Evaluating the advantages of Cisco ACI over traditional Three-Tier Architecture

Bachelor's Thesis

Degree Programme in Computer Applications

Spring 2025

Amin Hamidi Younessi

DP Bachelor's degree in Computer Application

Author Amin Hamidi Younessi Year 2025
Subject Evaluating the advantages of Cisco ACI over traditional Three-Tier Architecture
Supervisors Jawad Yasin

Between 2005 to 2013, network infrastructure mainly in the data centers faced issues like scalability, automation, and segmentation. On the other hand, software developers need more bandwidth these days for their applications which are improving every day. Traditional data centers cannot provide the needs of today's software developers and handle their applications traffic. Even by using technologies like EtherChannel or link aggregation, it cannot provide the best solution for the data centers.

Traditionally, data centers and networks have been designed based on hierarchical designs that are named three-tier model (three layers). The recommended three-tier design data center by Cisco consists of a core layer, aggregation layer and an access layer. Network management, maintenance, correlating information between different vendors and devices, different levels of expertise, human errors and differences in protocols used to implement a network are the most challenges in front of companies.

The aim of this research is to compare traditional data centers with Software-Defined Network based solutions for the data center with a thought in mind to help anybody who wants to be more familiar with Cisco Application Centric Infrastructure (ACI). During some parts of this thesis, it will moreover be learned how data centers can take advantage of the latest AI technology like Deep Learning (DL).

The research takes a practical approach and focuses on a real-world solution. First, concepts about network, data center challenges, Software-Defined Networking are explained. After that, the project described how to configure Cisco ACI and how to use automation tools in order to decrease boring tasks and configuration in an enterprise network solution. This practical thesis was created based on the ATBIT company's customer data center and it is not only a configuration approach, but also a reference for migration to any data center.

This thesis tried to provide a good practical reason to choose SDN base technologies over traditional ones, and it shows us the different ways of automation and automation tools in network infrastructures.

Keywords Cisco ACI, Software-Defined Networking, Tenant, Application Centric Infrastructure, Programmability, EPG, Contract, VXLAN, Machine Learning (ML), Deep Learning (DL), Data Center, Network Automation.
Pages 69 pages and appendices 3 pages

Table of Contents

List of Figures

Tables

Commands

Appendices

1 Introduction

Most companies and service providers are using traditional design in their data centers called Three-tier architecture (*Small Enterprise Design Profile (SEDP)—Network Foundation Design*, n.d.). For high performance and scalable network designs, the three-tier architecture model is a commonly utilized strategy. While the three-tier network architecture works well for smaller businesses, it can become an obstacle to manage the large enterprise environments (*Small Enterprise Design Profile (SEDP)—Network Foundation Design*, n.d.). Network engineers often face limitations, especially when it comes to scaling. Imagine trying to add more switches to a busy data center – with a traditional three-tier setup, this is not always a straightforward process and might even require significant redesigns. Another key concern is performance. Think about how quickly software evolves; in the past, these rapid advancements often left behind the capabilities of the existing network design, leaving it struggling to provide the necessary bandwidth. Furthermore, one of the biggest challenges in front of the three-tier architecture is security. If an attacker manages to bypass security components, for instance, Access Control Lists (ACLs), Intrusion prevention system/intrusion detection systems (IPS/IDS), firewalls or Next-Generation Firewalls on network devices and subsequently gain access to a web server within the same network segment (broadcast domain), this could enable them to target other servers within that segment, such as the database server. ATBIT is a company (*AtBit Network - About*, n.d.) in the United States that provide cloud native solutions for companies from small to enterprise scale. In a large enterprise environment, it is common to have hundreds of infrastructure nodes like switches, routers, firewalls, IPS/IDS, load balancers, servers and storages. Managing this many devices can be a significant annoyance for network engineers, as it can lead to errors in configuration. As a result, the possibility of incorrect configurations going unnoticed for long periods of time is high. On the other hand, these days in data centers there might be different vendors of devices, virtual appliances, virtualizations, and clouds that need a high level of expertise and maximum number of network experts and administrators for the maintenance (Dagenhardt et al., 2018).

Software-Defined Networking (SDN) transforms network management by employing abstraction. This allows for dynamic and programmable network configuration, leading to benefits like enhanced segmentation, simplified deployment, flexible grouping, improved network performance, and more effective monitoring, particularly in cloud computing environments compared to traditional networking (*Cisco Application Policy Infrastructure Controller (APIC) - Cisco Application Centric Infrastructure (ACI) Design Guide*, n.d.).

Recognizing the limitations of static traditional network architectures, Software-Defined Networking (SDN) emerged as an approach. Its fundamental design principle involves centralizing network intelligence within a single component achieved through the segregation of packet forwarding (data plane) from routing logic (control plane) which handles routing decisions (*AI Solutions on Cisco Infrastructure Essentials | Cisco U. Path*, n.d.-a). In SDN environments, the control plane is characterized by one or more controllers, which effectively concentrate the network's intelligence. These components act as the brains of the SDN network, containing its operational logic.

- How can Cisco ACI improve data center design?
- How can Cisco ACI solve traditional 3-Tier Architecture data center security issues?
- Is Cisco ACI compatible with cloud native and virtualization solutions?

2 Research Methodology

This research consists of a comparative analysis approach, evaluating the practical differences and advantages of Cisco Application Centric Infrastructure (ACI) over traditional three-tier data center architectures. The study combined emulation, simulation, and real-world observation gained through the researcher's involvement in a live migration project from a traditional infrastructure to Cisco ACI for an organization. This unique blend of hands-on experience, lab-based testing, and literature review aimed to provide a broad and practical understanding of the benefits and challenges associated with adopting ACI.

2.1 Data collection

Data for this research were collected through a multi-faceted approach, incorporating both controlled lab environments and real-world insights like real-world migration experience, emulation and simulation, automation testing, and literature review. Real-world migration experience obtained directly from involvement in a project assisting an organization with the migration from a traditional three-tier data center infrastructure to Cisco ACI provided invaluable practical data. This included observing the planning, implementation, configuration, and troubleshooting aspects of the migration process. Specific focus was placed on the steps involved in transitioning port aggregation technologies, the scalability considerations during the migration and the initial automation workflows implemented in the ACI environment. Moreover, emulated Cisco Nexus 7000 and 9000 series switches to model and analyze traditional port aggregation technologies (vPC, MLAG, EtherChannel) and Layer 2 connectivity using Cisco IOS in eve-ng and VMware workstation. Also, benefited Cisco ACI sandbox to explore ACI's policy-driven model, focusing on the creation of Tenants, EPGs, and the implementation of network policies, independent of the live migration environment. Furthermore, automation testing by the Cisco ACI simulator's REST API was interacted with using Postman, and automation scripts were developed and executed using TCLSH, Python and Ansible to compare automation capabilities. Finally, literature review by a comprehensive review of scholarly articles, Cisco official documentation (Cisco U, Cisco.com, Cisco Live), RFCs, and the Cisco DCACI study guide provided the theoretical and vendor-specific context for the practical findings.

2.2 Evaluation criteria

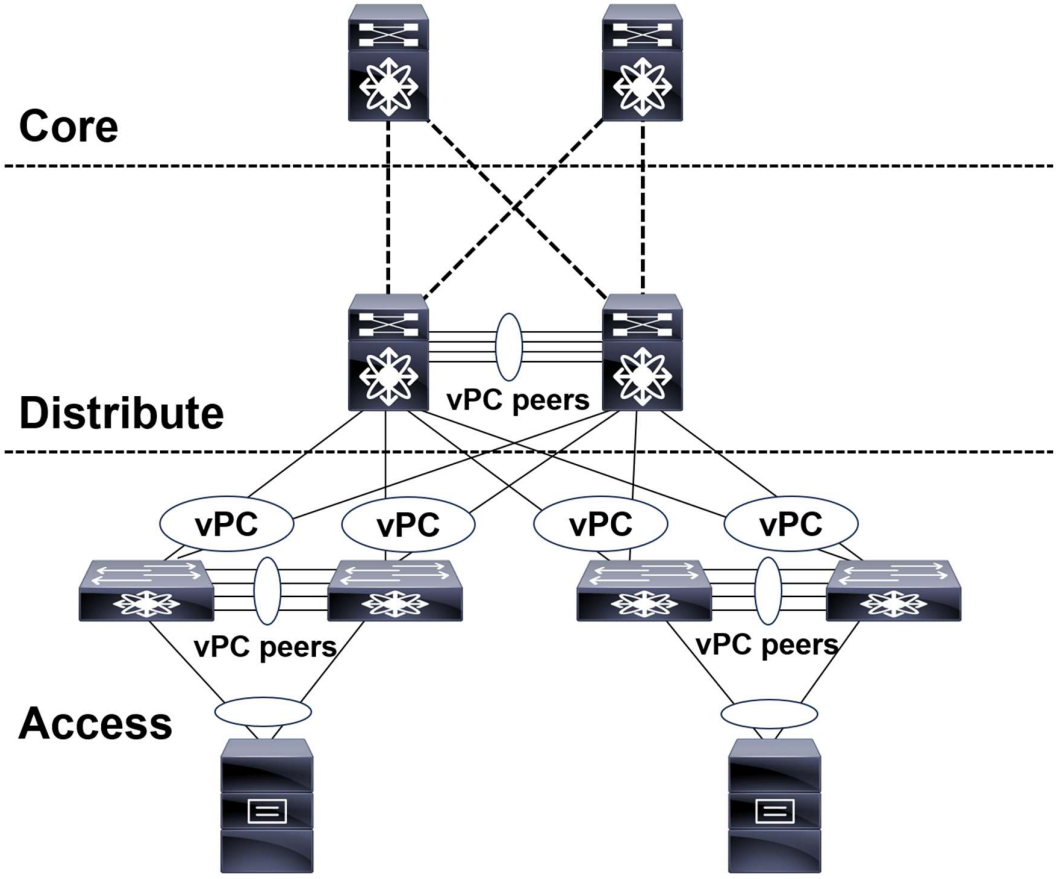
The evaluation of Cisco ACI and the traditional three-tier architecture was based on the following key metrics, informed by both the lab environments and the real-world migration experience like port aggregation technology transition by comparing the process and complexity of migrating port aggregation from vPC and Ether Channel to the Cisco ACI constructs. Assessing the changes in network design and scalability achieved through the ACI migration to show scalability and design evaluation. VRF to tenants and EPGs creation for analyzing the process of migrating logical network segmentation from traditional VRFs to ACI Tenants and EPGs. Furthermore, this thesis evaluated the implementation and impact of automation tools and workflows in the newly deployed ACI environment compared to the previous traditional setup infrastructure network automation, configuration and operational complexity. Finally, the thesis aims not only to describe the concept and differences between traditional and SDN networks, but also it shows insights into the Cisco fabric configuration by a practical scenario.

As Cisco ACI is a data center solution, evaluating its performance needs specific hardware, such as Cisco Nexus Spine and Leaf switches. While this thesis utilized emulated environments like Cisco Sandbox and EVE-NG, which leverage genuine Cisco IOS and NX-OS images, it was not possible to assess real-world bandwidth compared to traditional data centers. Furthermore, a comprehensive evaluation would typically involve the Cisco Nexus Dashboard, another Cisco product that falls outside the scope of this thesis.

3 Three-Tier network design and its challenges

Traditional networks and data centers are designed based on three famous layers called Core, Distribute, and Access layers which is considered as a hierarchical design (*Small Enterprise Design Profile (SEDP)—Network Foundation Design*, n.d.) Figure 1 illustrates the three-tier design which is utilized in traditional networking.

Figure 1 Three-Tier Network Architecture



Within the three-tier architecture, the access layer is where servers are located. This layer typically contains layer 2 switches, which are usually the lowest-cost switches with the least bandwidth (Ahmadi, 2020, p. 5). The second layer, the distribution layer, is responsible for providing connectivity between different broadcast domains, such as VLANs and acts as the gateway for each VLAN. Furthermore, the distribution or aggregation layer switches are responsible to carry the layer 2 traffic along each broadcast domain and act as a layer 2 device. Simultaneously, distribute layer switches must function as layer 3 devices to route packets between different broadcast domains. Additionally, packets from each network or

VLAN will be transited by layer 3 switches to the core layer for traffic destined outside the data center, typically using Dynamic Routing Protocols like Open Shortest Path First(OSPF) (Moy, 1997) or Enhanced Interior Gateway Protocol (EIGRP) (Savage et al., 2016). In the distributed layer, a spanning-tree protocol must be in place to prevent layer 2 network loops. As seen in Figure 1, multiple links exist between most devices from the distribution layer to the access layer, resulting in a Layer 2 loop. Normally network engineers must take care about link failure. From below, technologies that provides link aggregation in Cisco and multi-vendor environment will be introduced:

3.1 EtherChannel

Sometimes within the data center environment, two switches need to be connected to each other by more than one physical or logical link in the cloud native and virtual environment by this thought in our mind to increase a bandwidth between them. Another reason could be high-availability and preventing link failure. The problem is, when two neighbor switches connect to each other with parallel links, a layer two loop will occur and by default all Cisco switches that benefit from Spanning-Tree Protocol (STP) or Rapid Spanning-Tree protocol will block one or more ports to prevent layer two loop. Layer two loops can cause three main problems. First, layer 2 loops can cause a broadcast, multicast, or unicast storm. Second, MAC-table instability, and the third one is that multiple copies of each frame can reach to the same destination that cause performance issues or even failure (EtherChannels, n.d.).

By using a technology like EtherChannel which can operate at Layer 2 or Layer 3 depending on the scenario, two switches can connect via two or more parallel links that function as a single logical link. As a result, the bandwidth between two neighbor switches will increase. It has different protocols, Link Aggregation Control Protocol (LACP) (*Link Aggregation Control Protocol (LACP) (802.3ad) for Gigabit Interfaces*, n.d.) that is an IEEE protocol and Port Aggregation Protocol (PaGP) (*Configuring EtherChannels*, n.d.) that is a Cisco proprietary one. Some vendors have the port aggregation limits up to 32 and some of them 16 but a half of them could be active at the same time. In Cisco PaGP, the number of bundled ports cannot be more than 8. Figure 2 shows a simple scenario for the layer 2 EtherChannel in the eve-ng environment.

Figure 2 Layer 2 EtherChannel sample

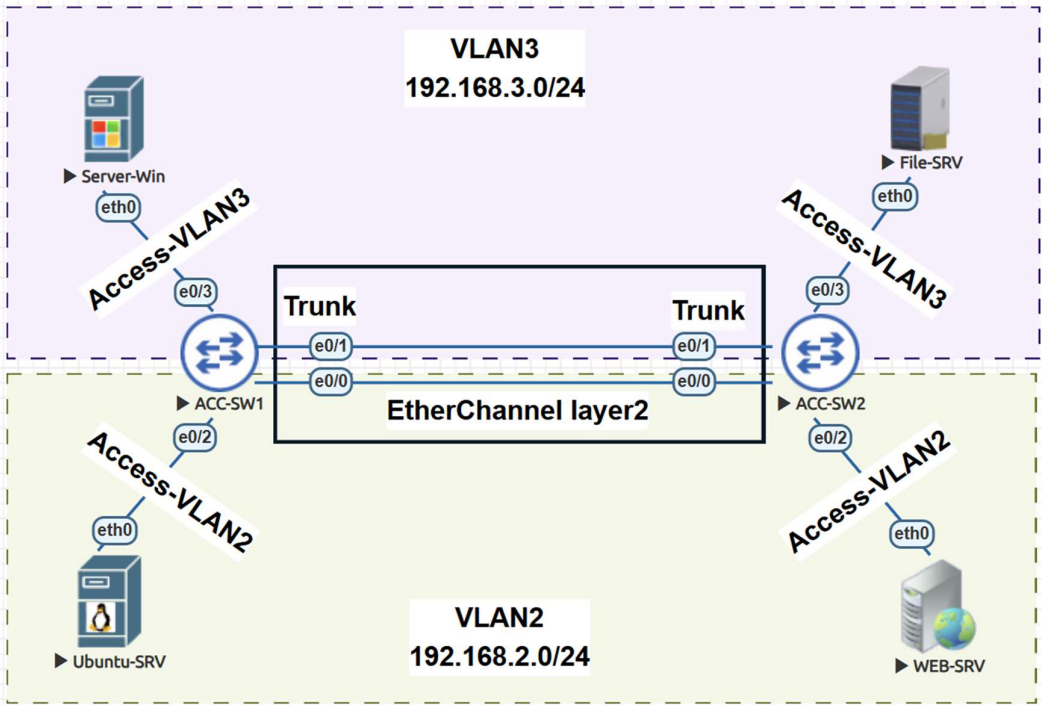
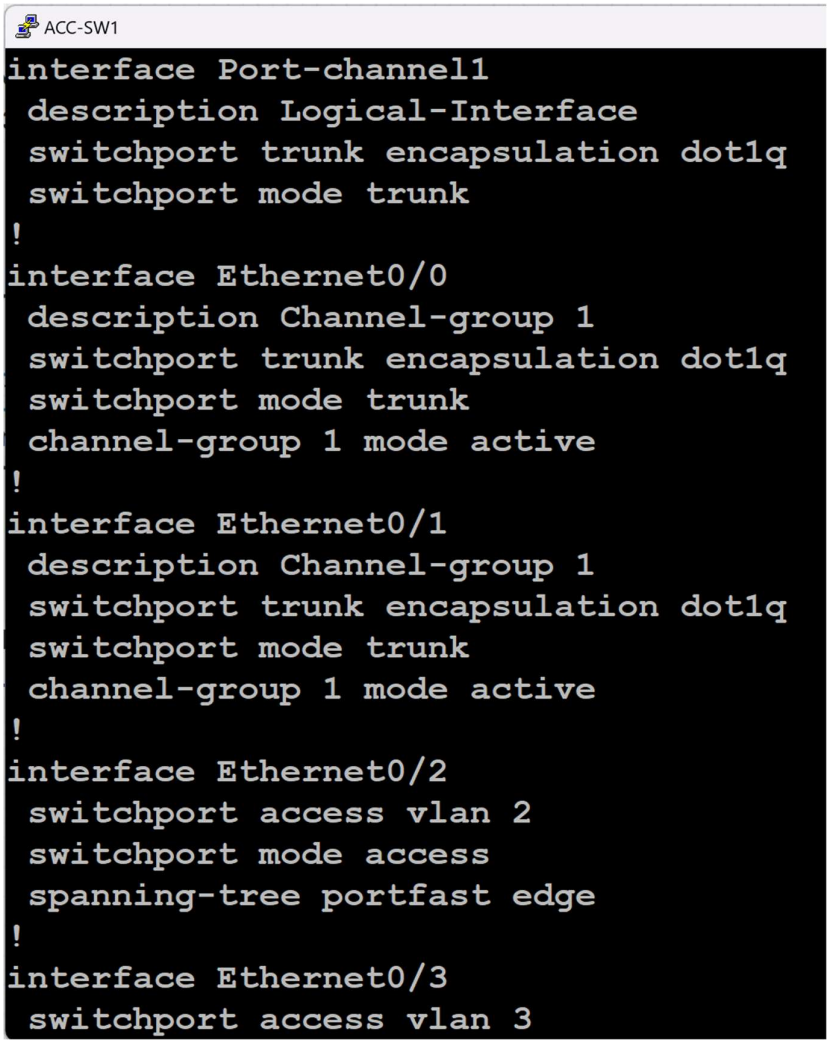


Figure 2 illustrates, Layer 2 EtherChannel to aggregate two links between Layer 2 switches. This was done to provide increased bandwidth, high availability, and protection against link failure between servers and services. Typically, Layer 2 EtherChannel is used between access switches and distribution switches, either within a data center or between different building floors.

A Layer 3 EtherChannel is a potential solution based on the scenario. The primary distinction between Layer 2 and Layer 3 EtherChannel is their operational layer. Layer 2 EtherChannel is used for Layer 2 connectivity, forwarding Layer 2 protocols like VLAN and ARP. Layer 3 EtherChannel, on the other hand, enables inter-network communication via routing, requiring Layer 3 devices. Both technologies provide increased bandwidth and high availability for various connection needs, including intra and inter-data center and branch connectivity. Link aggregation plays a crucial role in both conventional and SDN data center environments. The fundamental distinction between Layer 2 and Layer 3 EtherChannel lies in the specific network layer at which they operate. For example, in SDN based data centers, vPC is the main solution.

Furthermore, you can find the Layer 2 sample configuration as illustrated in Figure 3.

Figure 3 Layer 2 EtherChannel configuration sample



Moreover, EtherChannel could be used based on Layer 3 of the Open Systems Interconnection (OSI) model (*Ietf.Org/Rfc/Rfc1122.Txt*, n.d.). It can be used within the data center or to connect buildings together to achieve higher performance and prevent failure.

3.2 Virtual Port Channel (vPC)

Usually, a pair of Nexus switches in a virtual PortChannel (vPC) domain serves as the aggregation or core layer, establishing the boundary between Layer 2 switched Ethernet networks and Layer 3 routed networks. These switches are configured with numerous VLANs and manage routing for both east-west (inter-VLAN) and north-south traffic flows. This protocol is a Cisco proprietary protocol used on Cisco Nexus series switches. Any two

Cisco Nexus switches of the same model can form a single vPC pair. It is supported on Nexus switches such as the 5000, 7000, and 9000 series (*Understand Virtual Port Channel (vPC) Enhancements*, n.d.). Figure 4 shows Cisco Nexus 9000 series switches.

Figure 4 Cisco Nexus 9000 series switches (*Cisco Data Center Networking Solutions*, n.d.)

Virtual Port Channel (vPC) technology provides a way to aggregate two separate links from two different switches to a third switch. Usually, this protocol is used at the distribution and access layers of the data center. Network endpoints in the data center (switches, routers, servers, firewalls, NAS storage) are connected via vPC to the core layer in an active/active state (*Understand Virtual Port Channel (vPC) Enhancements*, n.d., p. 9). Technology like vPC is used in two main scenarios: within the data center, including Single-Sided vPC and Double-Sided or Multilayer vPC; and across data centers for Data Center Interconnect (DCI). Figure 5 illustrates a scenario based on vPC on Cisco Nexus 7000 and 9000 series switches in eve-ng environment.

Figure 5 vPC connection between three Nexus switches

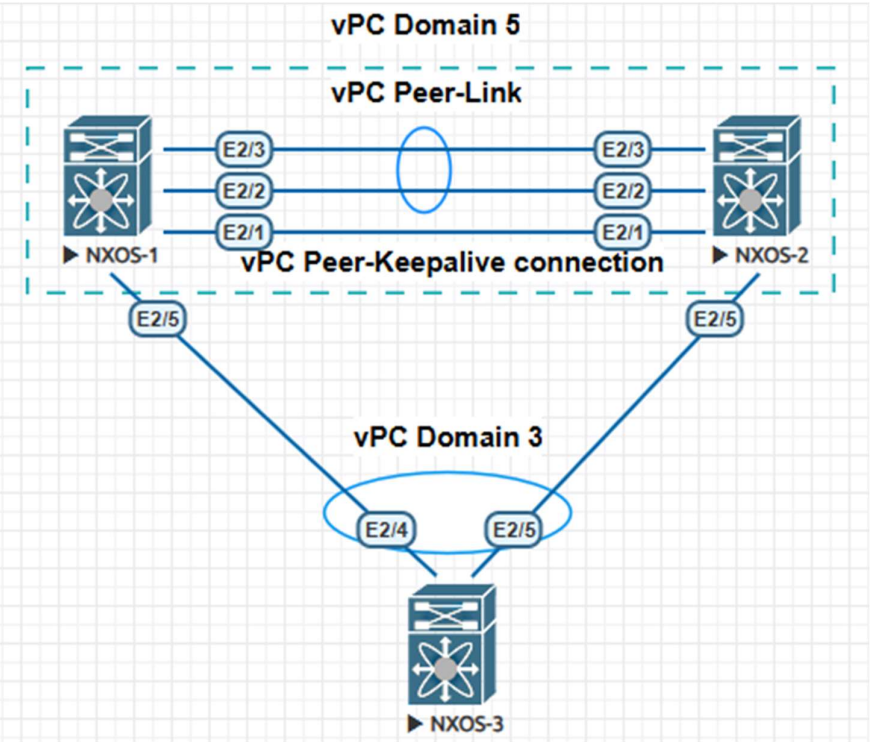


Figure 5 emulates a vPC connection between three Cisco Nexus switches. In this emulation, NXOS-1 and NXOS-2, assumed to be Cisco Nexus 7000 series switches. These Nexus switches have three physical connections to each other, one for vPC management packets and two for the vPC peer-link connection. Additionally, Ethernet interfaces 2/5 on both NXOS-1 and NXOS-2 are connected to NXOS-3, forming another vPC in a different vPC domain. Therefore, in enterprise data centers based on Cisco products, vPC is a technology, or protocol, that provides high availability and prevents link failures. The NXOS commands used in this chapter illustrated in Command 1, are provided for reference.

Command 1 Cisco NXOS configuration sample for vPC

NXOS-1 Configuration:

```
NXOS-1# configure terminal
NXOS-1(config)# vrf context VPC-
KEEPalive
NXOS-1(config-vrf)# interface
Ethernet 2/1
NXOS-1(config-if)# no switchport
NXOS-1(config-if)# vrf member VPC-
KEEPalive
```

NXOS-2 Configuration:

```
NXOS-2# configure terminal
NXOS-2(config)# vrf context VPC-
KEEPalive
NXOS-2(config-vrf)# interface
Ethernet 2/1
NXOS-2(config-if)# no switchport
NXOS-2(config-if)# vrf member VPC-
KEEPalive
```

```
NXOS-1(config-if)# ip address
10.1.1.100/24
NXOS-1(config-if)# end
NXOS-1#conf t
NXOS-1(config)# interface ethernet
2/2-3
NXOS-1(config-if-range)# switchport
NXOS-1(config-if-range)# switchport
mode trunk
NXOS-1(config-if-range)# channel-
group 1
NXOS-1(config-if-range)# end
NXOS-1#conf t
NXOS-1(config)# feature vpc
NXOS-1(config)# vpc domain 5
NXOS-1(config-vpc-domain)# peer-
keepalive destination 10.1.1.200
source 10.1.1.100 vrf VPC-KEEPalive
NXOS-1(config-vpc-domain)# exit
NXOS-1(config)# interface port-
channel 1
NXOS-1(config-if)# vpc peer-link
NXOS-1(config)# feature lacp
NXOS-1(config)# interface Ethernet
2/5
NXOS-1(config-if)# channel-group 2
mode active
NXOS-1(config-if)# interface port-
channel 2
NXOS-1(config-if)# vpc 3
NXOS-1(config-if)# end
NXOS-1#
```

```
NXOS-2(config-if)# ip address
10.1.1.200/24
NXOS-2(config-if)# end
NXOS-2#conf t
NXOS-2(config)# interface ethernet
2/2-3
NXOS-2(config-if-range)# switchport
NXOS-2(config-if-range)# switchport
mode trunk
NXOS-2(config-if-range)# channel-
group 1
NXOS-2(config-if-range)# end
NXOS-2#conf t
NXOS-2(config)# feature vpc
NXOS-2(config)# vpc domain 5
NXOS-2(config-vpc-domain)# peer-
keepalive destination 10.1.1.100
source 10.1.1.200 vrf VPC-KEEPalive
NXOS-2(config-vpc-domain)# exit
NXOS-2(config)# interface port-
channel 1
NXOS-2(config-if)# vpc peer-link
NXOS-2(config)# feature lacp
NXOS-2(config)# interface Ethernet
2/5
NXOS-2(config-if)# channel-group 2
mode active
NXOS-2(config-if)# interface port-
channel 2
NXOS-2(config-if)# vpc 3
NXOS-2(config-if)# end
NXOS-2#
```

Additionally, the NXOS-3 commands are listed below:

NXOS-3 Configuration:

```
NXOS-3# configure terminal
NXOS-3(config)# feature lacp
NXOS-3(config)# interface Ethernet 2/4-5
NXOS-3(config-if-range)# channel-group 15 mode active
NXOS-3(config-if-range)# end
```

To accomplish this task, network engineers must connect to each switch one by one and configure them manually. In large enterprise data centers, a network engineer may be responsible for managing hundreds or thousands of switches and devices, so manual configuration is time-consuming and can lead to misconfigurations.

3.3 Multi-Chassis Link Aggregation Group

Multi-chassis Link Aggregation Group (MLAG or MC-LAG) (Bailis & Kingsbury, 2014) is a form of Link Aggregation Group (LAG) where the member ports are located on different physical devices. The main advantage of this configuration is redundancy, ensuring network connectivity even if one of the interconnected chassis experiences a failure. It's important to note that the specific implementation and the communication protocol between the chassis are vendor-specific. MLAG is widely used in traditional networks that employ multi-vendor environments, and Cisco vPC is completely compatible with MLAG technology (*Multi-Chassis Link Aggregation Group - RouterOS - MikroTik Documentation*, n.d.).

3.4 Network modularization within the data center

Network modularization is a critical design principle in modern data centers, offering enhanced scalability, fault isolation and manageability. By dividing the network into discrete, functional units, the complexity of managing large-scale deployments is significantly reduced. This section of the thesis will explore the key aspects of network modularization, including its benefits, common architectures, and implementation considerations (Diamond, 2024).

Network modularization offers several key advantages. Scalability is enhanced, as modular designs allow for incremental expansion without disrupting existing operations. Fault isolation is improved by containing failures within specific modules, minimizing the impact of outages and increasing overall network resilience and availability. Management is also simplified, since dividing the network into smaller, more manageable units makes configuration, monitoring, and troubleshooting easier. Furthermore, modularization increases security by enabling the implementation of granular security policies, which limits the attack surface and improves the security posture. Finally, network performance is enhanced by reducing the size of broadcast domains and containing traffic within specific modules.

Several architectural approaches are commonly used to achieve network modularization within the data center. These architectures include Three-Tier Architecture and Spine-Leaf Architecture. Spine-Leaf provides high bandwidth, low latency and offers a high degree of modularity and scalability.

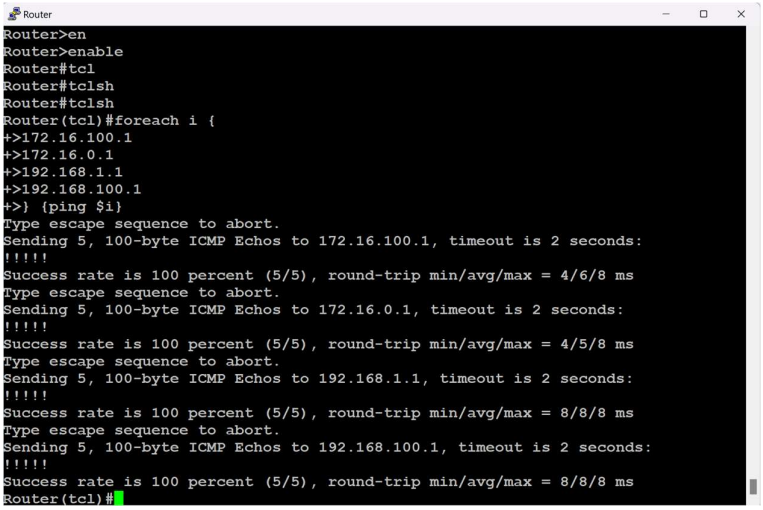
When implementing network modularization, several factors should be considered. The design of the connections between modules is necessary for ensuring high performance and availability. Proper IP addressing and routing protocols are essential for efficient traffic flow between modules. Highly detailed security policies should be implemented at module boundaries to restrict traffic flow and enhance security. Managing and configuring modular networks becomes easier with centralized tools.

3.5 Automation in traditional networks

For many years, Cisco Prime Infrastructure was a primary solution in enterprise network environments, used for network monitoring, management, configuration, and reporting. This comprehensive network management platform offers a unified web interface for administering any wireless and wired networks (*SMB50 ENG Switching.Pdf - This Module Describes Cisco Borderless Network Switching Solu6ons for Small and Midsize Customers and Helps | Course Hero*, n.d.). With Cisco Prime, network engineers were able to create configuration templates to push configurations to infrastructure network devices such as routers, switches, and wireless access points. Additionally, it offered a wizard-based process for deploying templates or utilizing tclsh (Tool Command Language Shell) script files to push configurations based on logical groups within the data center or infrastructure network environment. Furthermore, it employed a graphical user interface panel with real-time status and reporting capabilities. Moreover, many network engineers with knowledge of the Python programming language could leverage the built-in automation capabilities of Cisco routers and switches for their daily tasks, but the challenge was not all the network engineers had a knowledge of the programming language to automate configuration. Historically, protocols such as VLAN Trunking Protocol (VTP) have assisted network engineers in reducing repetitive manual configuration when deploying additional VLANs for new infrastructure or modifying network requirements. Operationally, within a VTP domain, network administrators were required to define all VLANs on a designated Cisco switch functioning as the VTP server. Subsequently, access layer switches configured as VTP clients would automatically receive VLAN database updates from the server, thus getting rid of the need for manual VLAN creation on each VTP client switch. Distribution layer switches typically assumed the role of VTP servers, automatically propagating newly defined VLANs to all associated VTP clients. Cisco IOS (Internetworking Operating System), from version 12.3(2)T and 12.2(25)S officially introduced the tclsh command. From testing connectivity with ping to creating OSPF templates and pushing them into multiple routers, tclsh was a helpful tool for some network engineers. Figure 6 shows a

code snippet that I used in the eve-ng (The multi-vendor emulated virtual environment for network, security and DevOps professionals) environment, Figure 6 presents a sample tclsh use cases for testing a connection with the simple ping command. This simple ping with tclsh is specifically used to test the connection or for troubleshooting purposes.

Figure 6 `tclsh` ping sample on the Cisco router



Another example utilizing `tclsh` could be an OSPF template for configuring multiple routers, thus avoiding the manual configuration of OSPF between them. Figure 7 illustrates a `tclsh` template designed to configure the hostnames, IP addresses, and OSPF settings of two routers, along with the resulting configuration.

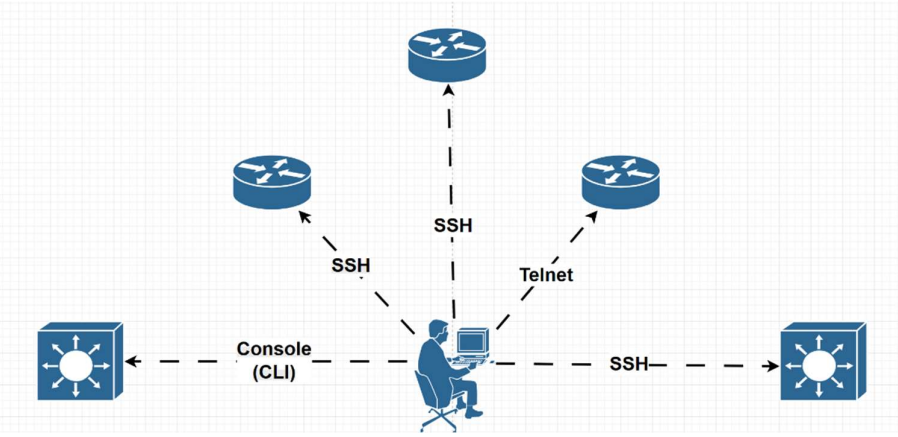
Figure 7 OSPF configuration sample between two Cisco routers using `tclsh`

```
#R1 Script
enable
tclsh
ios_config "hostname R1"
ios_config "interface GigabitEthernet0/0" "ip address 192.168.12.1 255.255.255.0" "no shutdown"
ios_config "router ospf 10" "network 192.168.12.0 0.0.0.255 area 0"
ios_config "end"
ios_config "write memory"
puts "Router1 configuration complete."
tclquit

#R2 Script
enable
tclsh
ios_config "hostname R2"
ios_config "interface GigabitEthernet0/0" "ip address 192.168.12.2 255.255.255.0" "no shutdown"
ios_config "router ospf 10" "network 192.168.12.0 0.0.0.255 area 0"
ios_config "end"
ios_config "write memory"
puts "Router2 configuration complete."
tclquit
```

Traditionally, network management and configuration relied on direct human interaction with network devices. Network engineers would use text editors to create a single configuration sample from a device. Subsequently, network engineers would copy this existing configuration into the same text file and then update it according to the specific requirements of the desired network device. Furthermore, network engineers would connect to devices via the console, Telnet, or SSH. Alternatively, a central controller server, such as Cisco Prime Infrastructure, might be used to push configurations onto the network devices. Figure 8 provides a visual representation of traditional network management.

Figure 8 Traditional network management



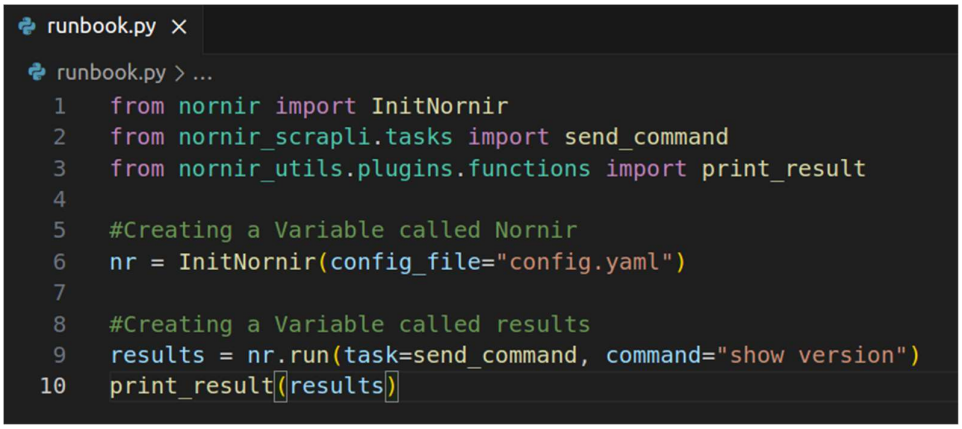
This method presents several challenges when applied to an enterprise network environment. Furthermore, companies often need to hire additional staff with advanced skills, such as Python programming and Ansible, to effectively manage their infrastructure network devices. One of ATBIT company's main customers is scaling their network from a medium size to an enterprise level. To manage their network, ATBIT primarily utilized Python and Ansible for their data centers. Consider a scenario where ATBIT manages not just one data center, but multiple data centers distributed globally from a central point. Hiring more skilled staff with programming language proficiency is a key challenge for them. Despite Python and Ansible being popular automation solutions today, they are not always the most suitable or scalable options for large enterprise networks and data centers.

This chapter covers two additional methodologies for infrastructure network automation. Specifically, this section will focus on Python and Ansible as tools for automating the configuration of Cisco-based data centers and network devices. The Python programming

language is widely used to automate infrastructure configuration. It benefits from a powerful framework called Nornir, which is fast and flexible because it is written in Python, making the integration of Nornir with Python easy. Furthermore, Nornir is a Python automation solution. With the Nornir ecosystem, by installing libraries such as Napalm, Scrapli, and Netmiko to automate networks. Nornir has two important abilities like concurrent task execution and inventory management. Moreover, Nornir does not require an agent to be installed on the devices that need to be configured automatically. A key difference between Nornir and Ansible is their execution speed, with Nornir being notably faster. It gives us host variable and group variable files which are written in Python programming language. The logic using with Nornir to execute tasks is called Runbook (*Welcome to Nornir's Documentation! — Nornir 3.5.0 Documentation*, n.d.). For visualization purposes, this thesis employs an EVE-NG environment running on VMware Workstation. Within EVE-NG, Cisco IOS images of devices such as routers and multi-layer switches are installed. Furthermore, an Ubuntu virtual machine, also hosted on the same VMware Workstation and connected to the Cisco virtual devices within EVE-NG, completes the lab setup. The Ubuntu desktop environment benefits from using VSCode to install extensions such as Python 3, Cisco IOS Syntax Highlighter, Better Jinja, and Remote - SSH, which simplifies this process. Now the important part is to create some useful YAML files and install Nornir in the same directory as YAML files. After that, some important files need to be created, for example, hosts.yaml file. In network automation with Python, especially when using frameworks like Nornir, to define and manage the inventory of network devices hosts.yaml file is necessary. Additionally, groups.yaml file is using to organize network devices into logical groups based on the same characteristics or roles. Another important file is config.yaml. This file is created for a structured, centralized, and easily manageable way to define and manage the essential settings and parameters. These settings and parameters are essential for the automation scripts to interact with network devices. Another useful file called defaults.yaml, it defines default values for variables and data that can be applied to all or a subset of network devices. The defaults.yaml file works in conjunction with hosts.yaml and group.yaml files to provide a hierarchical way of managing device attributes. Also, creating a virtual environment like .venv with this command `Python -m venv .venv` is recommended for automation projects to establish an isolated environment. This isolation is beneficial for different network automation projects that require project-specific packages. Moreover, it will avoid conflicts between Python packages installed globally on the same computer. Furthermore, creating a virtual environment provides a safe, sandboxed space to experiment with different libraries and versions without affecting the system's stability. Finally, by using `python3 -m pip install nornir` and `python3 -m pip install nornir-scrapli`, everything will be ready for creating our runbook.py file

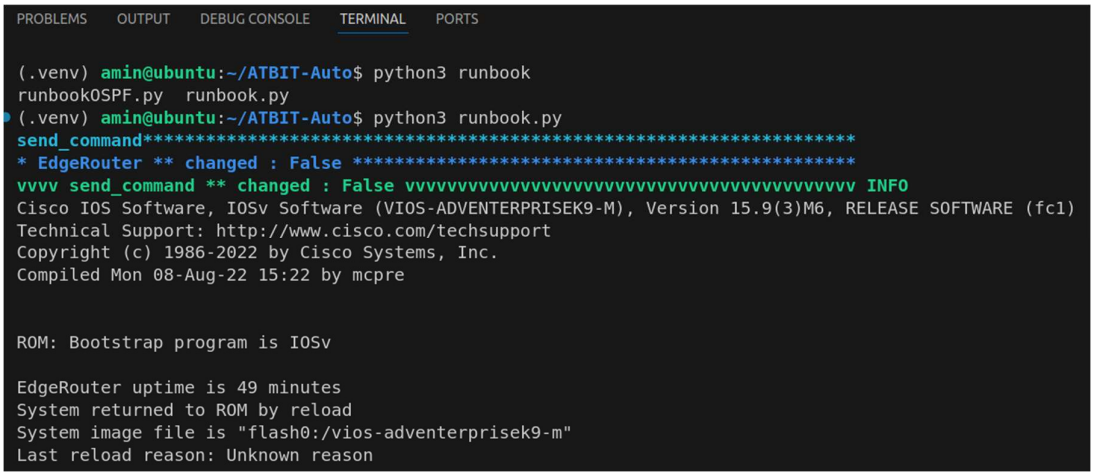
in order to start our network automation with Python. Figure 9 illustrates the simple runbook.py script, created in Python, to retrieve the IOS version from multiple Cisco devices simultaneously. This approach avoids the need to connect to each device individually and execute the `show version` command.

Figure 9 runbook.py script to retrieve the Cisco IOS version



Every time different commands and templates are pushed via Python runbook scripts, the response will be displayed in our VS Code terminal window. This response allows us to verify if the configuration was completed successfully or if an error occurred. Figure 10 provides a visual representation of the response messages displayed in VSCode each time our commands are successfully pushed to the Cisco devices.

Figure 10 Python runbook result in VSCode



In most cases, creating a general template is necessary. Consider a scenario where ATBIT company's customer needs to configure OSPF between multiple data center devices within one of their data centers. Either all 15 routers present at that time could be connected to, and the OSPF commands manually configured on each, or a template could be created in Python and the configuration pushed to all the routers simultaneously. Of course, a production environment typically contains far more than 15 routers. In large enterprise environments, network engineers often need to configure various protocols and rules simultaneously across hundreds or even thousands of devices. Without automation, this process will be significantly time-consuming. The data presented in the Command 2 illustrates the Python script used to create an OSPF template, ready to be pushed to multiple routers simultaneously based on the hosts.yaml file created suitable for the ATBIT project.

Command 2 Python script sample to create OSPF template

```
!
{% if host.test.OSPF is defined %}
router ospf {{ host.test.OSPF.process_id }}
 router-id {{ host.test.OSPF.rid }}
{% if host.test.OSPF.redistributing is defined %}
{% for policy in host.test.OSPF.redistributing %}
 redistributing {{ policy }}
{% endfor %}
{% endif %}
{% if host.test.OSPF.networks is defined %}
{% for network in host.test.OSPF.networks %}
 network {{ network }}
{% endfor %}
{% endif %}
{% endif %}
!
```

Ansible is another framework suitable for automation. These days, network engineers are using Ansible to automate their tasks and configurations instead of performing box-by-box configuration. It is an open-source automation tool written in Python language. It could be installed in a wide variety of operating systems like Linux distributions (Ubuntu, Red Hat, CentOS and many others). For example, if an engineer wants to configure OSPF across multiple routers, it will be a significant challenge. Network engineers can use Ansible and Ansible Tower to create an Ansible playbook, which serves as a template, and then push the configuration to all the routers. Ansible offers a multi-vendor automation solution that benefits not only software developers but also network engineers. These engineers leverage Ansible to simplify their daily tasks and establish a centralized platform for deploying configurations. Moreover, Ansible's agentless architecture eliminates the need to install software on network devices for configuration management. Instead, it utilizes

Secure Shell (SSH) to concurrently connect to and configure multiple devices. With Ansible network engineers can do critical tasks like provisioning infrastructure devices, managing complex configurations, enabling continuous integration and simplifying security processes automatically.

4 What is Cisco ACI?

As a leading Software-Defined Networking (SDN) solution, Cisco ACI relies on the Application Policy Infrastructure Controller (APIC) for management. This APIC is regarded as the central intelligence or brain of the Cisco ACI fabric. Cisco SDN is a software-driven methodology for managing infrastructure. This is a solution that is changing the way of configuration and deploying network infrastructures. Instead of logging in to the infrastructure via the Command-Line Interface (CLI), it will be necessary to log in to the controller and push all of our configurations to the network devices.

ACI uses a high-level object-oriented model that allows network engineers to manage their network devices and virtualizations, which can be fully configured by using an open Representational State Transfer (REST) API. Cisco ACI enables us to integrate physical devices and virtual machines, like any cloud-native environment or VMware ESXi, together with the benefits of programmability and AI. Cisco ACI simplifies network operations across deployment, management, and monitoring through its common policy framework. This approach enables rapid application deployment and causes to a considerable decrease in network administrative overhead. (Fordham, 2017).

In traditional network and data center environments, programmability may be the last thing that network engineers think about. However, in Cisco ACI, programming the fabric using API calls is preferred. The programmatic nature of ACI enables network agility; it allows network and security engineers to script out changes and decrease repeatable tasks to save time and enforce configuration standardization. For companies and engineers who are looking to decrease the need for box-by-box changes and configurations through automation and orchestration, ACI programmability offers more options.

4.1 Cisco Application Policy Infrastructure Controller

This central management capability allows operators to build fully automated and scalable multi-tenant infrastructures. The APIC offers a single point for policy-based configuration, utilizing group-based policies to simplify operations. It offers several key features include the defining and enforcing of application-centric policies, an open framework supported by northbound and southbound APIs, integration with third-party Layer 4-7 services, virtualization, and management tools, intelligent telemetry and visibility for applications and

tenants, scalable security for multi-tenant environments, and a consistent policy platform across physical, virtual, container, and cloud networking (*LTRDCN-2143*, n.d.).

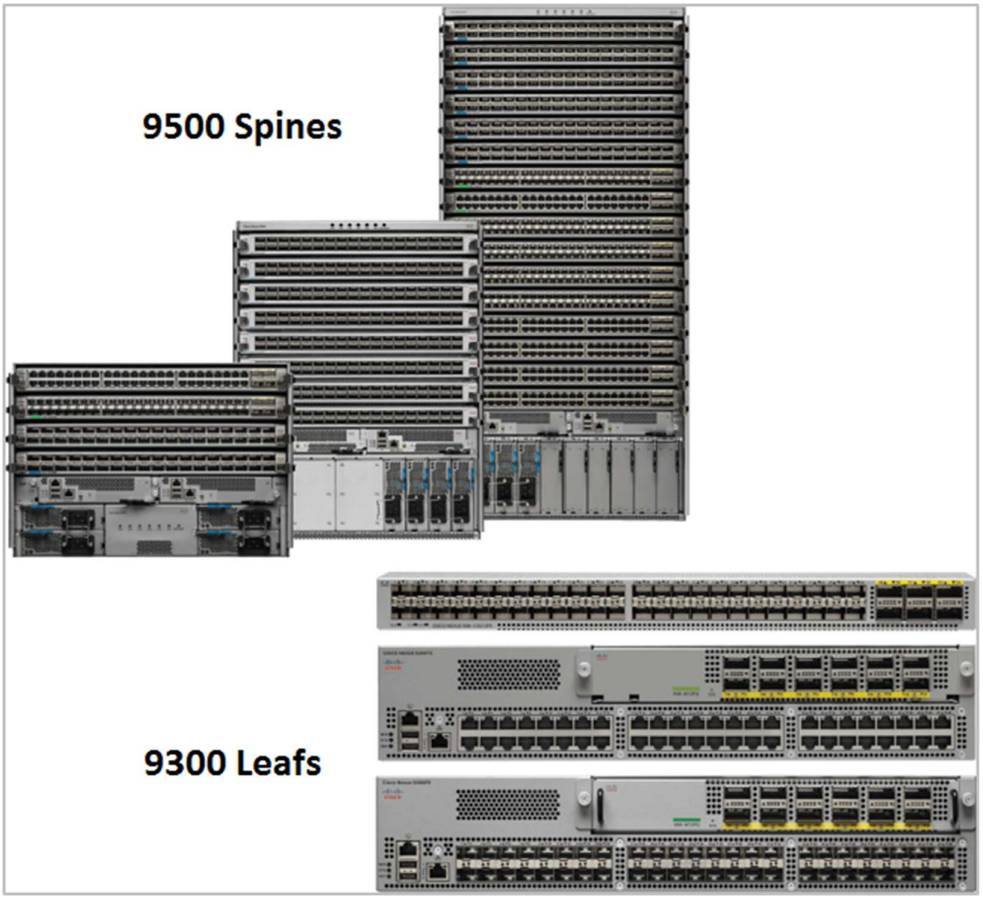
4.2 Cisco Fabric and Nexus Switch series suitable for the Cisco SDN

Cisco Nexus 9000 Series switches have two different modes: NX-OS and Application Centric Infrastructure (ACI). When using them in NX-OS mode, the Nexus 9000 switch utilizes a traditional OS to provide a platform for the existing network. This means that Cisco Nexus switches, when operating as NX-OS, must again be configured and managed individually via SSH or by using a tool like Ansible to centrally push configurations from an Ansible server to the Cisco Nexus switches. However, there is no controller present in this mode. Below, an example of an Ansible playbook can be seen that was created for one of ATBIT's customers to configure their VLANs and OSPF as one of their Dynamic Routing Protocols on their infrastructure devices, such as Cisco switches and routers.

Cisco Nexus 9000 Series devices can be configured in ACI mode. This means that the infrastructure devices will be controlled and managed centrally by a cluster of controllers, or more accurately, the Application Centric Infrastructure Controllers (APICs), of which there must be three within the same data center. ACI is Cisco’s primary SDN solution for the data center.

In an ACI fabric, Nexus switches operate based on the spine-and-leaf topology. Spine switches aggregate leaf switches, and leaf switches are used to connect to access layer devices. However, access layer devices, or let’s say endpoints here in ACI, are fundamentally different from how endpoints are conceived in traditional data centers. Within the ACI fabric, endpoints can include servers, routers, switches, firewalls, load balancers, cloud-native servers like OpenStack and Kubernetes, virtualization servers, and so on. Figure 11 provides a clear example of leaf and spine switches. These switches can be fully programmed by Cisco APIC and automate with its respective tools.

Figure 11 Cisco Nexus 9000 series switches (*Understanding ACI - Understanding ACI*, n.d.)



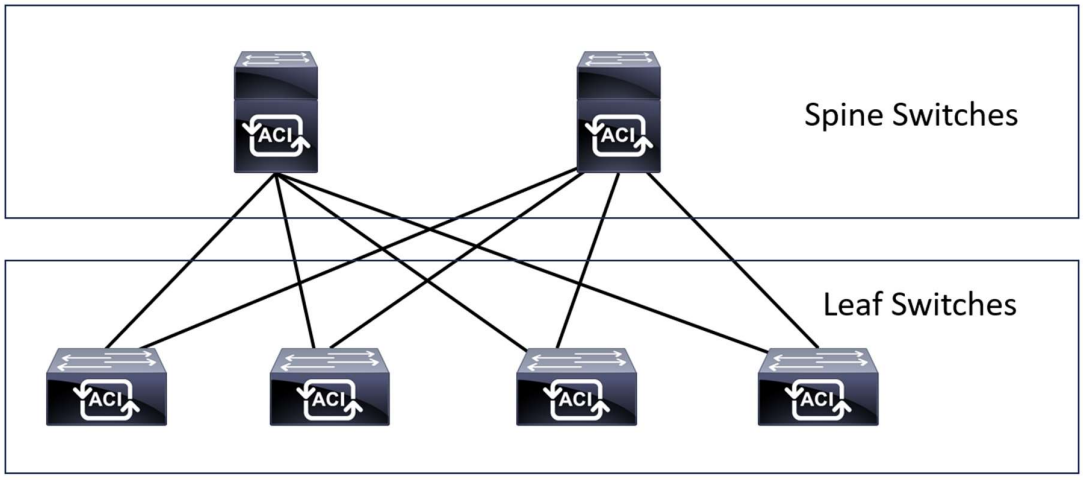
Cisco Nexus switches in ACI mode can only be programmed via Cisco (APIC) controllers that are known as an object-based policy engine of Cisco fabric. This information, formatted in either XML or JSON, resides in a profile to enable the enforcement of policies focused on applications, networking, and security.

4.3 Understanding ACI hardware and Topology

The Cisco ACI fabric is built upon the Clos (Charles Clos) leaf-and-spine topology. Within this fabric, each spine and leaf switch has a specific role. For instance, leaf switches serve as connectivity points for various hosts, including traditional out-of-band switches, servers, routers, firewalls, and cloud native solutions. Furthermore, it integrates with Rancher, an enterprise Kubernetes management platform and Kubernetes, the open-source container orchestration system. The Cisco fabric or the connections between spine and leaf switches,

differs from the approach used in traditional three-tier data center designs. Figure 12, shows a simple Cisco ACI fabric consisting of Clos-based leaf and spine topology. This topology is called two-tier leaf and spine topology and in enterprise environment it could be extended from two-tier topology to multi-tier topology that includes spine layer, tier 1 leaf switch and tier 2 leaf switch or more to provide a capability of vertical expansion for Cisco ACI that is useful for migration from traditional three-tier data center to the Cisco ACI. Moreover, this design can be stretched to the remote location if needed to interconnect remote data centers together with respect to the remote leaf switch functionality or solutions like Cisco ACI Multi-Site, Multi-Pod, vPod and public cloud integration that is not in the scope of this thesis.

Figure 12 Simple Clos-based leaf and spine topology



Cisco ACI controllers (APICs) reside outside the fabric and are responsible for provisioning, managing, and programming the entire infrastructure. Think of APICs as the brains of the ACI fabric. The critical tasks are handled by APICs, which sit outside the data path. By provisioning ability, network engineers can set up the network according to their desired configuration. Furthermore, the health and performance of the ACI fabric can be monitored by the management ability provided to the network by APICs. Instead of manual configuration, with APICs network engineers can program the leaf-and-spine switches. The Cisco fabric itself comprises leaf and spine switches, which are Cisco Nexus switches operating in ACI mode. These switches are programmable exclusively through an object-based Policy Engine (APIC). Typically, each Cisco fabric requires an APIC cluster, and the reason is to provide reliability. Having multiple APICs provides redundancy, so if one APIC encounters an issue, the others can take over, ensuring continuous network operation. This

cluster can consist of at least one controller, though a configuration of three APICs is recommended for redundancy. Figure 13 illustrates an example of Cisco APICs within the same fabric.

Figure 13 Cisco application centric infrastructure controller

Essentially, Cisco ACI combines a powerful and scalable hardware architecture (leaf-and-spine) with a centralized, policy-driven control system (APICs) to create a network that's not only fast and efficient but also highly programmable and adaptable. The Clos design, also known as the leaf-and-spine architecture, is a specialized network layout optimized for handling massive amounts of traffic, which is exactly what modern data centers need. Imagine a traditional network with a three-tier hierarchical structure, where devices are connected in layers. While this can work, it can also create bottlenecks and make it harder to scale. The leaf-and-spine design seems to be simple, but it has powerful implications. Later in this chapter, more information about the rules concerning leaf-and-spine switch connectivity can be found. However, for now, the main goals of this design, in comparison to three-tier architecture, will be focused upon. Because every leaf connects to every spine, traffic can be distributed very efficiently across the fabric, and with this design every single host is exactly two hops away from every other host in the fabric. There are multiple paths for data to travel, which minimizes congestion and maximizes throughput. In addition, the predictable path between any two devices (one leaf hop, one spine hop) ensures consistent and low latency, which is necessary for demanding applications. Moreover, adding capacity is relatively easy. More leaf switches can simply be added to add more access ports. To increase overall fabric capacity, more spine switches can be added. The architecture scales gracefully to handle growing network demands.

In the design and deployment of Cisco Application Centric Infrastructure (ACI) fabrics, engineers may strategically allocate specific roles to individual leaf switches to optimize performance and scalability. Several categories of leaf switches are typically considered for such dedicated hardware assignments. Border leaf switches serve as the crucial interface points, establishing both Layer 2 and Layer 3 connectivity between the internal ACI fabric and external networks. Furthermore, these border leaves can act as policy enforcement points, controlling traffic flow between internal and external endpoints.

Service leaf switches, in contrast, are designated for connecting to Layer 4 through Layer 7 network services, such as firewalls and load balancers, ensuring dedicated pathways for traffic requiring these advanced functions.

Compute leaf switches directly connect to server infrastructure within the ACI fabric. Notably, these leaves also enforce policies governing traffic exchange between endpoints residing within the local fabric. Finally, IP storage leaf switches provide dedicated connectivity to IP-based storage systems and can similarly enforce policies for traffic originating from or destined to locally connected endpoints.

While dedicating leaf switches to specific roles offers inherent scalability advantages by isolating functions and potentially increasing capacity for each, the network's overall size and requirements should be carefully evaluated. In scenarios where the scale does not necessitate dedicated hardware for each function, a practical approach involves, at a minimum, dedicating a redundant pair of leaf switches as border leaves for external connectivity. Moreover, the functionality of service leaves can be optionally consolidated with that of border leaves, leading to the deployment of a pair (or more) of combined borders/service leaf switches, particularly in smaller ACI deployments, to balance functionality with resource utilization.

4.4 Cisco ACI Fundamentals and Terminology

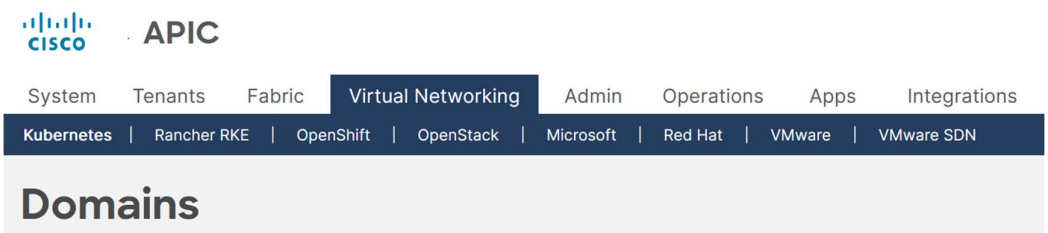
The foundational components of Cisco ACI will be explored in this chapter. To understand Cisco ACI, it is essential to know its exact components. Following that, understanding the main differences between traditional data centers and Cisco SDN. As mentioned earlier in this thesis, ACI functions are based on objects, and the Cisco Application Centric Infrastructure Controller (APIC) operates on the basis of an object-based policy engine. This enables network engineers to define the desired states of the fabric and entrust the

implementation to the controllers (APICs). It is also important to know that these policies are defined based on Tenants, which are objects within the Cisco fabric (*Cisco Application Policy Infrastructure Controller (APIC) - Cisco Application Centric Infrastructure (ACI) Design Guide*, n.d.).

One of the key rules in the Cisco fabric is that neither two leaf switches nor two spine switches should be directly connected to each other. Compared to the traditional architecture, where nodes like servers, virtual machines, or any type of endpoint that needs to connect to the access layer are added there, in the Cisco fabric, endpoints must connect directly to the leaf switches.

ATBIT company designs, implements, and provides customer service for organizations that want to use cloud solutions like Microsoft Azure, Google Cloud Platform (GCP), and Amazon Web Services (AWS), or even on-premise cloud solutions like OpenStack. One of the main key values of Cisco ACI is its support for multi-cloud environments, so ATBIT can offer a single central environment to manage and extend their customers' policies and security controls to the public cloud. One of the main challenges for ATBIT was that their customers might use different cloud environments and want to connect their data centers from different places in the world securely. Now imagine a company with different customers and data centers running on multiple public and on-premise cloud-native environments that must transmit data together. For example, if a large company built its data center on Google Cloud Platform (GCP) and another customer is using Amazon Web Services (AWS), these two customers' data centers should connect and be managed together with a single solution. Even if a larger data center is using Cisco APIC, the scenario is to connect all of these data centers together. Another important challenge is that the terms used in each environment differ, so ATBIT needs a solution with the ability to integrate with all of those technologies. Figure 14 illustrates a Cisco ACI integration sample. The image provided originates from the Cisco sandbox lab where work was being conducted.

Figure 14 Cisco ACI integration solutions



As Figure 14 shows, Cisco ACI can integrate with third party vendors. Moreover, it can integrate with other third-party tools like Terraform from HashiCorp and Red Hat Ansible (*Cisco Application Centric Infrastructure - Cisco Application Centric Infrastructure (Cisco ACI) Solution Overview*, n.d.).

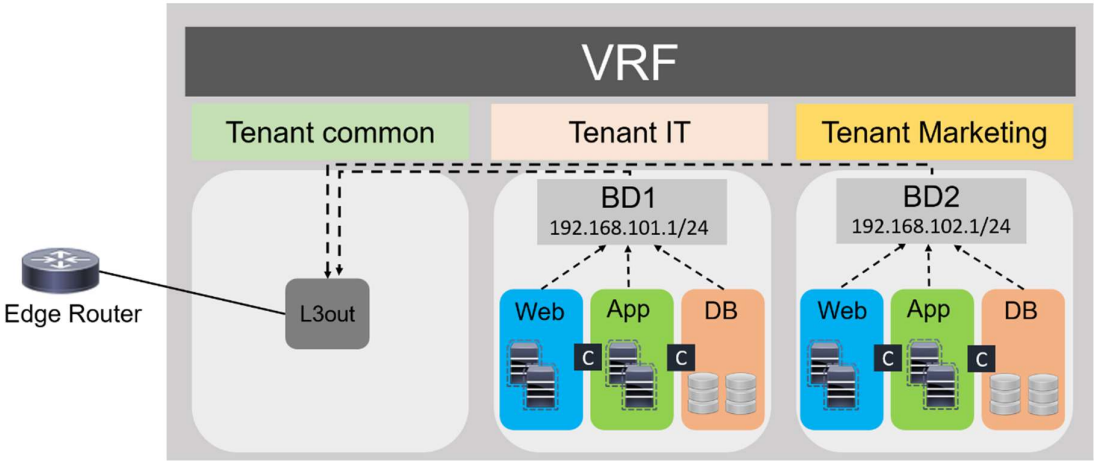
To understand the main difference between Cisco's SDN-based solution compared to the traditional architecture, discussing Cisco ACI terminology is necessary. The main components of an ACI deployment include leaf switches, border leaf switches, spine switches, and APIC controllers. Leaf switches are responsible for connecting network nodes into the fabric at the Top-of-Rack (ToR) or End-of-Row (EoR). Compared to the traditional architecture, leaf switches function like distributed Layer 3 switches (gateways for services) and are the precise points where policies defined by the APIC are enforced on the network connecting to external networks. However, border leaf switches are used to connect other network devices such as firewalls, routers, load balancers, or any non-ACI switches. Importantly, a smooth migration from the traditional architecture to an ACI network is allowed by border leaf switches. Furthermore, spine switches provide a non-blocking fabric capable of rapidly detecting failures and even re-routing packets. The main responsibility of these switches is to transit traffic between any two leaf switches. Now imagine a specific VLAN, like VLAN number 10, with different web services in the same network connected to different leaf switches. VLANs carry Layer 2 traffic, and each node in the same VLAN must connect to another within the same network using Layer 2 protocols like ARP and destination MAC addresses. Creating Layer 2 boundaries in a traditional environment is very risky and can lead to loop issues. But here, in Cisco ACI-based data centers, spine switches handle this Layer 2 traffic. Additionally, spine switches can extend the VLAN limitation from 4096 VLANs to an effectively unlimited number by using VXLAN. Another important component within the fabric is the APIC controller. As I mentioned before, it is the central point of management for fabric configuration.

Compared to traditional networks, the desired configuration and states of the fabric are defined by network engineers and administrators, but the implementation tasks are left to the controller. Cisco APIC then will take care that policies that take place for the network nodes or not. Policies, when created, must be defined for a specific object within the fabric called a Tenant (*ACI Terminology - Understanding ACI*, n.d.).

Tenants are the logical objects within the Cisco fabric that allow us to create departments, network domains, and application policies. Think of a tenant as a logical container for application policies, enabling network engineers to create different domains. Policies can

be attached to nodes within the same tenant or between different tenants. From a design perspective, a single-tenant or multi-tenant design is possible based on the company's scenario and scale. Moreover, tenants represent a customer or a company in a service provider environment. Even in an enterprise organization, tenants could represent different departments (*Mso-Configuration-Aci-Managing-Tenants-33x.Pdf*, n.d.). Figure 15 represents a brief logical view of tenants in the Cisco ACI fabric.

Figure 15 Logical view of tenants in the Cisco ACI fabric



Tenants are a top-level object in ACI that provide administrative control, network/failure domain separation and application policy isolation. The sub-objects within a tenant fall into two primary categories: Tenant Networking and Tenant Policy. While these categories are inherently related, discussing them separately can be beneficial for understanding their respective roles.

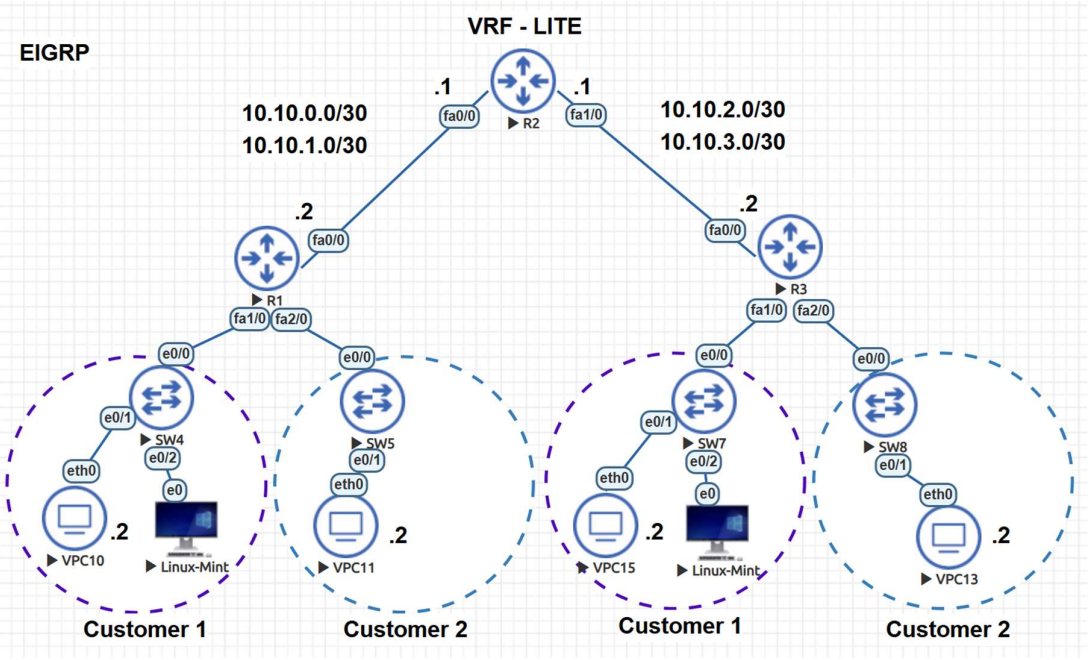
Within the Cisco fabric, there is a special tenant called the Common tenant. The purpose of using this tenant is to share its resources and services with other tenants, such as Domain Name Service (DNS), authentication services like Authentication, Authorization, and Accounting (AAA), or any other services that need to be shared. Furthermore, defining network objects and policies like Contracts and Filter rules across multiple tenants is possible with the Common tenant (*ACI Terminology - Understanding ACI*, n.d.).

Tenant Networking is another important term that needs to be understood before deep diving into the Cisco ACI. Let’s say, Tenant Networking is another object within the fabric. This tenant is responsible for providing layer two and layer three connections between hosts and services within the fabric. It consists of different Virtual Routing and Forwarding

(VRF), Bridge Domains, Subnets, and External Networks (*Cisco Application Centric Infrastructure - ACI Multi-Pod White Paper*, n.d.).

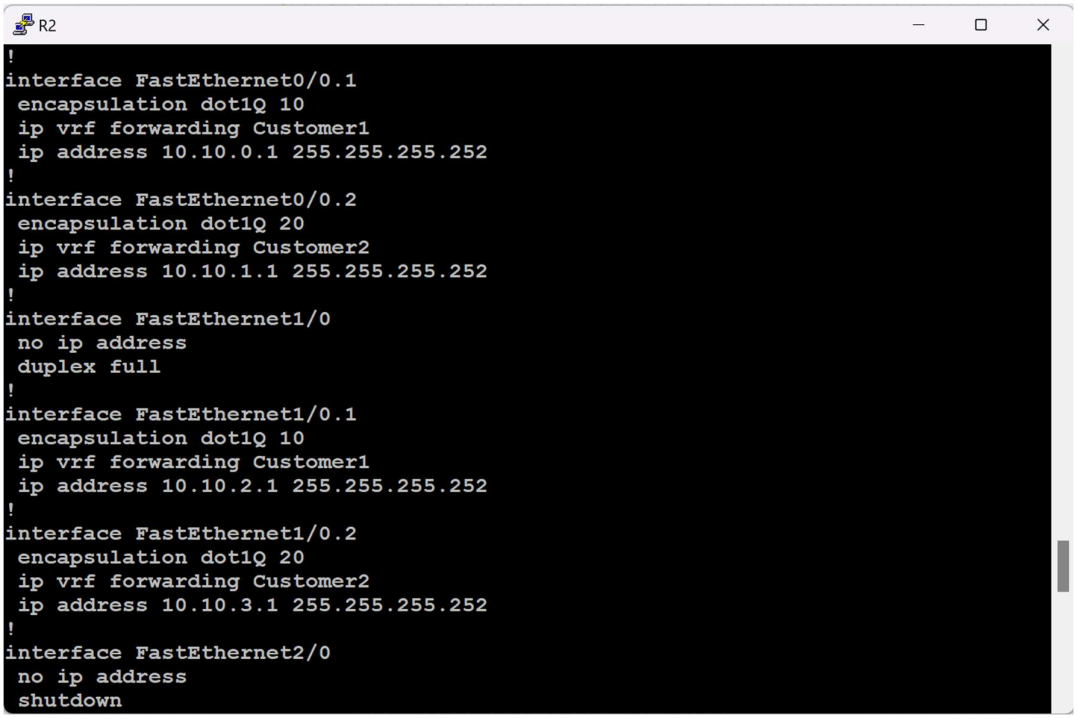
VRF, or Virtual Routing and Forwarding, is a well-known term that has been used in traditional networking for a long time. To connect different networks or broadcast domains, using a Layer 3 device called a router is essential. Each router can route packets based on its routing table. Routers typically have one main routing table that can be logically divided into unlimited, logically separated routing tables called VRFs. One of the main goals of using multiple VRFs is to isolate the traffic of multiple customers on the same router. By using this technique, the traffic of different customers with the same overlapping IP addressing ranges can transit within the same backbone. Figure 16 illustrates a scenario I designed in the EVE-NG environment to understand the logic behind VRFs in a traditional setup (Rahman et al., 2016).

Figure 16 VRF lite sample scenario in eve-ng



In Figure 16, routers R1, R2, and R3 acted as service providers to transit both customers 1 and 2 data centers. Enhanced Interior Gateway Protocol (EIGRP) (Savage et al., 2016) is used here as a dynamic routing protocol between service provider routers. Furthermore, due to the absence of Multi-Protocol Label Switching (MPLS) (Rahman et al., 2016) in this configuration, VRF-Lite was configured. A part of the configuration for this scenario has been provided in Figure 17.

Figure 17 Router R2 VRF configuration sample



The primary difference regarding VRF lies in the configuration methods employed by network engineers in traditional networking versus Cisco ACI, although the underlying concept remains consistent across both environments.

Within the Tenant Networking category, Bridge Domains play a crucial role. A Bridge Domain in ACI is essentially a Layer 2 forwarding construct. Think of it like a traditional VLAN, but with enhanced capabilities and a more policy-driven approach. Bridge Domains define a broadcast domain within the ACI fabric, allowing devices within the same bridge domain to communicate with each other via bridging. Bridge Domains provide a way to configure Layer 2 forwarding behavior, including how traffic is flooded, and whether or not unicast traffic is routed. Importantly, a Bridge Domain can span multiple leaf switches, providing flexibility in how a layer 2 network can be segmented. Unlike the way VLANs have been traditionally deployed, each Bridge Domain can contain multiple subnets (*Understanding ACI - Understanding ACI*, n.d.).

Another important term in Cisco ACI is called Subnets. Subnets are the layer 3 networks that provide IP space and gateway services for hosts to connect to the network (*Understanding ACI - Understanding ACI*, n.d.). Subnets play a crucial role in Cisco APICs by defining a layer 3 forwarding boundaries within the same Bridge Domains (BDs). Some

of the most important functions of Subnets in Cisco ACI are layer 3 connectivity and gateway, Anycast gateway, Switch Virtual Interface (SVI) creation, Layer 3 Out(L3Out) connections Virtual Routing and Forwarding (VRF), integration with Bridge Domains, VRF leaking, and Dynamic Host Configuration Protocol (DHCP) relay.

External Bridge Network or simply Layer 2 Outside (L2Out) is another terminology in Cisco Fabric. L2Out in Cisco ACI is useful to extend a Bridge Domain from the same ACI fabric to an external Layer 2 network. For example, if a legacy layer 2 infrastructure or existing switches or devices on a particular VLAN present and integration with applications or services within the ACI fabric on the same layer 2 segment is needed, then L2Out allows that VLAN to be extended into the ACI Bridge Domain. It means devices both inside and outside of the Cisco fabric can communicate with each other without any routing protocol and layer 3 process. L2Out is suitable for the migration scenarios from the traditional network designs to the Cisco SDN design.

Cisco ACI often needs to connect to networks beyond its fabric using External Routed Networks (L3Out). Cisco ACI commonly integrates with external networks through External Routed Networks (L3Outs). An L3Out serves as a Layer 3 gateway, enabling IP routing and the exchange of routing information (using protocols like OSPF, BGP or static routes) between the ACI fabric and external domains. This integration relies on External Endpoint Groups (External EPGs), which represent external network address spaces, and L3Out Instances, defining routing protocols, participating leaf switches and interfaces, and the isolating Virtual Routing and Forwarding (VRF) instance. Often, an optional External Bridge Domain provides the necessary Layer 2 connectivity for the routed interfaces on leaf switches to communicate with external routers. Furthermore, Contracts can be applied to External EPGs to control traffic flow between internal and external networks. There is an important consideration in Cisco ACI that each WAN router must support MP-BGP EVPN, OpFlex and VXLAN to connect ACI fabric to the Wide Area Network (WAN). Figure 18 visualizes the existing data center designed for one of the main ATBIT customers and shows the External End Point Group (EPG) connection with the Cisco fabric. These leaf switches act as the VXLAN Tunnel Endpoints (VTEPs) in the ACI fabric. As traffic enters the ACI fabric from a connected endpoint like servers, the ingress leaf switch encapsulates the original Layer 2 frame with a VXLAN header. This process involves adding the VXLAN Network Identifier (VNID), which correlates the traffic to its respective Bridge Domain and Tenant.

Figure 18 External End Point Group (EPG) design

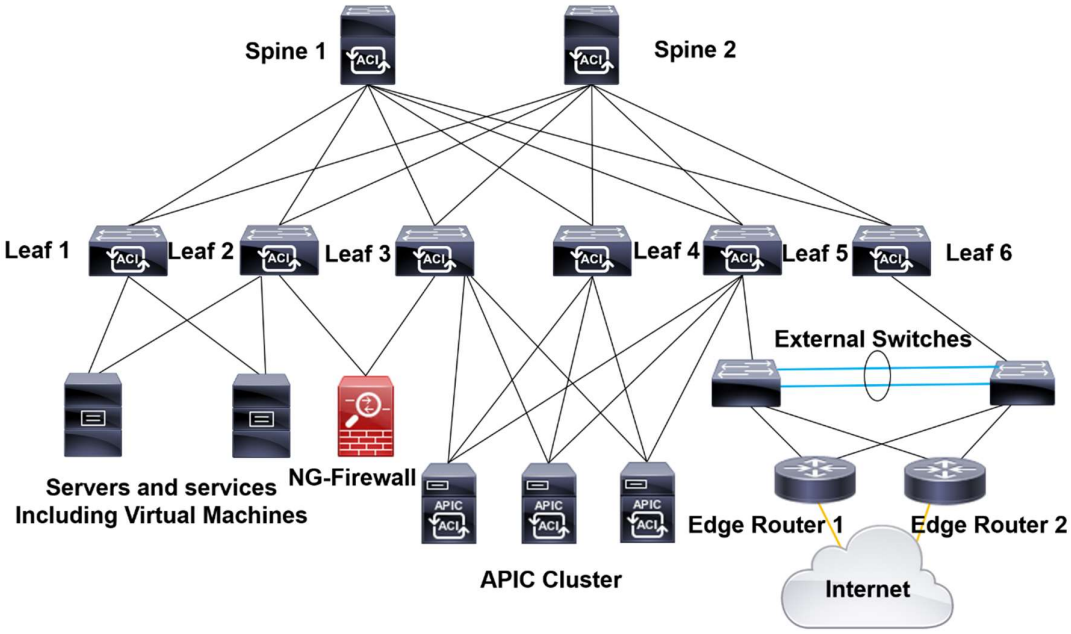


Figure 18 illustrates the simple Cisco ACI fabric architecture. At its core, there are two Spine switches lying centrally in the fabric and are shown by Spine 1 and Spine 2. Furthermore, Leaf Layer switches, consisting of Leaf 1 through Leaf 6, are present. The spine switches, characterized by their high-capacity and non-blocking nature, form the network's central transit fabric. Each leaf switch establishes a full-mesh connectivity with every spine switch, a design principle that ensures the existence of multiple equal-cost paths between any two endpoints within the fabric. This comprehensive interconnection is fundamental to achieving low-latency communication and maximizing available bandwidth across the network. Within this leaf layer architecture, if ATBIT customers require supplementary ports to connect their virtual machine servers, such as cloud-native or physical servers, the fundamental data center design remains unaffected. The flexibility to integrate new servers or services into any of these leaf switches is possessed by Spine switches. This adaptability extends to the transit of Layer 2 traffic, like VLANs. For instance, consider an OpenStack server environment distributed across Leaf 1 and Leaf 2, where each leaf supports multiple logical data centers with the capacity for up to 3000 VLANs (the system default being 4096 VLANs). If their requirement is to establish a backup for each of these OpenStack deployments, maintaining identical VLANs and IP address schemes (Subnets), this architecture facilitates a straightforward solution and can readily incorporate additional OpenStack servers, along with their associated network configurations, by connecting them to other leaf switches, such as Leaf 3 and Leaf 5. The Leaf Layer,

conversely, serves as the primary point of attachment for various endpoints. As depicted, this layer hosts critical resources such as servers and associated services, including virtualized environments, highlighting its role in supporting computational workloads. Furthermore, the strategic placement of a Next-Generation Firewall (NG-Firewall) connected to Leaf 3 underscores the integration of security services directly within the ACI fabric, enabling granular control and protection of data flows. Moreover, an Out-Of-Band (OOB) management cluster, composed of three fully redundant Application Policy Infrastructure Controllers (APICs) provides a unified management interface for the entire fabric. APICs are responsible for translating application-centric policies into network configurations, automating the provisioning and management of network resources, and providing comprehensive health monitoring of the infrastructure.

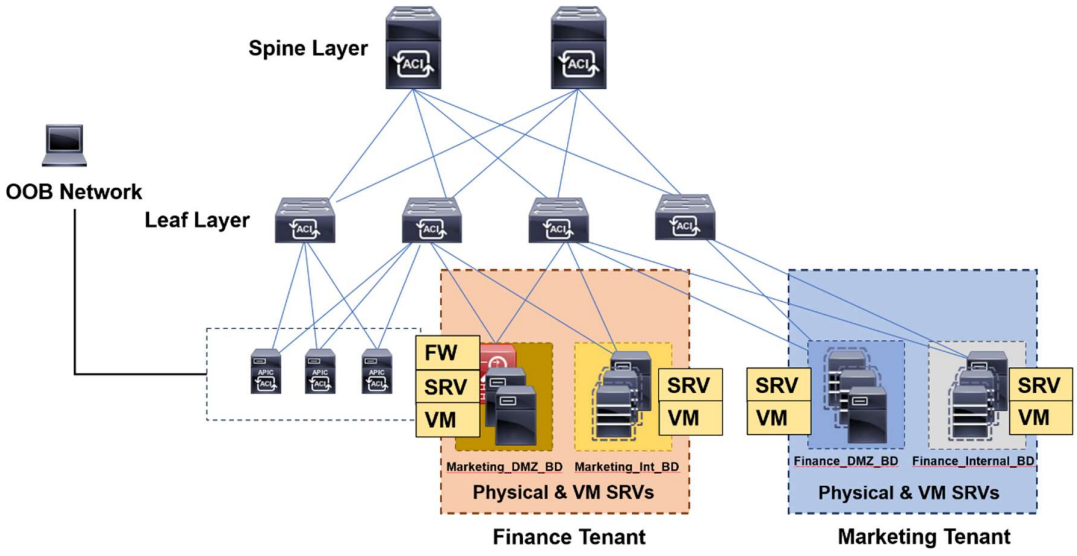
Connectivity beyond the ACI fabric is facilitated through External Switches, which are connected to specific leaf switches (Leaf 5 and Leaf 6 in this instance). These external switches, in turn, provide a gateway to Edge Routers (Edge Router 1 and Edge Router 2), which are responsible for routing traffic to external networks, as demonstrated by their connection to the Internet. This hierarchical yet interconnected design allows for seamless communication between resources within the data center and the external world. Think of the Edge router as a dual-home Internet Service Provider (ISP) designed to provide an internet connection redundancy over two different ISPs that is suitable for an enterprise network solution. By this design, their customer can leverage a Border Gateway Protocol (BGP) (Rekhter et al., 2006) to connect to two ISPs and provide a high available connection to their data center from outside. A primary requirement for their customer involved external connectivity from the internet, necessitating the establishment of an IPsec tunnel with their edge routers (Edge Router 1 and 2). However, ATBIT customer’s specified that only a single public IP address should be visible externally. Given that both edge routers were Cisco devices, High Availability solutions such as Hot Standby Routing Protocol (HSRP) (Li et al., 1998) and Gateway Load Balancing Protocol (GLBP) (*GLBP - Gateway Load Balancing Protocol*, n.d.) were presented. The customer ultimately selected HSRP to meet this redundancy and single IP address requirement.

Fundamentally, this architecture exemplifies a forward-thinking, flexible, and dependable methodology for data center networking, specifically refined to meet the requirements of present-day application deployment and operational control.

4.5 Isolating and securing data centers with ACI Tenant and VRF

In this chapter, I designed a scenario for one of ATBIT customers to simulate Cisco ACI in a lab environment. Consideration was being given to them to how Cisco ACI could isolate and secure important parts of their network. Cisco ACI supports different methodologies to achieve this goal, but at that time, I preferred to use tenants and Virtual Routing and Forwarding (VRFs) to create highly secure and isolated network segments (like departments). This method is particularly valuable in multi-tenant environments, such as data centers or cloud service providers, where multiple organizations or departments need to share the same physical infrastructure while maintaining strict separation of their data and traffic. ATBIT has a customer that started migrating from a traditional data center to Cisco SDN, and below a scenario that completely separates two different departments by using Virtual Routing and Forwarding (VRFs) within two different tenants is provided. The tenants are named after their respective departments. Multiple departments are present, and an example using the Finance and Marketing departments will be provided. Moreover, in Figure 19, a cluster of three APICs are visible, but in the provided Cisco LAB, only one APIC is used which is not recommended in a production environment.

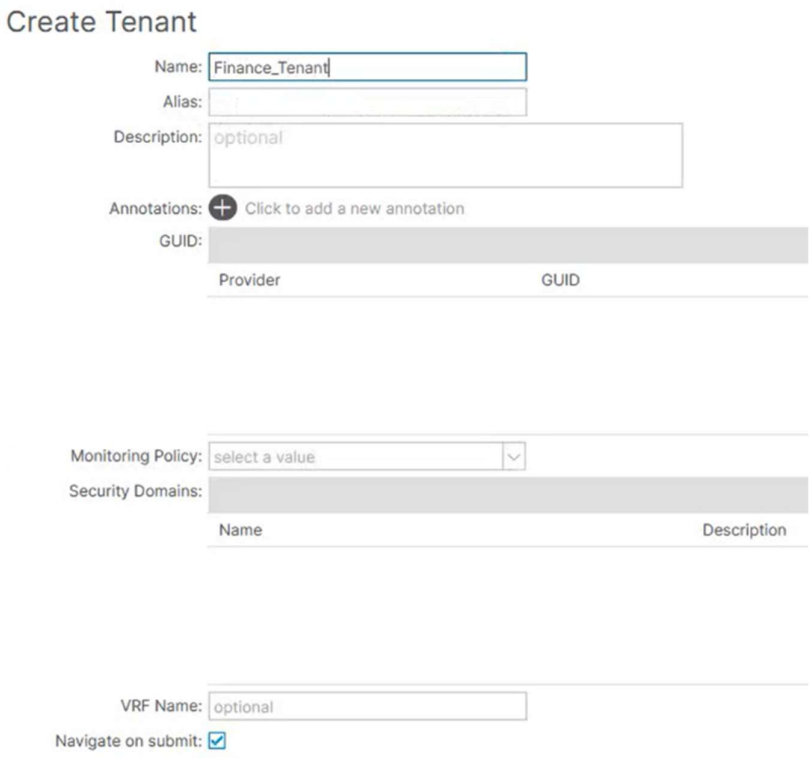
Figure 19 Isolating and Securing different departments in Cisco ACI



To configure the scenario, after logging into the Cisco APIC panel and navigating to the Tenants tab, new tenants called Finance and Marketing can be created. Figure 20 Creating

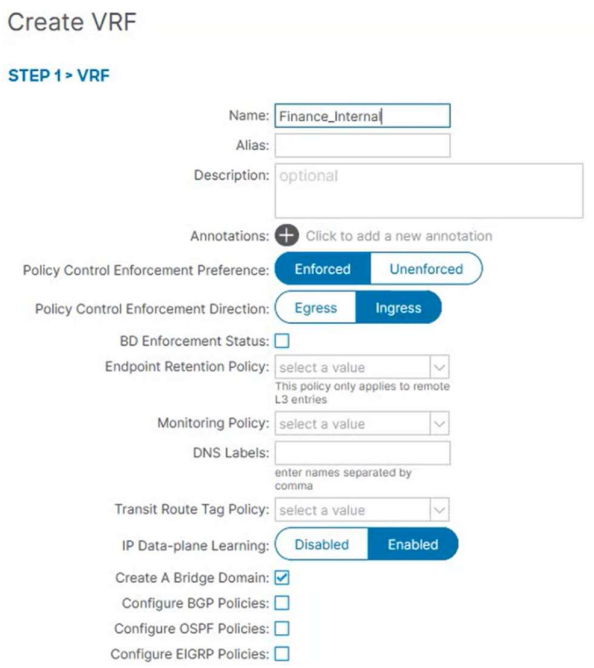
a new tenant in Cisco ACI fabric illustrates how to configure the new tenant in Cisco ACI fabric.

Figure 20 Creating a new tenant in Cisco ACI fabric



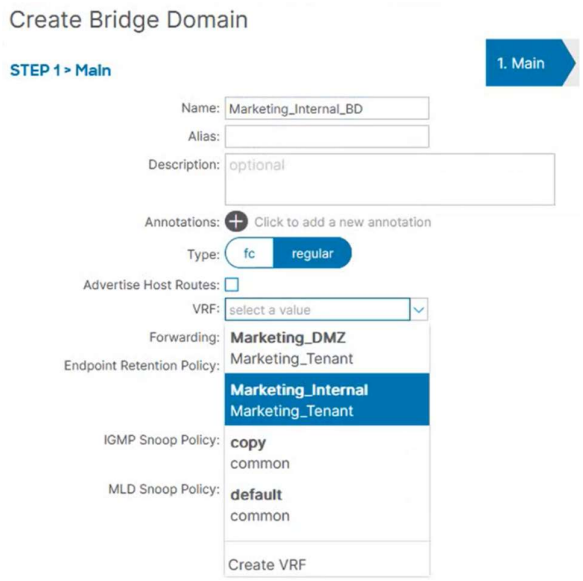
Upon creating a tenant, all objects are automatically created and available for modification. After creating the desired tenants, VRF can be configured under the Networking tab within the Finance tenant. Figure 21 demonstrates how a new VRF is created in Cisco ACI using the Graphical User Interface, compared with the traditional model presented earlier in this thesis. To understand Cisco ACI deeply, first it is required to understand relations between each object. For example, each Tenant may include multiple VRFs in the same scenario but there is no possibility to share the same VRF between multiple Tenants. Furthermore, you may prefer to have multiple bridge domain inside of one Tenants, so there is a one-to-many relationship between Tenant and the Bridge Domain as well as VRF. Another important rule in Cisco ACI object structure model is parent and child relations. It will help to understand the structure of configuration, even manually or by automation solutions like REST API. Each tenant acts as a parent for VRF, bridge domain, application profile, L3Out, contract and filter. It assumes each of them to be a child for the same tenant. In addition, VRF and bridge domain have a referential relationship together.

Figure 21 Creating a VRF in APIC



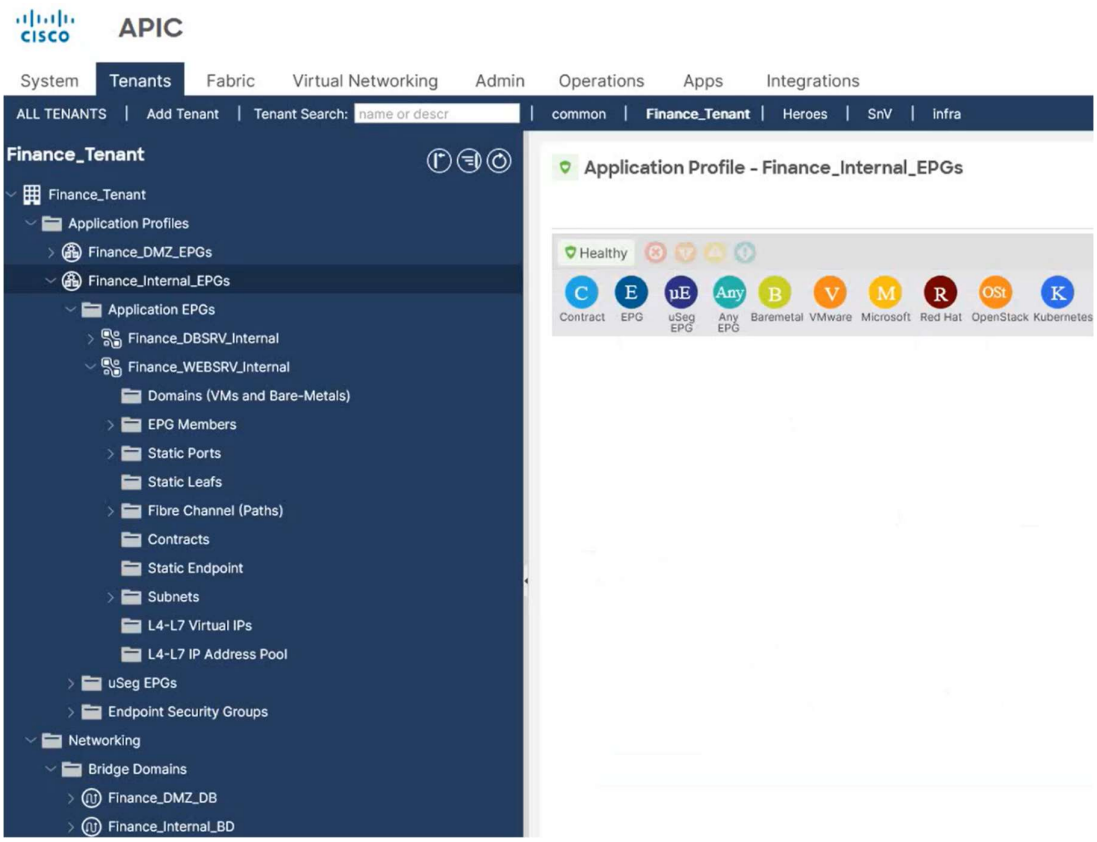
With specific VRFs created, the configuration and attachment of a new Bridge Domain to its corresponding VRF is vital. Figure 22 illustrates the procedures of creating a new Bridge Domain in the Cisco fabric.

Figure 22 Creating a new Bridge Domain in Cisco APIC



After a specific subnet is configured for the desired bridge domain and a correct gateway is set for network reachability, new EPGs are created under the Application Profiles section. Finally, by creating different Contracts that are similar to Access Control List (ACL) or simply firewall rules in the traditional networks, it can be determined which of these EPGs will act as the source of the packet and which will act as a destination of the packets. As shown in Figure 23, all data center components, such as devices, VMs, servers, and other objects, are accessible and can be monitored.

Figure 23 Finance tenant objects verification



5 How Artificial Intelligence can improve Cisco data centers

Cisco ACI primarily focuses on automation and policy-driven infrastructure and benefits significantly from automation and orchestration. Nowadays, Artificial Intelligence (AI) further enhances it by analyzing network patterns, anticipating needs, and automatically adjusting configurations to achieve optimal performance. Moreover, it enhances security through a capability called micro-segmentation, which provides a strong security foundation. Furthermore, AI improves this security mechanism by enabling advanced threat detection. Over the past years, Cisco has benefited from AI across its product lines; for example, it offers a solution called Cisco Digital Network Architecture (DNA) Center, which uses AI and machine learning for anomaly detection, predictive maintenance and traffic analysis. Even more, AI/ML can help many other industries like those addressed in Table 1.

Table 1 Industries examples that can improve by AI/ML

Healthcare	Finance	E-commerce	Manufacturing	Transportation	Education
Diagnosis and early detection	Customer service	Dynamic pricing	Predictive maintenance	Intelligence vehicles	Personalized learning
Robotic surgery	Trading analysis	Customer experience management	Quality control	Route optimization	Intelligent tutoring systems
Medical risk prediction	Fraud detection	Chatbots for customer support	Robotics and automation	Warehouse automation	Administrative tasks

Moreover, Cisco ACI has partnerships with companies like Nvidia to produce AI infrastructure solutions primarily for data centers. These characteristics help ACI to handle AI workloads (*AI Solutions on Cisco Infrastructure Essentials | Cisco U. Path*, n.d.-a).

5.1 AI Basics

AI systems now possess the ability to learn and improve their performance independently by processing data repeatedly. Early forms of these systems operated based on fixed instructions. A simple illustration is how early spam filters worked: AI systems can identify spam by checking for specific words in emails, following basic "if this, then that" rules.

However, current systems are much more advanced. Instead of just following set of rules, AI can learn from large amounts of data. This involves techniques where the system's ability improves as it is exposed to more examples. For instance, in image recognition, these systems are shown many labeled images, allowing them to learn to distinguish between different objects without being explicitly told every single difference. Self-driving vehicles offer another example; roads are learned to be navigated by them through the processing of data from numerous driving situations.

A significant recent development is the emergence of systems through which content can be created. Tools that can generate text or manipulate images might be familiar. This indicates a move beyond simply analyzing data to the independent generation of new content, such as text, images, or even music (*AI Solutions on Cisco Infrastructure Essentials | Cisco U. Path*, n.d.-b).

5.1.1 Machine Learning vs Deep learning Applications and Use cases

Machine learning (ML) and Deep learning (DL) have different use cases based on their strengths and weaknesses. Machine learning is defined as a subset of Artificial Intelligence (AI) wherein computational systems acquire knowledge from data without reliance on explicit programming (*0AD125C9B7294083A6684FD823756EE6.Pdf*, n.d.). Rather than relying on built-in rules, machine learning algorithms recognize patterns from data to generate predictions or make decisions. ML models are more suitable for environments where the data and requirements are relatively static. Machine Learning is usually suitable for applications requiring high interpretability and transparency. Table 2 presents use cases for Machine and Deep learning (*AI Solutions on Cisco Infrastructure Essentials | Cisco U. Path*, n.d.-b).

Table 2 Machine Learning and Deep Learning use cases

Machine Learning	Deep learning
Predictive Analytics	Image Recognition
Recommendation Systems	Speech Recognition
Anomaly Detection	Natural Language Processing
Credit Scoring	Autonomous Driving
Predictive Maintenance	Medical Image Analysis

Customer Segmentation	Algorithmic Trading

Within the automotive sector, ML algorithms are integral to predictive maintenance. These algorithms analyze sensor data and they can forecast equipment malfunctions, enabling the scheduling of proactive repairs (ai-admin, 2023). This technology is also widely applied in areas for instance anomaly detection, recommendation systems and predictive analytics.

ML models are used by marketers to segment customers and predict their buying behavior based on structured data like past purchases and demographic information. Having structured data and the desire for clear actionable insights makes ML appropriate for these types of applications (*AI Solutions on Cisco Infrastructure Essentials* | *Cisco U. Path*, n.d.-a).

Machine learning is frequently chosen for applications where understanding and explaining how a decision is made are crucial. For instance, in finance, ML models can be used for credit scoring because their decision processes can be readily understood and explained to relevant parties, which is often a legal requirement. The automotive industry also benefits from ML algorithms for predictive maintenance. These algorithms analyze sensor data to forecast when equipment might fail, allowing for timely repairs to be scheduled. More broadly, ML is a common approach in predictive analytics, recommendation systems, and identifying unusual patterns. Marketers utilize ML models to divide customers into groups and predict their purchasing habits based on organized information like past transactions and demographics. The availability of structured data and the need for clear, actionable insights make ML suitable for these kinds of applications.

On the other hand, deep learning excels when dealing with intricate, unstructured data and achieving high precision in tasks such as recognizing images and speech. In healthcare, DL models are applied to the analysis of medical images. Their ability to automatically extract relevant features from large datasets can significantly enhance the accuracy of diagnoses. Self-driving vehicles rely on DL models to process and interpret visual information from cameras, enabling them to identify objects, detect lane markings, and make driving decisions. Voice-activated assistants, such as Siri and Alexa, also use DL to understand and respond to spoken commands by analyzing speech patterns. Furthermore, more complex tasks like sentiment analysis can employ DL models, where unstructured data from sources like social media posts and reviews is analyzed by DL models to

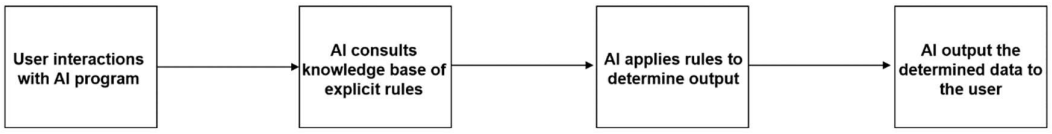
determine public sentiment towards products and brands.(*AI Solutions on Cisco Infrastructure Essentials* | *Cisco U. Path*, n.d.-a).

Cisco is actively integrating Artificial Intelligence (AI) capabilities into its data center solutions to address the increasing demands of modern workloads, particularly those related to AI and Machine Learning (ML) itself. This integration spans across their infrastructure, networking, and security offerings, aiming to create intelligent, efficient, and secure data center environments.

5.1.2 Traditional AI and its process flow

Traditional AI, often referred to as symbolic or rule-based AI, solves problems by applying explicitly coded rules and logical deduction. Unlike modern AI that learns from data, these systems' functions are based on a fixed body of knowledge and instructions directly input by human experts. The focus is on representing knowledge in a structured way, often using symbols, rules (like "if-then" statements), and logical frameworks. Figure 24 shows the traditional AI process flow:

Figure 24 Traditional AI process flow



Imagine a computer program being used that has been provided with a very specific set of instructions on how to solve a problem. This is similar to traditional AI. First, the AI program is interacted with by a user, and some information is given to it or a question is asked of it.

Next, instead of figuring things out on its own by looking at lots of examples, the AI looks at a special set of rules it was given by its creators. Think of it like a detailed instruction manual that tells it exactly what to do in different situations. These rules are like pre-written "if this happens, then do that" statements. The AI doesn't learn these rules. It's simply told what these rules are.

Then, the AI takes the user input and carefully follows those rules to figure out an answer or what to do next. It's like a detective using a fixed set of clues to solve a case. It goes

through the rules step-by-step. Finally, the AI shows the result on the user's screen and the answer it found by following the rules.

Fundamentally, traditional AI can be likened to a program that operates by strictly adhering to a set of pre-programmed instructions in response to input it receives. The ability to learn or improve autonomously is absent in this approach. Instead, the knowledge and rules that are explicitly provided to it are solely utilized.

5.1.3 Traditional AI Infrastructure

Traditional Artificial Intelligence systems, characterized by their reliance on explicitly defined rules and knowledge bases, required a distinct infrastructure compared to modern, data-driven approaches. The infrastructure supporting these early AI systems was often centered around powerful computing systems capable of executing complex logical operations and managing structured data. Key components included high-performance CPUs for symbolic processing, extensive memory to store knowledge bases, and specialized software environments for developing and running rule-based systems. Data storage was typically relational, organized in databases that could be efficiently queried using structured query languages. Networking infrastructure, while crucial, primarily focused on enabling communication between the system's components and end-users, rather than facilitating the massive data transfers associated with modern machine learning. In essence, traditional AI infrastructure was designed to support the predictable and rule-driven nature of these systems, emphasizing computational power and structured knowledge representation.

5.1.4 Modern AI Infrastructure

Modern AI models use deep learning and data-driven approaches that have large datasets and advanced algorithms. Therefore, modern AI requires a scalable infrastructure to adapt to complex, real-world applications that are too sophisticated for traditional AI methods. One key aspect is the development of AI-ready infrastructure. Cisco provides high-performance computing platforms, such as their UCS (Unified Computing System) servers with support for powerful GPUs from NVIDIA and AMD. These servers are designed to handle the intensive computational requirements of AI model training and inference. Complementing this, Cisco is building high-speed, low-latency networking fabrics based on their Nexus switches, including advancements like 800G Ethernet, to ensure efficient data

flow between AI compute nodes and storage. Technologies like Cisco Silicon One are being integrated into these networking solutions to optimize performance for AI workloads.

Furthermore, Cisco embeds AI-powered intelligence into data center management and operations. Tools like Cisco Intersight are leveraging AI and ML for tasks such as predictive analytics, automated workload placement, and infrastructure optimization. This helps in proactively identifying potential issues, improving resource utilization, and simplifying the management of complex AI deployments.

In the realm of security, Cisco is introducing AI-native solutions like Hypershield (*Unveiling a New Era of AI-Native Security with Cisco Hypershield - Cisco Video Portal*, n.d.). This architecture aims to embed security directly into the data center fabric, using AI-driven techniques for autonomous segmentation, threat detection and policy enforcement. This is crucial for protecting the sensitive data often involved in AI applications.

Cisco is also collaborating with key players in the AI ecosystem, such as NVIDIA, to create integrated solutions that simplify the deployment and management of AI infrastructure. These partnerships aim to deliver validated designs and full-stack solutions that combine Cisco's networking and computing strengths with specialized AI hardware and software. Ultimately, Cisco's focus on integrating AI into data centers is driven by the need to support the growing adoption of AI and ML across various industries. By providing AI-ready infrastructure, intelligent management tools, and AI-powered security, Cisco aims to empower organizations to build and operate data centers that can effectively handle the demands of AI workloads, accelerate innovation, and gain a competitive edge in the AI era.

Modern Artificial Intelligence, particularly the advancements in deep learning, relies heavily on specialized and scalable hardware to handle the immense computational demands of training and deploying complex models. Traditional central processing units (CPUs), while versatile, often struggle with the parallel processing required for matrix multiplications and other core AI operations. This has led to the development and widespread adoption of hardware accelerators like Graphics Processing Units (GPUs), Tensor Processing Units (TPUs), and Application-Specific Integrated Circuits (ASICs) (*AI Solutions on Cisco Infrastructure Essentials | Cisco U. Path*, n.d.-a).

GPUs, initially designed for graphics rendering, have proven remarkably effective for AI due to their massively parallel architectures. Thousands of smaller cores, capable of performing numerous calculations simultaneously, are contained within them, significantly

accelerating tasks like training deep neural networks. Their programmability through frameworks like CUDA and OpenCL has made them a versatile workhorse for a wide range of AI workloads.

TPUs, developed by Google, are custom-designed AI accelerators optimized specifically for the tensor computations prevalent in machine learning, particularly within the TensorFlow framework. Their architecture is tailored for the specific needs of neural network operations, offering enhanced performance and efficiency compared to general-purpose GPUs for these tasks. Different generations of TPUs have further refined their capabilities, focusing on increased speed, memory capacity, and interconnect bandwidth to support ever-larger and more complex models.

Application-Specific Integrated Circuits (ASICs) contain the highest level of hardware expertise. Their design is specially optimized for a particular AI task or for a specific model (Panchumarthy & Benala, 2025). While the potential for the highest performance and energy efficiency is offered by them, their development is costly and time-consuming, and the flexibility of GPUs and TPUs is lacked by them. ASICs are typically deployed for very high-volume, specific inference tasks where the model and its computational requirements are well-defined and unlikely to change rapidly. Examples include inference chips for edge devices or large-scale deployment of specific recommendation models.

The continuous evolution of these scalable hardware solutions is a critical enabler of progress in modern AI. Thanks to increasing computational power and efficiency, researchers and practitioners can train increasingly large and sophisticated AI models on massive datasets. This capability is powering advancements in areas like natural language processing, computer vision, and robotics. When choosing hardware, key considerations include the specific AI workload (training or inference), the deployment scale, cost implications, and the necessary degree of flexibility.

Modern AI infrastructure characteristics are:

Several key attributes define modern AI infrastructure, facilitating the efficient development, training, and deployment of AI models. These characteristics collectively address the specific demands of AI workloads, which frequently involve extensive datasets, intricate computations, and specialized hardware.

Given the significant variations in size and complexity of AI workloads, modern infrastructure must offer on-demand scalability. This necessitates the ability to easily provision or de-provision resources like processing power, memory and storage to handle changing workloads and increasing data volumes. Significantly, AI training heavily depends on high-performance computing (HPC) capabilities, including specialized hardware such as GPUs, TPUs, and FPGAs, alongside high-bandwidth interconnects and distributed computing frameworks, to expedite the processing of large datasets and intricate models. AI models are trained on vast amounts of data, necessitating robust and scalable storage solutions. Modern AI infrastructure incorporates high-capacity storage systems, often distributed across multiple nodes, with high-speed data access and efficient data management tools to handle the volume, velocity and variety of AI data. High-performance networking is crucial for AI infrastructure, especially in distributed training scenarios. Low-latency, high-bandwidth networks, such as InfiniBand or high-speed Ethernet, enable efficient communication between processing nodes, minimizing communication bottlenecks and reducing training times. A rich ecosystem of software frameworks and tools is essential for AI development and deployment. Modern AI infrastructure supports popular frameworks like TensorFlow, PyTorch, and MXNet, as well as tools for data preprocessing, model training, visualization and deployment. Cloud-native technologies, such as containerization (e.g., Docker, Kubernetes) and microservices architectures, are increasingly adopted in AI infrastructure. These technologies provide flexibility, portability, and scalability, allowing AI workloads to be easily deployed and managed across diverse environments, including on-premises, cloud, and edge. AI is a rapidly evolving field, with new models, techniques, and tools emerging constantly. Modern AI infrastructure must be flexible and adaptable to accommodate these changes, supporting a wide range of hardware and software options and enabling easy integration of new technologies. As AI becomes more integrated into critical applications, security becomes paramount (*Resources Tutorials - Page 7 of 10 - AskPython*, 2023). Modern AI infrastructure incorporates with comprehensive security features to safeguard sensitive information, avoid forbidden access, and confirm the integrity and reliability of AI systems.(*AI Solutions on Cisco Infrastructure Essentials | Cisco U. Path*, n.d.-a).

5.2 AI Infrastructure requirements

The development and deployment of AI applications place unique demands on the data center’s infrastructure. To support AI workloads effectively, there are some important key requirements that each data center needs to support. Cisco UCS servers are designed to

support a wide range of High-performance Computing (HPC) configurations, including the latest GPUs from NVIDIA. UCS Manager provides a unified management interface, simplifying the deployment and scaling of GPU-accelerated computing resources. Cisco also collaborates with partners like NVIDIA to deliver optimized solutions for AI workloads. Additionally, scalable storage is another necessary element within each AI-powered data center. Cisco offers scalable storage solutions to meet the demands of AI workloads. While Cisco provides storage solutions, it is also a partner with companies like Pure Storage to provide high-performance, all-flash storage arrays that are ideal for AI applications.

These solutions deliver the required capacity, throughput, and low latency for efficient data access. Moreover, Cisco data centers are supporting High-speed Networking by using Cisco Nexus switches. These switches offer high port densities, low latency, and high bandwidth, supporting technologies like RoCE (RDMA over Converged Ethernet) to accelerate communication between servers. ACI can automate the configuration and management of these networks, simplifying the deployment of distributed AI workloads. Besides, Cisco's infrastructure is designed to support a wide range of AI software frameworks and tools. While Cisco doesn't provide the AI frameworks themselves, their hardware and networking solutions are optimized to run these frameworks efficiently. Beyond that, Cisco recognizes the importance of hybrid clouds in AI deployments. Cisco solutions like Cisco Viptela for Software-Defined Wide Area Network (SD-WAN) and its cloud on Ramp can integrate with cloud platforms like AWS, Azure, and Google Cloud, allowing organizations to leverage cloud resources for AI workloads while maintaining on-premises infrastructure for sensitive data or specific performance requirements. Additionally, Cisco ACI Multi-Site Orchestrator (MSO) (Donnelley, n.d., p. 7) can manage cloud and on-premise solutions named in this chapter.

5.3 AI Network Architecture

The introduction of Artificial Intelligence (AI) is fundamentally changing the way network architectures are designed, managed, and optimized. AI presents a powerful solution for addressing the increasing complexity of modern networks, improving performance and security, and automating a range of network operations. This section of the thesis will explore the crucial elements of AI in network architecture, examining its applications, advantages, and challenges.

AI finds applications across numerous aspects of network architecture. For example, AI algorithms can analyze network traffic patterns, user behavior, and application demands to optimize network topology, capacity planning, and resource allocation (rajatsrinivas, 2024). Moreover, AI can predict network congestion, dynamically adapt traffic flows, and optimize routing protocols to enhance network performance and Quality of Service (QoS). Additionally, AI strengthens network security by identifying anomalies and potential threats, and by automating security responses; machine learning algorithms can learn to recognize malicious patterns and prevent attacks like DDoS, intrusion, and malware spread. AI can also automate routine network tasks such as configuration, provisioning, and troubleshooting, minimizing manual effort and boosting operational efficiency. AI-driven tools can analyze network performance data, pinpoint bottlenecks, and forecast potential failures, enabling proactive network maintenance and minimizing downtime.

Artificial Intelligence offers numerous advantages for network architecture. It can optimize network performance by dynamically adapting to evolving network conditions, thereby reducing latency, increasing throughput, and improving QoS. AI also strengthens network security by detecting and mitigating threats more effectively than conventional methods. Furthermore, it can automate network operations, minimizing human errors, lowering operational expenses, and enhancing network reliability. AI can also facilitate more efficient network scaling through optimized resource allocation and capacity planning. Lastly, AI delivers valuable insights and predictions, empowering network operators to make better-informed decisions. For example, Cisco's Nexus switches provide low-latency, high-bandwidth connectivity crucial for distributed AI training, while technologies such as RDMA over Converged Ethernet (RoCE) and enhanced fabric architectures enable efficient communication among GPU-accelerated servers.

While AI offers significant potential for network architecture, several challenges and considerations need to be addressed. AI algorithms must be trained on large quantities of high-quality data to operate effectively (*AI in Medicine*, n.d.). Implementing AI in network architecture can be complex and may require specialized expertise. Some AI models, notably DL, can be tough to analyze, making the understanding of how decisions are arrived at by them a difficult process. AI systems themselves can be vulnerable to attacks, such as adversarial attacks, which can compromise their performance and reliability. In AI-powered data centers, security is essential to keep sensitive information safe and ensure the integrity of AI systems (*AI in Medicine*, n.d.). Cisco Tetration Analytics provides comprehensive visibility and security analytics, playing a vital role in securing these environments. Telemetry data for every packet and flow originating within the data center is

captured by Cisco Tetration Analytics (Mohanty et al., 2024). This pervasive visibility is essential for Anomaly detection, Microsegmentation, Threat hunting as well as Compliance monitoring. Tetration uses machine learning to define a baseline for typical network behavior, enabling the detection of deviations that may indicate security threats or performance issues. Moreover, by understanding application dependencies, Tetration facilitates the implementation of microsegmentation policies, isolating critical AI workloads and limiting the impact of potential security breaches. Tetration's historical data and powerful analytics capabilities enable security teams to proactively hunt for threats, identify patterns of malicious activity, and respond quickly to security incidents. Furthermore, it provides the visibility and reporting capabilities needed to demonstrate compliance with industry regulations and security standards (Palanivel, 2019).

6 Network Automation methods used within Cisco ACI

Earlier in this thesis, Cisco's entry into the Software-Defined Networking (SDN) landscape with Application Centric Infrastructure (ACI) was established, emphasizing its design for reduced complexity compared to alternative solutions. The Cisco Application Policy Infrastructure Controller (APIC), the central management component of ACI, offers significant advantages through its extensive support for programmatic automation of fabric configuration and management. Cisco ACI can be configured and managed by the following list of automation tools: APIC REST API, Python SDK or Cobra SDK, Ansible, Terraform and GUI automation.

APIC REST API is the most fundamental and powerful way to interact with and automate Cisco ACI. The Application Policy Infrastructure Controller (APIC) exposes a comprehensive Representational State Transfer (REST) API. Standard HTTP/HTTPS methods (GET, POST, PUT, DELETE) can be used to programmatically configure, monitor, and manage all aspects of the ACI fabric. This API allows for granular control and the development of custom automation scripts and tools using various programming languages that Python is very popular.

Cisco provides another automation method named Python Software Development Kit (SDK), often referred to as the ACI Toolkit or Cobra SDK. This SDK simplifies interaction with the APIC REST API by providing Python classes and functions that map to ACI objects and API calls. It makes writing automation scripts in Python much more efficient and less prone to errors compared to directly manipulating raw HTTP requests. Cobra supports CRUD operations of Cisco ACI fabric. Cobra offers full functionality, so it is well suited for more complex queries with filters for network devices. Command 3 presents a sample for creating a new tenant in Cisco ACI by using Cobra:

Command 3 Sample code to create a new tenant with Cobra SDK in Cisco ACI

```
from credentials import *
import cobra.mit.access
import cobra.mit.request
import cobra.mit.session
import cobra.model.fv
import cobra.model.pol

# connect to the apic
auth = cobra.mit.session.LoginSession(URL, LOGIN, PASSWORD)
```

```
session = cobra.mit.access.MoDirectory(auth)
session.login()

# Create a Variable for your Tenant Name
tenant_name = "Marketing_Department"

# create a new tenant
root = cobra.model.pol.Uni('')
new_tenant = cobra.model.fv.Tenant(root, tenant_name)

config_request = cobra.mit.request.ConfigRequest()
config_request.addMo(new_tenant)
session.commit(config_request)
```

Against using variables like URL, LOGIN and PASSWORD inside the Cobra code directly, using an additional Python file (like logininfo.py) to store this sensitive information is recommended. Moreover, Cisco ACI uses a code generator assistance called ARYA (APIC Rest Python Adapter). It is a valuable tool designed to simplify the process of working with the APIC REST API and the Cobra SDK (Python SDK for ACI). The primary function of ARYA is to convert APIC object documents, which are typically in XML or JSON format, into equivalent Python code that leverages the Cobra SDK.

6.1 Cisco ACI built-in Automation tool

As discussed earlier in this chapter, Cisco ACI is using pure JSON inside the Cisco APIC and it will be used by Postman. Another important tool that mentioned earlier is Python programming with the Cobra SDK. Using solutions like Ansible or Terraform is another solution provided by third-party companies that can integrate with Cisco ACI. Another way of doing automation in ACI is ACI Toolkit. The Cisco ACI Toolkit is a Python library that simplifies the management and automation of Cisco Application Centric Infrastructure (ACI) fabrics. It provides a higher-level abstraction over the ACI REST API, allowing network engineers and developers to programmatically interact with the APIC and configure network policies using Python scripts. This facilitates more efficient and streamlined automation workflows, reducing the complexity associated with direct API interaction and enabling the creation of reusable automation code (*Introduction — Acitoolkit 0.1 Documentation*, n.d.).

6.2 Using API with Postman to automate Cisco Fabric configuration

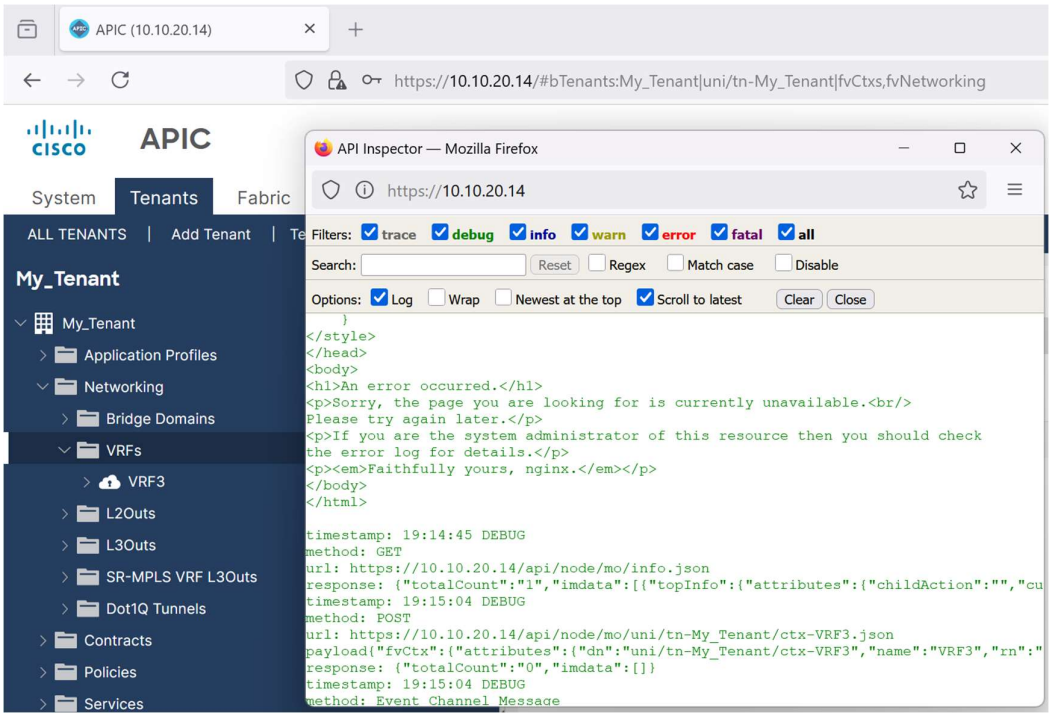
This section details a scenario in which the creation of multiple Virtual Routing and Forwarding (VRF) instances within a Cisco Application Centric Infrastructure (ACI) fabric is

automated using the Postman API client. This approach demonstrates the efficiency and scalability gains achieved by leveraging APIs for network configuration, compared to traditional manual methods.

The scenario involves the automated creation of three VRFs (VRF3 and VRFtest) within a specified tenant called My_Tenant on a Cisco ACI fabric. Postman, a popular API development environment, is used to send a series of API calls to the Cisco APIC (Application Policy Infrastructure Controller), the central management entity of the ACI fabric.

One of the most important tools in Cisco ACI is ACI Inspector. The ACI Inspector in Cisco ACI is a built-in tool within the Application Policy Infrastructure Controller (APIC) GUI that allows administrators to observe in real-time the REST API calls made by the GUI as users navigate and configure the ACI fabric. This provides a transparent view into how the APIC communicates with the underlying infrastructure, capturing the specific JSON payloads and HTTP methods used for each action, which is invaluable for understanding ACI's programmatic interface, troubleshooting configurations, and for learning how to automate tasks using the ACI REST API. To initiate API calls for APIC configuration, network engineers are required to construct a payload within Postman. While a comprehensive understanding of automation is typically beneficial, engineers can leverage tools such as the ACI Inspector in conjunction with the APIC GUI to accelerate this process by copying specific payloads from the ACI Inspector and transferring them into the body of the Postman request. Furthermore, the Postman Runner feature facilitates Cisco ACI automation by enabling the utilization of .CSV files. These files can be structured to define multiple objects, including VRFs, bridge domains, subnets, and VLANs, thereby allowing for automated configuration deployment via the Postman Runner. Figure 25, provides an overview of the Cisco ACI inspector.

Figure 25 Cisco ACI inspector



By performing any configuration through the GUI, the specific request (POST or GET) is shown by this inspector. After that, by copying the whole content of the ACI inspector to any text editor, it will be possible to poll the API request from the result generated by ACI inspector. For example, Command 4 shows the specific syntax that used by the specific URL and the payload for creating a new VRF.

Command 4 REST API using a JSON payload generated by the API inspector

```
url: https://10.10.20.14/api/node/mo/uni/tn-My_Tenant/ctx-VRF3.json
payload{"fvCtx":{"attributes":{"dn":"uni/tn-My_Tenant/ctx-
VRF3","name":"VRF3","rn":"ctx-VRF3","status":"created"},"children":[]}}
```

Second step is to obtain an authentication token from the configuration computer to the Cisco APIC. To make an Authentication from the Postman, it is important to send a POST request to the APIC. This token is required for subsequent API calls. The request includes the APIC username, password. Figure 26, shows an API call for this purpose.

Figure 26 POST request to make authentication with Cisco ACI

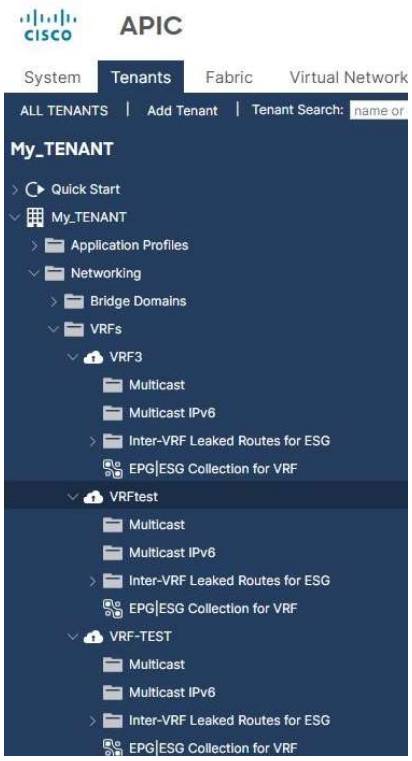
In this step, as shown in Figure 27, a POST request is sent to the APIC to create VRF3 within a tenant called My_Tenant. The request payload includes the necessary VRF configuration parameters, such as the VRF name and description.

Figure 27 Creating a new VRF with Postman

A similar POST request is sent to create VRFtest within the same tenant. Final step is to verify the created VRF inside the desired tenant. For this step, network engineers can easily verify created VRFs by using Cisco APIC GUI or sending a GET request to the APIC to retrieve a list of VRFs within the tenant.

The response will be parsed to confirm that VRFs have been successfully created. Figure 28 confirms that, the desired VRFs are created within My_Tenant.

Figure 28 Verifying VRF configuration in Cisco APIC



The results of this scenario demonstrate the effectiveness of using APIs and Postman to automate the creation of multiple VRFs. The process is significantly faster and less error-prone than manually configuring each VRF through the APIC GUI or CLI. The use of variables within Postman allows for easy modification of VRF names and other parameters, further enhancing efficiency. This automation approach can be easily extended to create a larger number of VRFs or to configure other ACI objects, highlighting the scalability and flexibility of API-driven network configuration. Moreover, network engineers can use a single Excel file containing all variables with their respective names within the Postman Runner. By using this method, not only VRFs but also any other variables like Tenants, bridge domains, contracts, interface parameters and more can be automated through a single Excel file. The ACI calls can then be run after uploading the Excel file into Postman.

7 Results and Conclusions

The architectural differences between traditional three-tier and Cisco ACI are fundamental and achieve a substantial effect on network operation and scalability. The traditional three-tier architecture, is characterized by a hierarchical structure comprising three layers that consists of the core layer, responsible for high-speed backbone connectivity, the aggregation layer, which provides connectivity between access and core layers, and the access layer, which serves as the point of connection for end devices. This model while well-established, can become complex and difficult to manage as network size and application demands increase (FS, n.d.).

In contrast, Cisco ACI employs a spine-leaf architecture. In this design, leaf switches connect to all spine switches, and spine switches connect to all leaf switches (*Cisco Application Policy Infrastructure Controller (APIC) - Cisco Application Centric Infrastructure (ACI) Design Guide*, n.d., p. 11). The Application Policy Infrastructure Controller (APIC) centralizes network management and policy enforcement. This architecture offers improved scalability and reduced latency compared to the traditional three-tier model (*NVIDIA Networking with Nutanix*, n.d., p. 11).

While in-depth performance testing was not conducted, observations were made regarding the scalability advantages of Cisco ACI based on its architectural design. The spine-leaf architecture of ACI, with its high degree of interconnectivity, provides increased bandwidth and reduced latency compared to the hierarchical three-tier design. Specifically, the backplane capacity of Cisco ACI-based switches supports high-speed data transfer and allows for the addition of more devices with minimal performance degradation. This contrasts with traditional three-tier networks, where adding devices can lead to increased congestion and reduced throughput, especially at the aggregation layer. The spine-leaf architecture inherently provides more paths for data to travel, reducing bottlenecks (LogicMonitor, 2024).

Security in traditional three-tier networks is often implemented using a combination of VLANs and Access Control Lists (ACLs). VLANs provide basic network segmentation, grouping devices into logical networks, while ACLs control traffic flow between VLANs and within VLANs. However, this approach has limitations in achieving granular security and can be complex to manage, especially in dynamic environments. Cisco ACI, on the other hand, utilizes a policy-based approach with Endpoint Groups (EPGs) and contracts to implement micro-segmentation. EPGs group endpoints based on application requirements,

and contracts define the communication allowed between EPGs. This provides a more granular and flexible security model compared to VLANs and ACLs. For example, in a traditional network, isolating a vulnerable server might require creating a new VLAN and configuring ACLs on multiple switches. In ACI, this can be achieved by placing the server in a dedicated EPG and defining a contract that restricts communication to only authorized applications.

During the configuration of EPGs, Tenants, and VRFs, it was observed that ACI's policy-driven model simplifies the implementation and enforcement of security policies. This minimizes the potential of misconfigurations and enhances the entire security posture of the network.

Cisco ACI significantly improves operational efficiency through its automation capabilities. Using tools like Ansible, Python, and Postman to interact with the APIC and automate network configuration tasks was explored.

For example, an OSPF scenario was automated using a Python runbook, demonstrating the ability to programmatically configure routing policies. Similarly, the creation of VRFs and Tenants was automated using Postman and the APIC's REST API, leading to a reduction in the time and effort required compared to box-by-box configuration in a traditional three-tier network. These examples highlight ACI's ability to automate daily tasks, reduce human error, and enable rapid deployment of network changes. This increased automation translates to significant time savings for network administrators and improved network agility.

8 Future Work

Building upon the practical evaluation presented in this thesis, which highlighted Cisco ACI's architectural advantages, enhanced security via microsegmentation, and significant gains in operational efficiency through automation, several key areas warrant further investigation.

Firstly, while qualitative observations on performance and scalability were noted, future research could conduct rigorous quantitative performance benchmarking of ACI against traditional architectures. This would involve extensive lab testing under varying load conditions to precisely measure latency, throughput, and resilience during simulated failures, providing concrete data for performance-critical applications.

Secondly, extending the automation concepts explored in this thesis, a crucial future direction involves deeply investigating the integration and practical application of Artificial Intelligence and Machine Learning (AI/ML) within Cisco ACI environments. This could focus on developing or assessing AI-powered solutions for predictive analytics to anticipate network issues, automating complex root cause analysis, or even enabling network segments with automated recovery, thereby transforming ACI from an automated system into an intelligent and adaptive infrastructure.

Furthermore, a comprehensive overall long-term cost analysis is recommended, moving beyond qualitative estimates to provide detailed financial modeling of CAPEX and OPEX for both architectures over a multi-year period. This would offer critical financial justification for ACI adoption. Finally, given the observed complexities of real-world migrations, future studies could benefit from longitudinal case studies of organizations post-ACI deployment, assessing the long-term operational shifts, skill set evolution, and sustained business benefits over several years of production use. Such research would provide invaluable insights into the ongoing journey of data center modernization.

References

0AD125C9B7294083A6684FD823756EE6.pdf. (n.d.). Retrieved May 27, 2025, from https://cdn.istanbul.edu.tr/file/JTA6CLJ8T5/0AD125C9B7294083A6684FD823756EE6

ACI terminology—Understanding ACI. (n.d.). Cisco DevNet Learning Labs Center. Retrieved April 14, 2025, from https://developer.cisco.com/learning/labs/sbx-intro-aci-01_understanding-aci/

Ahmadi, A. (2020). *CCNP data center application centric infrastructure 300-620 DCACI official cert guide* (1st ed.). Pearson Education, Inc.

AI in Medicine: The Future of Medical Science. (n.d.). Retrieved May 28, 2025, from https://mirror.xyz/pinkpitch.eth/8-ohI08c37b0LMLWqlxHh6nG3XBirgOttGSOoNJOOvs

AI Solutions on Cisco Infrastructure Essentials | *Cisco U. Path*. (n.d.-a). Retrieved April 1, 2025, from https://u.cisco.com/paths/ai-solutions-on-cisco-infrastructure-essentials-288

AI Solutions on Cisco Infrastructure Essentials | *Cisco U. Path*. (n.d.-b). Retrieved April 1, 2025, from https://u.cisco.com/paths/ai-solutions-on-cisco-infrastructure-essentials-288

ai-admin. (2023, December 5). Examples of AI Problems. *AI for Social Good*. https://aiforsocialgood.ca/blog/examples-of-ai-problems

AtBit Network—About. (n.d.). Retrieved April 17, 2025, from https://www.atbit.net/about

Bailis, P., & Kingsbury, K. (2014). The Network is Reliable: An informal survey of real-world communications failures. *Queue*, *12*(7), 20–32. https://doi.org/10.1145/2639988.2655736

Cisco Application Centric Infrastructure—ACI Multi-Pod White Paper. (n.d.). Cisco. Retrieved April 14, 2025, from https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-737855.html

Cisco Application Centric Infrastructure—Cisco Application Centric Infrastructure (Cisco ACI) Solution Overview. (n.d.). Cisco. Retrieved May 1, 2025, from https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/solution-overview-c22-741487.html

Cisco Application Policy Infrastructure Controller (APIC)—Cisco Application Centric Infrastructure (ACI) Design Guide. (n.d.). Cisco. Retrieved April 7, 2025, from

https://www.cisco.com/c/en/us/td/docs/dcn/whitepapers/cisco-application-centric-infrastructure-design-guide.html

Cisco Data Center Networking Solutions. (n.d.). Cisco. Retrieved April 7, 2025, from https://www.cisco.com/site/us/en/products/networking/cloud-networking/index.html

Configuring EtherChannels. (n.d.). [Bookmap]. Retrieved April 17, 2025, from https://www.cisco.com/c/dam/en/us/td/docs/switches/lan/catalyst9600/software/release/16-12/configuration_guide/lyr2/configuring_etherchannels.html

Dagenhardt, F., Moreno, J., & Dufresne, B. (2018). *Deploying ACI: The complete guide to planning, configuring, and managing Application Centric Infrastructure*. Cisco Press.

Diamond, J. (2024, October 28). Data Center Network Architecture—Key Components, Challenges and Insights. *Park Place Technologies*. https://www.parkplacetechnologies.com/blog/data-center-network-architecture/

Donnelley, R. (n.d.). *Cisco ACI Multi-Site Architecture White Paper*.

EtherChannels, L. (n.d.). *Restrictions for EtherChannels*.

Fordham, S. (2017). *Cisco ACI Cookbook*. Packt Publishing Ltd.

FS. (n.d.). *What Is an Aggregate Switch?* FS.Com. Retrieved May 7, 2025, from https://www.fs.com/blog/what-is-an-aggregate-switch-1340.html

GLBP - Gateway Load Balancing Protocol. (n.d.). Cisco. Retrieved May 8, 2025, from http://www.cisco.com/en/US/docs/ios/12_2t/12_2t15/feature/guide/ft_glbp.html

Ietf.org/rfc/rfc1122.txt. (n.d.). Retrieved April 17, 2025, from https://www.ietf.org/rfc/rfc1122.txt

Introduction—Acitoolkit 0.1 documentation. (n.d.). Retrieved May 3, 2025, from https://acitoolkit.readthedocs.io/en/latest/introduction.html

Li, D., Cole, B. A., Morton, P., & Li, T. (1998). *Cisco Hot Standby Router Protocol (HSRP)* (Request for Comments No. RFC 2281). Internet Engineering Task Force. https://doi.org/10.17487/RFC2281

Link Aggregation Control Protocol (LACP) (802.3ad) for Gigabit Interfaces. (n.d.). Cisco. Retrieved April 17, 2025, from https://www.cisco.com/c/en/us/td/docs/ios/12_2sb/feature/guide/gigeth.html

LogicMonitor. (2024, July 24). *What is Cisco ACI?* LogicMonitor. https://www.logicmonitor.com/blog/what-is-cisco-aci

LTRDCN-2143. (n.d.). Retrieved May 26, 2025, from https://aci-lab.ciscolive.com/lab/pod7/aci101/aciconcepts

Mohanty, S., Nanjundan, P., & Kar, T. (2024). *Artificial Intelligence in Forecasting: Tools and Techniques* (1st ed.). CRC Press. https://doi.org/10.1201/9781003399292

Moy, J. (1997). *OSPF Version 2* (Request for Comments No. RFC 2178). Internet Engineering Task Force. https://doi.org/10.17487/RFC2178

Mso-configuration-aci-managing-tenants-33x.pdf. (n.d.). Retrieved April 14, 2025, from https://www.cisco.com/c/en/us/td/docs/dcn/mso/3x/configuration/cisco-multi-site-configuration-guide-aci-331/mso-configuration-aci-managing-tenants-33x.pdf

Multi-chassis Link Aggregation Group—RouterOS - MikroTik Documentation. (n.d.). Retrieved April 17, 2025, from https://help.mikrotik.com/docs/spaces/ROS/pages/67633179/Multi-chassis+Link+Aggregation+Group

NVIDIA Networking with Nutanix. (n.d.).

Palanivel, K. (2019). Modern Network Analytics Architecture Stack to Enterprise Networks. *International Journal for Research in Applied Science and Engineering Technology*, *7*(4), 2634–2651. https://doi.org/10.22214/ijraset.2019.4480

Panchumarthy, R., & Benala, T. R. (2025). An Overview of AI Workload Optimization Techniques. In T. R. Benala, S. Dehuri, R. Mall, & M. N. Favorskaya (Eds.), *Boosting Software Development Using Machine Learning* (pp. 269–299). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-88188-6_12

Rahman, R. A., Zahari, N. A., Kassim, M., & Yusof, M. I. (2016). Virtual Routing and Forwarding-lite Traffic Management over Multi-protocol Layer Switching-Virtual Private Network. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, *8*(3), Article 3.

rajatsrinivas. (2024, August 20). AI/ML: The Network Gamechanger for User Experience and SLAs. *Alethea Communications Technologies*. https://aletheatech.com/ai-ml-the-network-gamechanger-for-user-experience-and-slas/

Rekhter, Y., Hares, S., & Li, T. (2006). *A Border Gateway Protocol 4 (BGP-4)* (Request for Comments No. RFC 4271). Internet Engineering Task Force. https://doi.org/10.17487/RFC4271

Resources Tutorials—Page 7 of 10—AskPython. (2023, November 6). https://www.askpython.com/resources/page/7

Savage, D., Ng, J., Moore, S., Slice, D., Paluch, P., & White, R. (2016). *Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)* (Request for Comments No. RFC 7868). Internet Engineering Task Force. https://doi.org/10.17487/RFC7868

Small Enterprise Design Profile (SEDP)—Network Foundation Design. (n.d.).

SMB50 ENG Switching.pdf—This module describes Cisco Borderless Network Switching Solu6ons for Small and Midsize customers and helps | Course Hero. (n.d.). Retrieved May 26, 2025, from https://www.coursehero.com/file/25209357/SMB50-ENG-Switchingpdf/

Understand Virtual Port Channel (vPC) Enhancements. (n.d.). Cisco. Retrieved April 17, 2025, from https://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/nx-os-software/217274-understand-virtual-port-channel-vpc-en.html

Understanding ACI - Understanding ACI. (n.d.). Cisco DevNet Learning Labs Center. Retrieved April 7, 2025, from https://developer.cisco.com/learning/labs/sbx-intro-aci-01_understanding-aci/

Unveiling a New Era of AI-native Security with Cisco Hypershield—Cisco Video Portal. (n.d.). Retrieved May 1, 2025, from https://video.cisco.com/detail/video/6351752711112

Welcome to nornir's documentation! —Nornir 3.5.0 documentation. (n.d.). Retrieved April 26, 2025, from https://nornir.readthedocs.io/en/latest/index.html

Appendixes 1 Data Management plan

1. **Data description**
    - **Data types:** The data for this thesis primarily consists of:
        - Configuration files and scripts (code) for network devices (Cisco IOS images).
        - Virtual network topologies and configurations from emulators (EVE-NG, VMware Workstation) and simulators (Cisco ACI simulator).
        - Output and results from network testing and analysis.
        - Documentation from Cisco learning platforms (Cisco U, Cisco DevNet).
        - Video recordings of network scenarios and demonstrations uploaded to the YouTube.
    - **File Formats:** The data will be stored in the following formats:

- Configuration files: Plain text (.txt).
- Virtual machine images: Various emulator-specific formats (e.g.,. ovf, .vmdk).
- Scripts: Python (.py), Postman collections (JSON).
- Documentation: PDF (.pdf), Lab .zip file can be import in eve-ng or pnetlab.

- **Metadata:** The following metadata will be recorded to ensure data understandability and reusability:
    - For configuration files and scripts: Description of the network scenario, device configurations, purpose of the script, and any dependencies.
    - For virtual machine images: Description of the network topology, operating systems used, software versions, and configuration details.
    - For test results: Description of the test methodology, parameters, and metrics.
    - For video recordings: Title, description of the scenario, date of recording, and relevant configurations.
    - For Cisco U/DevNet materials: Course name, module/lesson title, and date accessed.

2. **Data Collection**
    - **Collection Methods:** The data was generated based on:
        - Configuration and deployment of network devices and virtual environments using EVE-NG, VMware Workstation, and the Cisco ACI simulator.
        - Hands-on experimentation and testing of network configurations and features.
        - Utilization of Cisco U and Cisco DevNet online courses and sandboxes for learning and practical exercises.
        - Recording of network scenarios and demonstrations using screen recording software.
    - **Existing Data:** Data was obtained from:
        - Cisco U and Cisco DevNet courses: These platforms provided configurations, best practices, and sample scenarios. Access was granted through a Cisco account, with usage limited to educational purposes. Moreover, thesis created based on the client data center and simple scenarios related to it.
    - **Quality Control:** The following measures were taken to ensure data quality:
        - Verification of network configurations and scripts for accuracy and correctness.
        - Thorough testing of network scenarios to validate results, by ether tools like ping, traceroute for VRF scenario in traditional model and by receiving reply code and messages in practical automation parts.
        - Documentation of all steps taken in the configuration and testing process.
        - Review of configurations and results against Cisco best practices.

3. **Storage and Backup**
    - Data was stored during the research on:
        - Personal laptop: For configuration files, scripts, and documentation.
        - EVE-NG server and VMware Workstation: For virtual network topologies and configurations.
        - Cisco Cloud Simulator for ACI configurations and testing (for a specific period).
    - **Backup Procedures:** Data was backed up as follows:
        - Regular saving of configuration files and scripts on the personal laptop.
        - Saving of EVE-NG and VMware Workstation virtual network scenarios to the personal laptop local drive.
        - Downloading and saving of relevant output and results from the Cisco ACI simulator as an image or API codes from Cisco API Inspector built-in tool.
        - Uploading of video recordings to YouTube, which provides a form of cloud backup.
4. **Sharing and Reuse**
    - **Data Sharing:** The researcher intends to share the data through:
        - From YouTube: Video recordings of network scenarios.
        - GitHub repository (potential) or by a .zip file: Configuration files and scripts.
        - Upon request: Other data related to the thesis.
    - **Restrictions:** There are no restrictions on sharing the data because thesis information used by imaginary data based on the real-world scenarios and client scenarios.
5. **Responsibilities**
    - **Data Management Roles:** The researcher is responsible for managing all data related to this thesis.
    - **Responsibilities:** The researcher's responsibilities include:
        - Collecting and generating data.
        - Organizing and documenting data.
        - Storing and backing up data.
        - Preserving data for the long term.
        - Sharing data, as appropriate.
6. **Resources**
    - **Required Resources:** The following resources were used for data management:
        - Personal laptop (storage and processing).
        - EVE-NG server and VMware Workstation (data generation and storage).
        - Cisco Cloud Simulator for the ACI scenario data.
        - YouTube (video hosting and sharing).

- **Resource Availability:** The researcher had access to the necessary resources.