Harnessing Software Engineering Principles in Network Operations

The Application of DevOps in Campus Area Networks

Shivam Singh

March 26, 2025

A Proposal Presented to
The Department of the Information Science and Technology
College of Science Technology and Applied Arts of Trinidad and Tobago

In (Partial) Fulfillment
of the Requirements for the Degree
Bachelor of Science

Complementary content available at
`https://github.com/Shivam-S-Singh/COSTAATT_Final_Project`

Contents

1 Executive Summary

1.1 Overview

This document serves as a proposal to adopt a DevOps approach to managing the network infrastructure for the College of Science, Technology and Applied Arts of Trinidad and Tobago.

[Note: Complete Executive Summary once enough content is generated.]

Many of the software used is open source or freely available. NetDevOps

1.2 What this proposal does not cover ?

This solution does not include the implementation of the following elements which are common components in Campus Area Networks.

- Wireless Access
- Data Center Routing and Switching
- Voice and Collaboration

The reason these topics are not covered is because this proposal is not an all inclusive solution and is intended to provide a sense of direction for the organization to transition to utilizing the methodologies associated with Developer Operations. As such, I limited the proof of concept to wired network connectivity and security.

2 Introduction

2.1 Client Background & Relevance

COSTAATT is a public tertiary institution in Trinidad and Tobago that offers programs in the areas of Information Technology, Business and Nursing, just to name a few.

The solution being proposed can be applied to all types of organizations with sizeable networks as it is aimed towards addressing the management of a large-scale topology with limited human resources. The reason COSTAATT is an ideal candidate for this solution is because of their vast amount of networks and communication equipment that span multiple campuses across Trinidad and Tobago.

2.2 Problem

Critical services such as file sharing, active directory and email which was originally on premise is now being hosted on cloud based platforms or being migrated to decentralized infrastructure such as remote data centres. High availability, fault tolerant networks have become a requirement in order to ensure consistent access to these resources.

The traditional methods of managing the network has led to longer turnaround times with regards to troubleshooting issues, and scalability when making configuration changes to accommodate new devices or applications.

The following topics below are areas that present challenges within ***Network Operations*** that can be solved by adopting a ***Developer*** approach to the problem.

2.2.1 Documentation

During the planning phase in network design we typically use Visio, spreadsheets and word documents to map our infrastructure. Where Visio is used to visualize our topology and spreadsheets or word documents would be used for IP address or VLAN assignment information. Text files holding configuration information would be stored in a centralized location and would sometimes be altered and then ran on the device where changes are required. This has traditionally been the standard for documenting our network but it comes with its drawbacks, specifically with regards to maintaining the documentation after the network is modified.

Several issues faced with regards to maintaining this documentation are

- Manual updates are required by network operators
- Lack of accountability when changes are executed
- Limited collaboration between colleagues

Manual updates are required whenever a change is made on the network, e.g., a network device is added or a new route or VLAN has been provisioned on a device. This would mean an engineer would have to amend the topology in the Visio diagram and make changes in the related spreadsheets or word documents. This can be a time consuming task that is usually not a priority for the engineer. The task of updating the documentation sometimes gets pushed back to the point where it never gets completed. A consequence of this is when an incident occurs or configuration changes are requested you are stuck with unreliable documentation which can delay resolving the issue or carrying out changes.

There is a lack of accountability when changes are made. For instance, lets say an ACL entry was configured to block FaceBook for a specific part of the network, why was this change made and who was the engineer that made the change. This change could have been implemented as a request from a manager who finds that their staff is spending too much time on social media or from the IT team in response to slow speeds due to over utilization of an internet circuit. When a change is made, we should have the necessary details when we look back on this change. Who made the change, when was it executed and why was it executed. This context and details is instrumental in effectively managing our network.

Collaboration on documentation can also have its limitations when different teams are associated in the network management process. Someone in the Systems Administration team may have documentation that was shared to them via an email from the Network Engineering team. Changes may have been implemented since that last email correspondence and the Sysadmins would not be aware that they are operating with outdated information.

2.2.2 Production Network as the Single Source of Truth

The Single Source of Truth (SSoT) is the action of centralizing all data about Information Systems in a single location. In reference to what was mentioned earlier with regards to collaboration on documentation, when documentation is spread across multiple sources it can be difficult to determine the desired state of the network. How can we be sure that the documentation that we are looking at is the actual intended behaviour of our network and not obsolete or inaccurate information ?

When the *desired state* of the network does not match the existing *known state* this leads to an issue known as configuration drift. Configuration drift can lead to unpredictable system behaviour and can extend the time when carrying out maintenance tasks on the network. We are now forced to utilize the production network as our source of truth as we can no longer rely on our documentation to dictate our actions moving forward.

We appropriately refer to the single central location of network information as the ***Network Source of Truth*** **(NSoT)**.

2.2.3 Configuration Management

CLI-based configurations are executed to manually make changes on network devices. This is acceptable for the basic initial configuration of a network device but it does not scale well since this is prone to human error. We can make mistakes with regards to syntax and maintaining consistency when configurations have to be made across a fleet of devices.

Well designed network environments often utilize redundant switches or routers in order to have increased network availability. When implementing configuration changes in these environments certain tasks become repeatable, e.g., lets say we have to provision a new VLAN on our network, we would have to add the VLAN on all the related switches. You can argue that we can use VTP to simplify this process but this has its drawbacks which will be discussed in the design of our network. This task can quickly become time consuming since the engineer would have to manually make the change on each switch.

Networks are typically made up of devices from multiple vendors, each with their own proprietary operating systems and unique syntax. This can add an additional level of complexity as now the operator has to possess knowledge specific to each platform. This can contribute to the risk of misconfiguration and also delay the execution of changes on the network.

2.2.4 Testing on Production & Testing Frameworks

Testing is important in DevOps but this has been a huge limitation for network engineers since traditionally for testing we would have to utilize real equipment. Having a non-production environment can be costly and most organizations would not invest in additional hardware. Not having a test environment means that we would have to execute changes directly in our production network. Configuration changes can potentially have unpredictable results which could lead to performance issues or downtime, so having some form of a test environment may possibly mitigate these risks.

Loopback to writing the testing framework once this is clear.

2.2.5 Change Management

Network changes can lead to potential loss of services so the process of implementing configuration changes can at times be long and drawn out. This is due to the fact that changes typically require multiple individuals to review as well as sign off on the change before any action is taken. These reviews can involve technical write-ups and meetings with senior staff and or stakeholders. When preparing change documents or presentations for meetings the engineer must tailor it for both technical and non-technical staff which takes additional effort to complete. Meetings can also be delayed based on the availability

of the necessary stakeholders. The fact that senior members are also involved in the process means that their attention is now diverted to this process when it could be spent elsewhere on other critical issues or projects.

2.3 Solutions

2.3.1 Documentation: Version Control System (VCS)

Version control systems are often utilized by Software Engineers in order to track changes in their code and collaborate with fellow engineers on projects. It provides accountability because you can see who made the changes and when it was executed. This system also allows you to roll-back to previous software builds in the event that a change breaks something.

A version control system along with an Infrastructure as Code implementation can be utilized for our configuration change process. When changes are executed it becomes self documenting and is stored in a centralized location. You can say that documentation now becomes baked into the process and would no longer be an afterthought. We will be utilizing Git as our VCS of choice and use GitHub to keep our code centralized.

2.3.2 Production as the Single Source of Truth: IPAM System

We can utilize a separate system in order to log the desired state of our network. It will allow for efficient planning in the development of our network as we will be able to define VLANs, Subnets, Interface Assignments and associate these elements with one another. It also has features that would alert you if you try to add overlapping subnets.

NetBox has become an industry standard IPAM and DCIM system over recent years and has been instrumental in helping organizations build, maintain and scale their networks based on their changing requirements.

2.3.3 Configuration Management: Infrastructure as Code (IaC)

Infrastructure as Code is a strategy that involves using code to manage and configure resources such as servers or network devices instead of using manual options such as the CLI.

Using IaC eliminates the issue of syntactical errors associated with CLI-based input. There is improved consistency when executing changes across multiple devices. We can scale our network by simply modifying our code to accommodate new changes. Deployments are significantly faster because the provisioning process can be automated. The code can be used across multi-vendor platforms through the use of APIs.

Ansible is an agentless automation framework written in Python that was originally developed to configure servers but has made its way into network operations. We will use this platform to carry out our configuration changes on our devices instead of directly interacting with the device's CLI. When we say Ansible is agentless, it does not require the installation of agents on end devices in order to establish communication between the Ansible host machine and the device. Other available configuration tools may have this requirement. Ansible uses playbooks which define a list of task that

2.3.4 Testing on Production & Testing Frameworks: Automated Testing

3 Objectives

3.1 Building our Network

This is not an in-depth guide to designing networks, however it is important that we construct a system that is up to current standards. As we mentioned earlier the focus here will be wired connectivity as well as security. We will be utilizing the *NSA's Network Infrastructure Security Guide* as well as *Cisco's Campus LAN and Wireless LAN Solution Design Guide* to construct our network.

3.1.1 Deployment Strategy

When it comes to deploying networks, *Cisco's PPDIOO Lifecycle* has been instrumental to network engineers going back to the early 2000's. We will be adapting our deployment strategy by utilizing the PPDIOO model alongside DevOps principles.

1. **Prepare:** Determine the business requirements.
2. **Plan:** Identify the network requirements based on business goals.
3. **Design:** Provide blueprints that conceptualizes the completed product.
4. **Implement:** Network elements are deployed based on design specifications.
5. **Operate:** Maintain and monitor the networks functionality.
6. **Optimize:** Search for areas where operational efficiency can be improved.

The PPDIOO model is a lengthy process but it is designed to produce the best possible product for your client.

DevOps which has garnered a culture of *moving fast and breaking things* favours an approach of rapid development by utilizing project management concepts such as **Agile**. Agile was introduced to streamline the traditional methodology of producing software which follows the *Software Development Lifecycle (SDLC)*.

The following procedures are associated with the SDLC.

1. **Plan:** Determine the business requirements.
2. **Design:** Prototype solutions based on business goals.
3. **Implement:** Code the proposed solution.
4. **Test:** Check for bugs/errors and that it meets user requirements.
5. **Deploy:** The final product is packaged and released.
6. **Maintain:** Execute changes related to bug fixes, user issues and performance.

You may have noticed that the SDLC and PPDIOO Lifecycle share common stages with one another. Given this realization, we can safely assume that we can apply Agile concepts to our PPDIOO model in a similar manner.

Agile is a philosophy that focuses on delivering customer satisfaction through cross-team collaboration and being able to quickly adapt to changes. Instead of waiting for the completion of each phase in its entirety before advancing, we aim for small iterative changes throughout the process until the desired results are achieved.

Taking this philosophy into consideration, we can focus on the configurations of basic connectivity first, such as the VLANs, IP Interface Assignments and Routing. Then after we can shift our attention to configurations associated with redundancy and security, e.g., FHRP, STP and ACLs. The physical network will have to be completed before these configurations can be carried out.

3.1.2 Design Phase

We will adhere to these practices by breaking down the Preparation, Planning and Design phases into the following tasks. The idea here is we want to spend less time on the design phases and start the implementation phase as quickly as possible to deliver a minimum viable product.

1. **Gather Business Requirements / Translate to Network Specifications**
2. **Create a HLD of our Topology**
3. **Select Devices for Implementation**
4. **Utilize NetBox to Develop Logical Design**

3.1.3 Implementation Phase

The implementation phase is dependent on the configuration of the Control Node and its related assets. This will have to be completed before changes can be pushed to our devices. For this phase, we will be focused on the initial configs to allow management to our devices.

1. **Add Devices to GNS3 and *"Cable"* Equipment**
2. **Generate Initial Device Configurations**
3. **Confirm Connectivity between Devices and Control Node**
4. **Begin Configuring the Network**

3.2 Configuring our Control Node

3.2.1 Installing our Base Operating System

3.2.2 Installing necessary Software Packages

3.2.3 Configuring System Environment for Network Operations

3.3 Establishing the Configuration Work Flow

4 Resources

4.1 Host System

The following below details my system environment that will be used to simulate the network. An Ubuntu host machine will be used for this project. If you use GNS3 on Windows you will have to use a GNS3VM that is running on VMware. The reason for this is because Windows does not have native support to efficiently emulate the network OS and the GNS3VM (which is really an Ubuntu VM) handles this. The problem with this is that there is nested virtualization taking place and you won't utilize the full capabilities of your systems hardware. If you run GNS3 directly on Ubuntu then you will be able to make full utilization of your systems hardware.

OS: `Kubuntu 24.04.1 LTS x86_64`
Motherboard: `B450 AORUS PRO WIFI`
Kernel: `6.8.0-55-generic`
Shell: `bash 5.2.21`
Desktop Environment: `Plasma 5.27.11`
Window Manager: `KWin`
Terminal: `konsole`
CPU: `AMD Ryzen 5 3600 (12) @ 3.600GHz`
GPU: `NVIDIA GeForce GTX 1060 6GB`
Memory: `32GB`
Storage: `1000GB`

4.2 Design

4.2.1 Topology , Physical Rack and Change Process Flow

Draw.IO is used for the creation of diagrams such as flow charts and network designs. It is a great alternative to Visio and no license cost is attached.

4.2.2 Network Source of Truth

NetBox is utilized to document networks with relevant information such as IPAM and DCIM. NetBox is intended to be used as a single source of truth within network operations.

4.3 Network Components

4.3.1 Simulation Software

GNS3 stands for Graphical Network Simulator-3. It is a popular tool used for testing networks, studying for certifications and in our case to develop a proof of concept for our proposed design.

4.3.2 Network Operating Systems

The following associated files are used to simulate the Network Operating Systems of our selected hardware within GNS3.

Firewall: Fortinet FortiGate 100E
Image Version: `FGT_VM64_KVM-v7.0.14.M-build0601`

WAN Switch: Cisco Catalyst 2960C
Image Version: `vios_l2-adventerprisek9-m.ssa.high_iron_20200929.qcow2`

Router: Cisco ASR 1001-X
Image Version: `csr1000v-universalk9.17.03.08a-serial.qcow2`

Aggregator Switches: Juniper EX2300
Image Version: `vJunos-switch-24.4R1.9.qcow2`

Access Switches: Arista 7050T-52
Image Version: `vEOS-lab-4.33.1.1F.qcow2`

4.3.3 Network Services

Several IP Services will have to be deployed in order to operate our network. Each of these services will be installed within their own Docker container.

DNS: *BIND 9* is a domain name server which is used to map human readable names to IP addresses.

DHCP: *Kea DHCP* is a server that distributes IPv4 and IPv6 addresses to clients on modern networks.

Syslog: *syslog-ng* is an implementation of the syslog protocol which is used for log collection from a variety of sources, including network devices.

NTP: *Chrony* is a Network Time Protocol server used to keep the time settings on devices synchronized.

RADIUS: *FreeRADIUS* is a suite of software that allows centralized management for the authentication, authorization and accounting of network devices.

4.3.4 Virtualization / Containerization

Docker is a container based technology that is used to package applications within an isolated environment. This environment has all the necessary packages and dependencies installed in order for the application to function.

4.4 Infrastructure as Code

4.4.1 Automation Platform

Ansible is an automation framework that was originally developed for configuring servers, however, in recent years it has grown in popularity as a method for provisioning network devices.

4.4.2 Version Control System

Git is a distributed version control system that allows developers to track changes to their code.

GitHub is an online platform that allows developers to collaborate on projects and share their code.

4.4.3 CI & CD

4.4.4 Automated Testing

5 Activities

6 Measurement

[Note: Define measurements later on in project]

7 Budget

[Note: Use the cost for each representational device on the network. This will only cover the devices and not the racks or the cabling associated with its design as this is merely conceptual.]