A Web Driven SDN Orchestrator For The Provisioning of ACI Fabric and Lab Infrastructure

Matthew Gaynor

922830

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Bachelor of Science
of the
University of Portsmouth.

School of Computing
Engineering Project

2023

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Date: 2023

Abbreviations

ACI	Application Centric Infrastructure
API	Application Programming Interface
APIC	Application Policy Infrastructure Controller
BD	Bridge Domain
CLI	Command Line Interface
CX	Customer Experience
DDI	DNS, DHCP and IP Address Management (IPAM)
DMZ	Demilitarised Zone
DNS	Domain Name System
DPG	Distrubited Port Group
DVS	Distributed Virtual Switch
EPG	Endpoint Group
ERD	Entity Relationship Diagram
FEX	Fabric Extender
HTTP	HyperText Transfer Protocol
MVC	Model View Controller
NAT	Network Address Translation
NOS	Network Operating System
NTP	Network Time Protocol
OOB	Out of Band
ORM	Object Relational Mapping
OSPF	Open Shortest Path First
RADIUS	Remote Authentication Dial-In User Service
REST	Representational State Transfer
SPA	Singple Page Application
SVS	Solution Validation Services
ToR	Top of Rack
VM	Virtual Machine
VPN	Virtual Private Network
VRF	Virtual Routing and Forwarding
WAN	Wide Area Network

Abstract

The purpose of this project is to provide a solution to CX Labs UK within Cisco Systems LTD that will streamline the operation of DMZ lab-space. The application shall allow lab staff to prepare network infrastructure for an incoming project with little to no CLI interaction and it shall be web-based. ACI, VMware ESXi and vCenter shall be used to manage the networking and virtual machines respectively. The lab space will be represented by a small testbed containing the minimum quantity of networking equipment and compute resources required to accurately simulate the environment.

Acknowledgements

I would like to thank Cisco, my manager and good friend, Dave Smith, for his close help in allowing me to use equipment necessary for testing and developing the solution.

Consent to Share

I consent / do not consent for this project to be archived by the University Library and potentially used as an example project for future students.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Client

The client is CX Labs UK within Cisco Systems. CX Labs provides lab space for use by business units internal to the company. Most of the space is used for the testing of customer networks by SVS (Solution Validation Services). SVS provide bespoke testing services to customers wishing to use Cisco's expertise to test a range of situations, from regression and firmware testing to full upgrade and migration plans.
In order to match the customers environment as close as possible, a scaled down version of the customers network is usually recreated in the lab space managed by CX Labs. CX Labs hold many devices that cover most of the Cisco portfolio, which allows for the recreation of most networks. Most of this lab space is hosted within the internal Cisco corporate network, which requires any users to be employees of Cisco in order to access testbeds. More and more customers however are requesting remote access to their testbeds. To facilitate this, a fully isolated DMZ environment is provided, which allows direct WAN connectivity to a testbed, allowing for a VPN tunnel to be established and hence remote access granted to a testbed from any location to any permitted person.

1.2 The Problem

Currently the solution for the DMZ network infrastructure consists of 4 Nexus 9K devices, with FEXs for RJ45 connectivity and 2960S ToR switches. Whilst this solution is functional and works, the major downside is that there is no automation or configuration management solution deployed. This means over time, configuration drift occurs as manual changes are made, but not made the same to all switches. Unused VLANs are also not removed from the switches which leads to bloat and a larger configuration than is required. The same situation also occurs on vCenter with the configuration of the DPGs within the DVS where unused DPGs are never removed or are labelled incorrectly.
The manual configuration of the required routing and VPN termination as well as other lab requirements such as a NTP, DNS and RADIUS all take a lot of time to configure. This

time could be better utilised, such as preparing the physical rack space for the racking and stacking of new equipment.

1.3 Aims and Objectives

The aim of this project is to provide an easy to use dashboard that allows CX Labs to easily onboard and manage projects. This will be achieved by providing a web based interface that allows the user to create a new project, which will then automatically create the required configuration within ACI, vCenter, and other associated network infrastructure. Management of existing projects will also be supported, operations such as expanding a projects rackspace utilisation, or removing a project from the environment will be supported. The dashboard will also provide a view of the current projects utilisation of the lab space, as well as the current utilisation of the lab space as a whole.

1.3.1 Deliverables

- A web based dashboard that allows the user to manage and provision projects
- User guides
  - User guide for the dashboard
  - User guide for how to provision ACI, vCenter and other associated network infrastructure so that it will be compatible with the automation solution
- A report detailing the design and implementation of the solution

1.4 Constraints

Due to the fact that the testbed utilised for testing and development is in a remote datacenter, physical access to the testbed will be limited. This may result in delays if a problem with the physical infrastructure occurs which could lead to the project running over time. The solution will be designed to be as resilient as possible, but there is a chance that the solution may not be able to recover from a failure. There may also be a chance that access to the testbed is revoked, which would result in the project being delayed until access is restored.

External factors such as the availability of the testbeds internet uplink and power supply may also have an impact on the projects delivery within the required timescale.

1.5 Evaluation

To ensure that the project delivers on the required functionality, the following evaluation criteria will be used to evaluate the project:

- Does the dashboard allow for the easy creation, management and deletion of lab projects?

- Does the dashboard allow for a logical view of the entire lab space as a whole?
- Does the dashboard show project status and health of the associated devices?
- Does the automation platform create a schema in ACI that is easy to follow and troubleshoot should a problem arise?

Chapter 2

Literature Review

The aim of this literature review is to research and analyse existing solutions, documentation and research on the automation of networking infrastructure. To ensure this review is of maximum usefulness, it will also involve analysing best practises and standards when developing software and automation solutions. This will allow for the optimisation of the planning and implementation stages of the project that will subsequently follow.

Sources for this review will be from relevant books, online websites, professional publications and Request for Comments. Multiple services will be used to identify and obtain sources that can be used for the purposes of this review.

- Google Scholar
- IEEE Explore
- ResearchGate
- University of Portsmouth Library

Research was conducted that was related to the following set of topics:

- What is software defined networking?
- What types of software defined networking exist?
- What are the advantages and disadvantages?

2.1 Definition of Software Defined Networks

All industry experts and academics define software defined networking similarly, that is providing automation and intelligence to networks via the means of software and APIs. Kreutz et al. (2015) state that “SDN was originally coined to represent the ideas and work around OpenFlow at Stanford University”.
SDN is often defined using four pillars, Kreutz et al. (2015) define these as the following:

1. “The control and data planes are decoupled. Control functionality is removed from network devices that will become simple (packet) forwarding elements”
2. “Forwarding decisions are flow based, instead of destination based. A flow is broadly defined by a set of packet field values acting as a match (filter) criterion and a set of actions (instructions)”
3. “Control logic is moved to an external entity, theso-called SDN controller”
4. “The network is programmable through software applications running on top of the NOS that in-teracts with the underlying data plane devices.”

2.2 Overview of Software Defined Networks

Since this project is centred around software defined networking and the automation of these networks, it is critical to ensure that the principles and their method of operations are understood.

An official survey paper from the IEEE that analysed the state of SDN provides a good explanation as to why the need for network programmability arose in the first place. Conventionally, “Computer networks are typically built from a large number of network devices such as routers, switches and numerous types of middleboxes” Nunes et al., 2014. Nunes et al. (2014) go on to state that due to the large amount of manual configuration required to achieve the desired traffic flow, “network management and performance tuning is quite challenging and thus error-prone”. This leads to one of the solutions, Software Defined Networking. “Software Defined Networking (SDN) is a new networking paradigm which the forwarding hardware is decoupled from control decisions. It promises to dramatically simplify network management and enable innovation and evolution”. Software defined networking “is designed to use standardized application programming interfaces (APIs) to quickly allow network programmers to define and reconfigure the way data or resources are handled within a network.” Kirkpatrick, 2013

2.3 Types of Software Defined Networks

Software Defined Networking is a blanket term for any solution that tackles conventional networking with a programmatic view. There are “the two main delivery models: Imperative and declarative” CDW (2015). CDW (2015) go on to state that Imperative SDN is where “A centralized controller (typically a clustered set of controllers) functions as the network’s ‘brain”’ and that declarative SDN is where “the intelligence is distributed out to the network fabric. While policy is centralized, policy enforcement isn’t”. CDW (2015) give “a protocol such as OpenFlow explicitly telling network switches precisely what to do and how to do it” to be an example of imperative SDN.

Chapter 3

Methodology

Here the methodologies used to develop and test the software will be detailed. This will include the tools used, the development process and the testing process. As the primary artefact will be software-based, it makes sense to select an appropriate development cycle that will ensure that the artefact is delivered on time and to specification.

3.1 Development

Whilst there are many software development approaches, Kanban was chosen as the best methodology because only one person will be working on the development of the software, not a team of developers. Kanban is a simple and effective way to manage a single person's workload and is a good fit for this project. A solution such as Trello, Atlassian, 2023, can be utilised as it is free for small teams and is quick and easy to use. The Trello board that will be used for development is shown in figure 3.1.

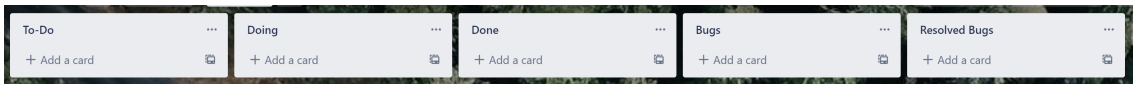

Figure 3.1: Trello Kanban board used to manage the development process

The project requirements will be broken down into smaller tasks, and each task will be added to the Trello board. The tasks will be added to the "To Do" column and then moved to the "Doing" column when the task is being worked on. Once the task is complete, it will be moved to the "Done" column. This will allow the project to be broken down into smaller tasks and will allow the project to be managed effectively. The tasks will be added to the board in the order that they are required to be completed, and will be worked on in that order. This will ensure that the project is completed in the correct order and will ensure that the project is completed in the required timescale. There are also 2 other lists, namely "Bugs" and "Resolved Bugs" which will allow for the ease of tracking bugs and ensuring that they are resolved during the development cycle.

3.2 Time Management

To ensure that the project keeps to time, a Gantt chart is to be used which will outline the key milestones of the project and when they should be met. To generate this chart, teamgantt, 2023, was used as it has a free tier and meets all of the requirements for this project. Figure 3.2 shows the Gantt chart for the project.

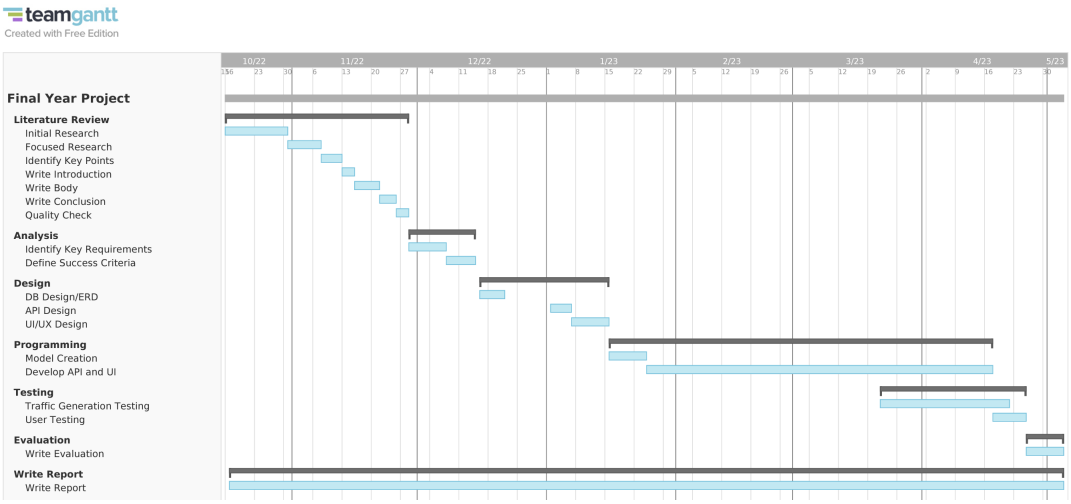

Figure 3.2: Gantt chart used to outline the key project milestones

As the project will follow the Kanban development methodology, some stages overlap and will occur simultaneously. This is because testing will be critical to influencing the development process and ensuring that features work correctly as they are developed. The tight timescale of this project also means that tight features be tested as they are written to prevent issues later in the project.

3.3 Project Management

To ensure secure storage, accessibility and history of code as it is written, Git will be used to manage the project. Git is a version control system that allows for the tracking of changes to code and the ability to revert to previous versions of the code. GitHub will be used to host the code and will allow for the code to be accessed from anywhere, GitHub also has a free tier which will be used for this project. The GitHub repository for this project can be found at https://github.com/mattg66/fyp.

Chapter 4

Requirements

This chapter will provide in more detail the requirements for the automation platform. The requirements will be split into 2 sections, namely the functional requirements and non-functional requirements. The functional requirements will outline the features that the platform must have, and the non-functional requirements will outline the requirements that the platform must meet to be successful. To prioritise the requirements, the MoSCoW method will be used. The method will allow for the requirements to be prioritised and will ensure that the most important requirements are met first. The MoSCoW method is a prioritisation method that splits requirements into 4 categories, namely Must Have, Should Have, Could Have and Won't Have.

4.1 Functional Requirements

Functional requirements are outlined by the IEEE as a function that a system or a system component must be able to perform. ("IEEE Standard Glossary of Software Engineering Terminology", 1990)

ID	Details	Priority
FR1	Visual representation of rack space	Must Have
FR2	Add and remove racks from the space	Must Have
FR3	Add and remove Terminal Servers from racks	Must Have
FR4	Add and remove Fabric Nodes from racks	Must Have
FR5	Add, remove and update projects	Must Have
FR6	Expand or contract a projects consumption of rack space	Must Have
FR7	Automate configuration of ACI fabric	Must Have
FR8	Deploy virtual router using vCenter API	Must Have
FR9	Deploy virtual services stack to provide remote access VPN	Could Have
FR10	Continuous monitoring of ACI and vCenter health	Won't Have
FR11	Terminal server automated management	Must Have
FR12	Login system to restrict access	Could Have

Table 4.1: Functional Requirements

FR1 - FR2 outlines the requirements to have the rack space visualised in the web application. The idea behind this is that the application will simplify the process of adding and removing racks which will allow for the rack space to be easily recreated in the application, and will also allow for the rack space to be easily updated if the rack space changes. It will also help show the utilisation of the space, and allow for project planning to be carried out more easily.

FR3 - FR4 outlines the ability to associate ACI nodes and terminal servers to racks, this is required so that the automation backend can push the required config out when a rack is onboarded into a project. This also adds the ability to add and remove nodes and terminal servers if any physical changes are required in the rack space.

FR5 provides the ability to manage projects present in the rack space and will provide the core automation functionality.

FR6 details the requirement to expand and contract a project's rack space utilisation, this will allow for the project to be scaled up or down as required which is a common occurrence.

FR7 outlines the core automation functionality of the platform. This is to automate the deployment of connectivity to the ACI fabric based on the selected rack space and associated fabric nodes.

FR8 outlines the deployment of a virtual router to the vCenter automation platform. This

will provide internet connectivity to the project network created by FR7.

FR9 provides the ability to automate the creation of a project services stack, this may include services such as VPN and NTP to name a few.

FR10 outlines the possibility of having continuous status monitoring of ACI and vCenter, however, due to the required time to implement this feature it has been marked as a Won't Have.

FR11 details the ability to also automate the terminal servers associated with racks which will ensure that terminal servers are connected to projects upon their onboarding.

FR12 details the possibility of implementing a login system, whilst this would be a useful feature and should be implemented at some point, the project will be hosted on a secure network that requires access to be granted, so the login system may be out of scope given the time restrictions.

4.2 Non-Functional Requirements

Non-functional requirements describe the nonbehavrioral characteristics of a system, capturing the properties and constraints under which a system must operate (Antón, 1997)

ID	Details	Priority
NFR1	Must be easy to use for staff with less technical knowledge	Must Have
NFR2	The system status should be easily visible to staff (e.g. errors, project status)	Must Have
NFR3	The system should be able to easily integrate with existing ACI fabric deployments	Could have

Table 4.2: Non-Functional Requirements

NFR1 outlines the requirement for the web application to be easy to use for less experienced team members. By abstracting away the networking and configuration through the use of automation, only a basic understanding of networking should be required for the platform to be used.

NFR2 shows that the system must report the status to staff via the use of status indicators. This should show the progress of the automation scripts as they progress through automating and applying the configuration to various elements of the network.

NFR3 outlines for the platform to be able to integrate with existing ACI fabric deployments. This will allow for the platform to be used in a production environment without the need to reconfigure and rearchitect the fabric. Ideally, the platform should be deployed alongside a new fabric in a greenfield deployment.

Chapter 5

Design

As the project has utilised an iterative approach with the Kanban methodology, the design has changed and been refined throughout the project. The design of the software will be broken down into 3 sections, namely the design of the web application and the design of the ACI and vCenter configuration. The design of the testbed will also be detailed.

5.1 Web Application

5.1.1 Architecture

A client-server architecture will be utilised to provide the user-facing experience, and the automation and data handling logic hosted on the server. By using this model, many clients can request and interact with data that is hosted on one central server. The backend and frontend that make up the application will be separate from one another, and will therefore be developed independently, with the frontend interacting with the backend via a REST API. This allows for the front end to be a SPA, which facilitates a better user experience due to the lack of page refreshes upon every request.

The server will process all requests generated by the front end and also make requests to the various APIs that will be required to automate the network deployment. The database will also store all data required by the server to generate the appropriate network configuration that is required to automate the deployment of projects to the network.

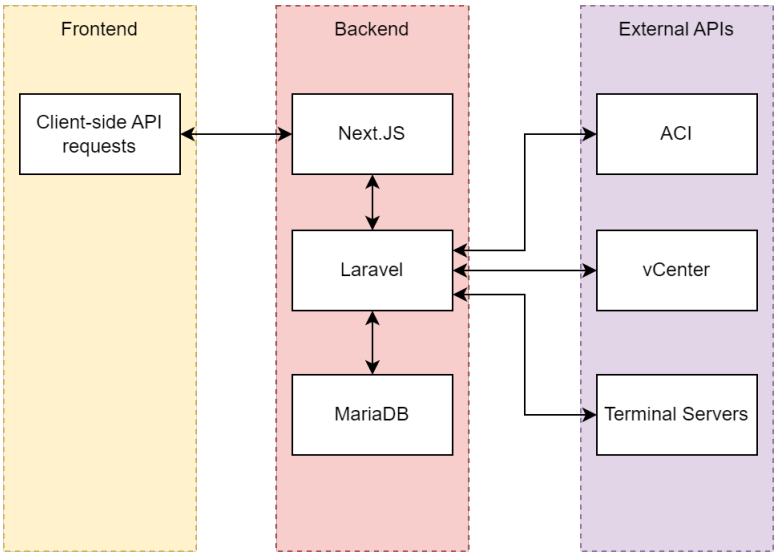

Figure 5.1: Web Architecture Design

5.1.2 Frontend

Next.js will be used to power the frontend of the application as it is an enhancement of React.js and provides server-side rendering and acceleration of pages. React uses a modular 'componentised' approach to building the frontend, which allows for the creation of reusable components that can be used throughout the application. This allows for the creation of a modular and scalable frontend that can be easily extended and maintained.

React.js also has an extensive library of open-source components and libraries that can be utilised to make developing the frontend easier and more feature complete. Due to the complexity of some required features, such as having a drag-and-drop interface for the recreation of the lab space, the use of a library such as React Flow will be required as the time required to develop such a feature would be out of the scope of this project.

5.1.3 Backend

The backend of the application will be written in PHP, using the Laravel PHP framework. Laravel is a popular PHP framework that will accelerate the development process, as it features an inbuilt ORM, API routing system and an authentication system that can be implemented. Laravel utilises SQL-based databases, and as such MariaDB will be used as the database for the application. Laravel also features an in-built HTTP client which will be required to interact with the ACI and vCenter APIs which are all REST-based.

Laravel follows the MVC architecture, however as the frontend is a React SPA, Laravel will only be used to provide and consume the data via its REST API.

5.1.4 Database Design

As the project will need to store data persistently, ensuring that the database has an appropriately designed schema will ensure that the data is stored in a way that is easily

accessible and can be queried efficiently. The design of the database was tweaked and refined throughout the development process, however, the final ERD is shown below in Figure 5.2.

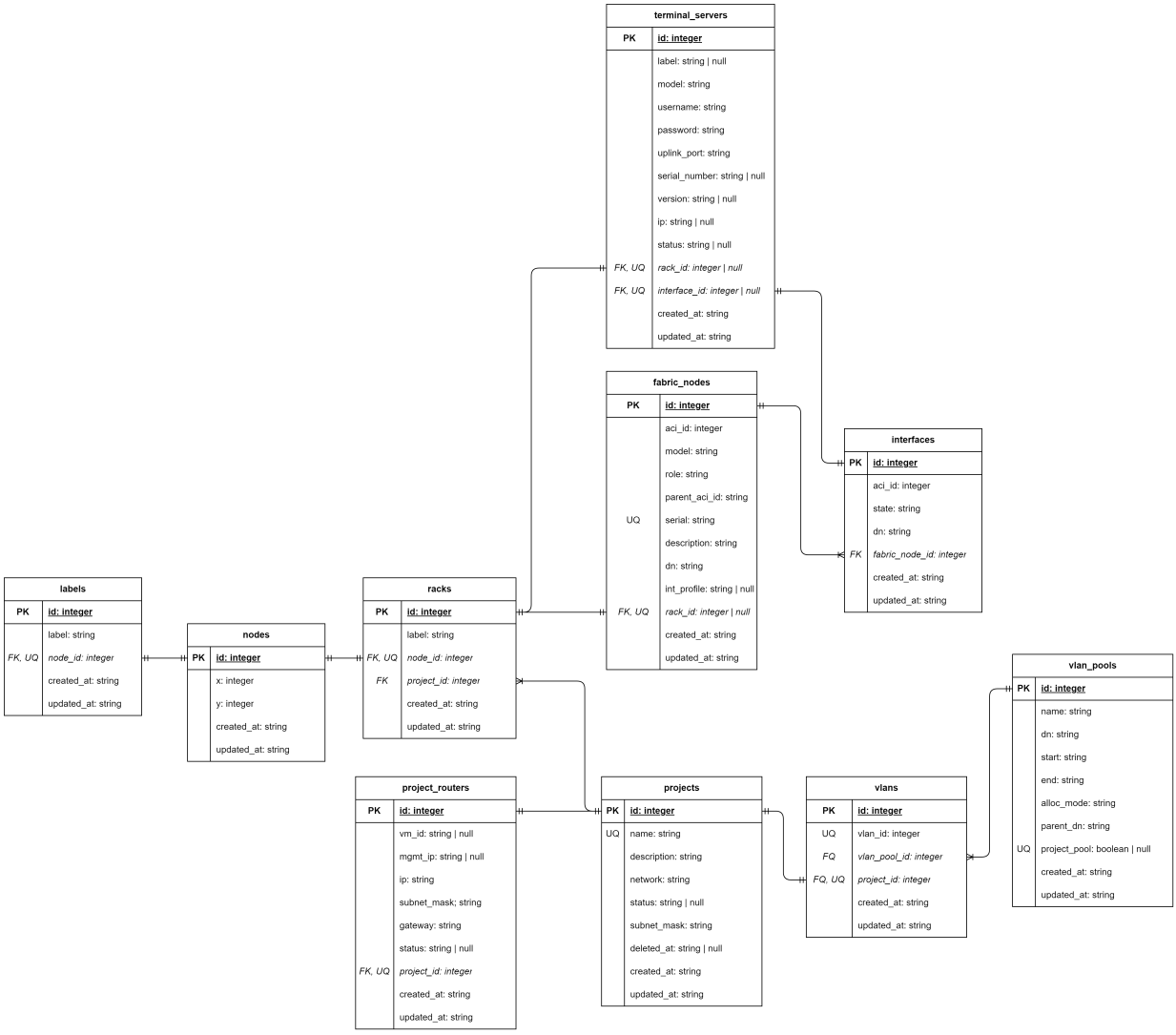

Figure 5.2: Database ERD

Nodes

Nodes provide the root of storing information about the layout of the rackspace. Each node can either have a rack or a label attached to it. Labels will serve as a place to store text information on the rackspace diagram for informational purposes only. Racks will represent a physical rack in the rackspace. By using a node table, the position of the node can be abstracted away from the other data which is more relevant to the automation side. A more refined and efficient query is also possible as all nodes can be easily retrieved by selecting all nodes and then joining the rack and label tables to get the relevant information.

Fabric Nodes and Interfaces

The fabric nodes and interfaces tables will be used to store information related to all fabric nodes that are attached to ACI. They will be automatically populated by the automation script which will retrieve information about the nodes and their associated interfaces from ACI. Each interface will tie to a fabric node so that all interfaces belonging to a fabric node can be easily retrieved. A role and parent ID field are also included in the fabric node table which allows for FEXs to also be stored as nodes. ACI treats FEXs as child nodes to leafs, hence why the parent ID and role field are needed to provide the correct differentiation between the two.

Terminal Servers

The terminal servers table will store information about the various terminal servers present in the rack space, these will be inserted manually via the web UI. A relation will also exist that associates an interface to a terminal server so that the automation scripts can appropriately configure the uplink ports on the ACI fabric.

Projects

The projects table will store all projects present in the application. Each project will link to the racks via a project ID field in the racks table. The project's private subnet will also be stored with the project.

Project Routers

A project router will have a one-to-one relationship with a project, allowing only one project router per project. This table will keep track of the virtual router that is created for each project, and will also store the WAN IP address assigned to the router.

VLANs and VLAN Pool

The VLAN pool table will store a list of all VLAN pools present in ACI. It will also store the selection that the user has made as to the VLAN pool that should be used by the automation platform for its endpoint groups. The VLANs table will be used to keep a record of which project is using which VLAN within a VLAN pool so that a VLAN cannot be used more than once.

5.2 Testbed

To support the development of the automation platform, it is necessary to have a network that can be used to test the automation functionality on real hardware and software that would be used in production. The network will be built using a scaled-down version of what could be deployed in a real scenario, the same also applies to the associated compute and storage resources.

5.2.1 OOB Network

To provide connectivity for management and day-zero configuration, it is important to have a reliable OOB network. The purpose of a OOB network is to provide access to networking and infrastructure that is external to the main network so that in the event of a failure, the devices can still be reached to rectify any problem that may have occurred. The OOB network will be a single layer 2 network, with a single switch providing connectivity to all devices. The switch will be a Cisco Catalyst 2960-XR, and will be connected to external infrastructure which will host a VPN server and a NAT router so that devices connected to the OOB network also have access to the internet. The VMs hosted on the ESXi server are also detailed in the topology shown in figure 5.3

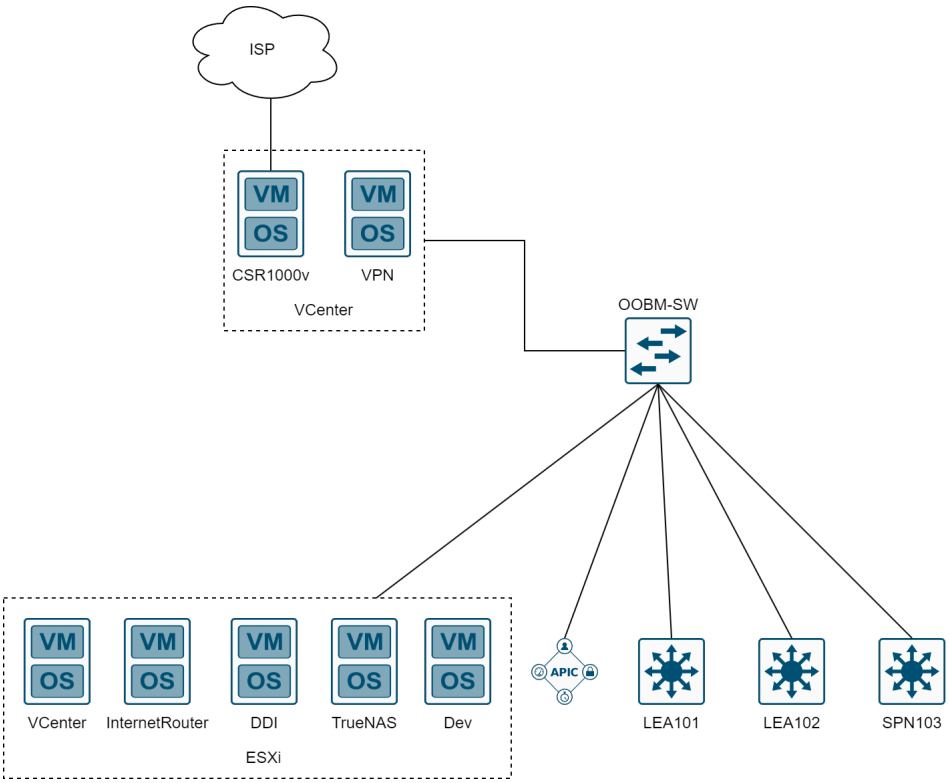

Figure 5.3: OOB Topology

5.2.2 ACI Fabric

As the automation platform is designed to automate ACI fabrics, a reduced ACI fabric will be deployed for development and testing. The base fabric will consist of one spine and two leafs, these being the N9K-C9336PQ and N9K-C93180YC-EX respectively. The leafs will also have a total of 3 FEXs attached to provide RJ45 connectivity to the fabric, and to also ensure that the automation platform will correctly support FEXs. The FEXs used will be the N2K-C2248TP-E-1GE. ACI also requires at least one APIC to function, so a single APIC M2 will be connected to the pair of leafs to provide overall administration and control over the fabric.

A single ESXi host will also be included in the network, which will be also dual-homed to both leafs to provide connectivity, and also test LACP functionality. The ESXi host will be used to host the automation platform and will also host the associated infrastructure required for the automation platform to function, such as vCenter.

Two routers simulating terminal servers will also be included in the design. Cisco routers can have additional modules inserted into them allowing them to provide console connectivity to devices via SSH or Telnet. Shown in figure 5.4 is the fabric topology that will be used for the testbed. Device names/hostnames are shown in the figure for clarity.

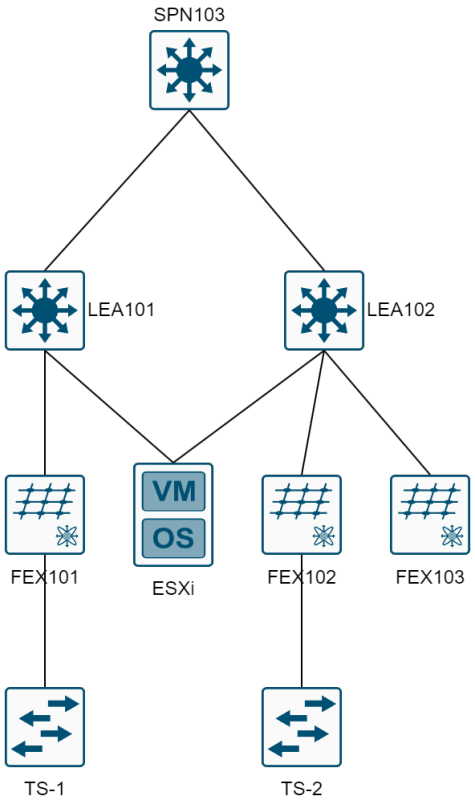

Figure 5.4: Fabric Topology

5.2.3 ACI Policy

To provide internal connectivity within the ACI testbed, the correct policy will need to be designed to facilitate this. As ACI is policy-based, conventional networking paradigms are shifted and abstracted behind ACI. Figure 5.5 shows the tenant and the policy it houses that will be created to provide connectivity within the testbed.

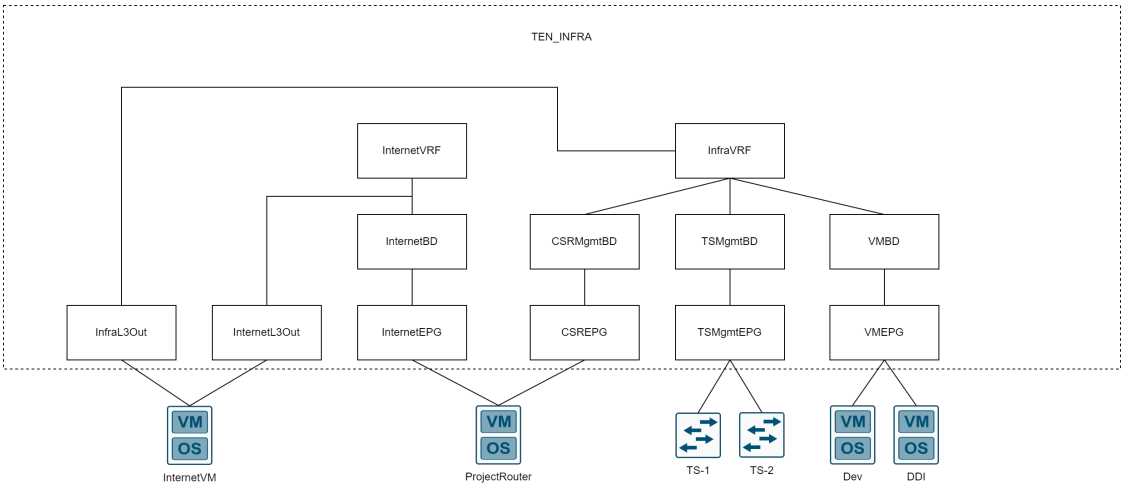

Figure 5.5: ACI Policy Overview

Infra VRF

The infra VRF will provide L3 routing ability between all associated bridge domains and EPGs for the sake of simplicity, all testbed connectivity for infrastructure will be able to communicate with one another.

InfraL3Out

The InfraL3Out will be used to advertise connectivity to outside the ACI fabric, the purpose of this is to provide internet connectivity via a NAT router as ACI can't provide NAT. This results in any packets destined for anything other than locally attached routes present in the InfraVRF will be routed out of the InfraL3Out.

OSPF will be used to share routes between ACI and the NAT router.

Infra Bridge Domains

Three bridge domains bring the ability to split connectivity into different subnets and broadcast domains. This allows for L2 functionality such as DHCP to be easily controlled and connectivity to be segmented. There will be a bridge domain to provide connectivity for the following:

- Virtual Project Routers
  - DHCP relay will be operational to forward DHCP discover requests onto the DDI server attached to the VM bridge domain, thus allowing newly created VMs to receive an IP address automatically, to facilitate being provisioned by the automation platform.
- Terminal Servers
  - Allows the automation platform to reach the terminal servers via their RESTCONF API to apply the configuration.

- Infrastructure and testing VMs
  - Provides connectivity to the fabric for testing and provides services such as DHCP.

Infra EPGs

Each EPG has a one-to-one relationship with a bridge domain and is used to provide connectivity via VMWare ACI integration and static port associations.

Internet VRF/BD/EPG

This VRF/BD/EPG will be used to emulate the project routers having a connection to the internet. This will be used to test the automation platform's ability to configure the project routers to have internet connectivity.

Connectivity will be provided via the InternetL3Out using OSPF for routing advertisements to the same NAT router that provides internet connectivity to the Infra VRF, although a different subnet will be used.

IP Addressing

Property	Network Address	Gateway
InternetBD	172.16.0.0/24	172.16.0.254
TSMgmtBD	172.16.1.0/24	172.16.1.254
CSRMgmtBD	172.16.2.0/24	172.16.2.254
VMBD	172.16.3.0/24	172.16.3.254
OOB	192.168.0.0/24	192.168.0.254

Table 5.1: Functional Requirements

References

Antón, A. (1997). Goal identification and refinement in the specification of information systems. *PhD Thesis, Georgia Institute of Technology*.

Atlassian. (2023). *Trello brings all your tasks, teammates, and tools together*. Retrieved January 3, 2023, from https://trello.com/

CDW. (2015). The future of networking arrives. *Commun. ACM*, 16–19. https : / / webobjects.cdw.com/webobjects/media/pdf/CDWCA/White-Paper-The-Future-of-Networking-Arrives-MKT2862CA.pdf.

Ieee standard glossary of software engineering terminology. (1990). *IEEE Std 610.12-1990*, 1–84. https://doi.org/10.1109/IEEESTD.1990.101064

Kirkpatrick, K. (2013). Software-defined networking. *Commun. ACM*, *56*(9), 16–19. https://doi.org/10.1145/2500468.2500473

Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14–76. https://doi.org/10.1109/JPROC.2014.2371999

Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, *16*(3), 1617–1634. https://doi.org/10.1109/SURV.2014.012214.00180

teamgantt. (2023). *Teamgantt is the refreshing solution that brings project scheduling software online*. Retrieved January 3, 2023, from https://trello.com/