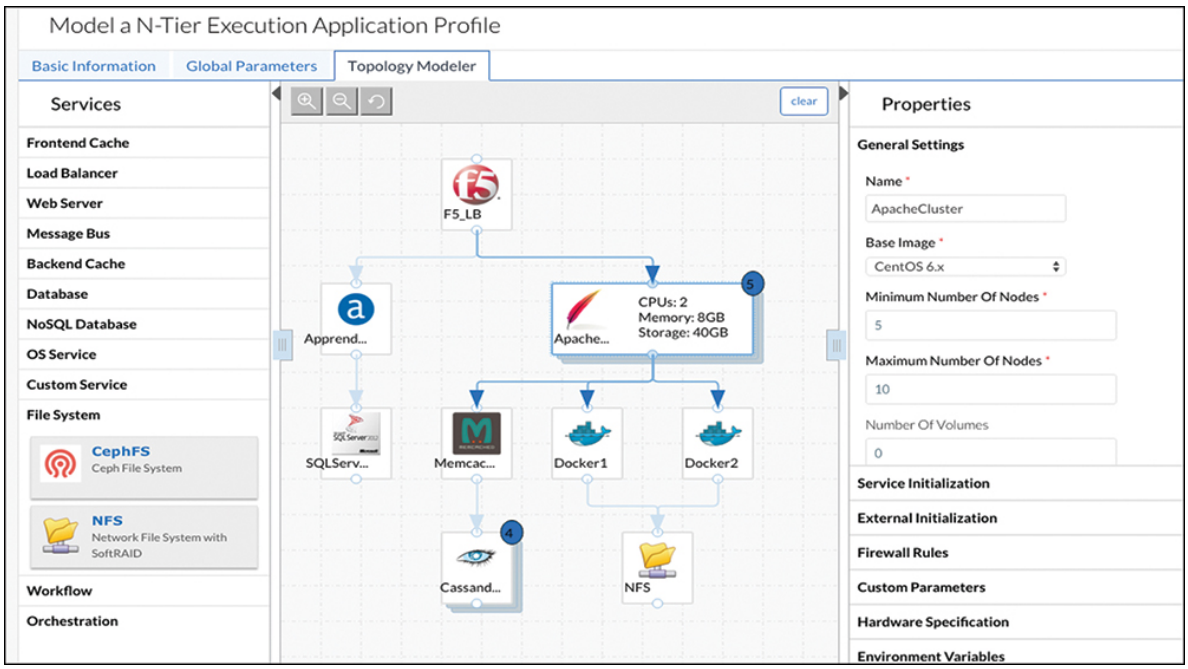


Figure 2-33 *CCC Built-in Drag-and-Drop Topology Modeler*

An application profile does not require any cloud-specific information to be provided or to hardwire the profile to any cloud infrastructure; it is completely cloud agnostic. For example, Figure 2-34 illustrates the traditional approach to deploy an application into different cloud environments via automation by using workflows and scripts that are

uniquely developed for each environment the application needs to be deployed on.

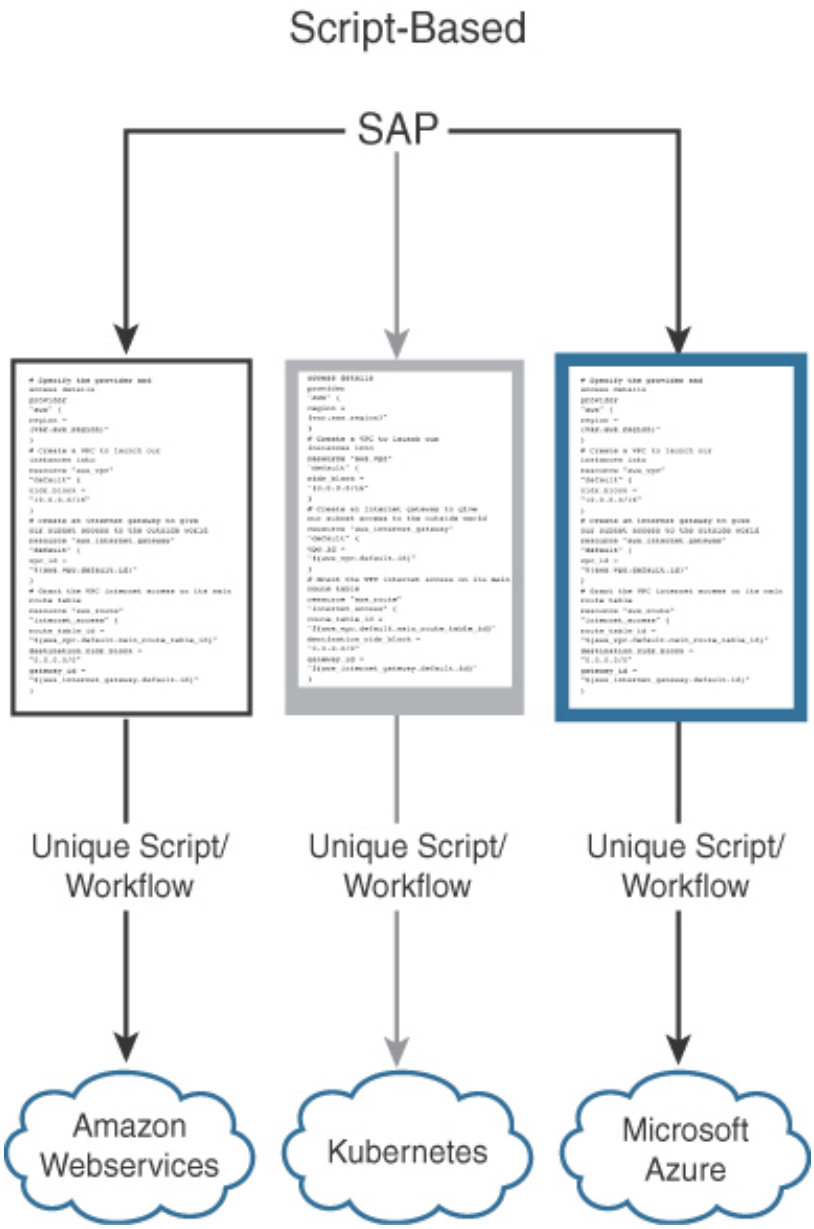


Figure 2-34 *Script-based Cloud Application Deployment*

Figure 2-35 illustrates how CloudCenter takes an application-centric approach, where the application profile is modeled once, and it can then be deployed and managed on any supported environment. This eliminates environment-specific automation that locks applications and workloads into a single cloud environment. CloudCenter abstracts the cloud and does the heavy lifting of understanding translating the application profile into the target cloud's native APIs.

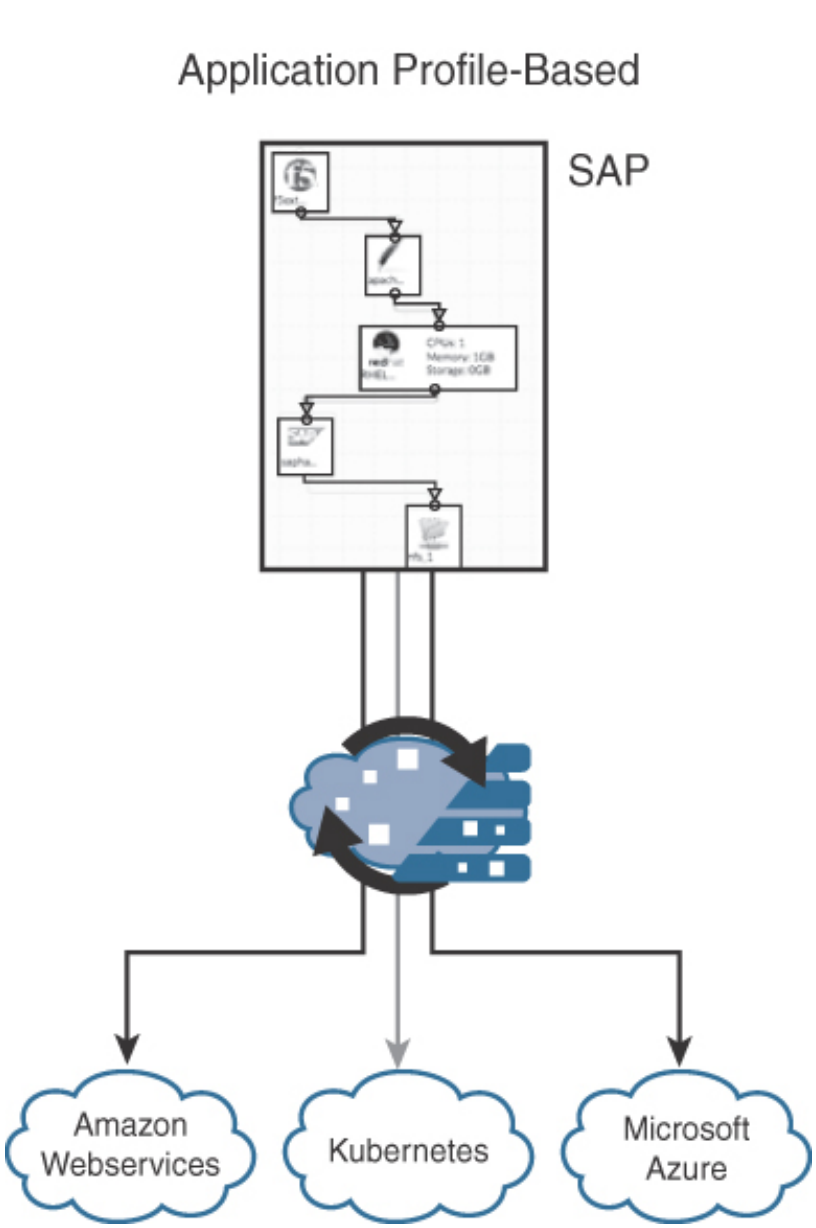


Figure 2-35 *CCC Cloud Application Deployment*

Deploy simply refers to clicking on the deploy button to deploy a new application profile as well as related components and data to any on-prem or cloud environment.

Manage refers to applying a wide range of application lifecycle actions to set policies to enable in-place scaling, support cross-environment bursting or high availability and disaster recovery, and start and stop the deployment

Cisco CloudCenter Orchestrator is a technology that decouples applications from the underlying infrastructure by abstracting the resources of the environment it is running on. One Orchestrator is deployed locally in each private cloud, public cloud, or on-prem environment and orchestrates the initial deployment of the application profile and all ongoing management requests that come from CCM. The CCM does not communicate directly with the cloud infrastructure; instead, it connects to the CCO via a REST API and sends it information and instructions, including application profiles, runtime policies, and application lifecycle management commands such as deploy, start, stop, and remove. The Orchestrator runs those commands and sends a status update back to

the CCM.

The Orchestrator abstracts the unique API and services offered by each cloud, and it uses the same communication mechanism back to the Manager regardless of the cloud on which the Orchestrator is installed.

Artifact Repositories

Organizations typically maintain application packages, data, and scripts in multiple repositories of their choice. Artifact repositories can be used to link to an existing repository to store and access files and to point to application binaries, scripts, and shared files. The preconfigured Cisco CloudCenter Network File System (NFS) options can be used to mount storage with multiple disks and encryption.

When an application or application profile is being modeled, the relevant repository can be selected to provide the relative path to the application packages, scripts, or files. The list of available repositories is displayed for user selection.

Multitenant

Cisco CloudCenter offers various multitenant models to support typical enterprise IT multicloud use cases, as shown in Figure 2-36.

- **Full isolation**: With Cisco CloudCenter, each tenant can be fully isolated from other peer tenants. In this way, two completely independent business units can use a single Cisco CloudCenter instance while strictly separating tenants.

- **Partial isolation**: Cisco CloudCenter offers an option for partial isolation between parent and child tenants. In some cases, a central IT organization may offer shared services, delivered either on-prem or through the CSP, that are consumed by various business units that are independent. For independent IT departments, the central IT organization may want to enforce OS image standards, require use of specific artifact repositories, or require a common rules-based governance framework.

- **Flexible sharing**: Cisco CloudCenter facilitates sharing within each tenant. Powerful features for sharing application profiles, application services, deployment environments, and more multiply the speed and agility benefits of an application-defined management solution.

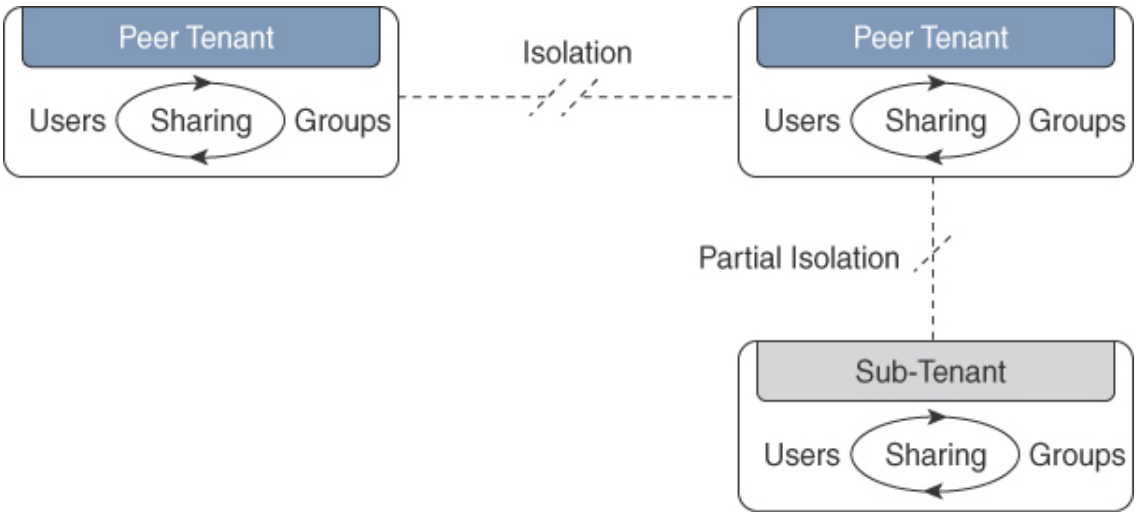


Figure 2-36 *CCC Multitenant Models*

Application Migration

As seen earlier in "Workload Migration" section of this chapter, migrating applications to the cloud can be overwhelming and time consuming, and organizations want to move to the cloud as fast as possible and with the least amount of downtime. In a typical on-prem-to-cloud migration, VMs are migrated one at a time. However, the problem is that applications are sometimes spread out across a group of virtual machines, and moving one of them into the cloud can create an orphaned state for an application. Therefore, it is necessary for all the VMs that make up the application to be migrated as a group.

It's a challenge to find all the application components for existing applications not created via CloudCenter (also known as brownfield applications), but with the help of application discovery tools, the application dependencies can be derived and the grouping of virtual machines that make up the application can be discovered. Once the application discovery is done, the migration of the application can be planned.

Planning for a multiphase approach for migration, as illustrated in Figure 2-37, is essential. This approach helps in the effective migration of applications by segregating the tasks according to their functional aspects. The Cisco CloudCenter solution simplifies the application migration process.

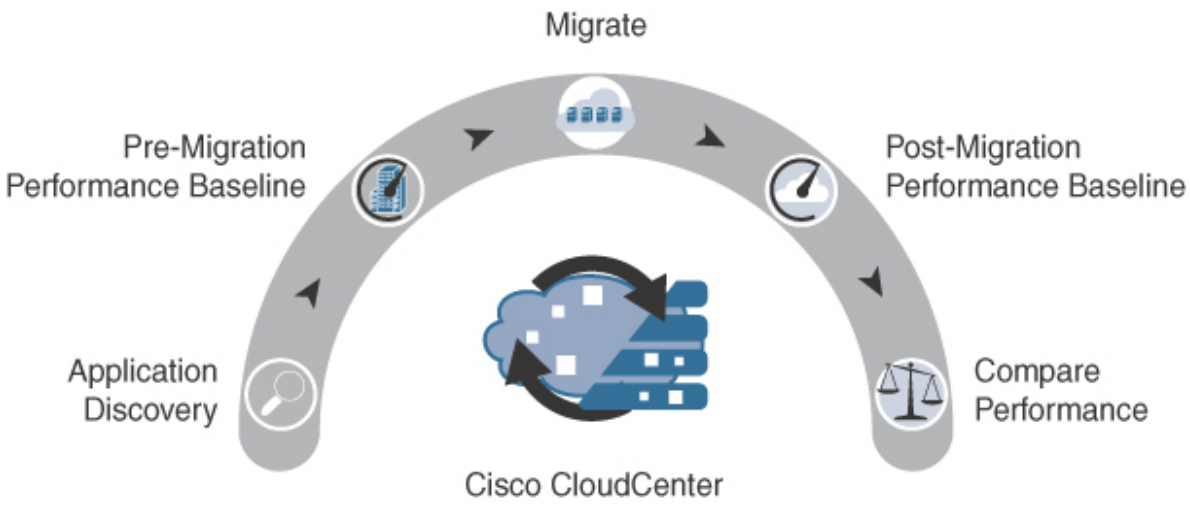


Figure 2-37 *Application Migration Flow with Cisco CloudCenter*

Application Discovery

Collecting information about the brownfield (legacy) applications in the existing environment is critical to understanding its interdependencies and planning how the application should be migrated. When migrating applications, the goal is to migrate the full application and avoid anything that would lead to an orphaned state for the application during or after the migration. To aid with the application discovery, iQuate iQcloud is recommended since it is the most advanced application discovery solution available.

iQuate provides an agentless SaaS service, which means there is no need to install agents on the VMs to monitor them; all that is required is to install an iQuate virtual appliance in the environment where the applications reside and to provide the login credentials for the VMs that need to be monitored. That is all it needs to generate a report with application discovery data.

Pre-migration Performance Baseline

To make sure that the application performance is the same or better in the public cloud, a pre-migration performance baseline needs to be performed. The application discovery data report generated by iQuate can be used to identify the VMs that make up the full application. The VMs are imported into CCC to enable AppD agents on them to capture the performance baseline of the application before migration based on technical and business metrics. Using technical and business metrics for the performance baseline allows measuring the performance not only at the technical level but also at the business level (for example, the end-user experience with the application, understanding business key performance indicators, and so on).

Migrate

After application discovery and initial performance baselining, the application can now be migrated. Two types of migrations can be performed:

- **Lightweight**: This type of migration is for applications that were created using a CCC application model. To migrate the application, CCC makes a backup of the application, its metadata, and its database in the current environment and redeploys the application in the target environment by restoring its metadata and database. This type of migration is lightweight since only the application is redeployed and not the full VM.

- **Heavyweight**: This type of migration is for brownfield applications. These applications need to be migrated using lift-and-shift-type migration tools. The lift-and-shift migration, as explained earlier in this chapter in the "Workload Migration" section, allows the application to be migrated without re-architecting or redesigning it, making it a less disruptive and quicker application migration.

Cisco CloudCenter uses CloudEndure Live Migration for heavyweight migrations, which is another SaaS service to migrate the application VMs to different clouds such as Amazon Web Services (AWS), Microsoft Azure, and Google.

Post-migration Performance Baseline

Once the application migration is complete, performance baselining is done again with AppD using the same technical and business metrics as the on-premises performance baselining.

Compare Performance

Using the pre-migration and post-migration performance baselines, you can compare the **application's** technical and business metrics. Technical metrics can be used by DevOps and application owners to gain confidence in the migration methodology, and the business metrics help in understanding end-user behavior and application interaction.

SUMMARY

In this chapter, a basic definition of cloud was provided using NIST's definition, which includes the following characteristics and models:

- Essential Characteristics

    - On-demand self-service: Portal
    - Measured service: Billing
    - Rapid elasticity: Scale out and scale in automatically
    - Resource pooling: Massive shared resources
    - Broad network access: Ubiquitous access for any device
- Service Models
    - Infrastructure as a Service (IaaS)
    - Platform as a Service (PaaS)
    - Software as a Service (SaaS)
    - Everything as a Service (XaaS)
- Cloud Deployment Models
    - Public cloud
    - Private cloud
    - Community cloud
    - Hybrid cloud
    - Multicloud (not defined by NIST)

Performance, scalability, and high availability for cloud were described with more focus on how to apply features and technologies (WAAS, PfR, and so on) to enhance application performance, which is directly correlated to what matters the most—the end-user experience.

How organizations share security risks and responsibilities with the CSPs was explained, as was security compliance guidelines, cloud threats, and recommendations on how to secure the cloud with Cisco solutions.

A workload migration strategy was also discussed, giving you a high-level methodology on the steps that need to be taken for successful workload migration into a cloud

environment.

Virtual machines and containers were described and compared. Also, a brief introduction was given showing how containers enable cloud native applications using microservices architectures.

The NFV architectural framework section introduced the concept of appliance virtualization, where networking elements can also be virtualized (called VNFs) and decoupled from their proprietary hardware. Cisco's NFVI platform was explained, which stems from the NFV framework and can be used to load VNFs of any kind, including third-party VNFs.

The various cloud connectivity options offered by the three leading CSPs—Microsoft Azure, GCP, and AWS—were discussed. These include direct connect, co-location, and IPSec VPN (the most popular). User-to-cloud access control using TrustSec SGTs and the CSR 1000V as well as other Cisco Security virtual platforms was shown. In addition, Cisco's SD-WAN solutions were included as the preferred way to access IaaS and SaaS services from the cloud using Cloud OnRamp, which is part of Cisco's cloud-first next-gen SD-WAN solution that enables multicloud.

In the last section, the Kubernetes architecture and its components are described, along with a high-level example of how pods can be deployed. OpenStack Queens release was described as a cloud operating system composed of multiple projects/services, of which the core ones are Horizon, Heat, Ironic, Nova, Neutron, Cinder, Swift, Keystone, and Glance.

Cisco CloudCenter (formerly CliQr) was explained, which is a multicloud platform for modeling, deploying, and managing applications on any cloud with a simple interface. We also described how to migrate lightweight and heavyweight application into the cloud using CCC along with iQuate and CloudEndure.

REVIEW QUESTIONS

1. Which of the following service models is *not* defined by NIST?

1. Software as a Service (SaaS)
2. Platform as a Service (PaaS)
3. Everything as a Service (XaaS)

4. Infrastructure as a Service (IaaS)

2 . Cisco CloudCenter supports application models with a mixture of containers and VMs. True or False?

1. True
2. False

3 . VNF and NFV are interchangeable. True or False?

1. True
2. False

4 . Which of the following workload migration strategies is the equivalent to lift-and-shift?

1. Rehosting
2. Replatforming
3. Repurchasing
4. Refactoring
5. Retain

5 . Which of the following Cisco SD-WAN components is optional?

1. vManage
2. vSmart
3. vBond Orchestrator
4. vAnalytics
5. vEdge Routers

REFERENCES

Gustavo A. A. Santana. *CCNA Cloud CLDFND 210-451 Official Cert Guide*. Indianapolis: Cisco Press, 2016.

Vinod Kataria. “MultiCloud Deployment of Self Operating Applications Using Cisco CloudCenter and AppDynamics APM” (presented at Cisco Live Barcelona, 2018).

Lloyd Noronha. “Getting poor Cloud performance? Cisco SD-WAN is here for you” (presented at Cisco Live, Barcelona, 2018).

Scott Ryan. “Intermediate—Demystifying Cloud Security Architectures” (presented at Cisco Live, Las Vegas, 2016).

Erik Vangrunderbeek, Kashif Zeeshan. “The Next Generation CCIE” (presented at Cisco Live, Las Vegas, 2017).

Naren Narendra. “Cisco Network Function Virtualisation Infrastructure (NFVI)” (presented at Cisco Live, Melbourne, 2018).

Arron Scott. “How to be a Hybrid Cloud ready IT Department” (presented at Cisco Live, 2017).

Markus Harbeck. “DNA Center—Network Automation easy, fast, reliable for everyone” (presented at Cisco Live, Barcelona, 2018).

Aaron Rohyans. “Understanding and Deploying Cisco SD-WAN” (presented at Cisco Live, Cancun, 2017).

Harris Haider, Rene Sobral. “Introduction to NFV orchestration using ETSI standard” (presented at Cisco Live, Barcelona, 2018).

Shannon McFarland. “Multicloud Networking—Design and Deployment” (presented at Cisco Live, Melbourne, 2018).

Jatin Sachdeva. “Introduction to Cloud Security” (presented at Cisco Live, Melbourne, 2018).

Frank Brockners. “Kubernetes Container Networking” (presented at Cisco Live, Barcelona, 2018).

Victor Moreno. “DNA Campus Fabric Automation—Policy Driven Manageability” (presented at Cisco Live, Barcelona, 2018).

Darrin Miller. “Advanced Security Group Tags: The Detailed Walk Through” (presented at Cisco Live, Barcelona, 2018).

Amazon Compute Service Level Agreement, https://aws.amazon.com/ec2/sla/.

Google Compute Engine Service Level Agreement (SLA), https://cloud.google.com/compute/sla?hl=en.

SLA summary for Microsoft Azure services, https://azure.microsoft.com/en-us/support/legal/sla/summary/.

FAQ: Discontinuation of third party vSwitch program (2149722), https://kb.vmware.com/s/article/2149722.

“Gartner Identifies Five Ways to Migrate Applications to the Cloud,” https://www.gartner.com/newsroom/id/1684114.

“Gartner Magic Quadrant for Application Performance Monitoring Suites,” https://www.gartner.com/doc/3868870.

“6 Strategies for Migrating Applications to the Cloud,” https://aws.amazon.com/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/.

Google Cloud Platform VPN, https://cloud.google.com/vpn/docs/concepts/overview.

Google Cloud Platform Interconnect, https://cloud.google.com/interconnect/.

Amazon Web Services VPN, https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Introduction.html.

Amazon Web Services Direct Connect, https://aws.amazon.com/directconnect/.

Microsoft Azure VPN Gateway documentation, https://docs.microsoft.com/en-us/azure/vpn-gateway/.

Microsoft Azure ExpressRoute documentation, https://docs.microsoft.com/en-us/azure/expressroute/.

Google cloud regions and zones, https://cloud.google.com/compute/docs/regions-zones/.

Azure regions and availability zones, https://azure.microsoft.com/en-us/global-

infrastructure/regions/.

AWS regions and availability zones, https://aws.amazon.com/about-aws/global-infrastructure/?nc2=h_l2_cc.

CSA, “The Treacherous Twelve: Cloud Computing Top Threats in 2016,” https://cloudsecurityalliance.org/group/top-threats/.

CSA, “Consensus Assessments Initiative Questionnaire (CAIQ),” https://cloudsecurityalliance.org/group/consensus-assessments/.

CSA, “Cloud Controls Matrix (CCM),” https://cloudsecurityalliance.org/group/cloud-controls-matrix/.

CSA, “Security Guidance for Critical Areas of Focus in Cloud Computing,” https://cloudsecurityalliance.org/guidance/.

CSA, “Security as a Service Implementation Guidance,” https://cloudsecurityalliance.org/group/security-as-a-service/.

NIST, “Risk Management Framework,” https://csrc.nist.gov/projects/risk-management/risk-management-framework-(RMF)-Overview.

Kubernetes documentation, https://kubernetes.io/docs/.

Docker documentation, https://docs.Docker.com.

Cisco documentation, https://www.cisco.com.

OpenStack, https://www.OpenStack.org.

Chapter 3

Foundational Network Programmability Methods

This chapter covers the following topics:

- Data Models
- Controller-Based Network Design
- Configuration Management Tools

COMMAND-LINE INTERFACE (CLI)

There are many different ways to connect to and manage your network. The most common method is to use the command-line interface, or CLI. This has been the most widely used way to configure your network for the last 30 years. However, the CLI, like anything, else has its pros and cons. Perhaps one of the most glaring and biggest flaws with using the CLI to manage your network is misconfiguration. Oftentimes, businesses have a high frequency of change in their network environment, and some of those changes can be extremely complex. When businesses have increased complexity in their network, the cost of something failing can be very high. This can stem from the increased time it takes to troubleshoot the issues in a complex network.

Failure in a network, however, doesn't necessarily mean software or a hardware component is to blame. A majority of network outages are caused by human beings. The causes for these outages are from misconfigurations due to a lack of understanding the complexity of a given network. Although not all outages or failures can be avoided, there are tools that can assist in lowering the number of outages caused by human error due to misconfigurations in the CLI. These tools will be discussed later in this chapter. Table 3-1 shows a brief list of common pros and cons associated with using the CLI.

Table 3-1 *CLI Pros and Cons*

Pros	Cons
Well known and documented.	Difficult to scale.
Commonly used method.	Large amount of commands.
Commands can be scripted.	Inflexible (you must know the command syntax).
Syntax help is available on each command.	Can be slow to execute commands.
Connection to the CLI can be encrypted (SSH).	Not intuitive.
	Can only execute a command at time.
	CLI and commands can change between software versions and platforms.
	Using the CLI can pose a security threat if you're using Telnet (clear text).

Context sensitive help is a structure built in to the Cisco CLI. When "?" is issued after each command, all available configuration help in that subconfiguration mode is

displayed. Figure 3-1 illustrates the context sensitive help available for the **show** command.

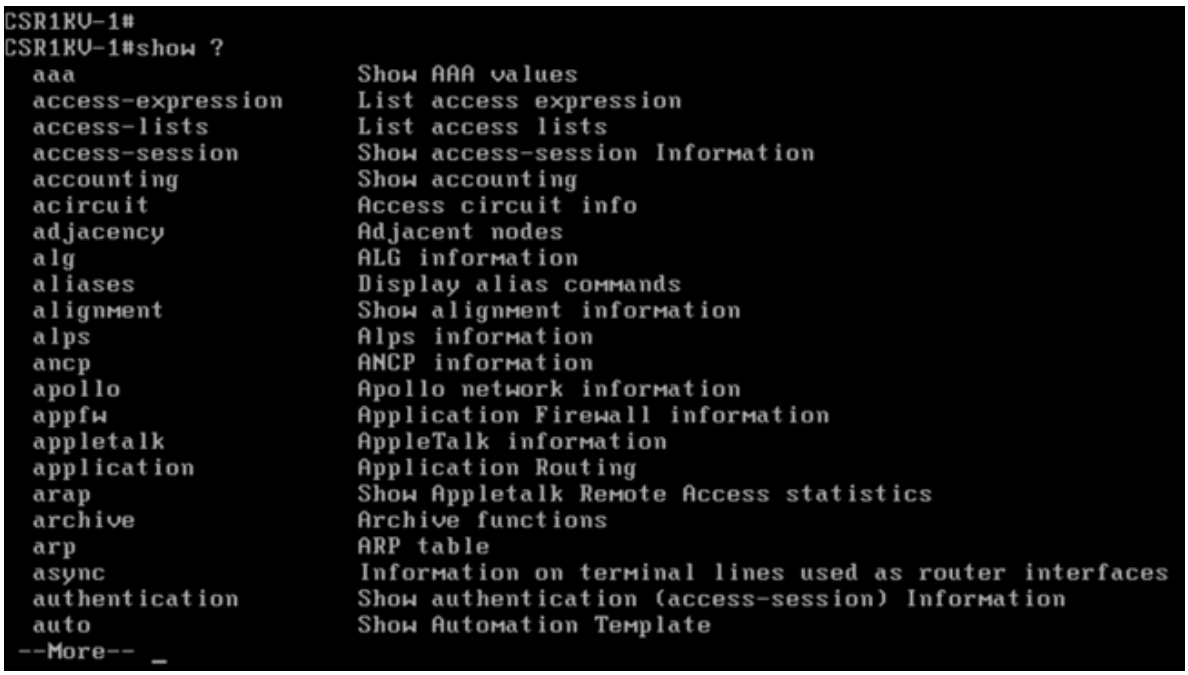


Figure 3-1 *CLI Context-Sensitive Help*

APPLICATION PROGRAMMING INTERFACE (API)

Another very popular method of communicating with and configuring your network is through the use of application programming interfaces, or APIs. APIs are used to communicate with various components of the network through software, and you can use these APIs to configure or monitor specific components of the network. There are multiple different types of APIs. However, the focus of this chapter will be on two of the most common APIs: northbound and southbound APIs. The following sections explain the differences between the two through the lens of network automation.

Northbound API

Northbound APIs are often used to communicate from a network controller to its management software. For example, Cisco DNA Center has a software graphical user interface (GUI) that is used to manage its own network controller. Typically, when a network operator logs in to a controller to manage their network, the information that is being passed from the management software is leveraging a northbound API. Best practices suggest that the traffic is encrypted between the software and the controller. Most types of APIs have the ability to use encryption to secure the data in flight.

Southbound API

An example of a southbound API would be when a network operator makes a change to a switch's configuration in the management software of the controller. Those changes would then be pushed down to the individual devices using a southbound API. These devices can be routers, switches, or even wireless access points. APIs interact with the components of your network through the use of a programmatic interface. Figure 3-2 illustrates the basic operations of a northbound API and a southbound API.

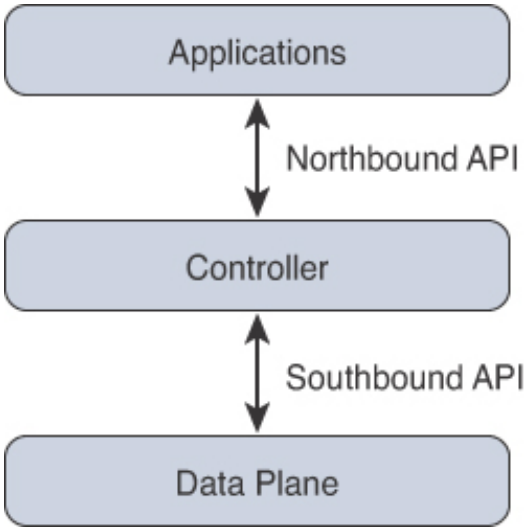


Figure 3-2 *Basic API Operations*

Representational State Transfer APIs (REST)

An API that uses REST is often referred to a RESTful API. What does this mean? RESTful APIs use HTTP methods to gather and manipulate data. Because there is a defined structure on how HTTP works, it offers a consistent way to interact with APIs from multiple vendors. REST uses different HTTP functions to interact with the data. Table 3-2 lists some of the most common HTTP functions and their associated use cases.

Table 3-2 *HTTP Functions and Example Use Cases*

HTTP Function	Action	Use Case
GET	Requests data from a destination	Viewing a website
POST	Submits data to a specific	Submitting login

	destination	credentials
PUT	Replaces data at a specific destination	Updating an NTP server
PATCH	Appends data to a specific destination	Adding an NTP server
DELETE	Removes data from a specific destination	Removing an NTP server

HTTP functions are very similar to the functions that most applications or databases use to store or alter data, whether it is stored in a database or within the application itself. These functions are called "CRUD" functions. CRUD is an acronym that stands for CREATE, READ, UPDATE, and DELETE. For example, in a SQL database, the CRUD functions are what are used to interact with or manipulate the data stored in the database. Table 3-3 lists the CRUD functions and their associated actions and use cases.

Table 3-3 *CRUD Functions and Example Use Cases*

HTTP Function	Action	Use Case
CREATE	Inserts data inside a database or application	Adding information to a database
READ	Retrieves data from a database or application	Reading information from a database
UPDATE	Modifies or replaces data in a	Changing a street address

	database or application	stored in a database
DELETE	Removes data from a database or application	Removing a customer from a database

Tools and Resources

Whether you are trying to learn how APIs interact with applications or controllers, to test code and outcomes, or are wanting to become a full-time developer, one of the most important pieces of interacting with any software via APIs is testing. Testing your code helps to ensure you are accomplishing the desired outcome that you hoped to achieve when executing the code. This section covers some tools and resources that will help you practice using APIs and REST functions. This also helps hone your coding skills in becoming a more efficient network engineer.

Introduction to Google Postman

Earlier, this chapter mentioned being able interact with a software controller using RESTful APIs. It also discussed being able to test your code to see if the desired outcomes are accomplished when executing it. Please keep in mind that APIs are software interfaces into an application or a controller. This means just like with any other device, you will need to authenticate to gain access to utilize the APIs. However, once you are authenticated, any changes you have access to make via the APIs will impact that application. This means that if you use a REST API call to delete data, that data will be removed from the application or controller, just as if you logged in to the device via the CLI and deleted the data. It is a best practice to use a test lab or the Cisco DevNet sandbox while learning or practicing any of these concepts. This way, there will be no accidental impact to your production environment.

> Note
>
> Cisco DevNet is covered in an upcoming DevNet section of this chapter.

Google has an application called Postman that allows us to interact with APIs using a console-based approach. Postman allows us to use various data types and formats to

interact with REST-based APIs. Figure 3-3 depicts the main Google Postman application dashboard.

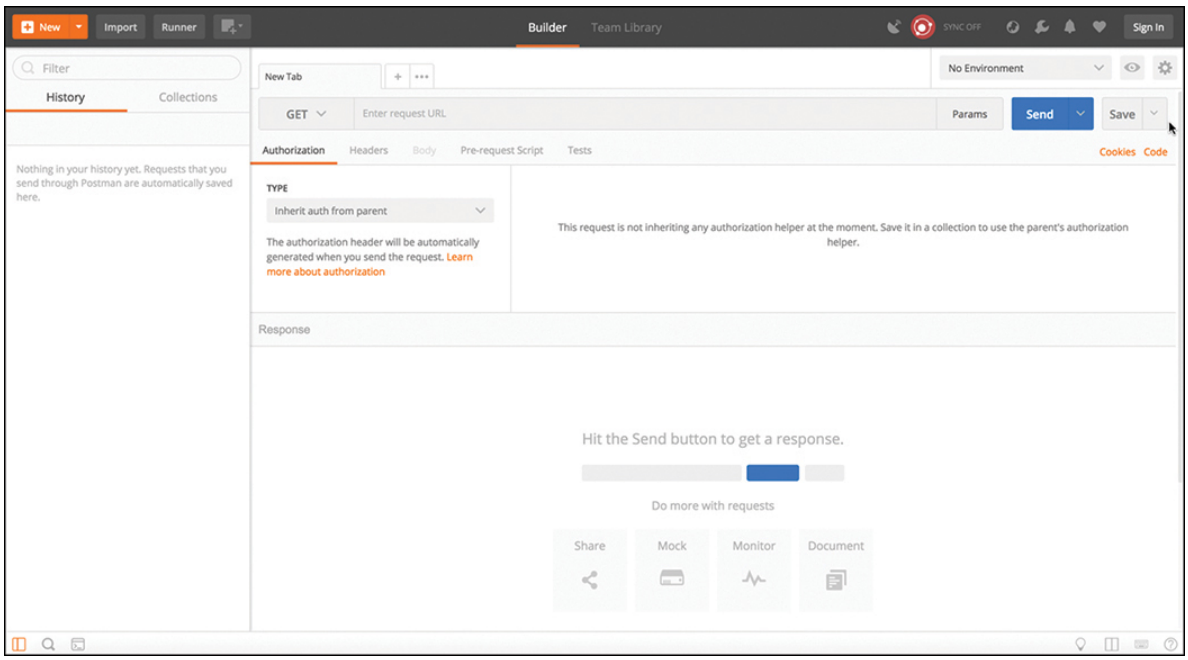


Figure 3-3 *Google Postman Dashboard*

> Note
>
> The screenshots of Google Postman used at the time of this writing may differ from the currently available version.

Within the Postman application, you can see the various sections you can interact with. Under the default configuration, the focus will be on using the "Builder" portion of the dashboard. The following sections are the ones that will require the most focus of our attention:

- History tab
- Collections window
- New tab
- URL bar

The History tab will show a list of all the recent API calls that were made using Postman. You have the option to clear your entire history at any time should you want to remove the complete list of API calls that have been made. This is done by clicking

the “Clear all” link at the top of the Collections window. You also have the ability to remove individual API calls from the history list by simply hovering the mouse over the API call and clicking the trash can icon from the submenu that pops up. An example of this is shown in Figure 3-4.

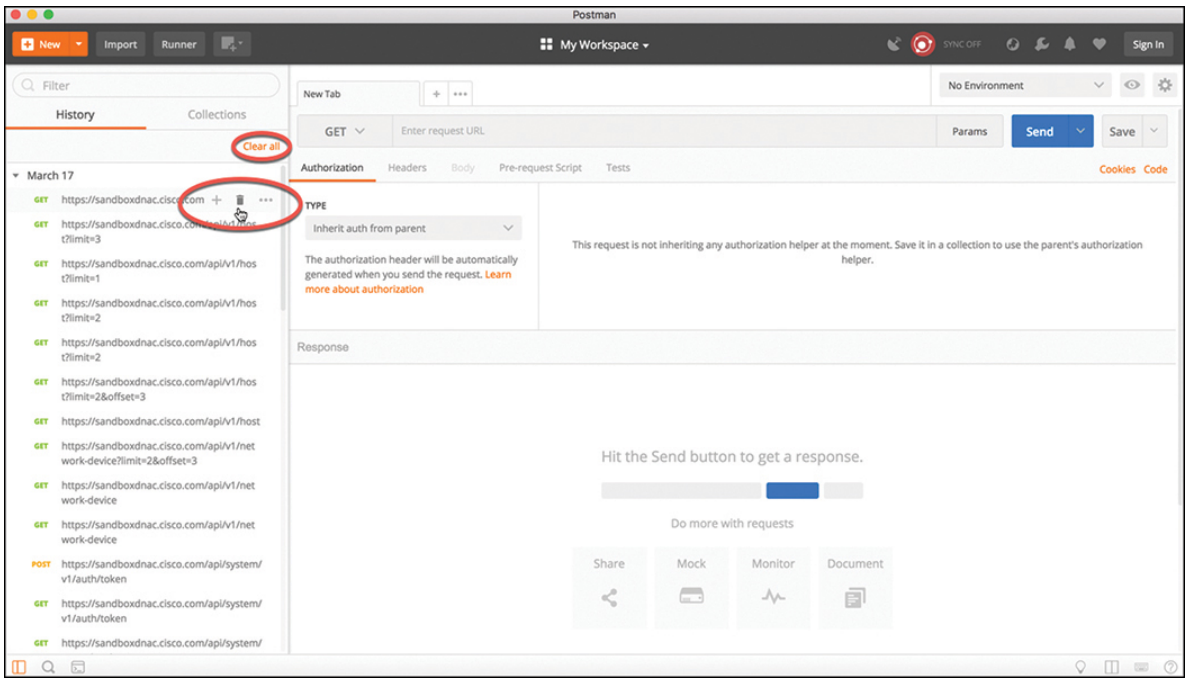


Figure 3-4 *Google Postman Clear History*

API calls can be stored into groups that are specific to a structure that fits your needs. These groups are called “collections.” Collections can follow any naming convention and appear as a folder hierarchy. For example, it’s possible to have a collection called DNA-C to store all of your DNA Center API calls in. This helps during testing phases, as API calls can easily be found and sorted by saving them into a collection. You can also select the collection to be a favorite by clicking the star icon to the right of the collection’s name. Figure 3-5 illustrates a collection called DNA-C that is selected as a favorite.

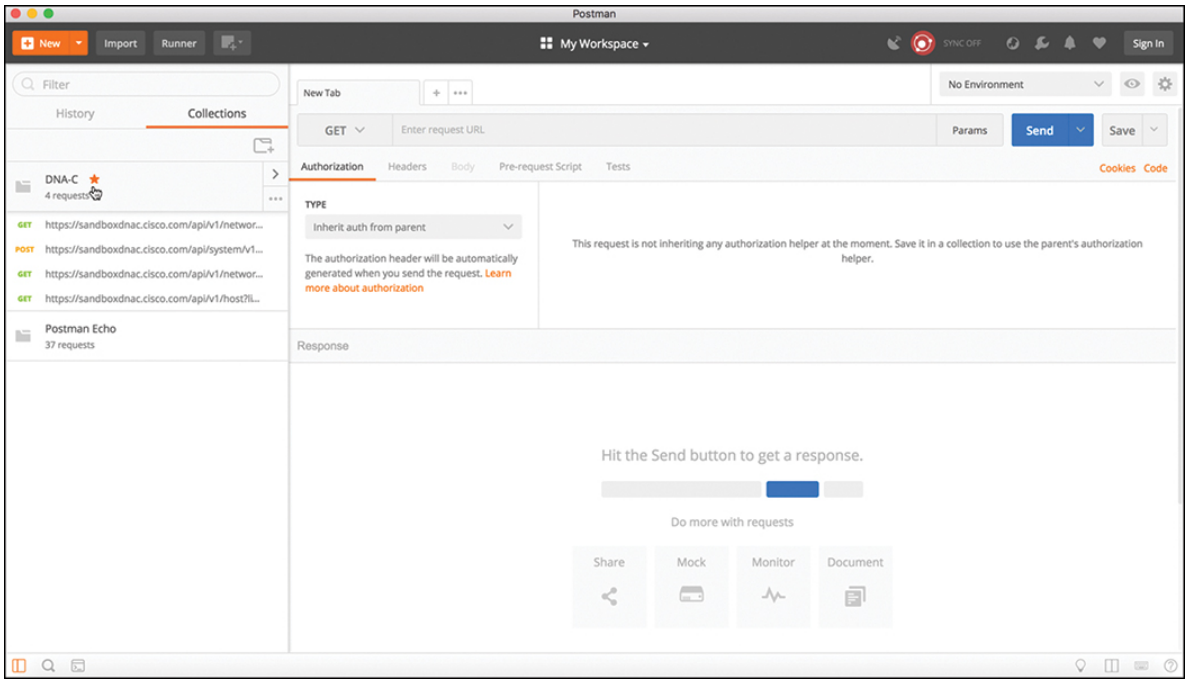


Figure 3-5 *Google Postman Collection*

Tabs are another very convenient way to work with various API calls. Each tab can have its own API call and parameters that are completely independent of any other tab. For example, this means you can have one tab open with API calls interacting with the DNA Center controller and another tab open that is interacting with a completely different platform, such as a Cisco Nexus switch. This is because each tab has its own URL bar to be able to use a specific API. Remember that an API call using REST is very much like an HTTP transaction. Each API call in a RESTful API maps back to an individual URL for a particular function. This means every configuration change or poll to retrieve data you make in a REST API has a unique URL for it, regardless of whether is a GET, POST, PUT, PATCH, or DELETE function. Figures 3-6 and 3-7 illustrate two different tabs using unique URLs for different API calls.

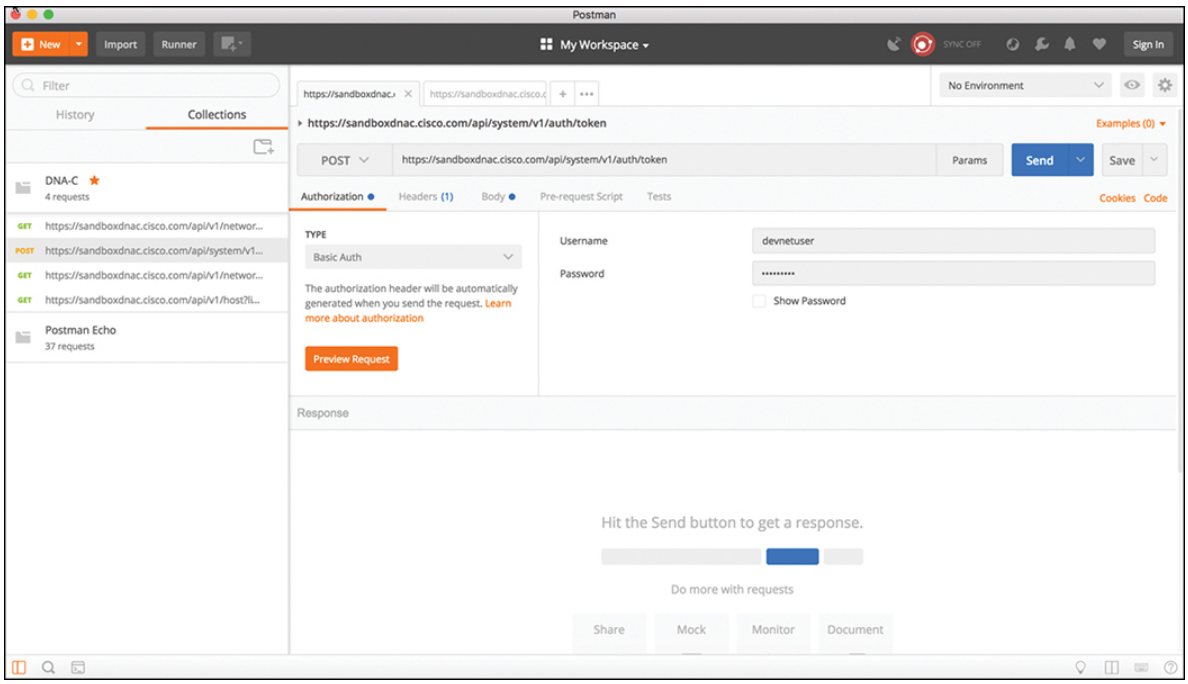


Figure 3-6 *Google Postman URL Bar with DNA Center Token API Call*

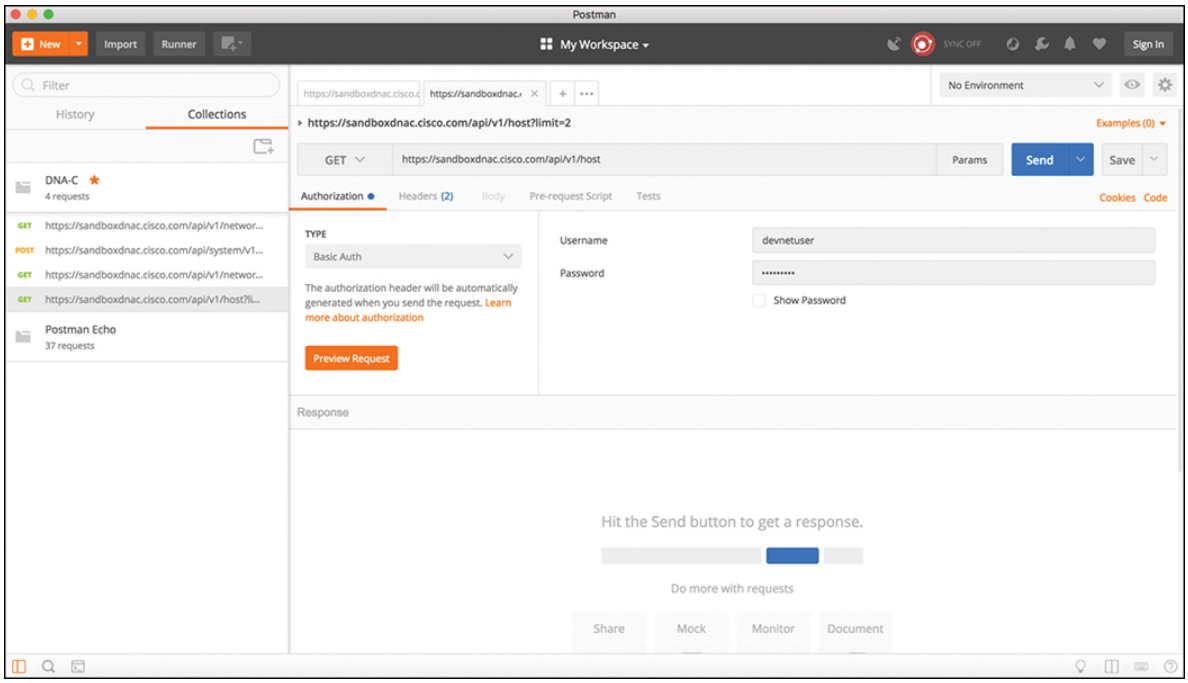


Figure 3-7 *Google Postman URL Bar with DNA Center Host API Call*

Data Formats (XML and JSON)

Now that the Postman dashboard has been shown, it's time to discuss two of the most common data formats used with RESTful APIs. The first one is called Extensible Markup Language (XML). This format may look familiar because it is the same format that is commonly used when constructing web services. XML is a tag-based language. This means that when you enter a tag within XML, it must begin with the < symbol. This also means it must end with the > symbol as well. For example, if you wanted to create a start tag named "interface," it would be represented as <interface>. One rule that goes along with XML is that if you start a section, you must end it. Or, in other words, for every beginning there is an end. Because you created a start tag called <interface>, the section will need to be closed by using an end tag. The end tag must contain the string of the start tag you are working with, followed by a / character. In this example, the end tag for <interface> would be <interface/>. Now that there is a start tag and an end tag, different types of code and parameters can be put inside or in between the tags. Example 3-1 depicts a snippet of XML output with both start and end tags as well as some various configuration parameters.

Example 3-1 *XML Code Snippet*

```
<users>
   <user>
     <name>root</name>
   </user>
   <user>
     <name>Jason</name>
   </user>
```

```
    <user>
      <name>Jamie</name>
    </user>
    <user>
      <name>Luke</name>
    </user>
  </users>
```

Notice that each section of Example 3-1 has a start tag and an end tag. The data is structured within a section called “users” that contains the following four individual users:

- root
- Jason
- Jamie
- Luke

Before and after each username is a start tag called <user> and an end tag called <user/>. The output also contains a start tag called <name> and an end tag called <name/>. These items start and end the tag that contains the actual user’s name. If you wanted to create another section to add on, you could simply follow the same logic as used in the previous example and build out more XML code. A key thing to keep in mind is that indentation of your XML sections is very important. For instance, if you didn’t use indentation, it would be much harder to read and follow each section in the XML output. This is why XML has been deemed so easy to read. Not only can humans read it, but applications can also read it. Another very common XML snippet is one that shows available interfaces on a device, such as a router or a switch. The Example 3-2 snippet, however, shows an XML code snippet without indentation to illustrate the difference in legibility. Although indentation is not required, it is certainly a recommended best practice.

Example 3-2 *Source Code for SIMPLE: A Very Simple OpenGL Program*

```
<interfaces>
 <interface>
 <name>GigabitEthernet1</name>
 </interface>
 <interface>
 <name>GigabitEthernet11</name>
 </interface>
 <interface>
```

```
            <name>Loopback100</name>
            </interface>
            <interface>
            <name>Loopback101</name>
            </interface>
            </interfaces>
```

The second data format that is important to cover is called JavaScript Object Notation (JSON). Although JSON has not been around as long as XML, it is currently taking the industry by storm. This data format is gaining popularity because it can be argued that JSON is much easier to work with than XML. It is simple to read and create, and the way the data is structured is much cleaner. JSON stores all its information in key-value pairs. There is much debate about whether JSON will eventually replace XML. Much like with XML, data that is indented is much cleaner in appearance and more legible. However, even without indentation, JSON is extremely easy to read. Like the name suggests, JSON uses objects for its format. Unlike XML, JSON objects start with { and end with }. These are commonly referred to as curly braces. Example 3-3 shows the same username example shown earlier in the XML section, but now in JSON format. It also can be read as having four separate key-value pairs—one for each user's name.

Example 3-3 *JSON Code Snippet*

```
{
  "user": "root",
  "user": "Jason",
  "user": "Jamie",
  "user": "Luke"
}
```

In the case of this JSON code snippet, you can see that the key used is "user" and the value for each key is a unique username.

Now that the XML and JSON data formats have been explained, it is important to circle back to actually using the REST API and the associated responses and outcomes of doing so. First, covering the different HTTP response codes and their importance is of top priority. Most Internet users have experienced navigating to a website and getting the dreaded "404 Not Found" error. What might not be clear to the users is what that error actually means. Table 3-4 lists the most common HTTP response codes as well as the reasons users may receive each one.

Table 3-4 *HTTP Response Codes*

HTTP Status Code	Result	Common Reason for Response Code
200	OK	Using a GET or POST to exchange data with an API.
201	Created	Creating resources via a REST API call.
400	Bad Request	Request failed because of a client-side issue.
401	Unauthorized	Client not authenticated to access site or API call.
403	Forbidden	Access not granted based on supplied credentials.
404	Not Found	Page at HTTP URL location does not exist or is hidden.

The first example covered in this section is the DNA Center Token API mentioned earlier in this chapter. The DNA Center Controller expects all incoming data from the REST API to be in JSON format. It is also important to note that the HTTP POST function is used to send the credentials to the DNA Center Controller. DNA Center uses an authorization concept of Basic Auth to pass a username and password to the DNA Center Token API to authenticate users. This API is used to authenticate users to the DNA Center Controller in order to make additional API calls. Just as users do when logging in to a device via the CLI, if the platform secured properly, they should be prompted for login credentials. The same method applies to using an API to authenticate to software. The key pieces of information necessary to successfully set up the API call in Postman are as follows:

1. In the URL bar, enter **https://sandboxdnac.cisco.com/api/system/v1/auth/token** to target the Token API.

2. Select the HTTP POST operation in the drop-down box.

3. Ensure under the Authorization tab that the type is set to Basic Auth.

4. Enter a username of **devnetuser** and a password of **Cisco123!**.

5. Select the Headers tab and enter **Content-Type** as the Key setting.

6. Select **application/json** as the Value setting.

Once Postman has been set up, you will need to select the Send button to pass the credentials to the DNA Center Controller via the Token API. Figure 3-8 illustrates the Postman setup required to authenticate with the DNA Center Controller.

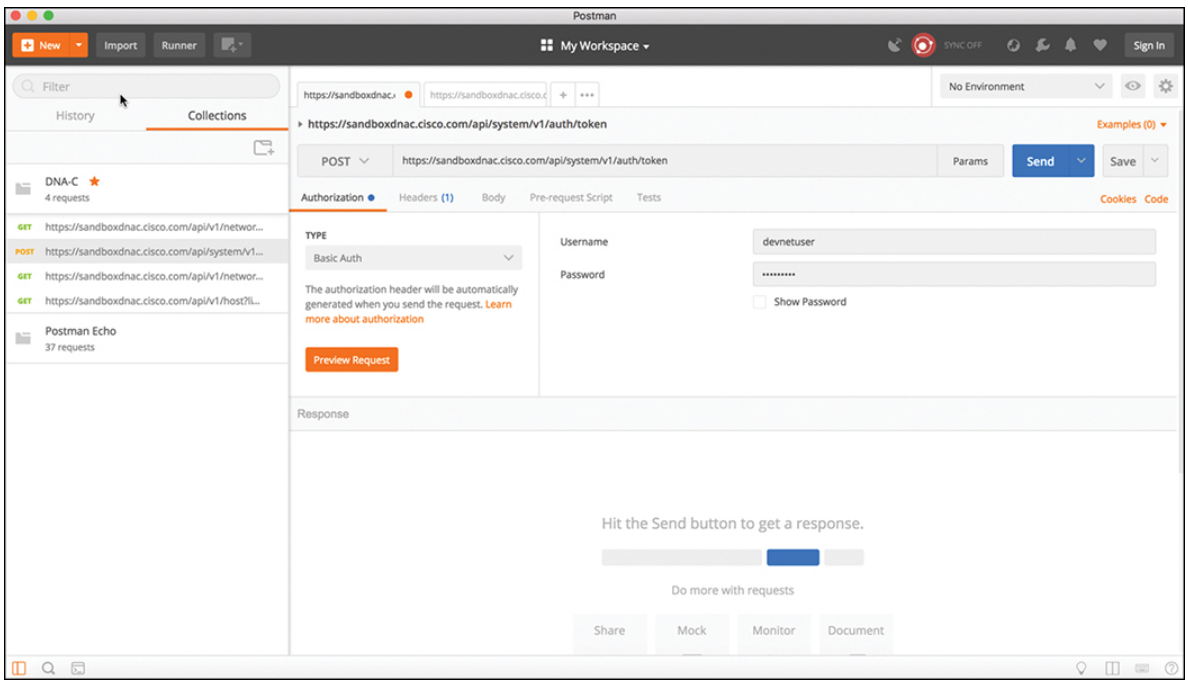


Figure 3-8 *Google Postman URL Bar with DNA Center Token API Call*

Once you successfully authenticate to the DNA Center Controller, you will receive a "token" that contains a string similar to the following:

Click here to view code image

```
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI1YTU4Y2QzN2UwNWJiYTA
wOGVmNjJiOTIiLCJhdXRoU291cmNlIjoiaW50ZXJuYWwiLCJ0ZW5hbnROYW1lIjoiVE5UM
CIsInJvbGVzIjpbIjVhMzE1MTYwOTA5MGZiYTY5OGIyZjViNyJdLCJ0ZW5hbnRJZCI6I
jVhMzE1MTlkZTA1YmJhMDA4ZWY2MWYwYSIsImV4cCI6MTUyMTQ5NzI2NCwidXNlcm5hb
WUiOiJkZXZuZXR1c2VyIn0.tgAJfLc1OaUwaJCX6lzfjPG7Om2x97oiTIozUpAzomM"
```

This token is needed for all future API calls to the DNA Center Controller. Think of it as a hash that is generated from your login credentials. This token will change every time an authentication is made to the DNA Center Controller. It is important to remember that once you are authenticated, the token you receive will be only usable for your current authenticated session to the controller. If another user authenticates via the Token API, they will receive a unique token in order for them to be able to utilize the APIs based on their login credentials. Figure 3-9 shows the response from DNA Center by issuing the POST operation to the Token API.

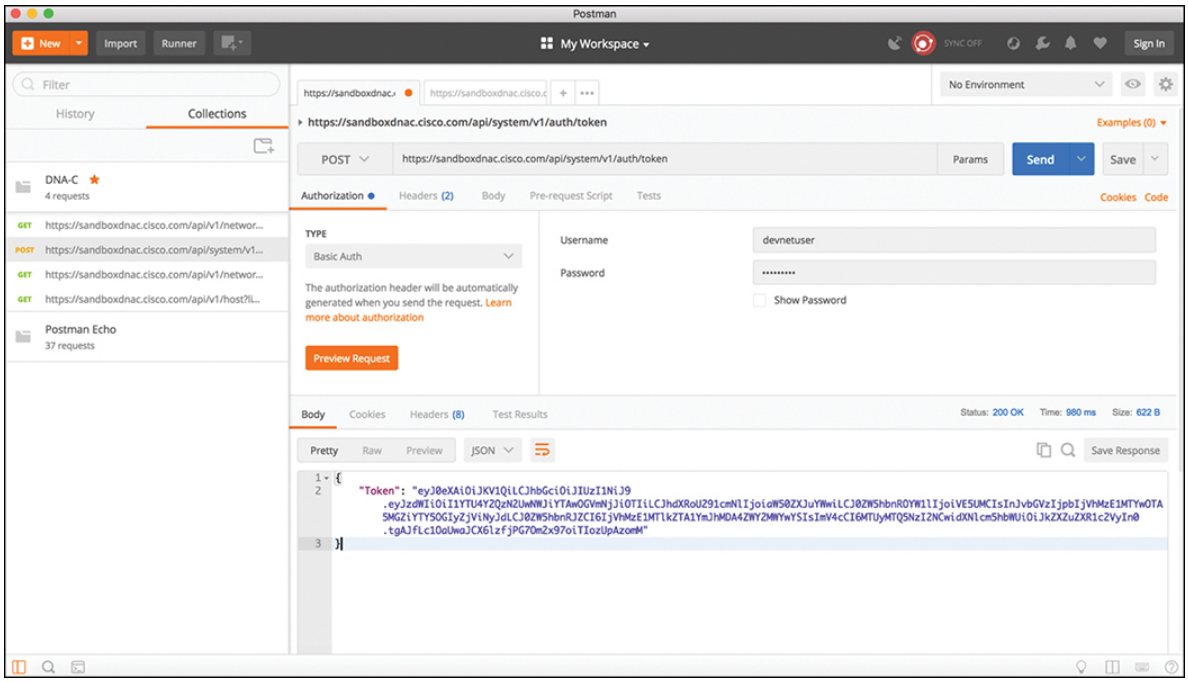


Figure 3-9 *DNA Center POST Operation*

You can see that an HTTP Status code of "200 OK" was received from the DNA Center Controller. Based on Table 3-4, you know that an HTTP Status code of 200 means that the API call completed successfully. In addition, you can also see how long it took to process the HTTP POST request. In this case, it was 980ms.

Now that you have successfully authenticated to the DNA Center Controller, you can now look at some of the other available API calls. The first API call covered in this section is the Network Device API, which allows users to retrieve a list of devices currently in inventory that are being managed by the DNA Center Controller. The first step is to prepare Postman to use the token that was generated when you authenticated to the controller. The following steps are necessary to leverage Postman to utilize the Network Device API:

1. Copy the token and click a new tab in Postman.

2. In the URL bar, enter **https://sandboxdnac.cisco.com/api/v1/network-**

device to target the Network Device API.

3. Select the HTTP GET operation in the drop-down box.

4. Select the Headers tab and enter **Content-Type** as the Key setting.

5. Select **application/json** as the Value setting.

6. Add another key and enter **X-Auth-Token**.

7. Paste the token in as the value.

The last step is to click Send to pass the token to the DNA Center Controller and perform an HTTP GET to retrieve a device inventory list using the Network Device API. Figure 3-10 illustrates the proper setup of Postman to use the Network Device API.

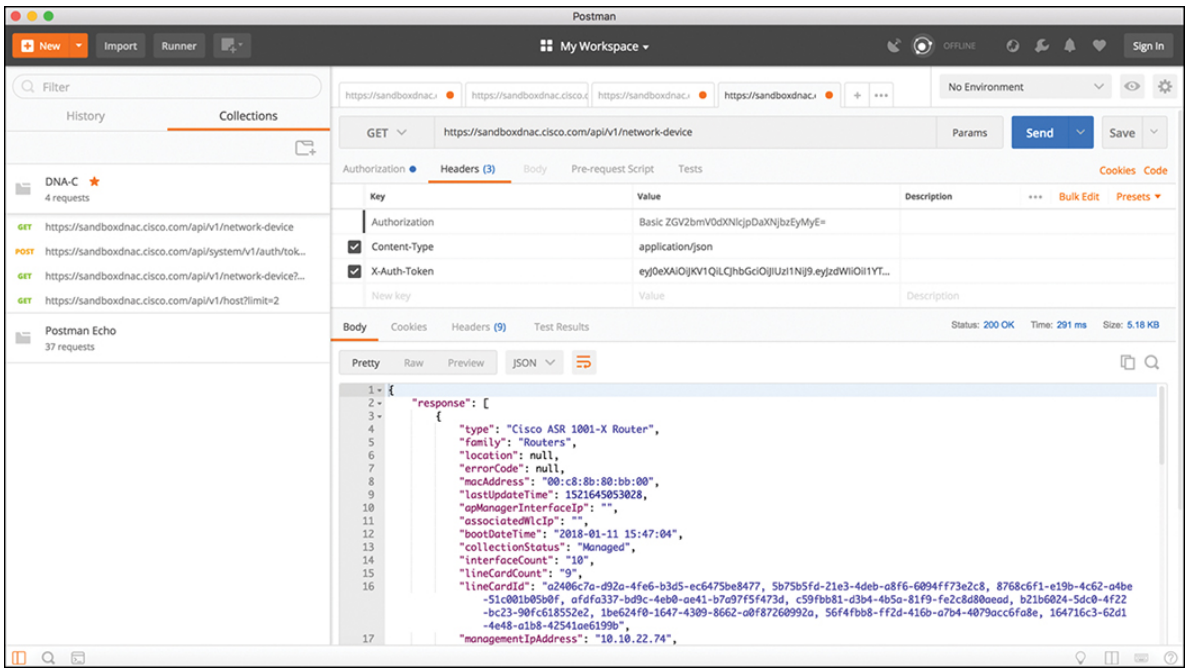


Figure 3-10 *Postman Setup to Retrieve Network Device Inventory via an API Call*

> Note
>
> The token will be different from the one in this book. It is unique to each authenticated user.

Based on the response received from the DNA Center Controller, you can see that you not only received an HTTP Response code of 200 OK, you also successfully retrieved the device inventory. Example 3-4 shows a list of devices in the inventory that were

pulled via the Network Device API.

Example 3-4 *Device Inventory Pulled via the Network Device API Call in Google Postman*

Click here to view code image

```
{
    "response": [
        {
            "type": "Cisco ASR 1001-X Router",
            "family": "Routers",
            "location": null,
            "errorCode": null,
            "macAddress": "00:c8:8b:80:bb:00",
            "lastUpdateTime": 1521645053028,
            "apManagerInterfaceIp": "",
            "associatedWlcIp": "",
            "bootDateTime": "2018-01-11 15:47:04",
            "collectionStatus": "Managed",
            "interfaceCount": "10",
            "lineCardCount": "9",
            "lineCardId": "a2406c7a-d92a-4fe6-b3d5-ec6475be8477, 5b75b5fd-
                    21e3-4deb-a8f6-6094ff73e2c8, 8768c6f1-e19b-4c62-a4be-
                    afdfa337-bd9c-4eb0-ae41-b7a97f5f473d, c59fbb81-d3b4-4
                    fe2c8d80aead, b21b6024-5dc0-4f22-bc23-90fc618552e2, 1
                    1647-4309-8662-a0f87260992a, 56f4fbb8-ff2d-416b-a7b4-
                    164716c3-62d1-4e48-a1b8-42541ae6199b",
            "managementIpAddress": "10.10.22.74",
            "memorySize": "3956371104",
            "platformId": "ASR1001-X",
            "reachabilityFailureReason": "",
            "reachabilityStatus": "Reachable",
            "series": "Cisco ASR 1000 Series Aggregation Services Routers"
            "snmpContact": "",
            "snmpLocation": "",
            "tunnelUdpPort": null,
            "waasDeviceMode": null,
            "locationName": null,
            "role": "BORDER ROUTER",
            "hostname": "asr1001-x.abc.inc",
            "upTime": "68 days, 23:23:31.43",
            "inventoryStatusDetail": "<status><general code=\"SUCCESS\"/><
            "softwareVersion": "16.6.1",
            "roleSource": "AUTO",
            "softwareType": "IOS-XE",
            "collectionInterval": "Global Default",
            "lastUpdated": "2018-03-21 15:10:53",
            "tagCount": "0",
            "errorDescription": null,
            "serialNumber": "FXS1932Q1SE",
```

```
            "instanceUuid": "d5bbb4a9-a14d-4347-9546-89286e9f30d4",
            "id": "d5bbb4a9-a14d-4347-9546-89286e9f30d4"

        },
        {
            "type": "Cisco Catalyst 9300 Switch",
            "family": "Switches and Hubs",
            "location": null,
            "errorCode": null,
            "macAddress": "f8:7b:20:67:62:80",
            "lastUpdateTime": 1521644291747,
            "apManagerInterfaceIp": "",
            "associatedWlcIp": "",
            "bootDateTime": "2018-01-11 14:42:33",
            "collectionStatus": "Managed",
            "interfaceCount": "41",
            "lineCardCount": "2",
            "lineCardId": "feb42c9f-323f-4e17-87d3-c2ea924320cb, 0f0c473e-
               4dcf-af11-9e7cf7216473",
            "managementIpAddress": "10.10.22.66",
            "memorySize": "889225360",
            "platformId": "C9300-24UX",
            "reachabilityFailureReason": "",
            "reachabilityStatus": "Reachable",
            "series": "Cisco Catalyst 9300 Series Switches",
            "snmpContact": "",
            "snmpLocation": "",
            "tunnelUdpPort": null,
            "waasDeviceMode": null,
            "locationName": null,
            "role"  : "ACCESS",
            "hostname":   "cat_9k_1.abc.inc",
            "upTime": "69 days, 0:15:51.44",
            "inventoryStatusDetail": "<status><general code=\"SUCCESS\"/><
            "softwareVersion": "16.6.1",
            "roleSource": "AUTO",
            "softwareType": "IOS-XE",
            "collectionInterval": "Global Default",
            "lastUpdated": "2018-03-21 14:58:11",
            "tagCount": "0",
            "errorDescription": null,
            "serialNumber": "FCW2136L0AK",
            "instanceUuid": "6d3eaa5d-bb39-4cc4-8881-4a2b2668d2dc",
            "id": "6d3eaa5d-bb39-4cc4-8881-4a2b2668d2dc"
        },
        {
            "type": "Cisco Catalyst 9300 Switch",
            "family":            "Switches and Hubs",
            "location": null,
            "errorCode":  null,
            "macAddress": "f8:7b:20:71:4d:80",
            "lastUpdateTime": 1521644755520,
            "apManagerInterfaceIp": "",
            "associatedWlcIp": "",
```

```
            "bootDateTime": "2018-01-11 14:43:33",
            "collectionStatus": "Managed",
            "interfaceCount": "41",
            "lineCardCount": "2",
            "lineCardId": "789e00f9-f52d-453d-86c0-b18f642462ee, 242debfd-
               4147-9bf6-574e488c5174",
            "managementIpAddress": "10.10.22.70",
            "memorySize": "889225280",
            "platformId": "C9300-24UX",
            "reachabilityFailureReason": "",
            "reachabilityStatus": "Reachable",
            "series": "Cisco Catalyst 9300 Series Switches",
            "snmpContact": "",
            "snmpLocation": "",
            "tunnelUdpPort": null,
            "waasDeviceMode": null,
            "locationName": null,
            "role"  : "ACCESS",
            "hostname":   "cat_9k_2.abc.inc",
            "upTime": "69 days, 0:22:17.07",
            "inventoryStatusDetail": "<status><general code=\"SUCCESS\"/><
            "softwareVersion": "16.6.1",
            "roleSource": "AUTO",
            "softwareType": "IOS-XE",
            "collectionInterval": "Global Default",
            "lastUpdated": "2018-03-21 15:05:55",
            "tagCount": "0",
            "errorDescription":   null,
            "serialNumber": "FCW2140L039",
            "instanceUuid": "74b69532-5dc3-45a1-a0dd-6d1d10051f27",
            "id": "74b69532-5dc3-45a1-a0dd-6d1d10051f27"
        },
        {
            "type": "Cisco Catalyst38xx stack-able ethernet switch",
            "family": "Switches and Hubs",
            "location": null,
            "errorCode":  null,
            "macAddress": "cc:d8:c1:15:d2:80",
            "lastUpdateTime": 1521644825918,
            "apManagerInterfaceIp": "",
            "associatedWlcIp": "",
            "bootDateTime": "2018-01-11 15:20:34",
            "collectionStatus": "Managed",
            "interfaceCount": "59",
            "lineCardCount": "2",
            "lineCardId": "15d76413-5289-4a99-98b6-fcacfe76b977, f187f561-
               4f30-b1a1-c6c6284bd075",
            "managementIpAddress": "10.10.22.69",
            "memorySize": "873744896",
            "platformId": "WS-C3850-48U-E",
            "reachabilityFailureReason": "",
            "reachabilityStatus": "Reachable",
            "series": "Cisco Catalyst 3850 Series Ethernet Stackable Switc
            "snmpContact": "",
```

```
            "snmpLocation": "",
            "tunnelUdpPort":   null,
            "waasDeviceMode": null,
            "locationName":   null,
            "role": "CORE",
            "hostname": "cs3850.abc.inc",
            "upTime": "65 days, 11:23:52.43",
            "inventoryStatusDetail": "<status><general code=\"SUCCESS\"/><
            "softwareVersion": "16.6.2s",
            "roleSource": "MANUAL",
            "softwareType": "IOS-XE",
            "collectionInterval": "Global Default",
            "lastUpdated": "2018-03-21 15:07:05",
            "tagCount": "0",
            "errorDescription": null,
            "serialNumber": "FOC1833X0AR",
            "instanceUuid": "8be78ab1-d684-49c1-8529-2b08e9c5a6d4",
            "id": "8be78ab1-d684-49c1-8529-2b08e9c5a6d4"
        }
    ],
    "version": "1.0"

}
```


Hopefully, you can start to see how powerful using APIs can be. Within a few moments, users are able to gather a tremendous amount of information about the devices currently being managed by the DNA Center Controller. One could even say that in the time it takes someone to log in to a device via the CLI and issue all the relevant **show** commands to gather this data, an API call can be used to gather it for the entire network within seconds. Talk about giving some time back to some engineers to do other things!

Manipulating data using filters and offset lists is very common with the use of APIs. A great example would be leveraging the Network Device API and only gathering the information on only a single device in the inventory. Perhaps it is the second device in the inventory. This is where the API documentation becomes so valuable. Most APIs that are created are documented as to what they can be used to accomplish.

In Postman, it is possible to modify the Network Device API URL and add **?limit=1** to the end of the URL to show only a single device in the inventory. We can also add the **&offset=2** command to the end of the URL to state that we want to see the second device in the inventory. Although it may seem confusing, the **limit** keyword simply states that a user only wants to retrieve one record from the inventory, and the **offset** command states that the user only wants that record to be the second record in the inventory. Figure 3-11 shows how a user can adjust the Network Device API URL in

Postman to show the second device in the inventory. Users can specify a filter that states that they only want to see a single device in the inventory (**?limit=1**) and to show the second device in the inventory (**&offset=2**).

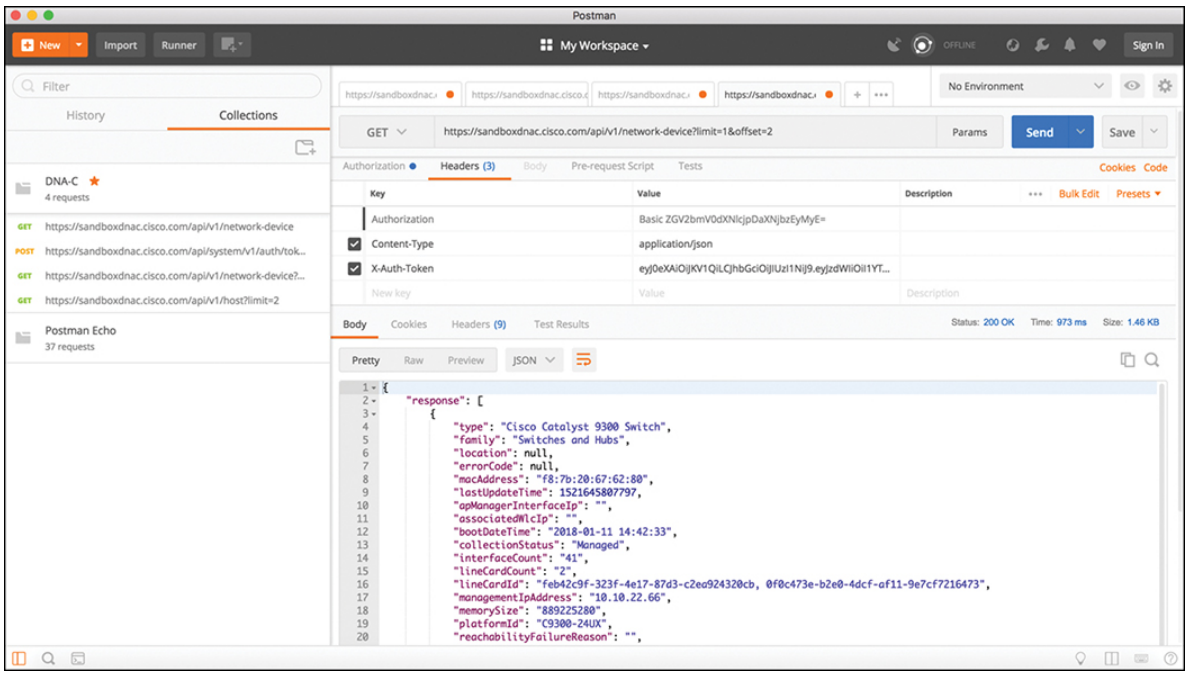


Figure 3-11 *Filter Output of Network Device Inventory*

Based on the response, you can see that the second device is consistent with the output you saw in the initial Network Device API call that was shown in Example 3-4. That device is a "Cisco Catalyst 9300 Switch" with the MAC address of f8:7b:20:67:62:80.

DATA MODELS AND SUPPORTING PROTOCOLS

This section covers some of the most common data models and tools and how they are leveraged in a programmatic approach. The data models and tools covered are as follows:

- Yet Another Next Generation (YANG) modeling language
- Network Configuration Protocol (NETCONF)
- ConfD

YANG Data Models

The Simple Network Management Protocol (SNMP) is widely used for fault handling and monitoring. However, it is not often used for configuration changes. CLI scripting is used more often than other methods. YANG data models are an alternative to SNMP Management Information Base (MIBs) and are becoming the standard for data definition languages. YANG was defined in RFC 6020 and uses data models. Data models are used to describe whatever can be configured on a device, everything that can be monitored on a device, and all the administrative actions that can be executed on a device, such as resetting counters and rebooting the device. This includes all the notifications that the device is capable of generating. All of these variables can be represented within a YANG model. Data models are very powerful in that they create a uniform way to describe data, which can be beneficial across vendors' platforms. Data models allow network operators to configure, monitor, and interact with network devices holistically across the entire enterprise environment.

YANG models use a tree structure. Within that structure, the models are constructed similar to the XML format and are built in modules. These modules are hierarchical in nature and contain all the different data and types that make up a YANG device model. YANG models make a clear distinction between configuration data and state information. The tree structure represents how to reach a specific element of the model. These elements can be either configurable or not configurable.

Elements all have a defined type. For example, an interface can be configured to be on or off. However, the interface state cannot be configured (for example, up or down). Example 3-5 illustrates a simple YANG module taken from RFC 6020.

Example 3-5 *YANG Model Example*

Click here to view code image

```
container food {
  choice snack {
      case sports-arena {
          leaf pretzel {
              type empty;
          }
          leaf beer {
              type empty;
          }
      }
      case late-night {
          leaf chocolate {
              type enumeration {
```

```
                        enum dark;
                        enum milk;
                        enum first-available;
                    }
                }
            }
        }
    }
```

The output in the previous example can be read as follows: You have food. Of that food, you have a choice of snacks. In the case that you are in the sports arena, your snack choices are pretzels and beer. If it is late at night, your snack choices are two different types of chocolate. You can choose to have milk chocolate or dark chocolate, and if you are in a hurry and do not want to wait, you can have the first available chocolate, whether it is milk chocolate or dark chocolate. To put this into more network-oriented terms, see Example 3-6.

Example 3-6 *Network-Oriented YANG Model*

Click here to view code image

```
    list interface {
        key "name";

        leaf name {
            type string;
        }
        leaf speed {
            type enumeration {
                enum 10m;
                enum 100m;
                enum auto;
            }
        }
        leaf observed-speed {
            type uint32;
            config false;
        }
    }
```

The YANG model shown here can be read as follows: You have a list of interfaces. Of the available interfaces, there is a specific interface that has three configurable speeds. Those speeds are 10Mbps, 100Mbps, and auto, as listed in the leaf named “speed.” The leaf named “observed-speed” cannot be configured due to the **config false** command. This is because as the leaf is named, the speeds in this leaf are those that were auto-detected (observed); hence, it is not a configurable leaf. This is because it represents the

auto-detected value on the interface, not a configurable value.

NETCONF

NETCONF was defined in RFC 4741 and RFC 6241. NETCONF is an IETF standard protocol that uses the YANG data models to communicate with the various devices on the network. NETCONF runs over SSH, TLS, or the Simple Object Access Protocol (SOAP). Some of the key differences between SNMP and NETCONF are listed in Table 3-5. One of the most important differences is that SNMP can't distinguish between configuration data and operational data, but NETCONF can. Another key differentiator is that NETCONF uses paths to describe resources, whereas SNMP uses object identifiers (OIDs). A NETCONF path can be similar to interfaces/interface/eth0, which is much more descriptive than what you would expect out of SNMP. Here is a list of some of the more common use cases for NETCONF:

- Collecting the status of specific fields
- Changing the configuration of specific fields
- Taking administrative actions
- Sending event notifications
- Backing up and restoring configurations
- Testing configurations before finalizing the transaction

Table 3-5 *Differences Between SNMP and NETCONF*

	SNMP	NETCONF
Resources	OIDs (Object Identifier)	Paths
Data Models	Defined in MIBs	YANG Core Models
Data Modeling	SMI (Structure of Management	YANG

Language	Information)	
Management Operations	SNMP	NETCONF
Encoding	BER (Bit Error Rate)	XML, JSON
Transport Stack	UDP	SSH/TCP

Transactions are all or nothing. There is no order of operations or sequencing within a transaction. That means it doesn't matter the order in which the configurations are done. It is completely arbitrary. Transactions are processed in the same order every time on every device. Transactions, when deployed, run in a parallel state and do not have any impact on each other.

Example 3-7 illustrates a NETCONF element from RFC 4741. This NETCONF output can be read as follows: There is an XML list of users named "users." In that list, there are the following individual users: root, Fred, and Barney.

> Note
>
> XML will be covered later in this chapter.

Example 3-7 *Example NETCONF Element*

Click here to view code image

```
<rpc-reply message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <data>
       <top xmlns="http://example.com/schema/1.2/config">
         <users>
          <user>
             <name>root</name>
```

```
            </user>
            <user>
              <name>fred</name>
            </user>
            <user>
              <name>barney</name>
            </user>
          </users>
        </top>
      </data>
    </rpc-reply>
```

An alternate way of looking at this type of NETCONF output is as a shopping list. Example 3-8 illustrates the shopping list concept and can be read as follows: You have a group of beverages. Of these beverages, you have soft drinks and tea. The available soft drinks are cola and root beer. Of the available teas, you have sweetened and unsweetened.

Example 3-8 *Shopping List NETCONF Example*

```
Beverages
    Soft Drinks
        Cola
        Root Beer
    Tea
        Sweetened
        Unsweetened
```

ConfD

Network operators today are looking for a way to manage their networks holistically, rather than managing them on a box-by-box basis. To do this, many network devices have northbound APIs that allow a management tool or suite of tools to interact with the devices across the network. This allows for service applications and device setup to be done uniformly across the campus environment. This type of automation introduces the concept of a transactional deployment model.

In the transactional deployment model, network operators deploy services or configurations to devices in an all-or-nothing fashion. For example, if a network operator was deploying Quality of Service (QoS) on a box-by-box basis, many configuration steps would have to be completed for QoS to function properly. Class maps must classify and match traffic, policy maps must be configured and applied to interfaces, and so on. If one of these steps is missed, QoS will not function properly. With a transactional model, you can deploy all the steps needed to enable QoS on all

devices simultaneously, and if for some reason the entire configuration isn't fully deployed, it can be rolled back entirely.

This is a much cleaner and less error-prone way to deploy certain features. In the transactional model, there wouldn't be any partially configured features resident in the campus devices, which ensures the integrity of the network as a whole. This also functionally makes the network act as a federated database rather than as a collection of separated devices.

Service applications can include things like VPN provisioning, QoS, firewall capabilities, and so forth. Device setup components can include configuration templates, scripts, and other device-specific operations. ConfD is a device-management framework that is very different from the traditional management tools. ConfD uses YANG data models to interact with various network devices. It can also use NETCONF, among other things, as a protocol to carry the different transactions to the equipment to be executed.

Once you have a YANG model for a device, ConfD automatically renders all the management protocols mentioned earlier. For example, you will automatically have a WEB UI into the YANG model of the device without having to program anything. The YANG model also supplies the configuration database (CBD, covered in the next section) schema, so the structure of the fields is taken from the YANG model, as well. Table 3-6 compares some of the differences between ConfD and other traditional management tools.

Table 3-6 *Comparison Between ConfD and Traditional Management Tools*

Traditional Management Tools	ConfD
SNMP agents	Data-model driven (YANG)
CLI	Any protocol
Feature lag of management tool	Executes transactions for desired features

ConfD has many different management protocols that can be used northbound to manage the product. Some of these protocols are shown in Figure 3-12. Those management protocols are as follows:

- NETCONF
- SNMP v1, v2c, v3
- REST
- CLI
- WEB UI

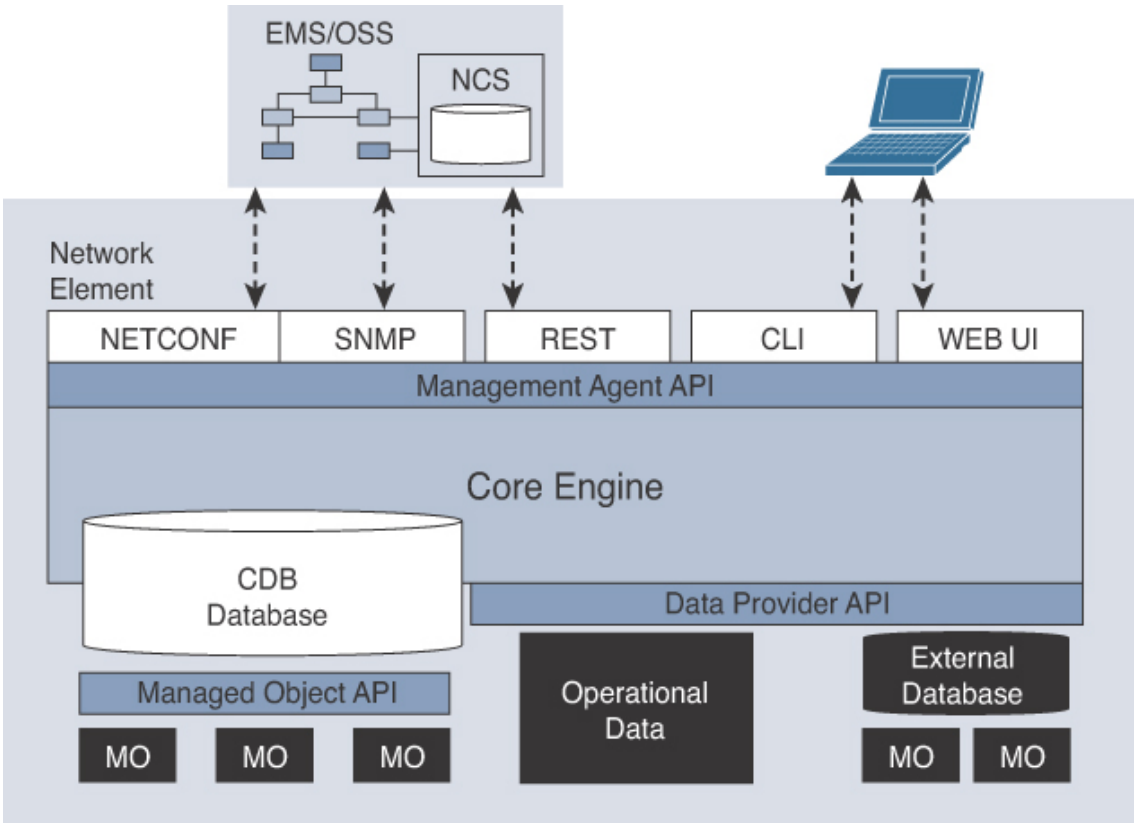


Figure 3-12 *ConfD Core Platform and Management APIs*

Some of the different components of ConfD are the Core Engine, CDB database, Managed Object API, and the Data Provider API. The CDB database is where the configurations are stored in the ConfD solution. The internal CDB database is optional, and network operators can choose to have external databases for their configurations or use both internal and external databases to fit their needs. This may make sense if you have a number of legacy applications already using an external database. Most often,

the internal CDB database is used.

> Note
>
> You can also store alarms, performance data, internal state, and other items in the CDB database.

Multiple components comprise the ConfD Core Engine, including the following:

- Transaction management
- Session management/authentication
- Role-based access control (RBAC)
- Redundancy/replication for HA
- Event logging/audit trails
- Validation syntax/semantic
- Rollback management

Upgrades and Downgrades

Once an operator has authenticated to ConfD, that operator is assigned one or more roles. These roles dictate not only what the operator can and cannot see, but what they can and cannot make changes to. These roles can be very granular in nature. This gives an organization the flexibility to have multiple tiers of operators that have job-specific functions tied to their roles. The audit-trailing capabilities of ConfD allow for robust reporting on which operator configured what, the commands that were executed, where they were executed, and when. This is especially important when you have multiple operators responsible in a campus environment. The validation component of ConfD is one of the most important. It can validate that the syntax of the commands is indeed valid before the commands are sent to the devices. More importantly, because of the YANG data models, ConfD can verify the validity of specific variables with other variables. That means if you need to make sure portions of the configuration are unique throughout your environment, ConfD can verify this across your whole environment (for example, it can check for duplicate IP addresses). Figure 3-13 illustrates how NETCONF uses YANG data models to interact with network devices and then talk back to management applications like ConfD.

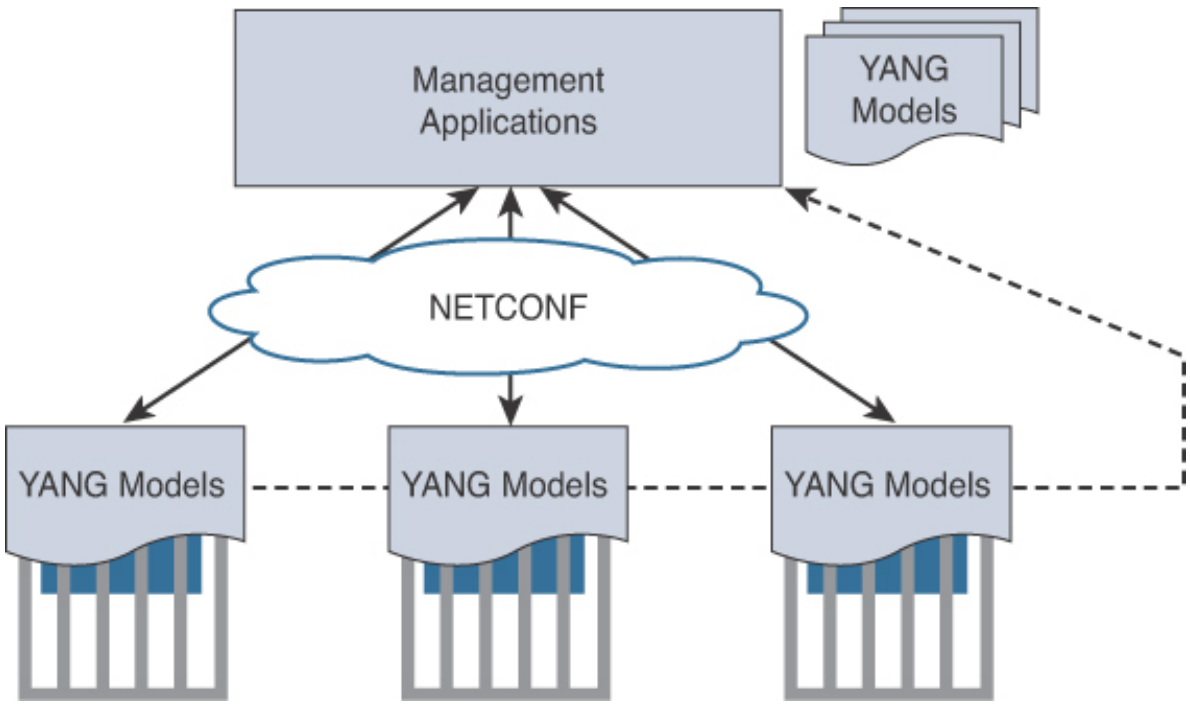


Figure 3-13 *NETCONF/YANG Interfacing with Management Applications*

The rollback management function is another very useful component of ConfD. For instance, after a network operator authenticates to the ConfD system and gets a role assigned to them, the rollback management component will track all the transactions made by that operator within their current session and create rollback files in the event that the transactions need to be reverted back to a previous state. This creates a timeline of transactions that allows the operator to pick a point in time to roll back to.

ConfD is designed to make your devices more manageable, programmable, and compliant with the industry standards. ConfD enables multiple ways to interact with your campus devices, such as NETCONF, SNMP, REST APIs, CLI, and web user interfaces. A key benefit of ConfD is its ability to hide network complexity and provide useful northbound interfaces that humans and applications can use. Using ConfD can save you time when interacting with devices because it is data model driven. All the interfaces of the data models are automatically rendered and require no programming by default. It has a self-contained database, although you can have external databases and a variety of feature-rich APIs.

DEVNET

The examples and tools discussed in this chapter are all available to use and practice at http://developer.cisco.com. This is the home for Cisco DevNet. DevNet is a single place to go when looking to enhance or increase your skills with APIs, coding, Python, and even controller concepts. In DevNet you will find learning labs and content that will help solidify your knowledge in network programmability. Regardless of whether you are just getting started or are a seasoned programmatic professional, DevNet is the place to be! In this section, DevNet will be covered from a high-level overview perspective, including the different sections of DevNet as well as some of the labs and content you can expect to interact with. Figure 3-14 shows the DevNet main page.

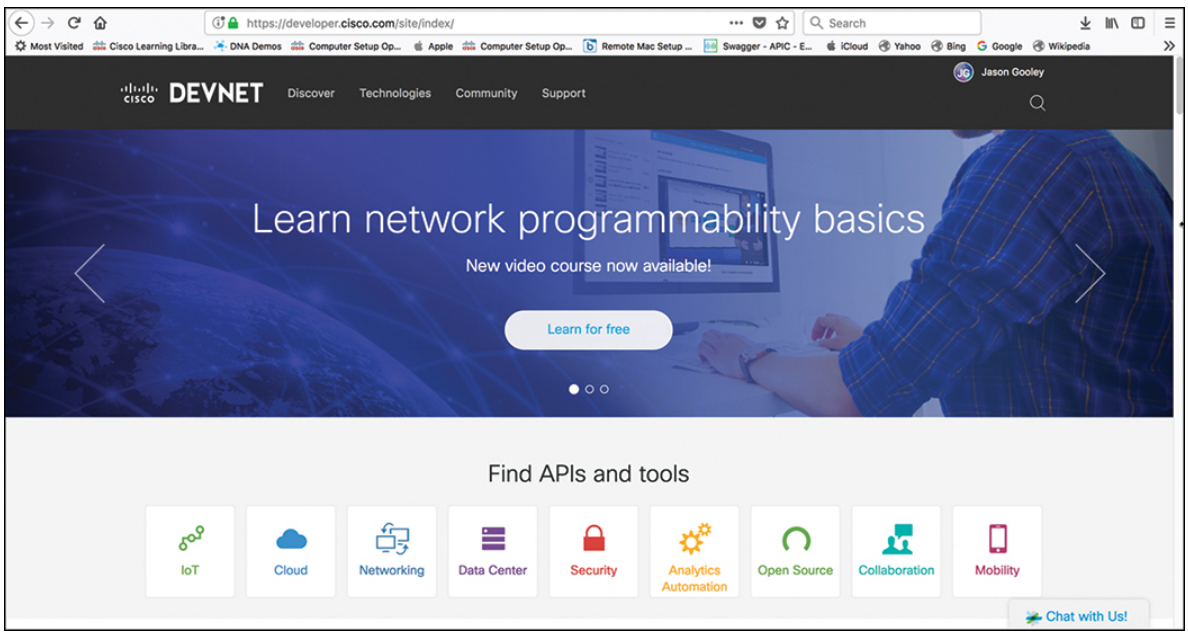


Figure 3-14 *DevNet Main Page*

Across the top of the main page you can see a few menu options. These menu options will be covered individually and are as follows:

- Discover

- Technologies
- Community
- Support

Discover

First is the Discover tab. This is where you navigate the different offerings DevNet has available. Under this tab you find subsections like guided learning tracks. These learning tracks guide you through various different technologies and the associated API labs. Some of the labs you can interact with are Programming the Digital Network Architecture (DNA), ACI Programmability, Getting Started with Cisco Spark APIs, and Introduction to DevNet, to name a few. Once you pick a learning lab and start the module, the website will track all your progress and allow you to continue where you left off. It's excellent for continuing your education over multiple days or weeks.

Technologies

The Technologies section allows you to pick relevant content based on the technology that you want to study and to dive directly into the associated labs and training. Figure 3-15 illustrates some of the networking content that is currently available.

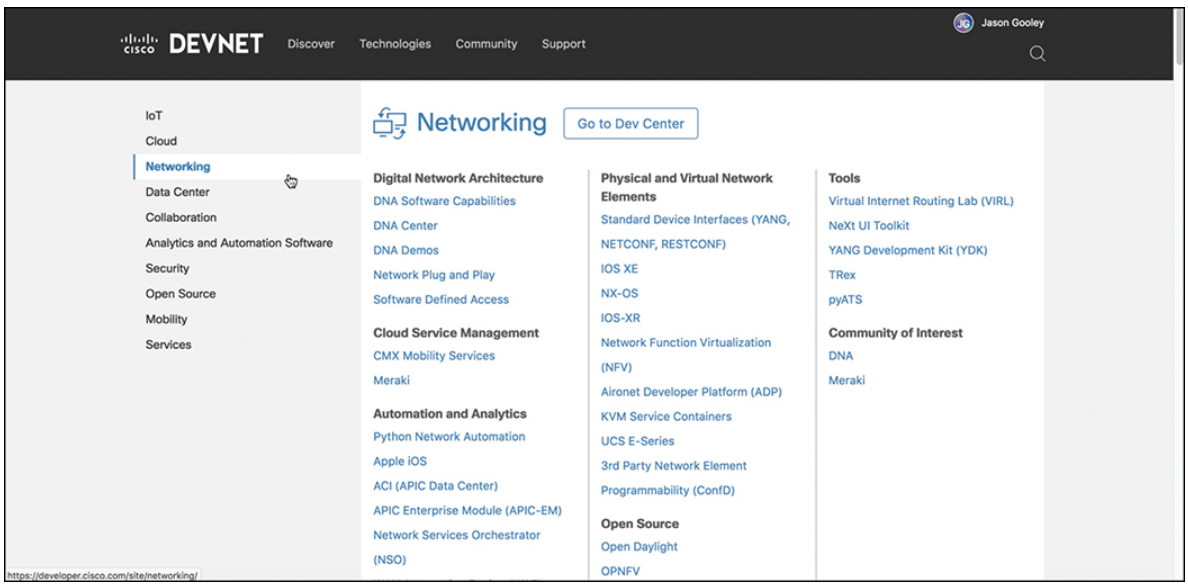


Figure 3-15 *DevNet Technologies Page*

> Note
>
> Available labs may differ from those shown in this chapter. Please visit developer. cisco.com to see the latest content available and to interact with the latest learning labs.

Community

Perhaps one of the most important sections of DevNet is Community, which is where you will have access to many different people at various stages of learning. You'll find DevNet ambassadors and evangelists there to help you at various stages of your learning journey. The Community section puts the latest events and news at your fingertips. This is also the place to read blogs, sign up for Developer Forums, and follow DevNet on all major social media platforms. This is the safe zone to ask any question, regardless of how simple or complex it may seem. Everyone has to start somewhere. The DevNet Community is the place to start for all things Cisco and network programmability. Figure 3-16 depicts some of the available options you have in the Community tab.

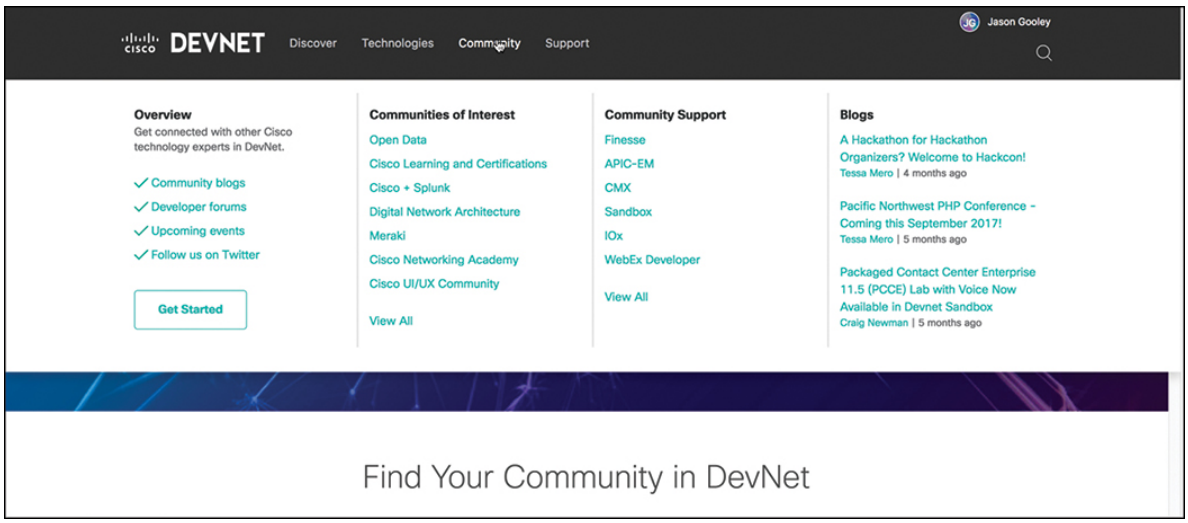


Figure 3-16 *DevNet Community Page*

Support

The Support section of DevNet is where you can post questions and get answers from some of the best in the industry. Technology-focused professionals are available to answer your questions both from a technical perspective and a theoretic perspective, meaning you can ask questions about specific labs or the overarching technology (for example, Python or YANG models). You can also open a case with the DevNet Support team and your questions will be tracked and answered within a minimal amount of time. This is a great place to ask one-on-one questions with the Support team as well as to tap into the expertise of the Support engineers. Figure 3-17 shows the DevNet Support page as well as where to open a Support case.

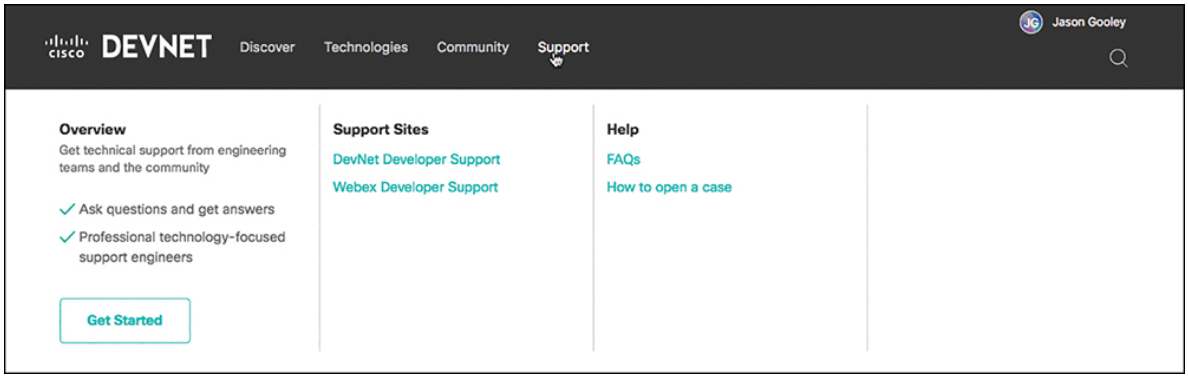


Figure 3-17 *DevNet Support Page*

CONTINUOUS INNOVATION AND CONTINUOUS DEPLOYMENT (CI/CD)

Oftentimes there is a stigma around automation and a hesitation to use it. It creates a fear that if businesses start deploying automation, engineers and developers will ultimately lose their jobs. This is simply not the case. Operations teams want to be able to streamline their efforts and alleviate the workload they have to face on a day-to-day basis. Ultimately, this will free up time for you to be able to help support your businesses plan for what to do for the next three to five years, as well as allow you to leverage the use of technology as a mechanism to meet the businesses strategic vision and be more agile.

Let's say, for example, your business is a software company. When developers work on creating and coding software, each development team is responsible for their own section or piece of the overall software suite or program. This means that there may be different teams of developers working on creating a specific portion of the software. Maybe they are creating a subset of features or developing a single use case into the software code. What ends up happening at some point is that all these developers have to consolidate all their separate versions of code into the overall codeset that makes up the software suite or program they are all developing. If each developer used different coding methods or error-checking methods, you could imagine that once all the code is put together into a single code set, it would likely be error-prone and might not function as desired. This is where Continuous Integration (CI) comes into play. If you were to use a common structure or guideline for every developer that includes error checking and testing, the codeset would have a higher likelihood of working the way the developers designed it to work. Continuous Integration automatically tests all of the code as it is loaded to ensure that any new changes to the code do not impact the overall codeset. Figure 3-18 illustrates an example of a common Continuous Integration workflow.

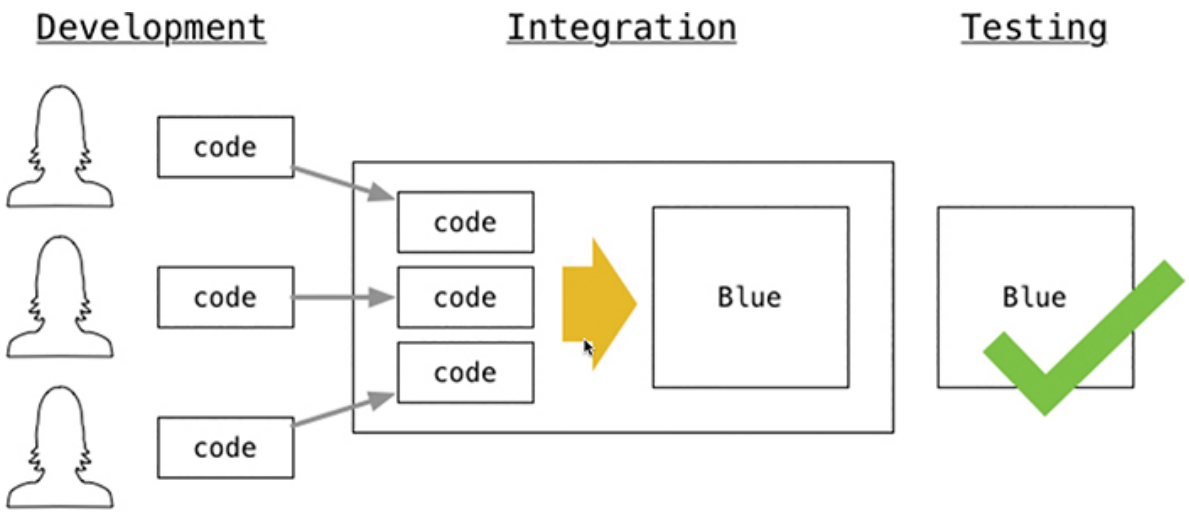


Figure 3-18 *Continuous Integration Workflow*

Once the changes in the code have been successfully tested, the code will need to be packaged up to be deployed to either a test development environment or production. This is where the code gets put into a container, along with any startup scripts, so that the software and any of its dependencies are all in the same location. This allows the developers to ensure that the code will launch and run consistently. Figure 3-19 depicts the process used once the developers' code has been consolidated and tested. This concept is called Continuous Packaging.

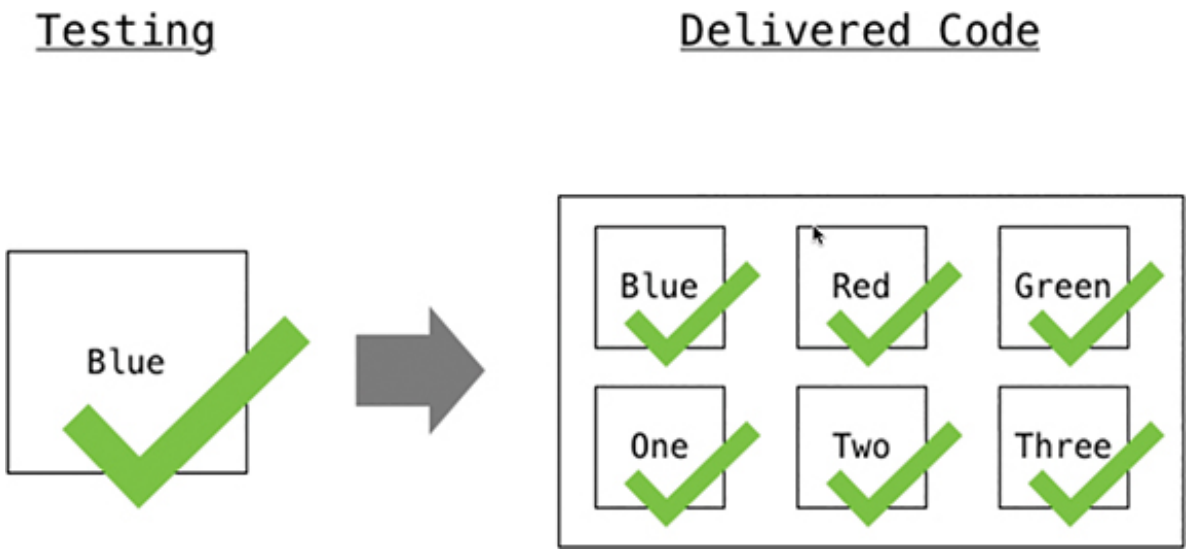


Figure 3-19 *Continuous Packaging Workflow*

Now that the developers' code has been packaged and put into a container, the package will need to be deployed, as mentioned earlier, to a development or production environment. Once this has been completed, the software will be available for people to use. The entire process that was just covered is called an automation pipeline. This pipeline gives us the ability to follow a structured process for software development. It starts with making sure all the various code components are integrated together and tested to ensure any changes did not impact the overall software solution. Then next step includes packaging up all the code components and scripts into a single location to ensure that the software has all the necessary pieces it requires to run. Finally, the

package is deployed to either a development environment or into production for users to be able to utilize the software. Figure 3-20 shows the final piece of the pipeline, which is Continuous Deployment (CD).

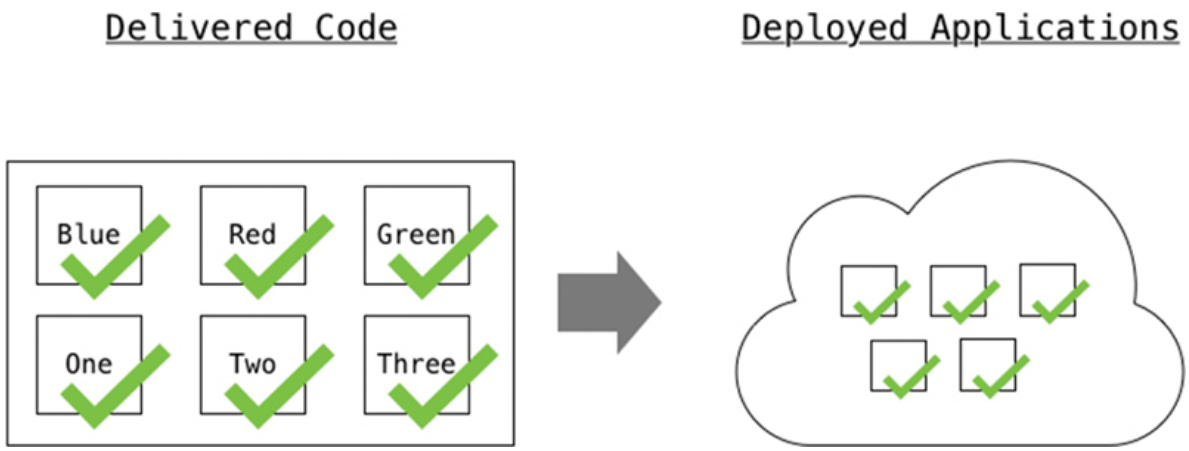


Figure 3-20 *Continuous Deployment Workflow*

Source Control Management

Now that we have discussed Continuous Integration and Continuous Deployment, the next topic to be covered is an important component that helps in making sure our software pipeline runs smooth and without issue. This component is called Source Control or Version Control Management. Source Control is a way to keep track of the various versions or revisions of code that are created, as well as any associated modifications. This allows developers to not only make sure they are all using the correct version of the code, but it also helps to ensure that if they needed to roll back to a previous version or revision of the code, they are more easily able to do so.

One of the most commonly used Source Control Management platforms is Git, which not only allows developers to have version control over their files, but also allows them to work cohesively with other developers on those files. This allows developers to accelerate working on software projects without sacrificing the integrity of the data. Git has two data structures that make it function. The first data structure is called an Index and is often referred to as the “cache.” This is because it caches the information of the working directory you are using for your files as well as the unsaved or uncommitted changes to those files in the working directory. The second data structure is called the Object Database, which contains four different types of objects. Table 3-7 lists these four different object types.

Table 3-7 *Object Types in the Object Database*

Object Type	Use Case

Blob (Binary Large Object)	Contains the contents of a file
Tree	Directory that contains filenames and references blobs or other tree objects
Commit	Links objects to create a history of timestamps, log messages, and names of objects
Tag	Used to store metadata that corresponds to a particular version or revision being tracked by Git

One of the most efficient and commonly adopted ways of using Git is to use GitHub. GitHub is a hosted web-based repository for Git. It has capabilities for bug tracking and task management as well. GitHub provides one of the easiest way to track changes on your files, collaborate with other developers, and share code with the online community. It is a great place to look for code to get started on the programmability journey. Oftentimes, other engineers or developers are trying to accomplish similar tasks and have already created and tested the necessary code. One of the most powerful features of using GitHub is the ability to rate and provide feedback on other developer's code. Peer review is encouraged when it comes to the coding community. Figure 3-21 shows the main GitHub web page.

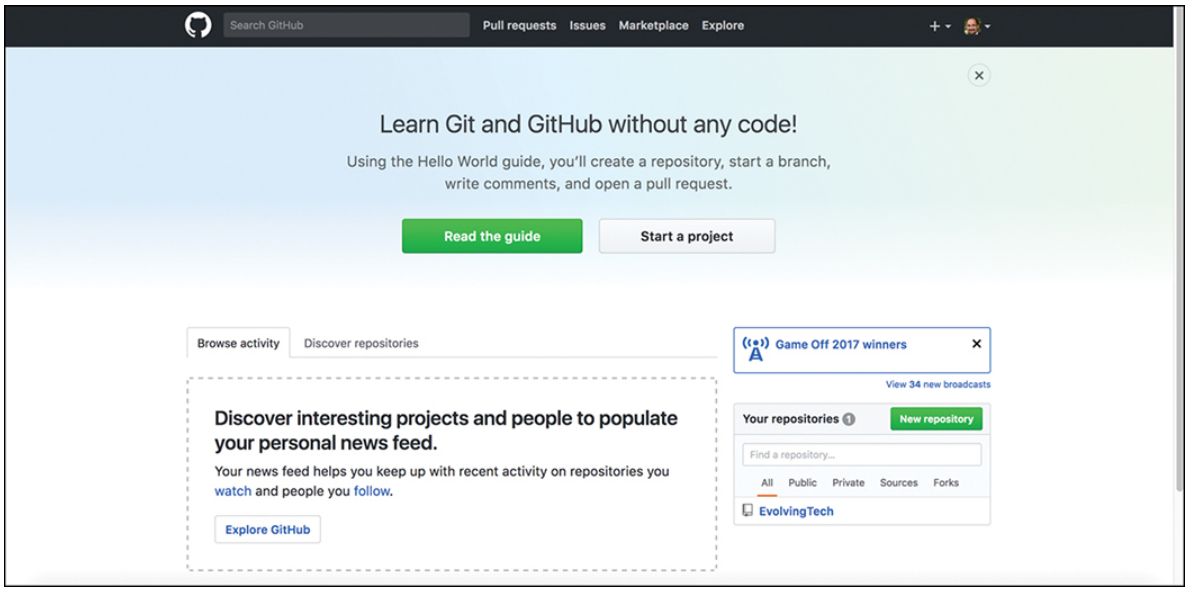


Figure 3-21 *GitHub Main Webpage*

Fortunately, a guide is available that steps you through how to create a repository, start a branch, add comments, and open a pull request. You can also just start a GitHub project after you are more familiar with the GitHub tool and its associated processes. Projects are repositories that contain code files. This is where you will have a single pane to create, edit, and share your code files. Figure 3-22 shows a repository called "EvolvingTech" that contains three files:

- EvolvingTech.txt
- JSON_Example.txt
- README.md

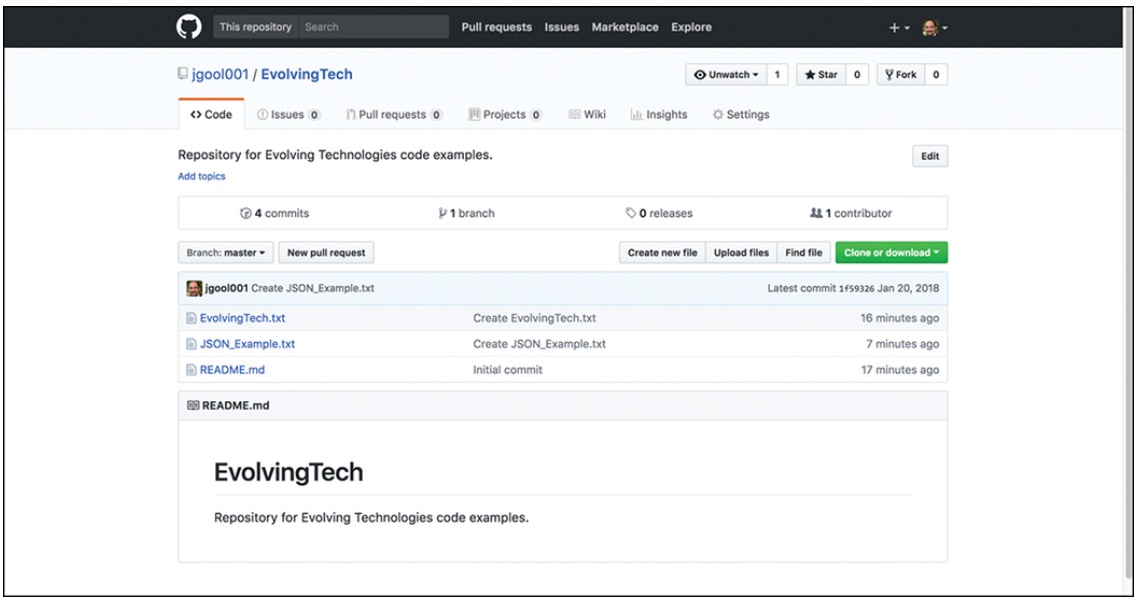


Figure 3-22 *GitHub EvolvingTech Repository*

GitHub also provides a great summary of commit logs, meaning when you save a change in one of your files or create a new file, it shows the details on the main

repository page. This can be seen in Figure 3-22 as well. By drilling down into one of the files in the repository, you can see how easy it is to edit and save your code. If you drill down into JSON_Example.txt, you will see its contents and how to exit the file in the repository. Once you click the filename JSON_Example.txt, you can see that the file has seven lines of code and is 76 bytes in size. Figure 3-23 shows the contents of the JSON_Example.txt file and the associated options for what you can do with the file.

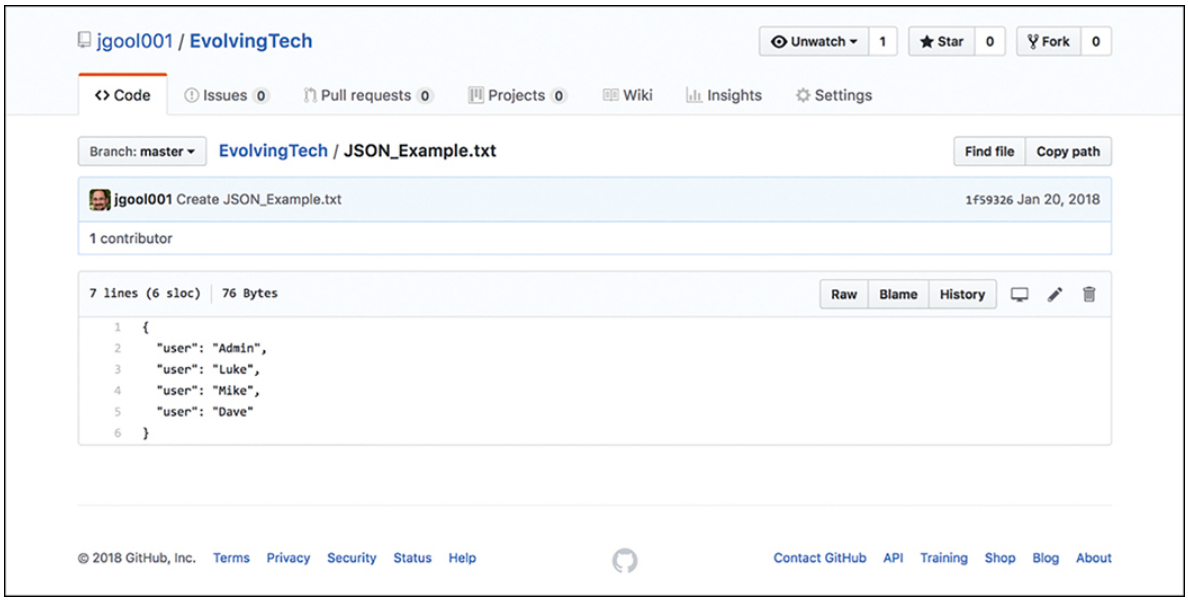


Figure 3-23 *JSON_Example.txt Contents*

The pencil icon allows you to go into editing mode so you can alter the file contents. This editor is very similar to any text editor. Developers can simply type into the editor or copy and paste code from other files directly into the editor. The example in Figure 3-24 shows the addition of another user named "Zuul." If we were to commit the changes, the file would be saved with the new user added to the file. Now that the file is available in the repository, other GitHub users and developers can contribute to the code, or add and delete lines of code based on the code we created. This is the true power of sharing your code. For example, if you have some code to add a new user via JSON syntax, someone could use that code and simply modify the usernames or add to the code to enhance it.

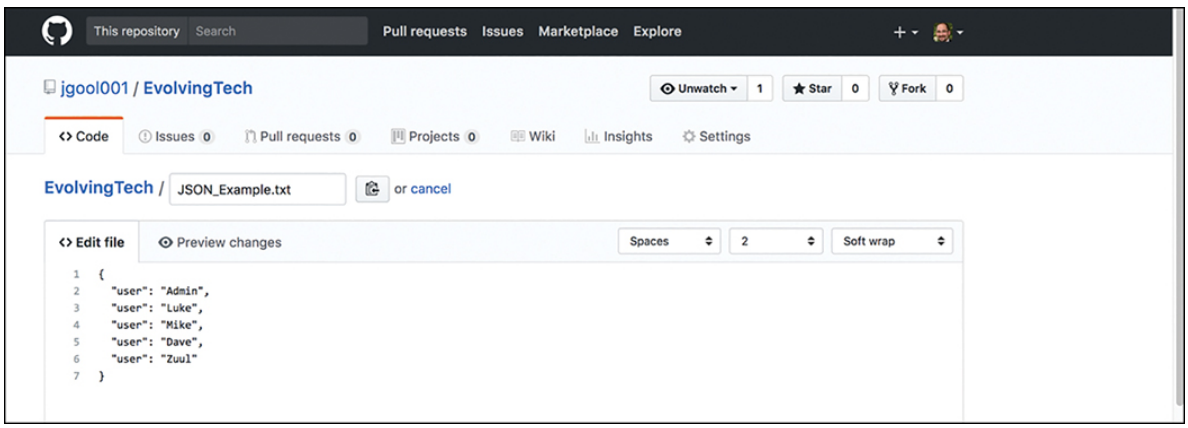


Figure 3-24 *Edit JSON_Example.txt Contents*

Ansible

Ansible is an automation tool that is capable of automating cloud provisioning, the deployment of applications, and configuration management. Ansible has been around for quite some time and was catapulted more into the mainstream when RedHat purchased the company back in 2015. Ansible has grown very popular due to the simplicity this product brings when deploying such services. Ansible is also open source as well as simple to use. Ansible was created with the following concepts in mind:

- Consistent
- Secure
- Highly reliable
- Minimal learning curve

Unlike many other automation tools, Ansible is an agentless tool. This means that no software or agent needs to be installed on the client machines to be managed. This is considered by some to be a major advantage of using Ansible versus other products. Ansible communicates via SSH for a majority of devices. It can also support WinRM and other transport methods to the clients it manages. Ansible also doesn't need an administrative account on the client. It can use built-in authorization escalation, like **sudo**, for example, when it needs to raise the level of administrative control. Ansible uses the concept of a "control station" to send all the requests from. This could simply be a laptop or a server sitting in a data center. It is quite literally the computer used to run Ansible and to issue the changes or requests from. This control station sends the requests to the remote hosts. Figure 3-25 illustrates the workflow.

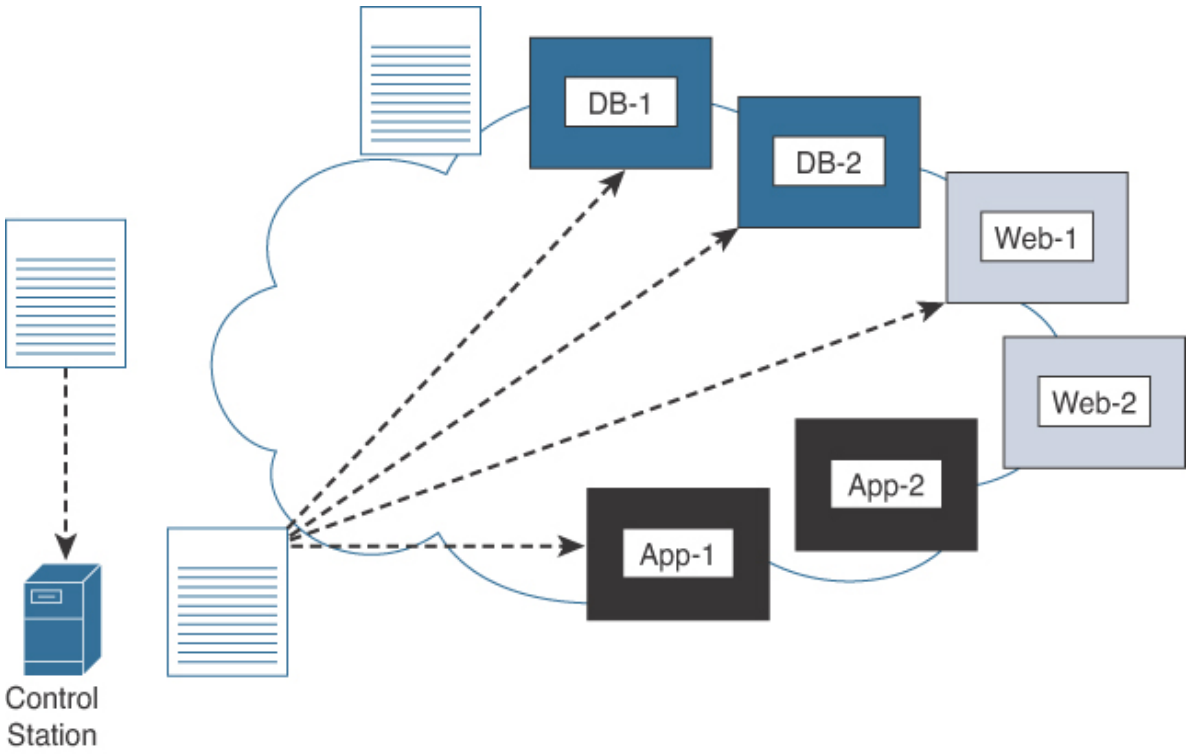


Figure 3-25 *Ansible Workflow*

These concepts are some of the reasons that administrators, developers, and IT managers seek to use Ansible. This allows for an easy ramp-up for users who aim to create new projects as well as sets the stage for a long-term automation initiatives and processes to further benefit the business. Previously we discussed the risk of human error and the impact it has on the business. Automation, by nature, reduces the risk of human error because we are taking known best practices that have been thoroughly tested in our environment and duplicating them in an automatic process. However, it is important to note that if a bad process or erroneous configuration is automated, it can be detrimental as well. When you're preparing to automate a task or set of tasks, it is important to start with the desired outcome of automating the task(s). Once a desired outcome has been documented, you can then move on to creating a plan to achieve the outcome. This process follows the PPDIOO (Prepare, Plan, Design, Implement, Observe, Optimize) methodology. Figure 3-26 outlines the PPDIOO lifecycle.

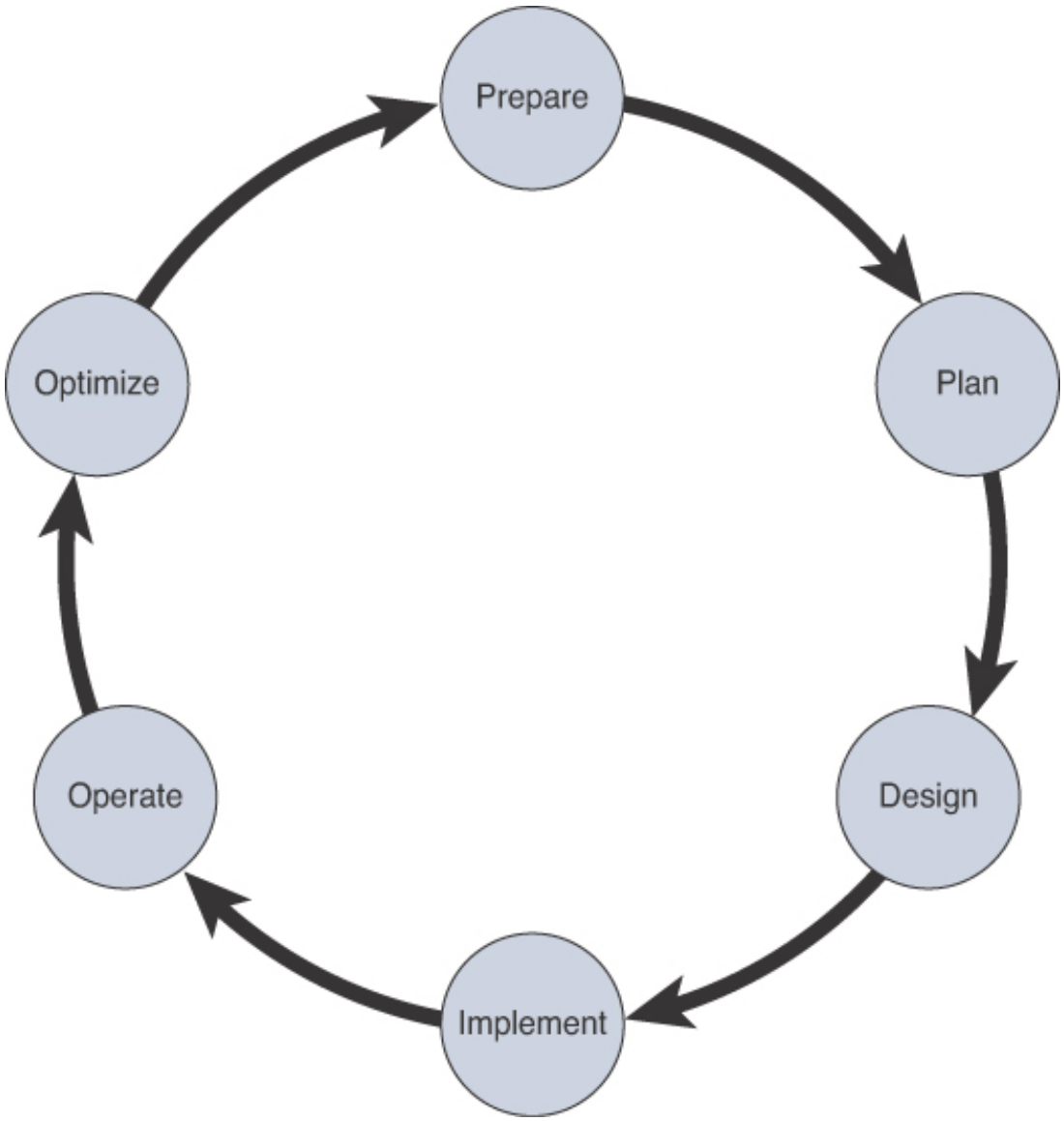


Figure 3-26 *PPDIOO Lifecycle*

Ansible uses a concept called “Playbooks” to deploy configuration changes or retrieve information from hosts within the network. You can think of Playbooks as structured sets of instructions. To use a sports analogy, it is very similar to what football players use to make different plays on the field during a game. For example, a Playbook contains multiple “plays,” and each play contains the “tasks” that each player must accomplish in order for that particular play to be successful. The same applies to Ansible Playbooks. Table 3-8 lists the structure of a Playbook and some commonly used examples of tasks.

Table 3-8 *Ansible Playbook Structure and Examples*

Components	Description	Use Case
Playbook	Set of plays for remote systems	Structure used by Ansible to enforce configuration and/or

		deployment steps.
Play	Ansible uses plays to group functions for single or multiple hosts	Grouping of a set of hosts to apply a policy or configuration to them.
Task	Call to an Ansible module	Log in to a device and issue a **show** command to retrieve an output.

Now that you know what the basic structure of a Playbook is, you need to further understand the language used to create Playbooks. Ansible Playbooks are written in YAML syntax. YAML stands for Yet Another Markup Language. Ansible YAML files usually begin with a series of three dashes (---) and end with a series of three periods (...). YAML files also contain lists and dictionaries. Example 3-9 illustrates a YAML file containing a list of musical genres.

Example 3-9 *YAML List Example*

```
---
# List of music genres
Music:
    - Metal
    - Rock
    - Rap
    - Country
…
```

YAML lists are very easy to read and consume. You can see based on the previous example that we have the ability to add descriptions to the YAML file by using the hash or pound sign (#) and adding text immediately following the it. You can also see the example of the starting --- and trailing …, which indicate the starting and end of the YAML file, respectively. More importantly, you can see that we started each line of the list with a dash and a space (-). Indentation is also important in YAML files.

YAML also has the concept of dictionaries. YAML dictionaries are very similar to JSON dictionaries because they also use key-value pairs. Remember from earlier in this

chapter that key-value pairs are represented by “key: value”. Example 3-10 depicts a YAML dictionary containing an example employee record.

Example 3-10 *YAML Dictionary Example*

Click here to view code image

```
---
# HR Employee record
Employee1:
    Name: John Smith
    Title: Network Architect
    Nickname: D'Bug
```

Lists and dictionaries can be used together as well in YAML. Example 3-11 shows a dictionary with a list in a single YAML file.

Example 3-11 *YAML Dictionary and List Example*

Click here to view code image

```
---
# HR Employee records
-  Employee1:
    Name: John Dough
    Title: Developer
    Nickname: Mr. D'Bug
    Skills:
      - Python
      - YAML
      - JSON
-  Employee2:
    Name: Jane Dough
    Title: Network Architect
    Nickname: Lay D'Bug
    Skills:
      - CLI
      - Security
      - Automation
```

YAML Lint is a free online tool used to check the format of YAML files to make sure they are in a valid syntax. Simply go to www.yamllint.com, paste the contents of a YAML file into the interpreter, and click Go. This will tell you if you have an error in your YAML file. Figure 3-27 shows the same YAML dictionary and list example, but it cleans up the formatting and removes the description line with the # sign.

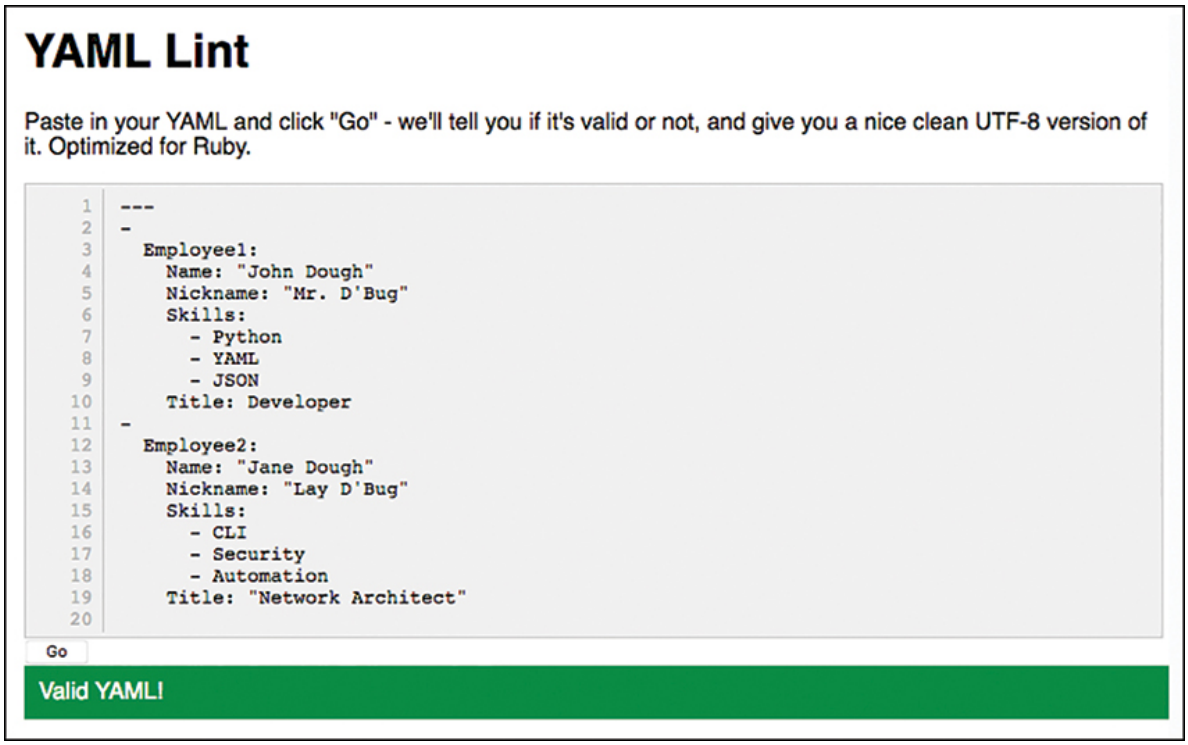


Figure 3-27 *YAML Lint Example*

Ansible has a CLI tool that can be used to run Playbooks or ad-hoc CLI commands on targeted hosts. This CLI tool has very specific commands needed in order to enable automation. The most common Ansible CLI commands are shown in Table 3-9 with their associated use cases.

Table 3-9 *Ansible CLI Commands*

CLI Command	Use Case
ansible	Runs modules against targeted hosts, ad-hoc CLI
ansible-playbook	Runs Playbooks
ansible-docs	Provides documentation on syntax and parameters via CLI
ansible-pull	Changes Ansible clients to “pull” modules from the control

	station versus the default “push” model
ansible-vault	Used to encrypt YAML files that contain sensitive data

Ansible uses an inventory file to keep track of the hosts it manages. The inventory can be a named group of hosts or a simple list of individual hosts. Hosts can belong to multiple groups and can be represented by either an IP address or a resolvable DNS name. Example 3-12 shows the contents of a host inventory file with the host 192.168.10.1 in two different groups.

Example 3-12 *Ansible Host Inventory File*

```
[routers]
192.168.10.1
192.168.20.1

[switches]
192.168.10.25
192.168.10.26

[primary-gateway]
192.168.10.1
```

Now that we have seen the fundamental concepts of Ansible and YAML, it is time to cover some useful examples. This section illustrates some basic examples of Ansible Playbooks used to accomplish common tasks. Imagine using a Playbook to deploy an interface configuration on a device without having to manually configure it. Perhaps taking this idea a step further, you could use a Playbook to configure an interface and deploy an EIGRP routing process. Example 3-13 illustrates the contents of an Ansible Playbook called ConfigureInterface.yaml. This Playbook will be used to configure the GigabitEthernet2 interface on a CSR1000V router. Leveraging the ios_config Ansible module, this Playbook will add the following configuration to the GigabitEthernet2 interface on the CSR1KV-1 router:

- description Configured by ANSIBLE!!!
- ip address 10.1.1.1

- subnet mask 255.255.255.0
- no shutdown

Example 3-13 *Ansible ConfigureInterface.yaml Playbook*

Click here to view code image

```
---
- hosts: CSR1KV-1

  gather_facts: false
  connection: local

  tasks:
   - name: Configure GigabitEthernet2 Interface
     ios_config:
       lines:
         - description Configured by ANSIBLE!!!
         - ip address 10.1.1.1 255.255.255.0
         - no shutdown
       parents: interface GigabitEthernet2

       host: "{{ ansible_host }}"
       username: cisco
       password: testtest
```

In order to execute this Playbook, the **ansible-playbook** command will be used to call the specific Playbook YAML file. In this case, the file is called ConfigureInterface. yaml. Figure 3-28 depicts the output from calling the Playbook from the Linux shell. The important things to note from the output are the PLAY, TASK, and PLAY RECAP sections. These sections list the name of the play and each individual task that gets executed in each play. The PLAY RECAP section lists the status of the Playbook that was executed. The output in Figure 3-28 shows that one play named CSR1KV-1 was launched, then a task called "Configure GigabitEthernet2 Interface." Based on the status of **ok=1**, the change was successful, and the **changed=1** status means that a single change was made on the CSR1KV-1 router.

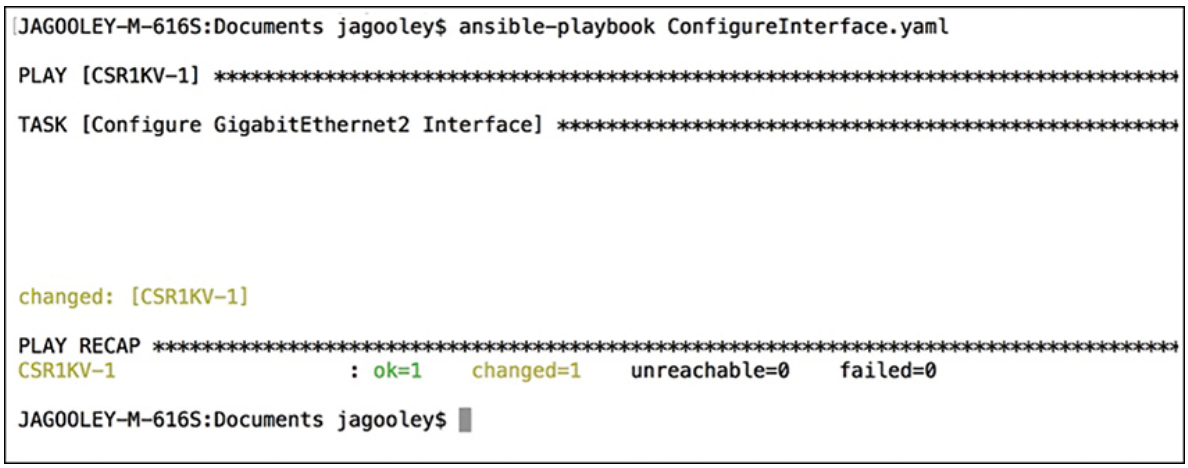


Figure 3-28 *Executing the ConfigureInterface.yaml Playbook*

Building out a Playbook can greatly simplify configuration tasks. Example 3-14 shows an alternate version of the ConfigureInterface.yaml Playbook named EIGRP_Configuration_Example.yaml, where EIGRP is added along with the ability to save the configuration by issuing a "write memory." These tasks are accomplished by leveraging the ios_command module in Ansible. This Playbook will add the following configuration to the CSR1KV-1 router:

On GigbitEthernet2:

- description Configured by ANSIBLE!!!
- ip address 10.1.1.1
- subnet mask 255.255.255.0
- no shutdown

On GigbitEthernet3:

- description Configured by ANSIBLE!!!
- no ip address
- shutdown

Global configuration:

- router eigrp 100
- eigrp router-id 1.1.1.1
- no auto-summary

- network 10.1.1.0 0.0.0.255

Save configuration:

- write memory

Example 3-14 *Ansible EIGRP_Configuration_Example.yaml Playbook*

Click here to view code image

```
---
- hosts: CSR1KV-1

  gather_facts: false
  connection: local

  tasks:
   - name: Configure GigabitEthernet2 Interface
     ios_config:
       lines:
         - description Configured by ANSIBLE!!!
         - ip address 10.1.1.1 255.255.255.0
         - no shutdown
       parents: interface GigabitEthernet2

       host: "{{ ansible_host }}"
       username: cisco
       password: testtest

   - name: CONFIG Gig3
     ios_config:
       lines:
         - description Configured By ANSIBLE!!!
         - no ip address
         - shut
       parents: interface GigabitEthernet3

       host: "{{ ansible_host }}"
       username: cisco
       password: testtest

   - name: CONFIG EIGRP 100
     ios_config:
       lines:
         - router eigrp 100
         - eigrp router-id 1.1.1.1
         - no auto-summary
         - network 10.1.1.0 0.0.0.255
      host: "{{ ansible_host }}"
       username: cisco
       password: testtest
```

```
    - name: WR MEM
      ios_command:
        commands:
          - write memory

        host: "{{ ansible_host }}"
        username: cisco
        password: testtest
```

Once the Playbook is run, the output will show the tasks as they are completed and their associated status. Based on the output in Figure 3-29, we can see that tasks with the following names are completed and return a status of "changed":

- Configure GigabitEthernet 2 Interface
- CONFIG Gig3
- CONFIG EIGRP 100

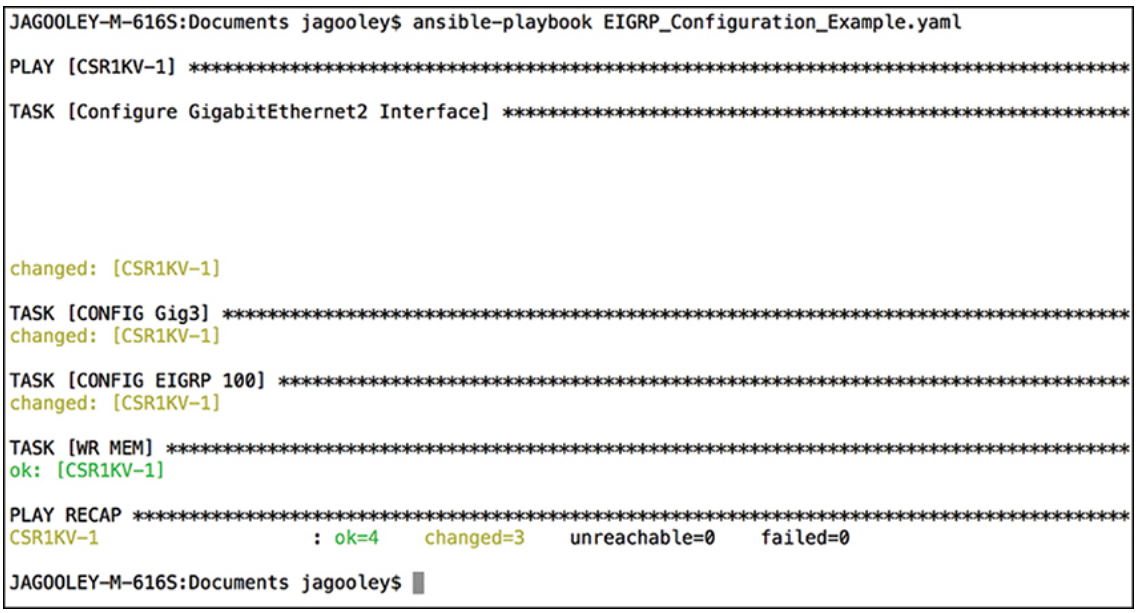


Figure 3-29 *Executing EIGRP_Configuration_Example.yaml Playbook*

Furthermore, the WR MEM task completes as well, which is evident by the "ok: [CSR1KV-1]" status. At the bottom of the output, we see in the PLAY RECAP section that we have a status of ok=4 and changed=3. This means that out of the four tasks, three actually modified the router and made a configuration change, while one task saved the configuration after it was modified.

Now that the EIGRP_Configuration_Example.yaml Playbook has been run against CSR1KV-1, it is best to verify the configuration to make certain that it was correctly applied. Example 3-15 lists the relevant sections of the startup configuration from

CSR1KV-1 to verify the tasks that were applied to the router.

Example 3-15 *Relevant Startup-config Post Running Playbook*

Click here to view code image

```
!
interface GigabitEthernet1
 ip address 172.16.38.101 255.255.255.0
 negotiation auto
 no mop enabled
 no mop sysid
!
interface GigabitEthernet2
 description Configured by ANSIBLE!!!
 ip address 10.1.1.1 255.255.255.0
 negotiation auto
!
interface GigabitEthernet3
 description Configured By ANSIBLE!!!
 no ip address
 shutdown
 negotiation auto
!
router eigrp 100
 network 10.1.1.0 0.0.0.255
 eigrp router-id 1.1.1.1
!
```

Since the last task in the Playbook is to issue the "write memory" command, verification involves quite simply issuing the **show startup-config** command with some filters to see the relevant configuration on the CSR1KV-1 router. Figure 3-30 illustrates the output from the **show startup-config | se GigabithEthernet2|net3|router eigrp 100** command.

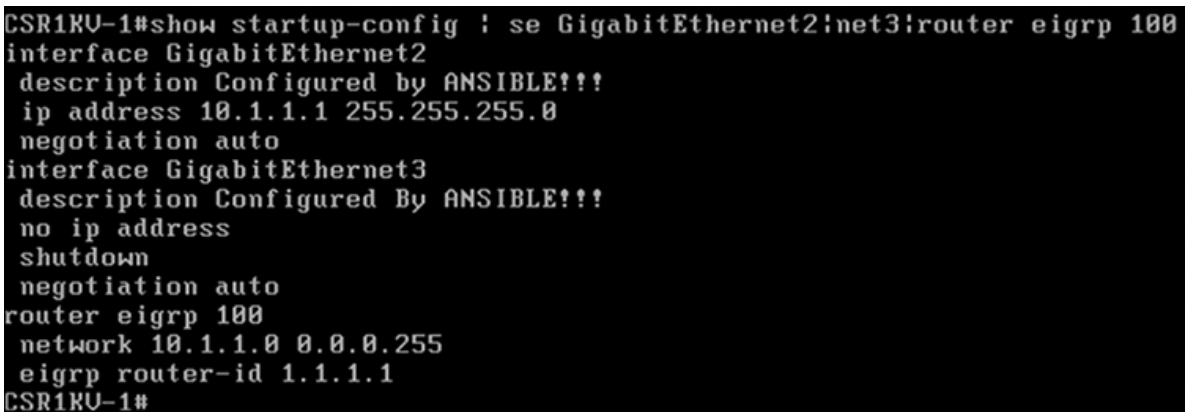


Figure 3-30 *Verifying EIGRP_Configuration_Example.yaml Playbook on CSR1KV-1*

gRPC

Modern-day applications require the ability to transparently communicate between the client and server in order to build systems that are connected. gRPC is a framework that allows for this type of communication structure. gRPC stands for gRPC Remote Procedure Call. gRPC is a project supported by the Cloud Native Computing Foundation (CNCF). The CNCF is responsible for creating cloud-ready software that is sustainable and open. gRPC is designed to be extensible and work with multiple data formats. These data types also include JSON and XML, among others. In addition to supporting multiple data types, another big benefit to gRPC is the support for many different programming languages. The following programming languages are supported at the time of this writing:

- Python
- C++
- Java
- Node.JS
- C#
- Ruby
- Go
- Objective-C
- Dart (beta)

> Note
>
> To learn more about gRPC-supported programming languages, visit https://grpc.io/docs/.

As you can see, this is a powerful technology that offers a wide support structure. The reason this is so important is that you can have multiple developers working on their own portion of a cloud project in their own programming language and still have the ability for them all to work together as a holistic system. For example, one developer

can be working on a section of the application in Go, and the code can interact and work with code developed by another developer using Python. Figure 3-31 depicts the typical communication flow between multiple programming languages.

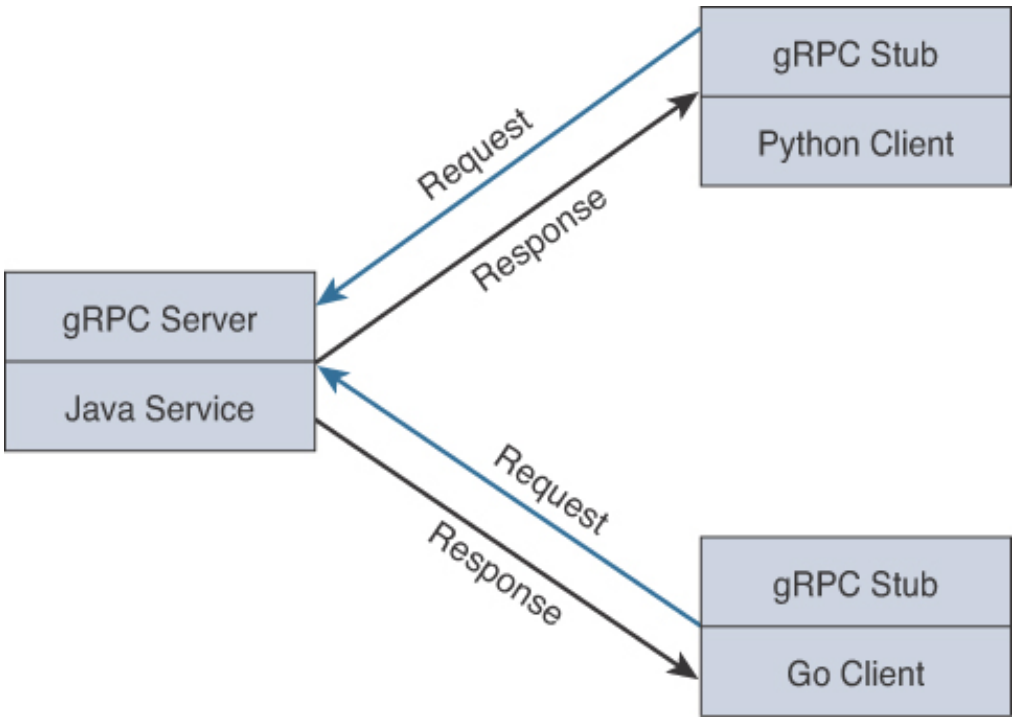


Figure 3-31 *gRPC Traffic Flow*

Google is one of the biggest consumers of this technology as of late. Google is using this for most of its cloud products and externally facing APIs. gRPC offers the ability to create a highly scalable and agile distributed system capable of handling workloads from different programming languages, making it very appealing to large companies looking to develop cloud-scale applications.

Like with other remote procedure calls, the data being sent across the connection needs to be serialized. gRPC has the ability to leverage HTTP/2. HTTP/2 offers quite a few benefits over its predecessor HTTP1.1. This is especially important when serializing multiple data streams over a single TCP connection. This is called multiplexing. Table 3-10 illustrates some of the key differences between the two versions of HTTP.

Table 3-10 *HTTP/2 vs. HTTP 1.1*

HTTP/2	HTTP 1.1
Can send multiple requests via a single TCP connection (multiplexing)	Requires new TCP connection for each request

Faster load times ~2 to 4 times faster	Slower load times due to lack of multiplexing
Non-blocking	Subject to Head of Line (HOL) blocking issues
Ability to use compressed headers across multiple transactions	Uses headers for every transaction
Capable of bi-directional communication	Unidirectional traffic flow

In order to understand the benefit gRPC has with data serialization, it is important to cover protocol buffers. Protocol buffers are Google's open source method of serializing structured data. Structured data can be a variety of data formats such as JSON. First, you have to define the data to be serialized into a text file with a proto extension. In Python, we define objects by name. Proto files are used to apply an integer to a definition. This means when serializing the data, instead of having to send the whole name you defined, you can simply send a number that represents the defined object. For example, the proto file contains a list of fields that just map the object names to numbers. Example 3-16 shows the content of a sample proto file.

Example 3-16 *Proto File Example*

```
message Employee {
  string EmployeeName = 1;
  int32 EmployeeID = 2;
}
```

Based on this example, we can send a 1 or a 2 instead of having to send the whole field name, like EmployeeName or EmployeeID, and still have the same outcome. This can save a significant amount of time and reduce latency as well when serializing the data. Essentially, one can send a single number that represents a much larger field or definition of structured data.

SUMMARY

This chapter covered some of the foundational skills needed to get started with network programmability. This chapter also covered some of the key capabilities and benefits from using a programmatic approach to managing a network. The tools covered in this chapter are available online and are very useful in terms of building skill and expertise. DevNet is a single place where network operators and developers can visit to practice any of the technology and examples covered in this chapter. You will often hear the following advice in various forms in reference to programmability: “Start small, but just start!” It is best to practice using a sandbox environment and just build code and run it to see what you can accomplish. The best way to learn any of these topics is to get started and practice. You are only limited by your imagination and coding skills! Remember to have fun and keep in mind that programmability is a journey, not a destination. Separate your learning into smaller, more manageable chunks. You will get better with practice and time.

REVIEW QUESTIONS

1. Configuring network components via the CLI is considered a best-practice approach when dealing with a large number of devices.

1. True
2. False

2. To authenticate with Cisco’s DNA Center, which type of HTTP request method must be used?

1. PUT
2. PATCH
3. GET
4. POST
5. HEAD

3. What does CRUD stand for?

1. CREATE, RESTORE, UPDATE, DELETE

2. CREATE, READ, UPDATE, DELETE
3. CREATE, RETRIEVE, UPDATE, DELETE
4. CREATE, RECEIVE, UPLOAD, DOWNLOAD
5. CREATE, RECEIVE, UPLOAD, DELETE

4. What software allows for testing of RESTful APIs?

1. Google Postman
2. Ansible
3. TELNET
4. XML
5. JSON
6. SSH

5. Which of the following is in JSON data format?

```
{
  "user": "root",
  "user": "Jason",
  "user": "Jamie",
  "user": "Luke"
}
```

```
<users>
  <user>
    <name>root</name>
  </user>
  <user>
    <name>Jason</name>
  </user>
  <user>
    <name>Jamie</name>
  </user>
  <user>
    <name>Luke</name>
  </user>
```

```
        </users>
```

3. 1. `root`
   2. `Jason`
   3. `Jamie`
   4. `Luke`
4. `[users[root|Jason|Jamie|Luke]]`

6 . What is the HTTP status code for Unauthorized?

1. 201
2. 400
3. 401
4. 403
5. 404

7 . What are some of the benefits of Source Control Management?

1. Keeping track of various versions of code
2. List of developers involved with code revisions
3. Collaborating with other developers and sharing code
4. Accelerating the speed of working on software projects
5. Real-time telemetry of the software database
6. Automatic blocking of malicious code

8 . What are the PPDIOO lifecycle components?

1. Prepare, Plan, Design, Implement, Observe, Optimize

2. Prepare, Plan, Design, Implement, Operate, Optimize

3. Prepare, Plan, Design, Implement, Operate, Optimize

4. Plan, Prepare, Design, Implement, Observe, Optimize

5. Prepare, Plan, Design, Integrate, Observe, Optimize

9 . Ansible uses the TAML syntax for creation of Playbook files that start with three dashes (---).

1. True

2. False

10. What is the proper command for executing a Playbook using Ansible?

1. ansible-playbook ConfigureInterface.yaml

2. ansible ConfigureInterface.yaml

3. play ansible-book ConfigureInterface.yaml

4. play ansible-book ConfigureInterface.taml

REFERENCES

RFC 4741, “NETCONF Configuration Protocol,” R. Enns, Ed. Network Working Group, https://tools.ietf.org/html/rfc4741, December 2006.

RFC 6020, “YANG—A Data Modeling Language for the Network Configuration Protocol (NETCONF),” M. Bjorklund, Ed., IETF, https://tools.ietf.org/html/rfc6020, October 2010.

RFC 6241, “Network Configuration Protocol (NETCONF),” R. Enns, M. Bjorklund, J. Schoenwaelder, A. Bierman, Eds. IETF, https://tools.ietf.org/html/rfc6241, June 2011.

Appendix

Answers to Review Questions

CHAPTER 1

1. a. Explanation: Processing should occur as close to the source of the data (sensor/thing) as possible. If an edge node can perform the computation, it should be used first.

2. a. Explanation: IoT compute processing can be staged in a hierarchical fashion. An edge node can send data to a fog node, and a fog node can send data to another fog node or toward a data center or cloud provider.

3. a. Explanation: IoT networks are more complex because they have a larger scale and more attack vectors than traditional IT networks.

4. b. Explanation: Availability is the most significant factor due to financial impacts to the business.

5. b. Explanation: OT networks connect to robots, power grids, refineries, and other markets where a compromise could result in the loss of life. Therefore, security must be considered to prevent malicious attackers from causing personal injury.

CHAPTER 2

1. c. Explanation: NIST only provides definitions for SaaS, PaaS, and IaaS.

2. a. Explanation: CCC's application models support a mixture of customized services, images, and containers.

3. b. Explanation: Network Functions Virtualization (NFV) is an architectural framework, whereas VNF is the software version of an NF (that is, a virtual load balancer).

4. a. Explanation: Rehosting (aka lift-and-shift) involves redeploying applications to an IaaS or Bare Metal as-a-service (BMaaS) environment without making any changes to the applications.

5. d. Explanation: vAnalytics is an optional analytics and assurance service that includes many advanced monitoring capabilities.

CHAPTER 3

1. b. Explanation: Configuring a large number of devices via the CLI is not only time consuming, it also leads to an increase in human error, ultimately putting the business at risk.

2. d. Explanation: To authenticate to Cisco's DNA Center Controller, a POST operation must be used. This is because we are sending the login credentials to the controller to be verified.

3. b. Explanation: CRUD stands for CREATE, READ, UPDATE, and DELETE. These are the common actions associated with the manipulation of data. For example, a database uses these actions.

4. a and b. Explanation: Postman and Ansible both have the ability to specify a RESTful API or URL to be used to practice or test API functions. Telnet and SSH are transport methods, whereas XML and JSON are data structures.

5. a. Explanation: A JSON data format is built from key-value pairs. For example, "user": "Jason" is a key value pair, where user is the key and Jason is the value.

6. c. Explanation: An HTTP status code of 401 is for Unauthorized, meaning

incorrect login credentials or not having valid authentication to a destination. See the following table for more HTTP Response codes.

HTTP Status Code	Result	Common Reason for Response Code
200	OK	Using a GET or POST to exchange data with an API.
201	Created	Creating resources via a REST API call.
400	Bad Request	Request failed because of a client-side issue.
401	Unauthorized	Client not authenticated to access site or API call.
403	Forbidden	Access not granted based on supplied credentials.
404	Not Found	Page at HTTP URL location does not exist or is hidden.

7. a, b, and c. Explanation: Source Control Management can help with many different aspects of software delivery. The most common are code revisions, associated developers, sharing of code, and becoming more agile in the development process. Real-time telemetry of the software database and automatic blocking of malicious code have nothing to do with SCM.

8 . b. Explanation: PPDIOO consists of six components: Prepare, Plan, Design, Implement, Operate, and Optimize. Please refer to Figure 3-29 for more information.

9 . b. Explanation: Ansible uses Yet Another Markup Language (YAML) for the creation of Playbook files. TAML doesn't exist.

10. a. Explanation: ansible-playbook <filename.yaml> is the correct command to execute a Playbook. Playbooks are built from Yet Another Markup Language (YAML) files. TAML files do not exist.

REGISTER YOUR PRODUCT at PearsonITcertification.com/register

Access Additional Benefits and SAVE 35% on Your Next Purchase

- Download available product updates.
- Access bonus material when applicable.
- Receive exclusive offers on new editions and related products. (Just check the box to hear from us when setting up your account.)
- Get a coupon for 35% for your next purchase, valid for 30 days. Your code will be available in your PITC cart. (You will also find it in the Manage Codes section of your account page.)

Registration benefits vary by product. Benefits will be listed on your account page under Registered Products.

PearsonITcertification.com–Learning Solutions for Self-Paced Study, Enterprise, and the Classroom

Pearson is the official publisher of Cisco Press, IBM Press, VMware Press, Microsoft Press, and is a Platinum CompTIA Publishing Partner–CompTIA's highest partnership accreditation.

At **PearsonITcertification.com** you can

- Shop our books, eBooks, software, and video training.
- Take advantage of our special offers and promotions (pearsonitcertifcation.com/promotions).
- Sign up for special offers and content newsletters (pearsonitcertifcation.com/newsletters).
- Read free articles, exam profiles, and blogs by information technology experts.
- Access thousands of free chapters and video lessons.

Connect with PITC – Visit PearsonITcertifcation.com/community

Learn about PITC community events and programs.

PEARSON IT CERTIFICATION

Addison-Wesley • Cisco Press • IBM Press • Microsoft Press • Pearson IT Certification • Prentice Hall • Que • Sams • VMware Press