MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Document

Research SDN (Software-Defined Networking) and build test models.

GFA23IA06	
Group Member	Phan Huyền Trâm - SE151121 Phạm Thanh Tân - SE151045 Phạm Quang Linh - SE150083 Trần Doãn Anh - SE150630 Mai Duy Nam - SE151347
Supervisor	Mai Hoàng Đỉnh
Ext Supervisor	
Capstone Project code	FA23IA06

- Hồ Chí Minh, 12/2023 -

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1. Project Information

Capstone Project Name: Research SDN (Software-Defined Networking) and build test models.

Project Group Name: GFA23IA06

Timeline: 04/09/2023 – 02/12/2023

1.2. The Participants

1.2.1. Instructors

Full name	Phone	E-Mail	Title
Mai Hoàng Đỉnh			Lecture

Table 1.2.1. Supervisors

1.2.2. Team Members

Full name	Student code	Phone	E-mail	Role in Group
Phan Huyền Trâm	SE151121			Leader
Phạm Thanh Tân	SE151045			Member
Phạm Quang Linh	SE150083			Member
Trần Doãn Anh	SE150630			Member
Mai Duy Nam	SE151347			Member

Table 1.2.2. Team members

1.3. The problems

1.3.1. Overview

A robust network infrastructure in an enterprise provides several advantages, including connectivity and information sharing, enhanced collaboration, rapid service deployment, efficient resource management, data security, integration of new services, and improved user experience.

The current capacity of the Internet is quickly becoming inadequate to handle the increasing volumes of traffic generated by new services and technologies such as mobile devices, virtualization, cloud services, and big data. This is due to the growing number of users, sensors, and applications. Globally, the total number of Internet users is projected to grow from 3.9 billion in 2018 to 5.3 billion by 2023 at a CAGR of 6 percent. In terms of population, this represents 51 percent of the global population in 2018 and 66 percent of global population penetration by 2023.

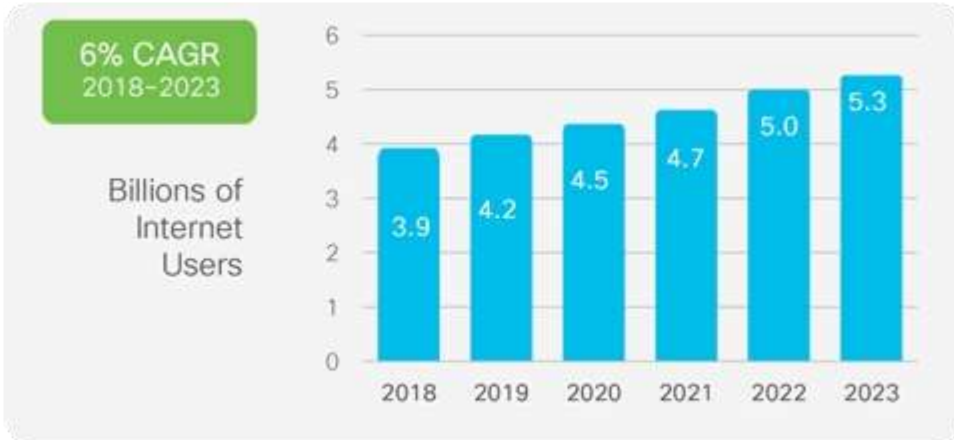


Figure 1.3.1. Growth in Internet users

In addition, to implement network-wide policies and support new services, managers today must configure thousands of network devices and protocols, which makes it difficult to apply a consistent set of QoS, security, and other policies. Networks become vastly more complex with the addition of thousands of network devices that must be configured and managed. These devices have their control and forwarding logic parts both integrated in monolithic, closed, and mainframe-like boxes. Consequently, only a small number of external interfaces are standardized (e.g., packet forwarding) but all their internal flexibility is hidden. The internals differ from vendor to vendor, with no open software platform to experiment with new ideas. Traditional networks based on static Ethernet switches arranged in a hierarchical tree topology are no longer suitable for the dynamic computing and storage needs of modern hyper-scale data centers, campuses, and carrier environments.

A lack of standard open interfaces limits the ability of network operators to tailor the networks to their individual environments and to improve either their hardware or software. Hence, there is a need for a new network equipment architecture that decouples the forwarding and control planes of the routers to dynamically associate forwarding elements and control elements. These infrastructures should also enhance network speed, scalability, and reliability to meet the demands of today and the future.

1.3.2. Solution

Software-Defined Networking (SDN) provides a new way of designing, deploying, and managing networks. A new concept that is used by famous technology corporations such as Google, Facebook... It does this by decoupling the network's control (logical) plane from the data (forwarding) plane, allowing network administrators to programmatically control and optimize network resources via open interfaces. This approach directly addresses several key problems with the current state of networking as below:

- **Complexity and Management Overhead**

SDN's centralized control function can simplify network design and operation by consolidating the control plane into a single logical point. Network administrators can manage the entire network from this central point, reducing the burden of configuring and managing thousands of devices individually. In SDN, the control plane is abstracted and centralized, making it possible to configure and manage the entire network from a single point. This simplifies the implementation of network-wide policies and the configuration of thousands of devices.

- **Lack of Flexibility**

SDN is fundamentally more adaptable than traditional networking technologies. By leveraging programmability, SDN allows network administrators to dynamically adjust network traffic, implement complex network-wide policies, and integrate new services more easily. Network administrators can program SDN to dynamically control the network, adjust traffic flow, implement complex policies, and introduce new services. This reduces manual configuration, increasing efficiency and reducing errors.

- **Vendor Lock-in**

SDN supports open protocols, such as OpenFlow, that enable multi-vendor hardware and software solutions. This breaks the cycle of vendor lock-in and promotes a more competitive and innovative networking market. Because SDN uses open protocols, it can work with any vendor's hardware or software, promoting a more innovative and competitive market.

- **Scalability and Adaptability**

With the separation of the control and data planes, SDN enables dynamic, on-demand scalability to handle increasing traffic volumes and network complexity, making it a more suitable approach for large-scale, dynamic environments such as cloud services, big data, and mobile applications. SDN can scale on demand to manage growing traffic and network complexity. It can quickly adapt to changes in business needs and technology trends, making it ideal for modern, hyper-scale data centers, campuses, and carrier environments.

CHAPTER 2: PROJECT MANAGEMENT PLAN

2.1. Problem Setting

2.1.1. Name of Capstone Project

English: Research SDN (Software-Defined Networking) and build test models.

Vietnamese: Nghiên cứu SDN (Software-Defined Networking) và xây dựng mô hình thử nghiệm.

2.1.2. Problem Abstraction

Networks become vastly more complex with the addition of thousands of network devices that must be configured and managed. One of the solutions to handle this situation is SDN (Software-Defined Networking) which offers benefits such as improved network management, scalability, agility, security, and performance optimization for enterprises. It empowers organizations to efficiently manage their networks, adapt to changing requirements, and embrace new technologies and innovations.

2.1.3. Project Overview

Currently, the traditional architecture is inadequate for meeting the computing needs of the features, traditional architecture approaches based on manual configuration of proprietary devices are cumbersome and error-prone, and they cannot fully utilize the capability of physical network infrastructure. Due to the scale, heterogeneity, and complexity of current and future computer networks, traditional methods for configuration, optimization, and troubleshooting will become ineffective and, in some cases, this is not enough. One of the fundamental features of Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) in particular is the self-managed service. In addition, effective network access to enterprise resources is becoming essential to meet today's computing requirements. SDN is touted as a most promising solution for those problems. SDN's main principle is to separate the control plane from the data plane to enable flexible and effective network management and operation by software. Specifically, devices (e.g., switches and routers) in the data plane perform packet forwarding, based on rules installed by controllers. Controllers in the control plane oversee the underlying network and provide a flexible and efficient platform to implement various network applications and services. Under this new paradigm, innovative solutions for specific purposes (e.g., network security, network virtualization) can be rapidly implemented in the form of software and deployed in networks with real traffic.

In this project, we will concentrate on creating a simulation model to test out SDN solutions. On the PNETLab platform, we will create a straightforward simulation of the Kubernetes model that corresponds to the real model used by enterprises. We will use CNI Calico to administer the network, K8s to manage resources, centralized management system, and K9s to display the model.

2.2. Project Organization

2.2.1. Solution Process Model

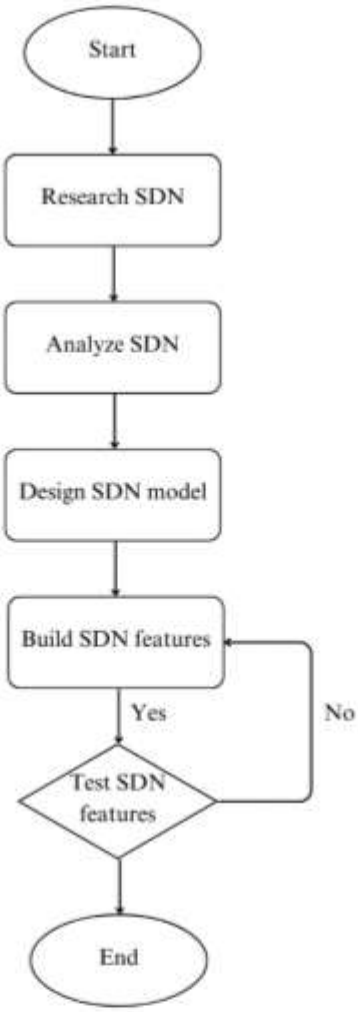


Figure 2.2.1. Solution Process Model

In this model, there are several phases:

Phase 1: Meet & Plan

During this phase, we organize and participate in group meetings to discuss topics, propose goals, and plan the implementation of this project.

Phase 2: Research SDN

During this phase, we collect documents and articles related to our topic to gain a deeper understanding of the SDN concept and its structure.

Phase 3: Analyze SDN and Design SDN model

After having a basic knowledge of the SDN concept, we discuss and analyze how it works. Then we find tools, and environments that suit our expectations to build a test model.

Phase 4: Build SDN feature

We choose and develop some outstanding features of SDN.

Phase 5: Test SDN feature

After building the model and its features, we create a test plan for each feature to ensure it works as we expect.

Phase 6: Feedback

We present and collect feedback from instructor and review board.

2.2.2. Roles and Responsibilities

Phase	Full name	Role	Responsibilities
Planning	Phan Huyền Trâm	Team Leader, Consultant, Document	Delegating tasks Analyzing goals Planning milestones Monitoring working progress Setting working rules
	Phạm Thanh Tân	Technical, Consultant, Document	Reviewing the plan Collecting documents Identifying project process model
	Phạm Quang Linh	Technical, Consultant, Document	Reviewing the plan Collecting documents Identifying resources
	Trần Doãn Anh	Technical, Consultant, Document	Reviewing the plan Collecting documents Identifying project process model
	Mai Duy Nam	Technical, Consultant, Document	Reviewing the plan Collecting documents Identifying resources
Risk Assessment	Phan Huyền Trâm	Team Leader, Consultant, Document	Analyzing the risks of the organization. Identifying scope for the assessment.

			Assigning tasks among members. Reviewing the document.
	Phạm Thanh Tân	Technical, Consultant, Document	Analyzing the risks of the organization. Documenting.
	Phạm Quang Linh	Technical, Consultant, Document	Analyzing the risks of the organization. Documenting.
	Trần Doãn Anh	Technical, Consultant, Document	Analyzing the risks of the organization. Documenting.
	Mai Duy Nam	Technical, Consultant, Document	Analyzing the risks of the organization. Documenting.
Risk Analysis	Phan Huyền Trâm	Team Leader, Consultant, Document	Proposing risk migration plan. Documenting. Reviewing.
	Phạm Thanh Tân	Technical, Consultant, Document	Proposing risk migration plan. Documenting.
	Phạm Quang Linh	Technical, Consultant, Document	Proposing risk migration plan. Documenting.
	Trần Doãn Anh	Technical, Consultant, Document	Proposing risk migration plan. Documenting.
	Mai Duy Nam	Technical, Consultant, Document	Proposing risk migration plan. Documenting.
Solution Deployment	Phan Huyền Trâm	Team Leader, Consultant, Document	Developing Solution. Reviewing. Documenting.
	Phạm Thanh Tân	Technical, Consultant, Document	Developing Solution. Documenting.
	Phạm Quang Linh	Technical,	Developing Solution.

		Consultant, Document	Documenting.
	Trần Doãn Anh	Technical, Consultant, Document	Developing Solution. Documenting.
	Mai Duy Nam	Technical, Consultant, Document	Developing Solution. Documenting.

Table 2.2.1. Roles and Responsibilities

2.2.3. Tools and Techniques

No	Tools and Techniques	Function
1	PNETLab	PNETLab (Packet Network Emulator Tool Lab) is a platform that allows you to download and share labs with the community.
2	Kubernetes	Orchestrates containerized applications to run on a cluster of hosts
3	Ubuntu	OS Platform to run all these services
4	k9s	A terminal-based UI to manage Kubernetes clusters that aims to simplify navigating, observing, and managing our applications in K8s
5	SSH	Connect to VMs in cluster of hosts
6	HA Proxy	HAProxy acts as a Load Balancer in a system, distributing requests to services, ensuring load balancing, connection control, high availability, and security.
7	Calico	Calico defines and manages network connectivity for containerized applications, ensuring that each container has a unique IP address and allowing for secure communication and access controls through network policies.
8	Metric Server	Metric Server in Kubernetes collects real-time CPU and memory metrics for pods and nodes, supporting Horizontal Pod Autoscaling (HPA) and enabling efficient resource management and scaling decisions within the cluster.
9	Container	In Kubernetes, containers provide lightweight, isolated environments for running applications, ensuring portability, scalability, and efficient resource utilization across clusters.

10	Apache Bench	It is a tool for benchmarking your Apache Hypertext Transfer Protocol (HTTP) server.
11	Keepalived	Used to maintain high availability by managing a virtual IP in a cluster, ensuring uninterrupted service by automatically redirecting network traffic to a backup node if the primary node fails.
12	Prometheus	An open-source monitoring system with a dimensional data model, flexible query language, efficient time series database and modern alerting approach.
13	Alertmanager	The Alertmanager handles alerts sent by client applications such as the Prometheus server. It takes care of deduplicating, grouping, and routing them to the correct receiver integration such as email, PagerDuty, or OpsGenie. It also takes care of silencing and inhibits alerts.

Table 2.2.2. Tools and Techniques

2.3. Project Management Plan

2.3.1. Tasks

2.3.1.1. Brainstorming

Description: Each member discussed the content and scope of the project.

Distribution: Each member shows their opinion and debates the best solution.

Resources needed: All knowledge related to network, SDN and its components in general, internet.

Dependencies and Constraints:

- Each member is assigned different tasks based on their ability.
- Understanding the project ideas and objectives.
- Communication

Risks: Conflicts between team members.

2.3.1.2. Researching

Description: All members researched the information about the SDN solution.

Distribution: Gathering information about SDN solution including concept, architecture.

Resources Needed: Documentation about the traditional networking and SDN, internet.

Dependencies and Constraints: Gathering relevant information carefully, share knowledge for other members of the team

2.3.1.3. Developing Solutions

Description: The team builds integrated system to experiment with SDN solutions.

Distribution: PNETLab, Kubernetes, Calico

Resources Needed: Determined timeline, research results, internet.

Dependencies and Constraints: Find and summarize information, share knowledge with other members of the team.

2.3.1.4. Evaluating

Description: The team evaluates whether the system is stable and efficient enough.

Distribution: Document the methods used in the project.

Resources Needed: Report form provided by FPT University.

Dependencies and Constraints: Reporting format.

2.3.2. Task Sheet: Assignments and Timetable

Task	Start date	End date
Project Initiating	05/09/2023	11/09/2023
Research on SDN	12/09/2023	22/09/2023
Research Kubernetes architecture	12/09/2023	22/09/2023
Reporting and proposing solutions	23/09/2023	02/10/2023
Write and submit report No.1	25/09/2023	27/09/2023
Write and submit report No.2	25/09/2023	27/09/2023
Design a Network system	03/10/2023	05/10/2023
Set up lab environment	06/10/2023	10/10/2023
Install Kubernetes cluster	11/10/2023	15/10/2023
Install K9s	11/10/2023	15/10/2023
Configure Kubernetes Cluster	16/10/2023	18/10/2023
Write configuration to YAML	18/10/2023	20/10/2023
Discuss the SDN demo scenario	19/10/2023	22/10/2023
Discuss about risk assessment and management	19/10/2023	22/10/2023
Identify risks of SDN solution	19/10/2023	22/10/2023
Write and submit report No.3	22/10/2023	23/10/2023
Write and submit report No.4	22/10/2023	23/10/2023
Develop support tool	21/10/2023	11/11/2023
Design test plan	07/11/2023	10/11/2023
Test the system	12/11/2023	15/11/2023
Write and submit report No.5	16/11/2023	21/11/2023

Write and submit report No.6	22/11/2023	23/11/2023
Complete demo	24/11/2023	30/11/2023
Complete fully report	26/11/2023	02/12/2023
Practice presentation	02/12/2023	12/12/2023

Table 2.3.1. Assignments and Timetable

2.3.3 All Meeting Minutes

Subject	SPM401
From Date	04/09/2023
Lecture	Mai Hoàng Đỉnh
Time	13h30 – 15h
Location	Room 029 – FPT HCM Campus
Attendees	Phan Huyền Trâm Phạm Thanh Tân Phạm Quang Linh Trần Doãn Anh Mai Duy Nam
Absent	

Table 2.3.2. All Meeting Minutes

CHAPTER 3: RISK ASSESSMENT

3.1. The Essential of Assessment

Risk is where vulnerabilities and threats intersect. At its core, risk refers to the possible impact of damage or loss of business assets and data. For that reason, Risk Assessment (RA) is one of the most important tasks that businesses need to focus on and implement to minimize its impact. Risk Assessment include several procedures:

- Identify potential hazards
- Consider the risks associated with each hazard
- Evaluate the risk and apply control measure
- Record significant findings
- Review and update Risk Assessments

3.2. Identity Sensitive Assets

3.2.1. Asset Classification

Assets are classified based on their sensitivity level, the possibility of exploitation, and estimated impact.

3.2.1.1. Information Asset Classification

Asset Categories	Classifications	Details	Likelihood of asset being exploited	Impact
Information	Non - Confidential	Corporation's registered office address, and the names and addresses of its directors	Most of this information is already public and cannot be used for any other purposes	No harm to the company and the information is validated before it is made public
	Confidential	Personal Information (Home address, phone, birthdate …)	This data is protected at an elevated level to prevent exploitation.	Cause great harm to the target being exploited
	Restricted	Financial account numbers of credit card numbers, personal	Although this information has the highest level of security, it continues to be the most	Information leaks of this type can result in identity theft, news coverage, and public exposure,

		medical and medical insurance information, passwords	frequently exploited.	as well as inflicting significant harm and financial expenses to the company.

Table 3.2.1. Information Asset Classification

3.2.1.2. Other Asset Classification

Asset Categories	Classifications	Details	Impact
Physical	Network	Switch, Router	Can cause unplanned operational downtime, product loss, customer churn, brand reputation damage, broken vendor or supplier relationships, and loss of investor confidence.
	Computing	Laptop, PC	
	Storage	Hard Disk, Clouds	
Software	Applications, tools	PNETLab, Pfsense, VM ESXI	Can lead to large fines from vendors for non-compliance, unnecessary spending on licenses, and cyber security vulnerabilities.
	OS	Linux, Windows, Switch OS	

Table 3.2.2. Other Asset Classification

3.2.2. SDN System Characterization

SDN has an open network architecture with separate control and forwarding planes, which enables centralized control and network programmability. SDN has three planes (Data, Control and Application) with two APIs used for communication between the planes.

By centralizing network state in the management level, SDN gives network managers the versatility to configure, take care of, secure, and also enhance network sources using compelling, automatic SDN programs. Therefore, Enterprises and vendors gain vendor-independent management of the whole network from a single logical perspective, which greatly simplifies network layout and functionality.

3.3. Risk Identification

3.3.1.Threat Identification

A threat is anything that could potentially cause harm to the system. In an SDN environment, these threats could arise from diverse sources.

Potential Threat	Describe Threat
Unauthorized Access	In SDN, control plane and data plane are decoupled. If attackers gain unauthorized access to the control plane, they could control network flows, change network policies, or even shut down the network
Denial of Service (DoS) Attacks	If a threat actor floods the network with excessive data packets, it can overload the SDN controller, resulting in a DoS attack
Insider Threats	Individuals with legitimate access to the SDN could misuse their permissions to cause harm.
Spoofing Attacks	Attackers can impersonate legitimate users or devices to manipulate the SDN controller.
Malware	Threat actors could introduce malicious software into the network, impacting network performance or stealing sensitive data.

Table 3.3.1. Threat Identification

3.3.2. Vulnerability Identification

Vulnerability refers to weaknesses that could be exploited to compromise the system. In SDN, these can include:

Potential Vulnerability	Describe Vulnerability
Inadequate Security Measures	Insufficient protection mechanisms for the control plane or improper authentication processes can lead to vulnerabilities.
Lack of Encryption	If data transmission between the control plane and data plane is not encrypted, it becomes vulnerable to interception.
Outdated Software	Older software versions may have known vulnerabilities that can be exploited.
Misconfigurations	Incorrectly configured elements of the SDN, such as firewalls or flow tables, can lead to vulnerabilities.

Poorly Implemented SDN Protocols	SDN protocols, such as OpenFlow, if not implemented correctly, can be manipulated by attackers.

Table 3.3.2. Vulnerability Identification

Risk identification is the first step in risk management. After identifying potential threats and vulnerabilities, they need to be assessed, and appropriate mitigation strategies need to be put in place to manage these risks.

3.4. Risk Analysis

3.4.1. Impact Assessment

Software-Defined Networking (SDN) has significantly impacted the networking field since its inception. Here are some key areas where SDN has made an impact.

Area	Describe	Improvement
Network Programmability	SDN has brought programmability to network infrastructure, allowing network administrators to control and manage their networks through software.	This enables dynamic network configuration, centralized management, and automation, which simplifies network operations and reduces manual configuration tasks.
Centralized Control	SDN separates the control plane from the data plane for a centralized view and control over the entire network.	This centralized control enables better network visibility, easier policy enforcement, and efficient resource allocation.
Network Virtualization	SDN facilitates network virtualization, which enables the creation of virtual networks that are logically isolated from each other, even if they share the same physical infrastructure.	This capability enhances network scalability, security, and multi-tenancy support.
Cost Efficiency	SDN can contribute to cost savings by reducing the need for expensive proprietary networking hardware and simplifying network management.	It helps optimize resource utilization, reduce downtime, and minimize operational costs.
Innovation and Experimentation	SDN's programmable nature encourages innovation by enabling the development and	It provides a platform for network researchers and developers to experiment with

	deployment of new network services and applications.	new protocols, algorithms, and network architectures.
Cloud and Data Center Networking	SDN has revolutionized networking in cloud environments and data centers.	It provides the flexibility and scalability needed to support dynamic workloads, orchestration, and resource management in highly virtualized environments.
IoT and Edge Computing	SDN can play a crucial role in managing and securing the networks in IoT and edge computing deployments.	It allows for centralized management, efficient traffic routing, and integration with other technologies such as network function virtualization (NFV) to support edge computing capabilities.

Table 3.4.1. Impact Assessment

Overall, SDN has had a transformative impact on networking, bringing increased flexibility, scalability, efficiency, and security to modern network infrastructures. It has paved the way for further advancements in network technology and continues to evolve as new use cases and challenges emerge.

3.4.2. Likelihood Assessment

The process of examining the likelihood and effects of risk occurrences is known as likelihood assessment. This evaluation's outcomes are then used to evaluate risk in order of most-to-least-critical relevance.

Level	**Probability Characteristic**	**Range of Probability**
Very Low	Rare event, may occur in exceptional cases	0 – 0.15
Low	The risk event may occur within 4 years	0.15 - 0.4
Medium	The risk event may occur within 2 years	0.4 - 0.6
High	The risk event can occur within 1 year	0.5 - 0.85
Very High	Risk event. Most likely to happen within 1 year, there is statistics of multiple occurrences of the event within 1 year in the past	0.85 - 1

Table 3.4.2. Likelihood Assessment

3.4.3. Risk Determination

3.4.3.1. Risk-Level Matrix

A risk-level matrix can be a helpful tool to assess and prioritize risks associated with Software-Defined Networking (SDN). Here is an example of a risk-level matrix considering both the likelihood and potential impact of risks:

Impact level	High	Medium	High	High
	Medium	Medium	Medium	High
	Low	Low	Medium	High
		Low	Medium	High
	Likelihood level			

Table 3.4.3. Risk-Level Matrix

3.4.3.2. Description of Risk Level

When assessing the risk level of Software-Defined Networking (SDN), it is important to consider the potential risks and their impact on the network infrastructure and operations. Here is a description of each risk level associated with SDN:

Risk Level	Description	Example
Low Risk	Risks categorized as low are those with a low likelihood and low potential impact on SDN. These risks may not pose an immediate threat to the network, but they still require monitoring and management.	Minor software bugs or occasional network congestion that can be addressed through routine monitoring and performance optimization.
Medium Risk	Risks categorized as medium have a moderate likelihood or potential impact on SDN. These risks may require more proactive monitoring and management to prevent their escalation. Mitigation measures should be implemented to reduce the likelihood or impact of these risks.	Intermittent controller failures, limited interoperability between different SDN components, or moderate security vulnerabilities that can be addressed through proactive measures and network resilience planning.
High Risk	Risks categorized as high are those with a high likelihood of or potential impact on SDN. These risks pose a significant threat to the network infrastructure and operations and require immediate attention and action. High-risk scenarios should be carefully managed and	Severe security breaches, widespread controller failures, or critical interoperability issues can disrupt network services and require immediate remediation measures.

	mitigated to minimize their impact or eliminate them altogether.	

Table 3.4.4. Description of Risk Level

It is important to note that risk levels may change over time as new vulnerabilities are discovered, technologies evolve, and threat landscapes evolve. Regular risk assessments and monitoring are essential to identify and address emerging risks in SDN deployments. By understanding the risk levels associated with SDN, organizations can prioritize their efforts and allocate resources effectively to manage and mitigate potential risks.

3.5. Control Identification and Assessment

3.5.1. Control Methods

3.5.1.1. Technical

Penetration Testing: Perform controlled simulated attacks to assess the effectiveness of the implemented controls and identify any exploitable vulnerabilities. Penetration testing helps uncover potential security gaps and provides insights into the overall security posture of the SDN environment.

Traffic Analysis: Monitor and analyze network traffic within the SDN environment to detect any anomalies or suspicious activities. Intrusion detection and prevention systems, as well as network traffic analysis tools, can be employed to identify potential security breaches or abnormal behavior.

Vulnerability Assessments: Conduct regular vulnerability assessments of the SDN infrastructure to identify potential security weaknesses. This involves using scanning tools and techniques to identify vulnerabilities in the SDN components, such as controllers, switches, and network applications.

Secure Code Review: Evaluate the security of the code used in SDN applications, controllers, and other components. Conducting secure code reviews helps identify potential vulnerabilities and coding errors that could be exploited by attackers.

3.5.1.2. Non-technical

Policy and Procedure Reviews: Review existing security policies, procedures, and guidelines to ensure they are comprehensive, up-to-date, and aligned with SDN security requirements. Assess the effectiveness of policies in enforcing access controls, managing user privileges, and protecting sensitive data.

Risk Assessments: Perform regular risk assessments to identify potential threats, vulnerabilities, and associated risks to the SDN environment.

Security Awareness and Training: Promote security awareness and provide training to SDN users and administrators. This helps ensure that individuals are knowledgeable about security best practices, understand the risks involved, and follow proper procedures to maintain the security of the SDN infrastructure.

Incident Response Planning: Develop and test an incident response plan specific to SDN to ensure a swift and coordinated response in the event of a security incident.

3.5.2. Control Types

- **Preventative controls** in Software-Defined Networking (SDN) are measures implemented to proactively mitigate risks and prevent security incidents from occurring.
- **Detective controls** in Software-Defined Networking (SDN) are implemented to identify and detect security incidents, anomalies, or unauthorized activities within the SDN environment.
- **Corrective controls** in Software-Defined Networking (SDN) are measures implemented to address and mitigate the impact of security incidents, vulnerabilities, or unauthorized activities within the SDN environment.
- **Compensating controls** in Software-Defined Networking (SDN) are measures implemented to address security requirements when the primary controls are not feasible or cannot be implemented

Preventative	Detective	Corrective	Compensatory
Access Control	Network Traffic Monitoring:	Incident Response	Network Segmentation
Network Segmentation	Intrusion Detection Systems (IDS)	System Recovery	Virtual Private Networks (VPNs)
Threat Intelligence and Detection	Centralized Logging and Monitoring	User Access Management	Redundancy and High Availability
Secure Configuration Management	Security Auditing	Security Awareness and Training	Third-Party Security Services
Security Monitoring and Logging	Threat Intelligence Integration		Cloud-based Security Services
	Incident Response and Forensics		

Table 3.5.1. Control Types

3.5.3. Risk Monitoring and Controlling

While the project team and team leader are carefully categorizing all the risks, we are aware that there is always a set of inherent risks associated with some aspect of the project. The project team leader will choose the best course of action once we have accurately categorized and structured all the potential risks. Risk monitoring and controlling refers to the complete process of detecting these risks and devising strategies to deal with them.

To determine whether risk mitigation strategies have been implemented as anticipated, their effectiveness and whether they need to be updated, this procedure entails tracking and describing identified risks. It also involves determining whether new or lingering threats that were not previously recognized have materialized.

CHAPTER 4: RISK MANAGEMENT PLAN

4.1. Objectives of RMP

4.1.1. Lists of Threats / Vulnerabilities

The list of vulnerabilities brought on by threats is shown in the table below so that threat management can be carefully planned.

No.	Threats	Vulnerability
1	Disaster	- Flood, Fire…etc. - Natural disasters (earthquakes, hurricanes, etc.).
2	Equipment malfunction	- Hardware Error. - Aging (Too old). - Storage equipment losing memory
3	System problem	- Damage caused by a third party. - Administrators change system configuration causing errors or down services. - Physical damage caused by individuals trying to destroy the data center or workstations.
4	Unauthorized access	- Weak authorized access control. - Un permission user can read or change information. - Password policy weak. - User forgets the password.
5	Malicious program	- Using the cracking program. - User violates the Acceptable Uses Policy. - OS, Software out of date or licenses.
6	Supplier problem	- Electronic problem. - Network ISP problem.

Table 4.1.1. Lists of Threats/Vulnerabilities

4.1.2. Costs Associated with Risks

We define the cost in 4 levels below:

Level 0 (Non):

- The risk has been resolved or does not affect the organization's assets, mission, or interests.
- The cost to resolve the risk is below 10% of the acceptable line.

Level 1 (Low):

- The risk has mostly been resolved but may affect some services that are not part of the organization's main mission.
- The risk does not interrupt the working of the organization.
- The cost to resolve the risk is between 10% and 40% of the acceptable line.

Level 2 (Medium):

- The risk has not been resolved and may affect some services that are part of the organization's main mission.
- The risk may interrupt the working of the organization.
- The cost to resolve the risk is between 50% and 80% of the acceptable line.

Level 3 (High):

- The risk has not been resolved and affects most services in the organization's main mission.
- The risk significantly interrupts the working of the organization.
- The cost to resolve the risk exceeds 90% of the acceptable line or is beyond the acceptable line.
- These levels provide a framework for categorizing risks based on their impact and the associated costs to resolve them. Organizations can use these levels to prioritize and allocate resources effectively for risk mitigation and management.

Every risk entails a cost. Equipment, processes, and survey results may be impacted by the damage. It establishes the risks and displays the expenses incurred. One of the things to undertake while conducting the inquiry is the table below.

Risk	**Cost Level**
User forgets the password	Level 0 (Non)
Aging equipment	Level 1 (Low)
Hardware error	Level 1 (Low)
Password policy weak	Level 1 (Low)
OS, Software out of date/licenses	Level 1 (Low)
Weak authorize access control	Level 2 (Medium)
Un permissioned user access	Level 2 (Medium)
Damage caused by third party	Level 2 (Medium)
Electronic problem (Supplier)	Level 2 (Medium)
Network ISP problem (Supplier)	Level 2 (Medium)
Physical damage	Level 3 (High)
Flood, fire, natural disasters	Level 3 (High)
Malicious programs	Level 3 (High)
Unauthorized access	Level 3 (High)

Table 4.1.2. The Cost Associated with Risks

4.1.3. Recommendations to Reduce Risks

Following threat management in the plan, we have suggested a way to lessen or eliminate the threat. The list of threats suggested to participants for prevention, and associated costs are shown in the table below.

Risk	Recommendations	Cost Level
Equipment malfunction	- Implement proactive maintenance schedules. - Regularly monitor hardware performance and health. - Maintain an inventory of spare parts and replacement equipment.	Level 1 (Low)
System problem	- Implement strong access controls. - Regularly review and update system configurations. - Implement intrusion detection and prevention systems.	Level 2 (Medium)
Unauthorized access	- Enforce a strong password policy. - Implement multi-factor authentication. - Regularly review and update user access privileges.	Level 2 (Medium)
Supplier problem	- Establish strong service level agreements (SLAs) with suppliers. - Regularly monitor and assess supplier performance and security. - Have contingency plans in place to mitigate disruptions caused by supplier issues.	Level 2 (Medium)
Disaster	- Implement a robust disaster recovery plan including data backups, off-site storage, and redundant systems. - Conduct regular drills and exercises to test the effectiveness of the plan. - Protect critical infrastructure and equipment against potential disasters.	Level 3 (High)
Malicious program	- Keep operating systems and software up to date with security patches. - Implement robust antivirus and anti-malware solutions. - Educate users about safe computing practices.	Level 3 (High)

Table 4.1.3. List of Recommendations to Reduce the Risks

4.1.4. Cost-Benefit Analysis (CBA)

The CBA is dependent upon the implementation's timing and specifics. Comparing various times shows the difference in cost and hence makes a reasonable choice.

Risk	Recommendations	Cost Level	Potential Benefits
Equipment malfunction	- Implement proactive maintenance schedules. - Regularly monitor hardware performance and health. - Maintain an inventory of spare parts and replacement equipment.	Level 1 (Low)	- Reduced downtime and disruptions due to equipment failures. - Improved performance and reliability of equipment. - Extended lifespan of equipment resulting in cost savings from delayed replacements or repairs.
System problem	- Implement strong access controls. - Regularly review and update system configurations. - Implement intrusion detection and prevention systems.	Level 2 (Medium)	- Enhanced security and reduced risk of unauthorized access or breaches. - Minimized disruptions and downtime caused by system issues. - Protection of sensitive data and information. - Compliance with regulatory requirements.
Unauthorized access	- Enforce a strong password policy. - Implement multi-factor authentication. - Regularly review and update user access privileges.	Level 2 (Medium)	- Increased security and reduced risk of unauthorized access. - Protection of confidential data and information. - Compliance with data protection regulations.
Supplier problem	- Establish strong service level agreements (SLAs) with suppliers.	Level 2 (Medium)	- Improved reliability and availability of supplier services.

	- Regularly monitor and assess supplier performance and security. - Have contingency plans in place to mitigate disruptions caused by supplier issues.		- Minimized disruptions and delays caused by supplier problems. - Mitigated financial losses and reputational damage.
Disaster	- Implement a robust disaster recovery plan including data backups, off-site storage, and redundant systems. - Conduct regular drills and exercises to test the effectiveness of the plan. - Protect critical infrastructure and equipment against potential disasters.	Level 3 (High)	- Minimized downtime and data loss in the event of a disaster. - Fast recovery and restoration of services. - Protection of critical business operations and customer trust.
Malicious program	- Keep operating systems and software up to date with security patches. - Implement robust antivirus and anti-malware solutions. - Educate users about safe computing practices.	Level 3 (High)	- Enhanced security and reduced risk of malware infections. - Protection of sensitive data and information. - Prevention of unauthorized access and data breaches. - Compliance with data protection and privacy regulations.

Table 4.1.4. Cost-Benefit Analysis (CBA)

4.2. Assigning Responsibilities

Full name	Role	Responsibilities
Phan Huyền Trâm	Leader	Project Planning, Offering Solutions, Tracking Progress Management, Assigning Tasks
Phạm Thanh Tân	Technical Developer	Developing, Testing and Reporting
Phạm Quang Linh	Technical Developer	Developing, Testing and Reporting
Trần Doãn Anh	Technical Developer	Developing, Testing and Reporting
Mai Duy Nam	Document Writer	Document writer, Quality Assurance

Table 4.2.1. Assigning Responsibilities

4.3. Describing Procedures and Schedules for Accomplishment

Phase	Sub-Phase	Task
Planning and Design	Requirement Analysis	Understand network requirements, potential solutions, and create a roadmap
	Network Design	Develop the network design suitable for SDN
	Vendor Selection	Choose SDN vendors based on requirements and network design
Infrastructure Setup	Hardware Setup	Acquire and set up the necessary hardware
	Software Installation	Install the SDN controller and relevant software applications
	Testing	Validate initial setup functionality and compatibility
Implementation	Network Programming	Define network operation using software
	Network Integration	Integrate the programmed network with existing infrastructure
	Validation	Test new network setup for performance, security, reliability

Deployment	Live Deployment	Begin deploying the SDN solution on the live network environment
	Monitoring	Regularly monitor the network for any issues or anomalies
	Optimization	Continually optimize the network based on the monitoring reports
Maintenance and Support	Troubleshooting	Address any issues or problems in the network
	Updates and Upgrades	Keep the network updated with the latest software updates and patches
	Training	Train IT staff to manage the SDN network

Table 4.3.1. Describing Procedures and Schedules for Accomplishment

4.4. Reporting Requirements

4.4.1. Present Recommendations

The requirements in the previous part can all be used to deal with project risks. But for the time currently, here are some suggestions for effective security:

- Update software and operating systems with security patches.
- Implement a robust disaster recovery plan including data backups, off-site storage, and redundant systems.
- Install robust access controls
- Continually check the health and performance of hardware.

4.4.2. Document Management Response to Recommendations

Assessing the benefits and costs leads to finishing this work. The team decides whether to approve and implement. The group arrived at the following conclusions and suggestions.

Recommendation	Action By	Management Response
Update software and operating systems with security patches.	Technical Team	Agreed
Implement a robust disaster recovery plan including data backups, off-site storage, and redundant systems.	Technical Team	Agreed
Install robust access controls	Technical Team	Agreed

Continually check the health and performance of hardware.	Technical Team	Agreed

Table 4.4.1. Management Response to Recommendations

4.4.3. Document and Track Implementation of Accepted Recommendations

No.	Task	Accepted	Rejected
1	Update software and operating systems with security patches.	X	
2	Implement a robust disaster recovery plan including data backups, off-site storage, and redundant systems.	X	
3	Install robust access controls	X	
4	Continually check the health and performance of hardware.	X	

Table 4.4.2. Track Implementation of Accepted Recommendations

4.5. Plan of Action and Milestones

Action Plans and Milestones are a way of breaking down goals into specific steps that can be followed and tracked easily.

Action	Author	Start date	End date
Brainstorming		5/9/2023	11/9/2023
Initiating Project	Group	5/9/2023	8/9/2023
Identifying goals, planning and delegating tasks	Phan Huyền Trâm, Phạm Thanh Tân	9/9/2023	11/9/2023
Researching		12/9/2023	2/10/2023
Research on SDN	Phan Huyền Trâm	12/9/2023	22/9/2023
Research Kubernetes architecture	Phạm Quang Linh	12/9/2023	22/9/2023
Research Kubernetes features	Phạm Thanh Tân	12/9/2023	22/9/2023
Research K9s	Trần Doãn Anh	12/9/2023	22/9/2023
Research Calico CNI	Mai Duy Nam	12/9/2023	22/9/2023
Reporting and proposing solutions	Group	23/9/2023	2/10/2023
Developing Solution		3/10/2023	15/11/2023
Design a Network system	Group	3/10/2023	5/10/2023
Build the virtual environment	Group	6/10/2023	10/10/2023

Install Kubernetes Cluster	Group	11/10/2023	15/10/2023
Install K9s	Group	11/10/2023	15/10/2023
Install Calico	Group	11/10/2023	15/10/2023
Install Kubernetes feature	Group	11/10/2023	15/10/2023
Configure Kubernetes Cluster	Group	16/10/2023	18/10/2023
Write configuration YAML file	Group	18/10/2023	20/10/2023
Develop support tool	Group	21/10/2023	11/11/2023
Test the system	Group	12/11/2023	15/11/2023
Evaluate		16/11/2023	
Write complete the report	Group	16/11/2023	2/12/2023
Present the project	Group	27/11/2023	12/12/2023

Table 4.5.1. Plan of Action and Milestones

4.6.Charting the Progress of an RMP

4.6.1. Milestone Plan Chart

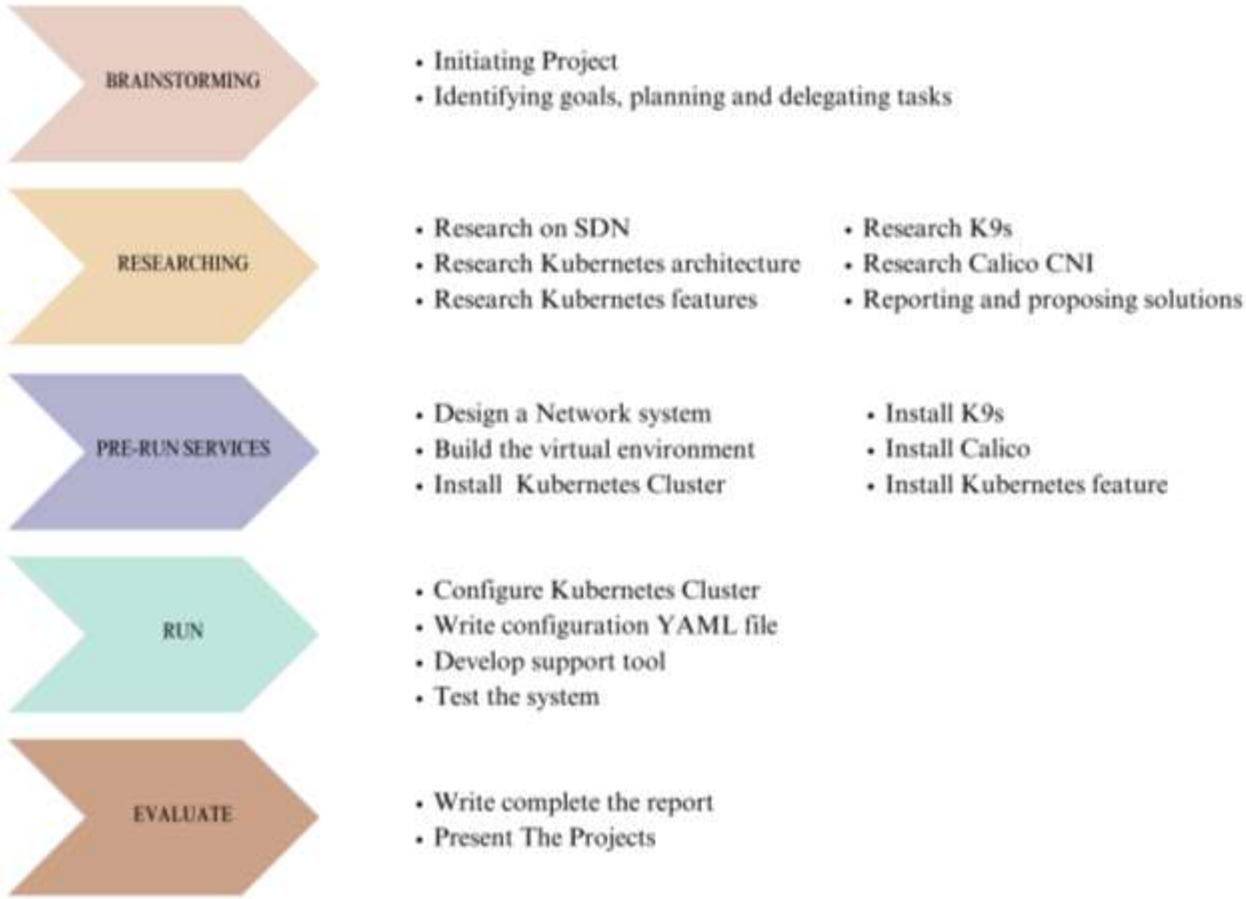


Figure 4.6.1. Milestone Plan chart

4.6.2. Gantt Chart

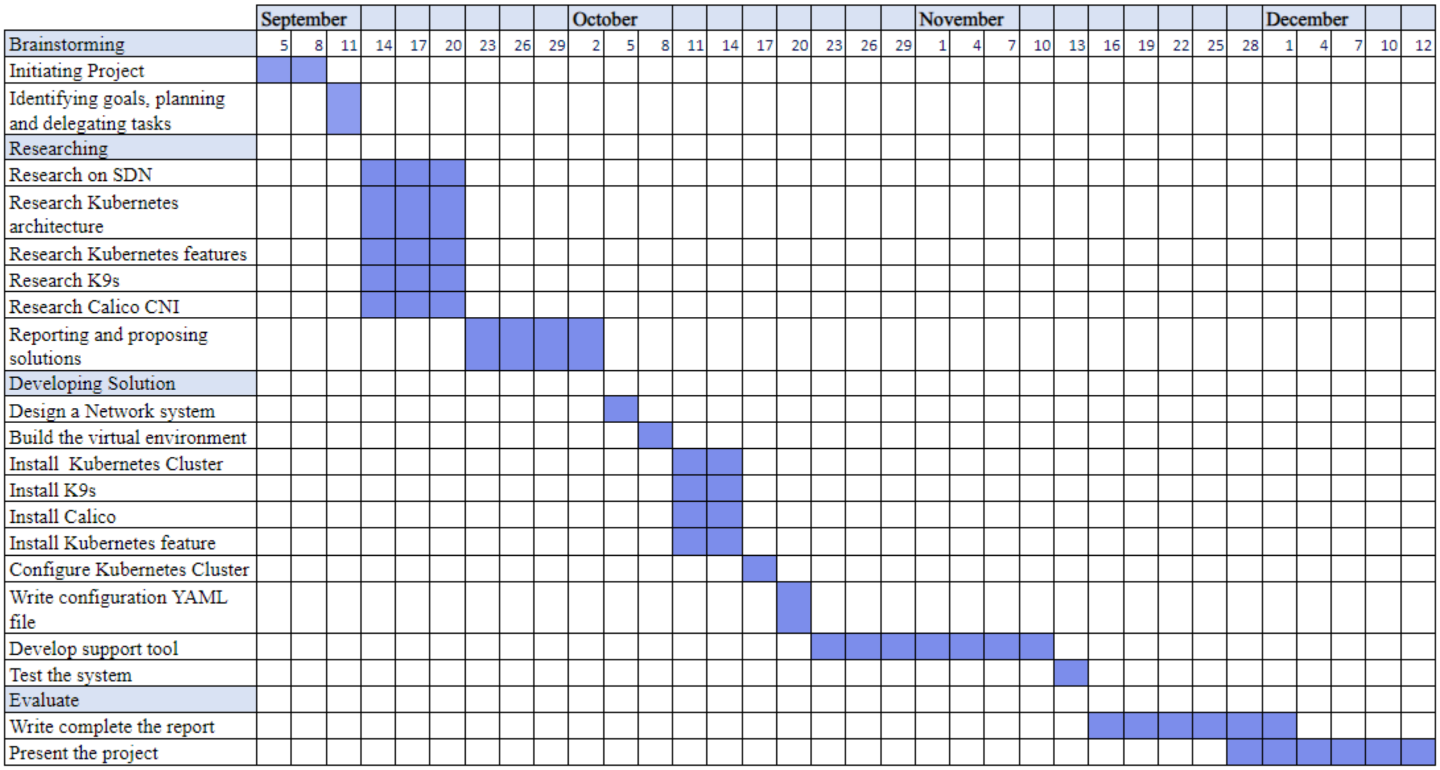


Figure 4.6.2. Gantt chart

4.7.Tools and Practices

No	Tools and Techniques	Function
1	PNETLab	PNETLab (Packet Network Emulator Tool Lab) is a platform that allows you to download and share labs with the community.
2	Kubernetes	Orchestrates containerized applications to run on a cluster of hosts
3	Ubuntu	OS Platform to run all these services
4	k9s	A terminal-based UI to manage Kubernetes clusters that aims to simplify navigating, observing, and managing our applications in K8s
5	SSH	Connect to VMs in cluster of hosts
6	HA Proxy	HAProxy acts as a Load Balancer in a system, distributing requests to services, ensuring load balancing, connection control, high availability, and security.
7	Calico	Calico defines and manages network connectivity for containerized applications, ensuring that each container

		has a unique IP address and allowing for secure communication and access controls through network policies.
8	Metric Server	Metric Server in Kubernetes collects real-time CPU and memory metrics for pods and nodes, supporting Horizontal Pod Autoscaling (HPA) and enabling efficient resource management and scaling decisions within the cluster.
9	Container	In Kubernetes, containers provide lightweight, isolated environments for running applications, ensuring portability, scalability, and efficient resource utilization across clusters.
10	Apache Bench	Is a tool for benchmarking your Apache Hypertext Transfer Protocol (HTTP) server
11	Keepalived	Used to maintain high availability by managing a virtual IP in a cluster, ensuring uninterrupted service by automatically redirecting network traffic to a backup node if the primary node fails
12	Prometheus	An open-source monitoring system with a dimensional data model, flexible query language, efficient time series database and modern alerting approach.
13	Alertmanager	The Alertmanager handles alerts sent by client applications such as the Prometheus server. It takes care of deduplicating, grouping, and routing them to the correct receiver integration such as email, PagerDuty, or OpsGenie. It also takes care of silencing and inhibits alerts.

Table 4.7.1. Tools and Practices

CHAPTER 5: DEVELOPMENT AND IMPLEMENTATION PLAN

5.1. Risk Response Planning

5.1.1. Major Risk Treatment

Risk	Risk Treatment
Failures or Disaster	Develop a robust business continuity and disaster recovery plan
Hardware Error	Ensure that the SDN solution supports the hardware devices deployed in the enterprise network or evaluate the need for hardware upgrades or replacements.
Unauthorized access	Define and enforce granular access controls to restrict unauthorized access to critical components and resources
System change-making error	Multi-vendor strategy: Design an SDN architecture that supports multi-vendor environments
Malicious actors	- Encrypt communication channels between SDN controllers, switches, and other network devices. - Regularly update and patch.
Performance and Scalability	- Conduct a detailed analysis of network traffic patterns, bandwidth requirements, and performance expectations - Implement QoS mechanisms

Table 5.1.1. Major Risk Treatment

5.1.2. Risk Mitigation Plan (RPM)

Job	Time	Cost
Design a Network system	3 days	Manpower
Build the virtual environment	4 days	Manpower
Install Kubernetes Cluster	5 days	Manpower
Install K9s	5 days	Manpower
Install Calico	5 days	Manpower
Install Kubernetes feature	5 days	Manpower
Configure Kubernetes Cluster	3 days	Manpower
Write configuration YAML file	3 days	Manpower
Develop support tool	3 weeks	Manpower

Table 5.1.2. Risk Mitigation Plan

5.2. Theoretical Basis

5.2.1. SDN

5.2.1.1. Introduction

In SDN, the software is decoupled from the hardware. SDN moves the control plane that determines where to send traffic to software and leaves the data plane that actually forwards the traffic in the hardware. This allows network administrators who use software-defined networking to program and control the entire network via a single pane of glass instead of on a device-by-device basis.

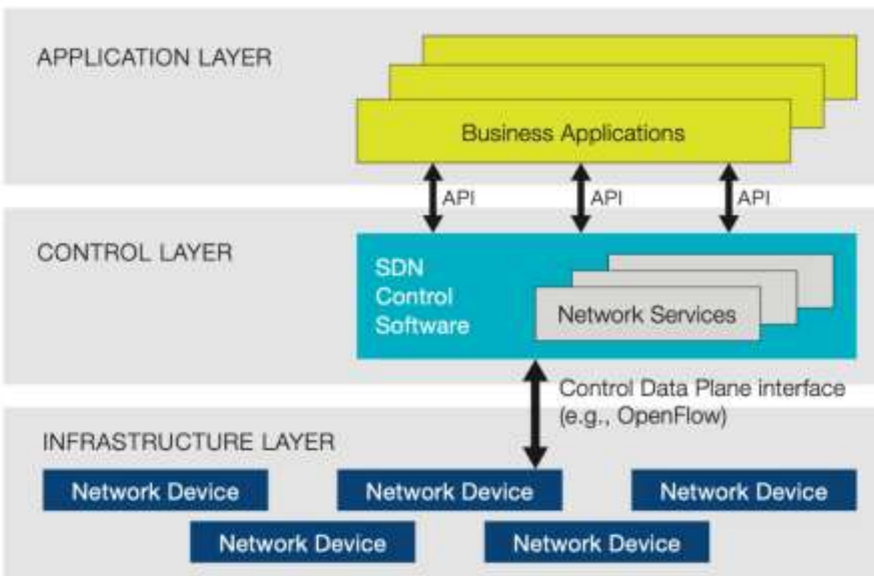


Figure 5.2.1. Software Defined Networking (SDN) Architecture

There are three parts to a typical SDN architecture, which may be in different physical locations:

- **Applications**, which communicate resource requests or information about the network as a whole
- **Controllers**, which use the information from applications to decide how to route a data packet
- **Networking devices**, which receive information from the controller about where to move the data

Physical or virtual networking devices actually move the data through the network. In some cases, virtual switches, which may be embedded in either the software or the hardware, take over the responsibilities of physical switches and consolidate their functions into a single, intelligent switch. The switch checks the integrity of both the data packets and their virtual machine destinations and moves the packets along.

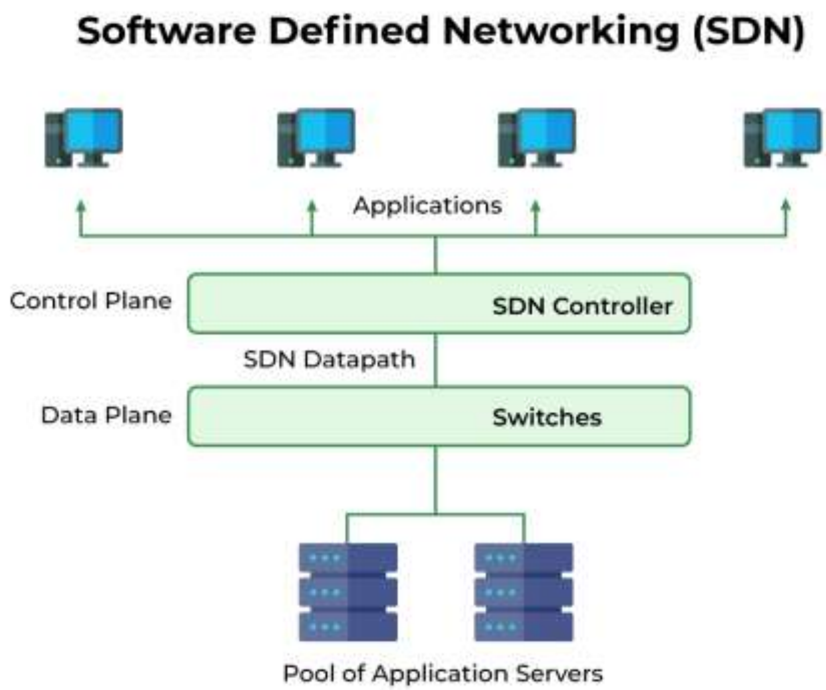


Figure 5.2.2. Software Defined Networking

5.2.1.2. SDN Data Plane

The data plane, also known as the forwarding plane, is one of the key components in Software-Defined Networking (SDN).

- It is responsible for the forwarding and processing of network traffic.
- It consists of network devices, such as switches or routers that forward packets based on predetermined rules.
- The data plane performs tasks like packet forwarding, filtering, traffic classification, and Quality of Service (QoS) enforcement.

5.2.1.3. SDN Control Plane

The control plane is responsible for managing and controlling the behavior of the network devices in an SDN architecture. It includes the logic, protocols, and formulas that govern network-wide management and decision-making.

- It decouples the control logic from the network devices and centralizes it in a software-based controller.
- The control plane establishes and maintains network state information, determines forwarding behavior, and calculates optimal paths for network traffic.
- It communicates with the data plane using protocols like OpenFlow to program the forwarding rules.

5.2.1.4. SDN Management Plane

The management plane is responsible for the overall management and operation of the SDN infrastructure. It handles tasks related to configuration, monitoring, provisioning, performance management, and security management of the SDN environment.

- The management plane facilitates the configuration of network devices, controllers, and applications in the SDN environment.
- It provides tools and interfaces for network administrators to define and manage network policies, rules, monitoring, analysis, optimization, and parameters.
- It includes mechanisms for access control, authentication, authorization, and encryption of network communications.

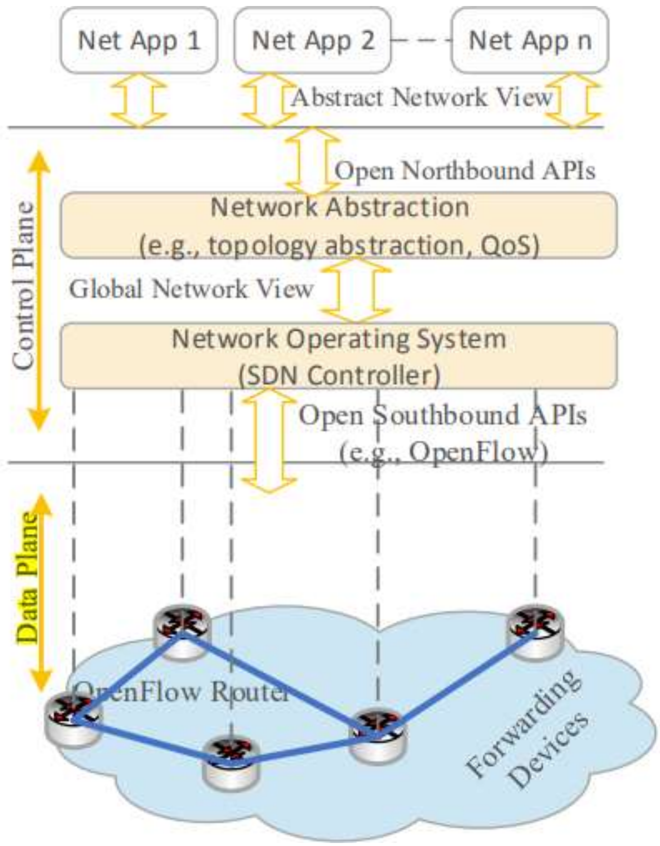


Figure 5.2.3. Simplified view of an SDN architecture

Figure 5.2.3 depicts the SDN architecture illustrating the separation between the applications, control plane and data plane:

- Applications use the northbound API supported by the control plane to enforce their policies in the data plane without directly interacting with the data plane.

- The interface between the control and data plane is supported by southbound APIs, where a SDN controller will use these APIs to communicate with the network equipment in the data plane.

5.2.2. Kubernetes

5.2.2.1. Introduction

Kubernetes, or "k8s" for short, is an open-source container orchestration system. It manages containerized applications across clusters of nodes for deploying, monitoring, and scaling containers. Containers run on a shared operating system (OS) across hosts but are isolated from each other unless users choose to connect them.

In Kubernetes, there are several concepts such as "deployments" and "services" that allow user to declare the desired state of an application. Kubernetes resources can be created directly on the command line but are usually specified using Yet Another Markup Language (YALM).

5.2.2.2. Kubernetes architecture

Kubernetes uses a client-server architecture. Groups of machines are networked together in multiple data centers. Each of those machines hosts one or more Docker containers. Those worker machines are called nodes. Other machines run special coordinating software that schedules containers on the nodes. These machines are called masters. A Kubernetes cluster is a set of physical or virtual machines and other infrastructure resources that are used to run applications. In other words, clusters are collections of masters and nodes.

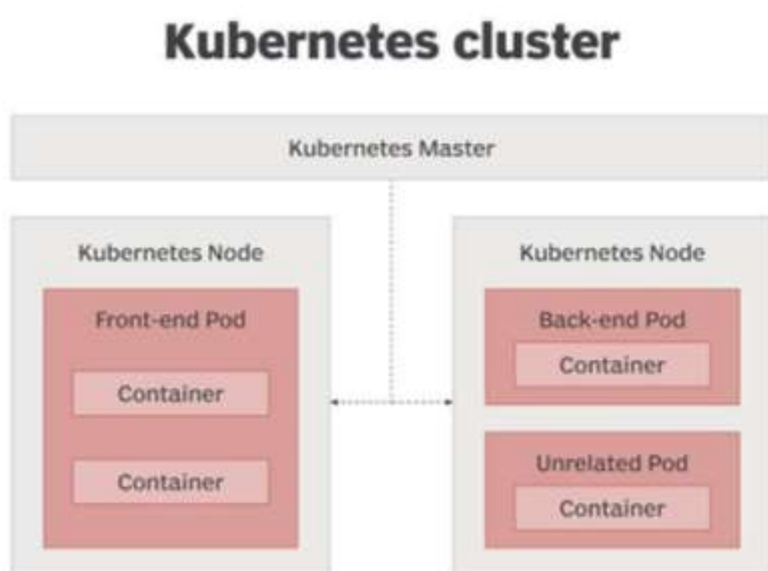


Figure 5.2.4. Kubernetes Cluster

Master

- ***API Server:*** nearly all the components on the master and nodes accomplish their respective tasks by making API calls. These are handled by the API Server running on the master.
- ***Etcd:*** Etcd is a service whose job is to keep and replicate the current configuration and run state of the cluster by distributed key-value store
- ***Scheduler and Controller Manager:*** These processes schedule containers (pods) onto target nodes. They also make sure that the correct numbers of these things are always running.

Worker node

- ***Kubelet:*** a special background process that respond to commands from the master to create, destroy, and monitor the containers on that host
- ***Proxy:*** a simple network proxy that manages networking rules so connections to service IP addresses are correctly routed to pods
- The master node is also usually configured as a worker node within the cluster. Therefore, the master node also runs the standard node services such as: kubetlet, kube proxy, container run time.

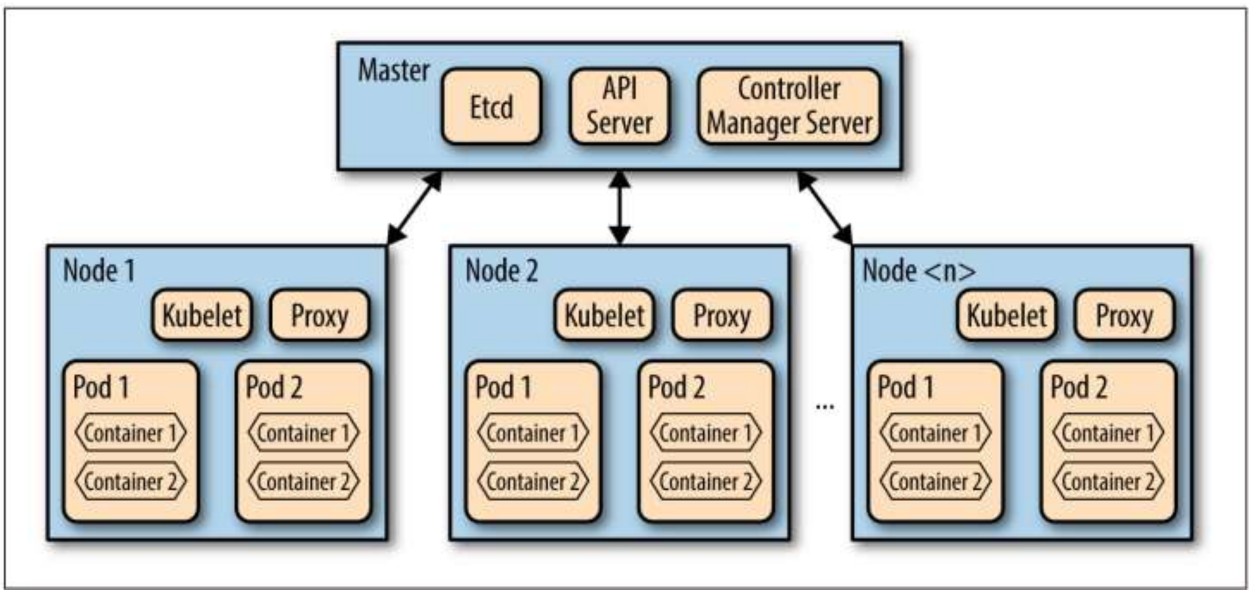


Figure 5.2.5. Details of Kubernetes components

Pods

A pod is purely a Kubernetes concept, which is a collection of containers and volumes that are scheduled together. The kubelet configures the container engine to place multiple containers in the same network namespace so those containers share an IP address and communicate with one another via localhost.

5.2.2.3. Other concepts

Kube-DNS

All services and pods (when enabled) on Kubernetes are assigned a domain name that is resolved by the DNS. Kube-DNS is a pod and service on Kubernetes that handles and resolves DNS of all services in the cluster.

Volumes

A volume is a virtual file system that containers can see and use. In Kubernetes, volume is defined at the pod level because:

- ***Duration:*** The data can survive the death and rebirth of any container in the pod addresses data loss when a container dies.
- ***Communication:*** Since volumes exist at the pod level, any container in the pod can see and use them. That makes moving temporary data between containers super easy.

Namespace

Namespaces are the mechanism by which we can separate groups of resources (such as Pods, Deployments or Services...) within the same cluster. The names of these resources are unique within that Namespace but may be used in other Namespaces.

ReplicaSet

One of the control loops available in Kubernetes that ensures that the desired number of pods are running.

5.2.3. Container Networking Interface (CNI)

5.2.3.1. Introduction

The Container Network Interface (CNI) is a plug-in which is used for managing network connectivity in Kubernetes clusters. As an essential component of the Kubernetes system, CNI enables seamless communication and connectivity between containers and external networks.

A CNI plug-in is not a K8s component, and it does not depend on K8s. In a K8s cluster, a CNI simply acts as a middleware between pods and the container engine being used. It serves as a bridge between the container runtime and the network plugins, allowing for the dynamic configuration of networking for Kubernetes pods. CNI plugins handle tasks such as assigning IP addresses, creating network interfaces, and setting up network routes for containers.

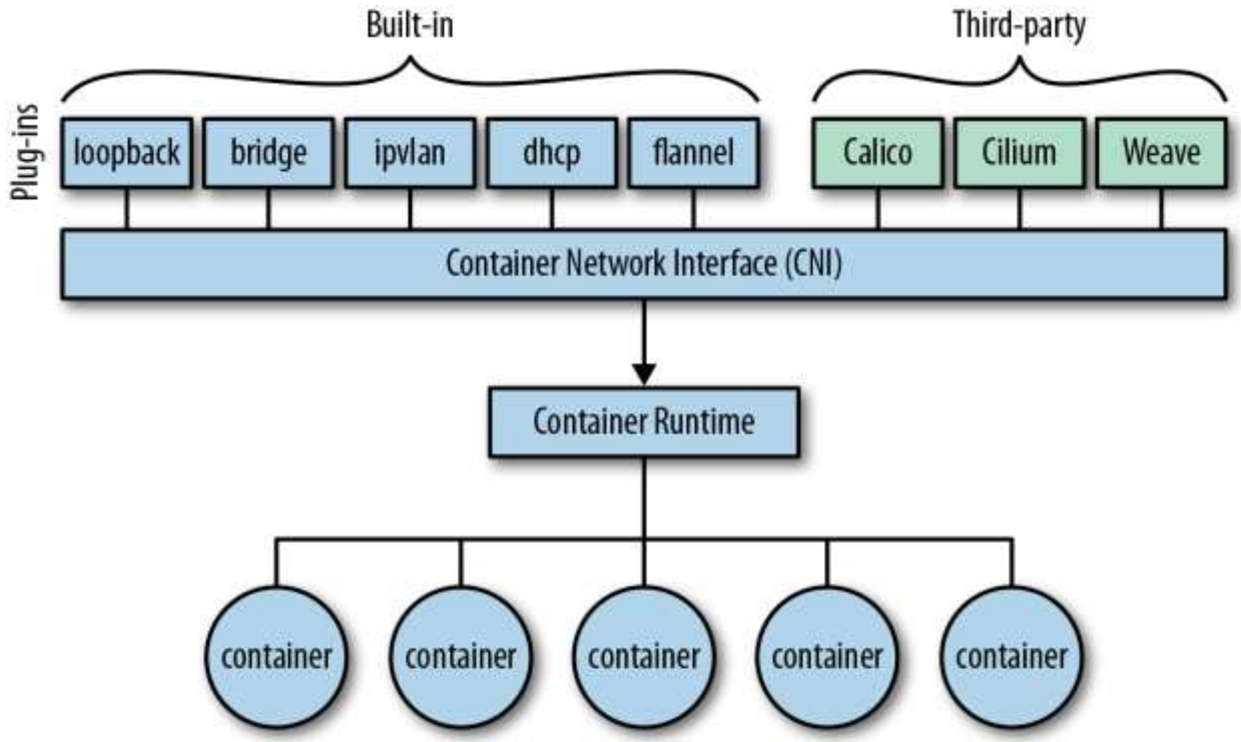


Figure 5.2.6. CNI components

5.2.3.2. CNI operation

When a pod is created in Kubernetes, the container runtime calls the CNI plugins and requests that the pod be added or removed from the cluster network. CNI works with other plug-ins to establish network connectivity for the pod. The plugins can be written in various programming languages and communicate with the container runtime via standard input and output. They leverage the Linux networking stack to configure networking for containers.

Figure 5.2.7. CNI operation

5.2.4. Kubernetes Networking

5.2.4.1. Introduction

Kubernetes is a container orchestration platform. These containers need to be able to communicate with each other as well as with other parts of the infrastructure, such as external services and databases. Therefore, the Kubernetes networking is an integral part of management and makes communication easier and more secure.

The difference Kubernetes components use different networking methods to communicate. They include container-to-container communication, Pod-to-Pod communication, Pod-to-service communication, and External-to-service communication.

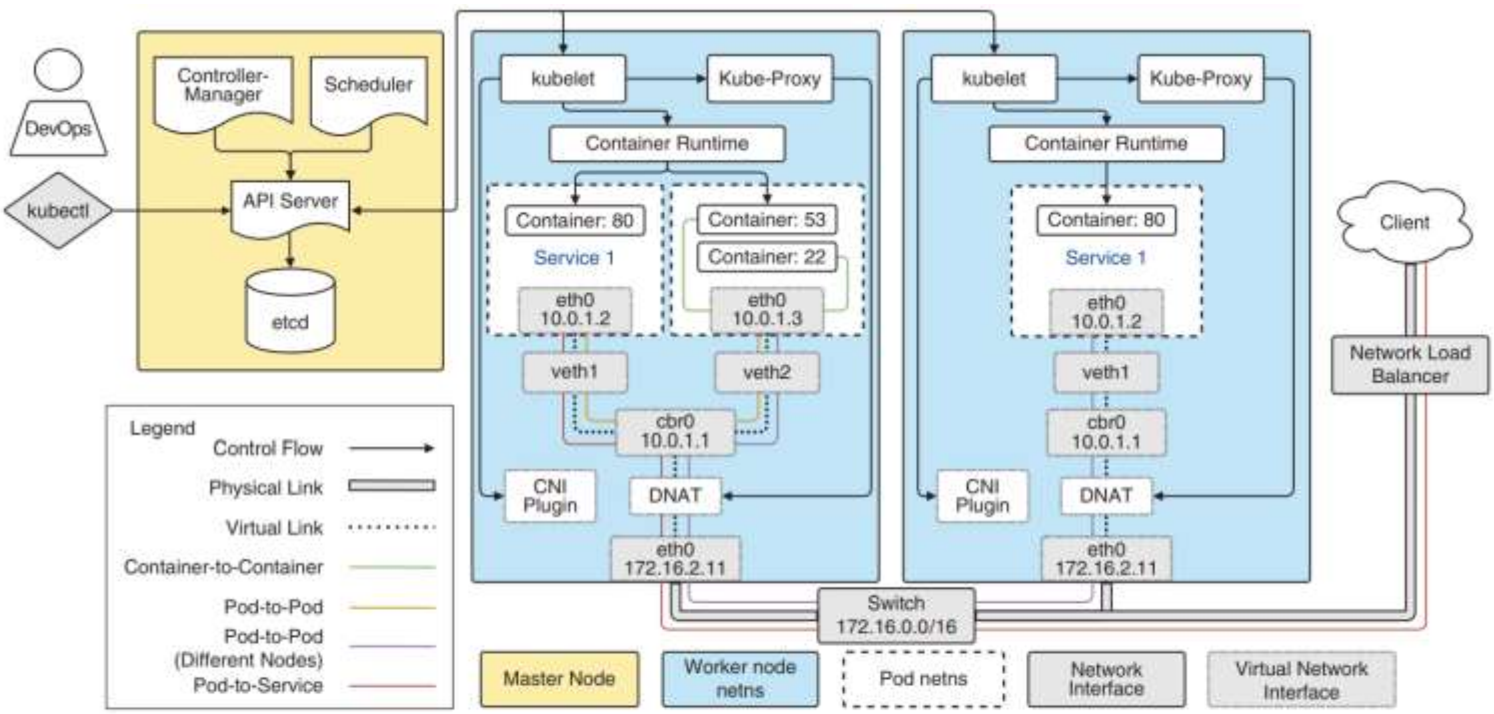


Figure 5.2.8. Kubernetes Networking methods

5.2.4.2. Container-to-container Networking

The simplest scenario consists of communication between containers within the same pod. Containers within the same pod share the same network namespace and communicate over localhost. It represented by the green line in the Figure 5.2.4.1

5.2.4.3. Pod-to-Pod Networking

Pod-to-Pod communication is the foundation of Kubernetes. There are two cases in these methods: two pods communicate in the same worker node (yellow line in Figure 5.2.4.1) and two pods communicate in different worker nodes (purple line in Figure 5.2.4.1). All pods in a Kubernetes cluster reside in a single, flat, shared network-address space that means all pods within a Kubernetes cluster are assigned unique IP addresses and can access every other pod at the other pod's IP address.

Communication between pods in the same node

Kubernetes creates a virtual Ethernet adapter (veth) for each pod, and it is connected to the network adaptor of the node.

In a Kubernetes node, there is a network bridge called cbr0, which facilitates the communication between pods in a node. All the pods are a part of this network bridge. When a network request is made, the bridge checks for the correct destination (pod) and directs the traffic.

Communication between pods in the different nodes

When the requested IP address cannot be found within the pod, the network bridge directs the traffic to the default gateway. This would then look for the IP within the cluster level.

Kubernetes keeps a record of all the IP ranges associated with each node. Then an IP address is assigned to the pods within the nodes from the range assigned to the node.

5.2.4.4. Pod-to-service Networking

A service is an abstraction that routes traffic to a set of pods. In other words, a service will map a single IP address to a set of pods called ClusterIP service. All pods used by an application share a common label that K8s uses for grouping the pods.

When a ClusterIP Service is created, it gets a static IP which never changes during the lifetime of the Service. Instead of connecting directly to Pods, clients connect to the ClusterIP Service through its IP address. The ClusterIP Service then makes sure that one of the Pods receives the connection, regardless of where the Pod is running and what its IP address is.

5.2.4.5. External-to-service Networking

There are several ways to expose services to external clients in Kubernetes:

NodePort

This service assigns the service to a static port on every node in the cluster. It can be accessed from outside the cluster using the node's IP address and the statically assigned port number. The traffic will route like this: External client → Worker Node IP → NodePort → ClusterIP → Service Pod.

Load Balancer

K8s exposes the service through a cloud-provider's load balancer. Requests arriving at the cloud-provider's load balancer are subsequently routed to a NodePort service, which in turn routes it to a ClusterIP service. The traffic goes through a path like this: External client → Loadbalancer → Worker Node IP → NodePort → ClusterIP Service → Pod.

5.3.Project Implementation

5.3.1. Preparation

Equipment/Tool	Version
Ubuntu	22.04
OpenVPN	1.6.0.02712
Pfsense	2.6.0
Pnetlab	4.2.10
window	10
HTML console	None
Vm Exsi	6.5.0 (Build 5310538)
WinSCP	5.21.3
K9s	0.27.04
Calico	v3.26.0
HAProxy	2.4.22
Kubernetes	1.28.x
Apache Benchmark	2.4
Prometheus	2.47.2
Telegram	
Alert Manager	0.26.0

Table 5.3.1. Preparation

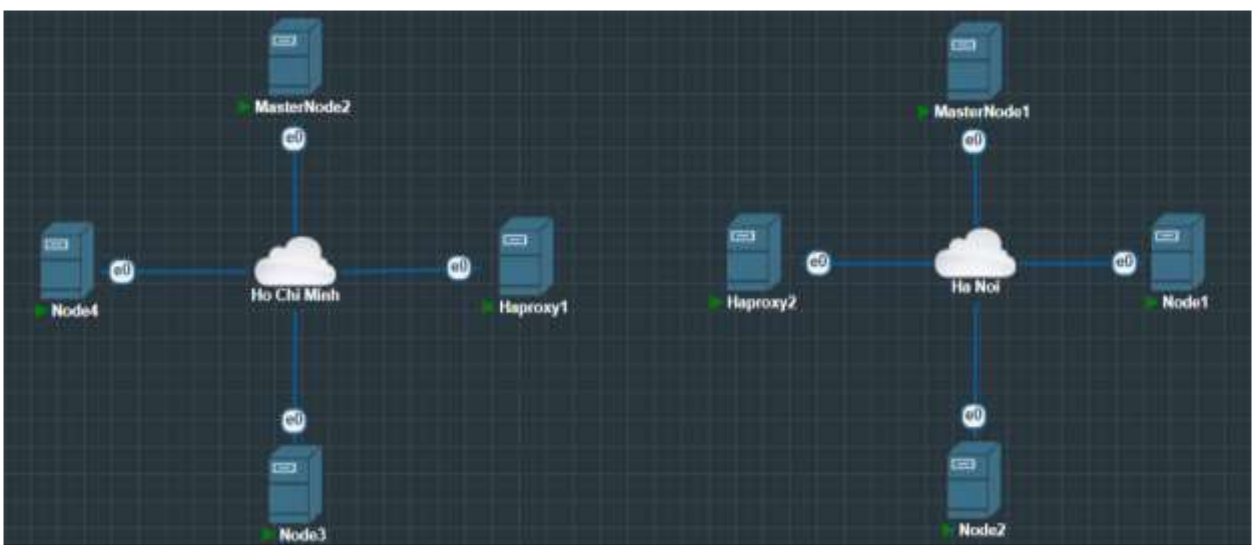


Figure 5.3.1. SDN Network model

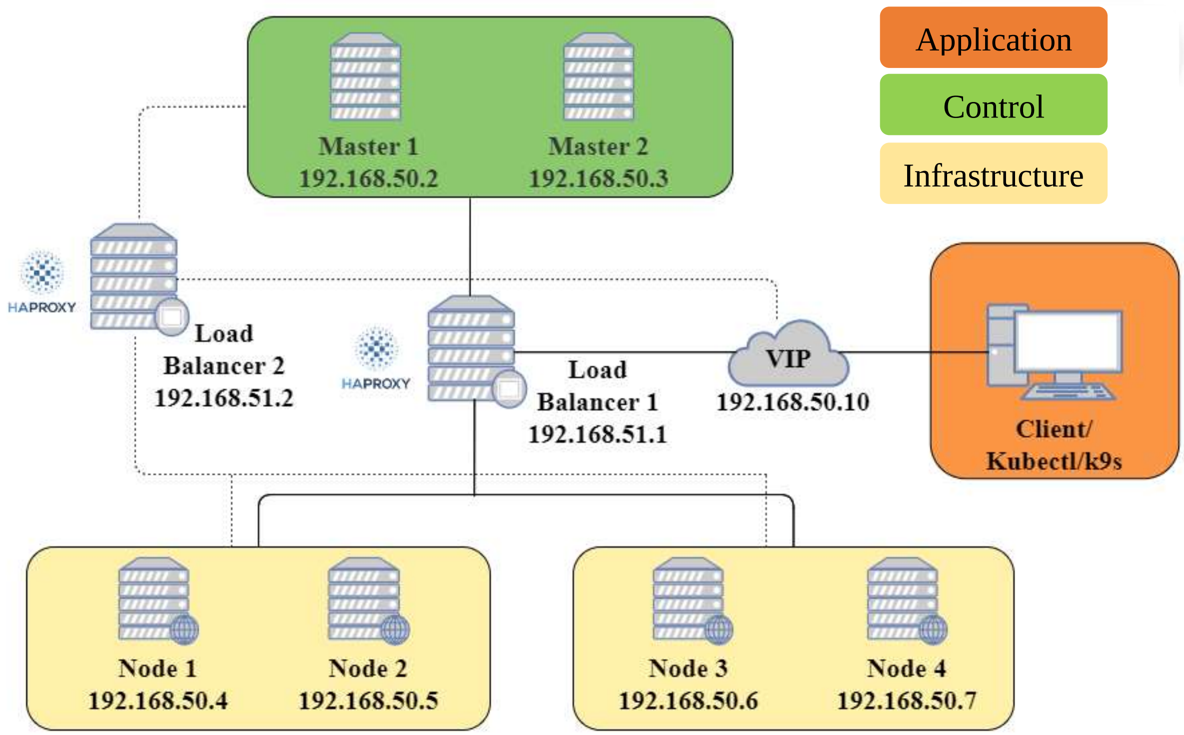


Figure 5.3.2. Detailed 3-layer SDN Network model

Node	Domain name	IP
Node1k8s	node1k8s.k8sdemo.com	192.168.50.4
Node2k8s	node2k8s.k8sdemo.com	192.168.50.5
Node3k8s	node3k8s.k8sdemo.com	192.168.50.6
Node4k8s	node4k8s.k8sdemo.com	192.168.50.7
MasterNode1	k8scontroler1.k8sdemo.com	192.168.50.2
MasterNode2	k8scontroler2.k8sdemo.com	192.168.50.3
HAProxy1		192.168.51.1
HAProxy2		192.168.51.2
Virtual IP		192.168.50.10

Table 5.3.2. Information of devices

5.3.2 Install and Configure

5.3.2.1. Configure HAProxy

Install HAProxy

```
kube@Haproxy1:~$ sudo apt install haproxy -y
[sudo] password for kube:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
haproxy is already the newest version (2.4.22-0ubuntu0.22.04.2).
0 upgraded, 0 newly installed, 0 to remove and 44 not upgraded.
kube@Haproxy1:~$
```

Figure 5.3.3. Install HAProxy 1

```
kube@Haproxy2:~$ sudo apt install haproxy -y
[sudo] password for kube:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
haproxy is already the newest version (2.4.22-0ubuntu0.22.04.2).
0 upgraded, 0 newly installed, 0 to remove and 24 not upgraded.
kube@Haproxy2:~$
```

Figure 5.3.4. Install HAProxy 2

Start HAProxy by the command*: sudo systemctl start haproxy*

Edit HAProxy configure file that apply for 2 master nodes

```
frontend kubernetes
  bind 192.168.50.10:6443
  mode tcp
  option tcplog
  default_backend kubernetes-master-nodes
backend kubernetes-master-nodes
  mode tcp
  option tcp-check
  balance roundrobin
  server k8s-master-1 k8scontroler1.k8sdemo.com:6443 weight 1 check
  server k8s-master-2 k8scontroler2.k8sdemo.com:6443 weight 1 check
```

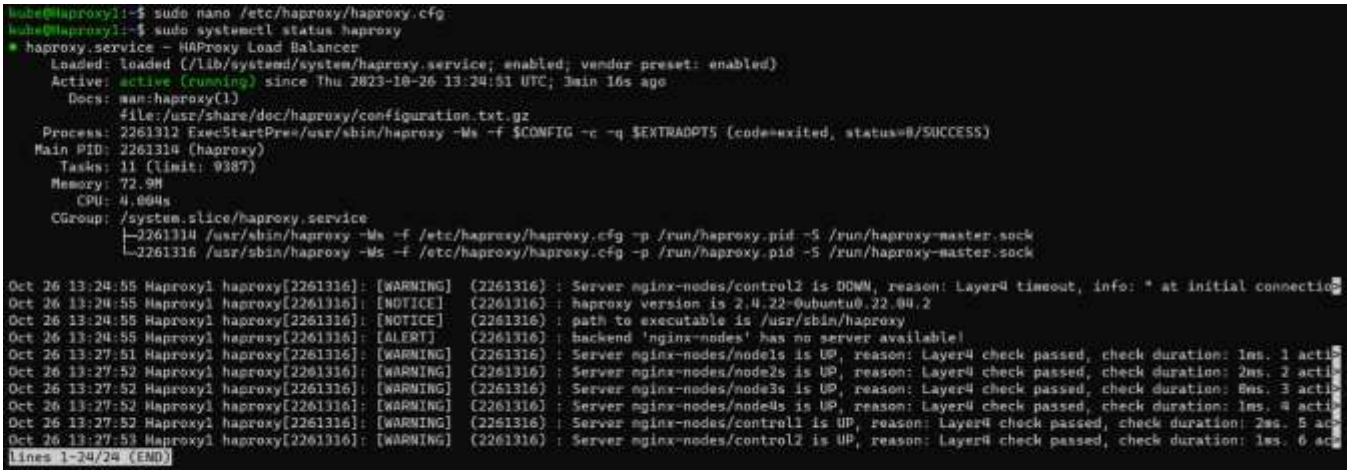
Figure 5.3.5. HAProxy Configure File

Edit HAProxy's Monitor page

```
frontend stats
    mode http
    bind 192.168.50.10:9000
    stats enable
    stats uri /status
    stats refresh 10s
    stats admin if LOCALHOST
```

Figure 5.3.6. HAProxy's monitor page

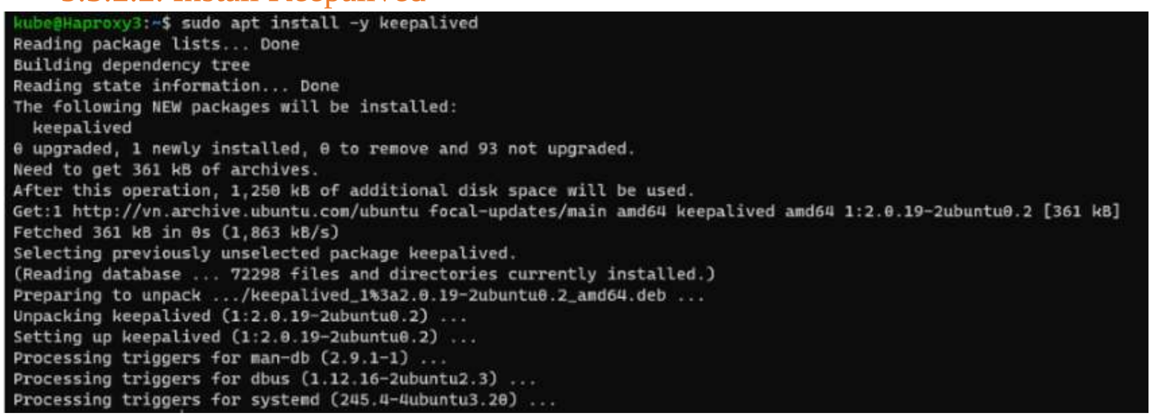
Restart and check the status



```
kube@Haproxy1:~$ sudo nano /etc/haproxy/haproxy.cfg
kube@Haproxy1:~$ sudo systemctl status haproxy
● haproxy.service - HAProxy Load Balancer
     Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-10-26 13:24:51 UTC; 3min 16s ago
       Docs: man:haproxy(1)
             file:/usr/share/doc/haproxy/configuration.txt.gz
    Process: 2261312 ExecStartPre=/usr/sbin/haproxy -Ws -f $CONFIG -c -q $EXTRAOPTS (code=exited, status=0/SUCCESS)
   Main PID: 2261314 (haproxy)
      Tasks: 11 (limit: 9387)
     Memory: 72.9M
        CPU: 4.004s
     CGroup: /system.slice/haproxy.service
             ├─2261314 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master.sock
             └─2261316 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master.sock

Oct 26 13:24:55 Haproxy1 haproxy[2261316]: [WARNING]  (2261316) : Server nginx-nodes/control2 is DOWN, reason: Layer4 timeout, info: " at initial connectio>
Oct 26 13:24:55 Haproxy1 haproxy[2261316]: [NOTICE]   (2261316) : haproxy version is 2.4.22-0ubuntu0.22.04.2
Oct 26 13:24:55 Haproxy1 haproxy[2261316]: [NOTICE]   (2261316) : path to executable is /usr/sbin/haproxy
Oct 26 13:24:55 Haproxy1 haproxy[2261316]: [ALERT]    (2261316) : backend 'nginx-nodes' has no server available!
Oct 26 13:27:51 Haproxy1 haproxy[2261316]: [WARNING]  (2261316) : Server nginx-nodes/node1s is UP, reason: Layer4 check passed, check duration: 1ms. 1 acti>
Oct 26 13:27:52 Haproxy1 haproxy[2261316]: [WARNING]  (2261316) : Server nginx-nodes/node2s is UP, reason: Layer4 check passed, check duration: 2ms. 2 acti>
Oct 26 13:27:52 Haproxy1 haproxy[2261316]: [WARNING]  (2261316) : Server nginx-nodes/node3s is UP, reason: Layer4 check passed, check duration: 0ms. 3 acti>
Oct 26 13:27:52 Haproxy1 haproxy[2261316]: [WARNING]  (2261316) : Server nginx-nodes/node4s is UP, reason: Layer4 check passed, check duration: 1ms. 4 acti>
Oct 26 13:27:52 Haproxy1 haproxy[2261316]: [WARNING]  (2261316) : Server nginx-nodes/control1 is UP, reason: Layer4 check passed, check duration: 2ms. 5 ac>
Oct 26 13:27:53 Haproxy1 haproxy[2261316]: [WARNING]  (2261316) : Server nginx-nodes/control2 is UP, reason: Layer4 check passed, check duration: 1ms. 6 ac>
lines 1-24/24 (END)
```

Figure 5.3.7. HAProxy status

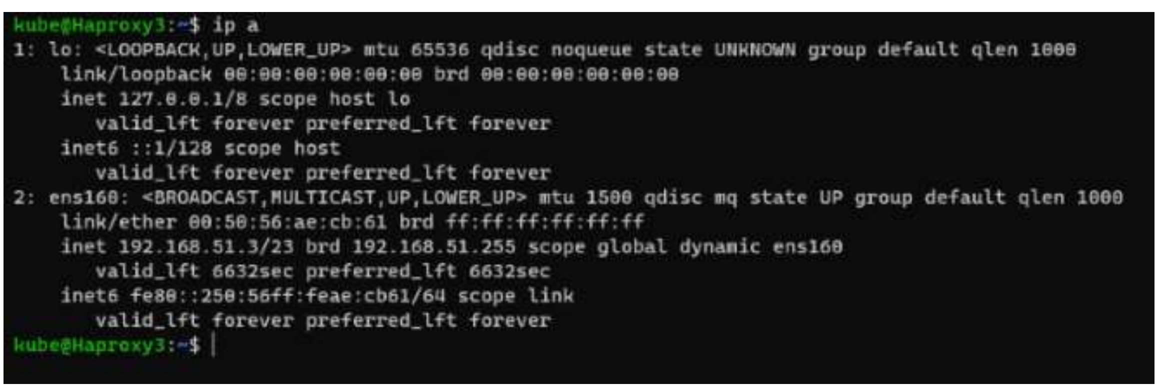
5.3.2.2. Install Keepalived



```
kube@Haproxy3:~$ sudo apt install -y keepalived
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  keepalived
0 upgraded, 1 newly installed, 0 to remove and 93 not upgraded.
Need to get 361 kB of archives.
After this operation, 1,250 kB of additional disk space will be used.
Get:1 http://vn.archive.ubuntu.com/ubuntu focal-updates/main amd64 keepalived amd64 1:2.0.19-2ubuntu0.2 [361 kB]
Fetched 361 kB in 0s (1,863 kB/s)
Selecting previously unselected package keepalived.
(Reading database ... 72298 files and directories currently installed.)
Preparing to unpack .../keepalived_1%3a2.0.19-2ubuntu0.2_amd64.deb ...
Unpacking keepalived (1:2.0.19-2ubuntu0.2) ...
Setting up keepalived (1:2.0.19-2ubuntu0.2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for dbus (1.12.16-2ubuntu2.3) ...
Processing triggers for systemd (245.4-4ubuntu3.20) ...
```

Figure 5.3.8. Install Keepalived

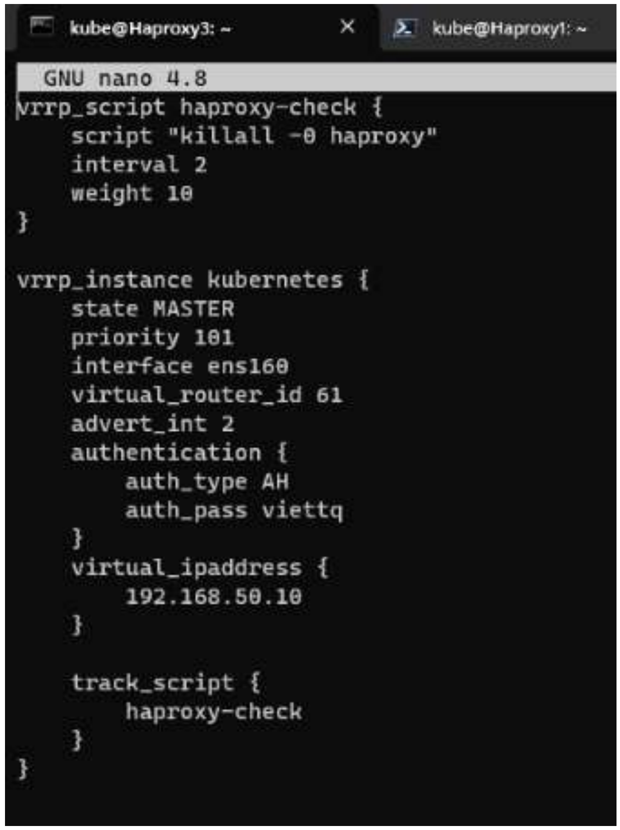
Identify Keepalived port as ens160



```
kube@Haproxy3:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:ae:cb:61 brd ff:ff:ff:ff:ff:ff
    inet 192.168.51.3/23 brd 192.168.51.255 scope global dynamic ens160
       valid_lft 6632sec preferred_lft 6632sec
    inet6 fe80::250:56ff:feae:cb61/64 scope link
       valid_lft forever preferred_lft forever
kube@Haproxy3:~$ |
```

Figure 5.3.9. Indentify Keepalived port

Edit Keepalived configure file and set priority number




```
vrrp_script haproxy-check {
    script "killall -0 haproxy"
    interval 2
    weight 10
}

vrrp_instance kubernetes {
    state MASTER
    priority 101
    interface ens160
    virtual_router_id 61
    advert_int 2
    authentication {
        auth_type AH
        auth_pass viettq
    }
    virtual_ipaddress {
        192.168.50.10
    }

    track_script {
        haproxy-check
    }
}
```

Figure 5.3.10. Configuration for Active

Configuration for Backup 1

```
vrrp_script haproxy-check {
   script "killall -0 haproxy"
   interval 2
   weight 10
}

vrrp_instance kubernetes {
   state BACKUP1
   priority 100
   interface ens160
   virtual_router_id 61
   advert_int 2
   authentication {
      auth_type AH
      auth_pass viettq
   }
   virtual_ipaddress {
      192.168.50.10
   }
```

Configuration for Backup 2

```
vrrp_script haproxy-check {
   script "killall -0 haproxy"
   interval 2
   weight 10
}

vrrp_instance kubernetes {
   state BACKUP2
   priority 99
   interface ens160
   virtual_router_id 61
   advert_int 2
   authentication {
      auth_type AH
      auth_pass viettq
   }
   virtual_ipaddress {
      192.168.50.10
   }
```

track_script {
haproxy-check
}
}

track_script {
haproxy-check
}
}

Start Keepalived service and check status

sudo service keepalived start
sudo service keepalived status

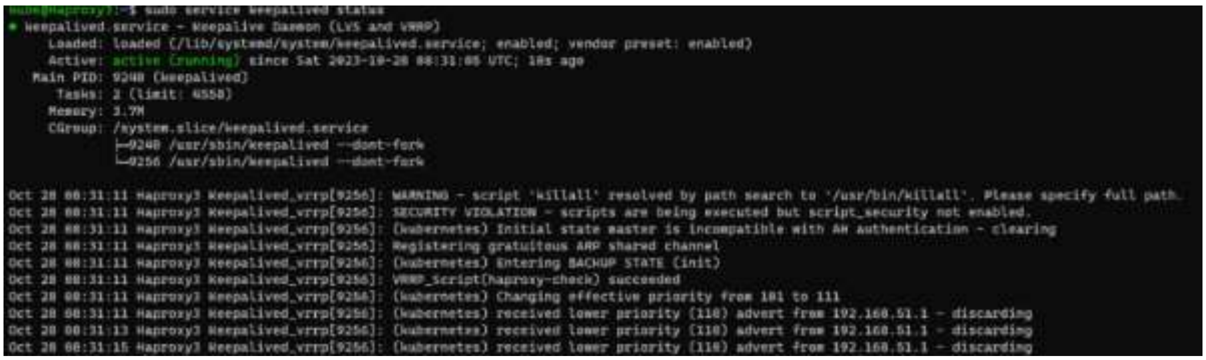


Figure 5.3.11. Keepalived status

5.3.2.3. Create Kubernetes cluster

Set up hostname for worker and master node with the command: “hostnamectl set-hostname”

Disable swap & add kernel parameters

Run the following commands on all the nodes

swapoff -a

sed -i '/ swap / s/^\(.\)$/#\1/g' /etc/fstab*

tee /etc/modules-load.d/containerd.conf <<EOF

overlay

br_netfilter

EOF

sudo modprobe overlay

sudo modprobe br_netfilter

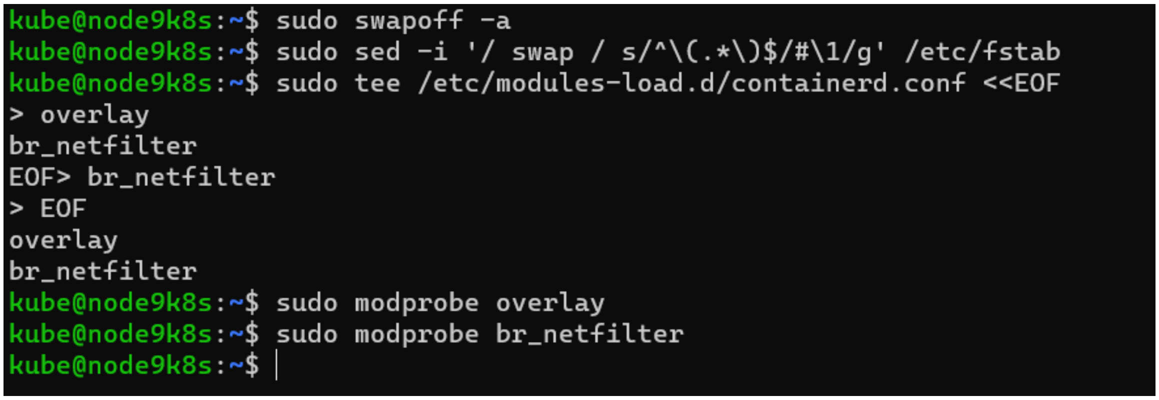


Figure 5.3.12. Swap disables

tee /etc/sysctl.d/kubernetes.conf <<EOF

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip_forward = 1

EOF

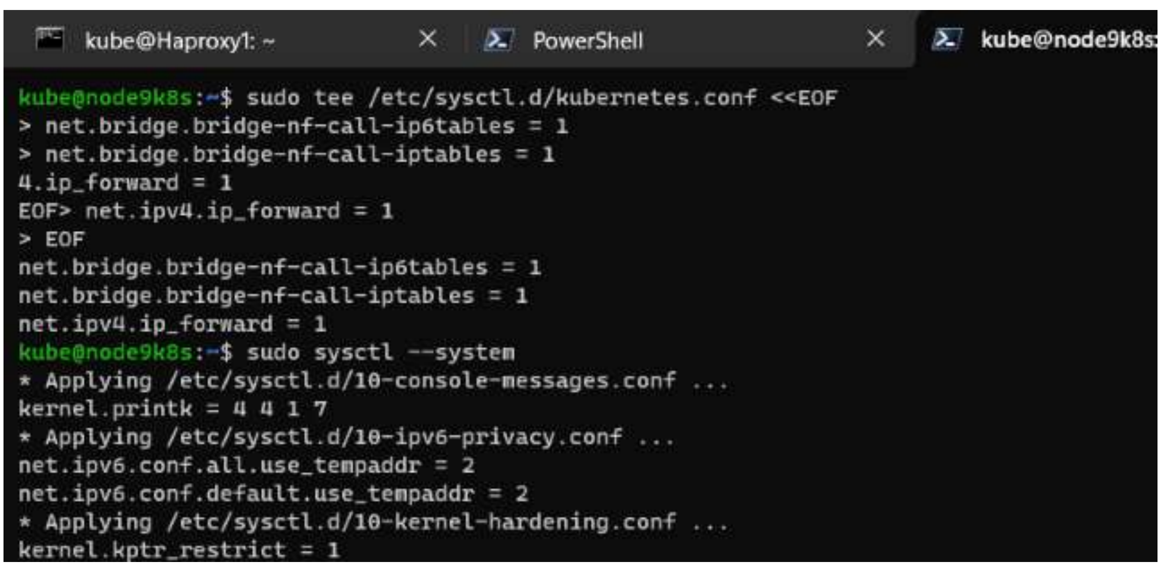


Figure 5.3.13. Add Kernel parameters

Install Containerd runtime

Install containerd dependencies

apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates

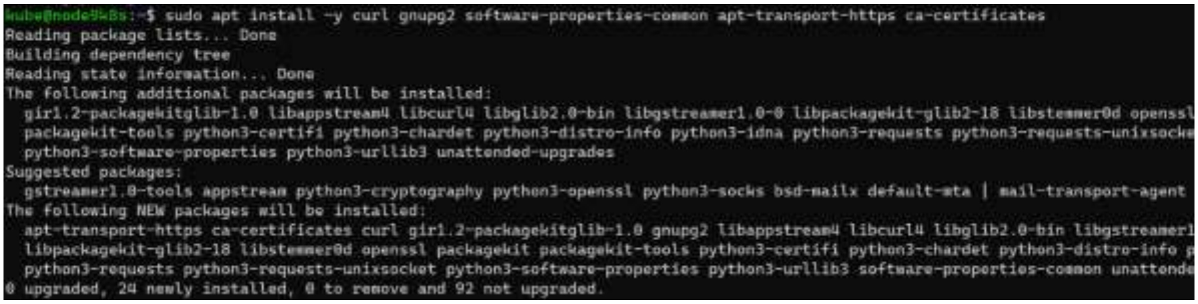


Figure 5.3.14. Container dependencies

Enable Docker repository

dearmour -o /etc/apt/trusted.gpg.d/docker.gpg

add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" ware-properties-common apt-transport-https ca-certificates

Figure 5.3.15. Enable Docker repository

Install containerd

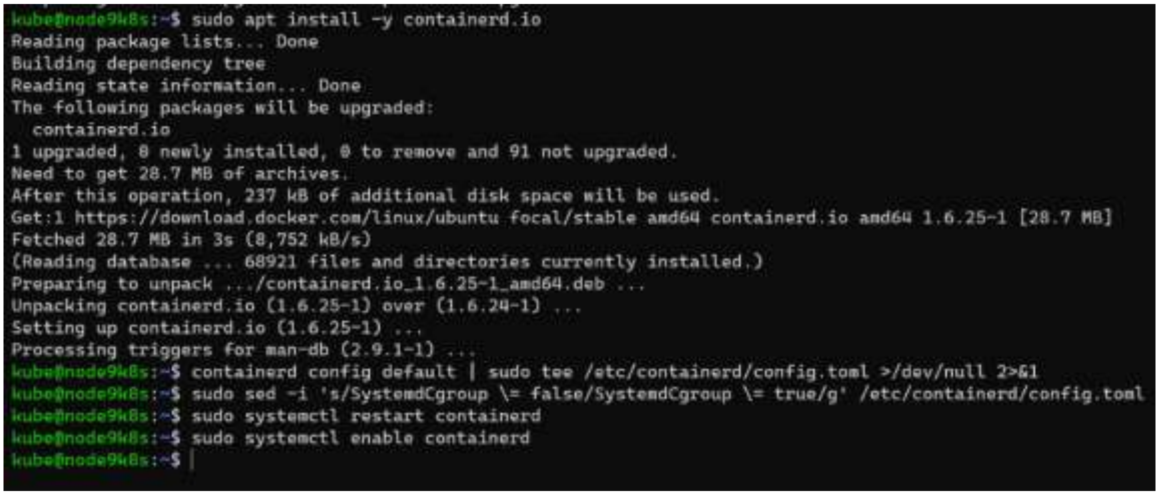
apt update

apt install -y containerd.io

Configure containerd so that it starts using systemd as cgroup

containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1

sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml



```
kube@node9k8s:~$ sudo apt install -y containerd.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  containerd.io
1 upgraded, 0 newly installed, 0 to remove and 91 not upgraded.
Need to get 28.7 MB of archives.
After this operation, 237 kB of additional disk space will be used.
Get:1 https://download.docker.com/linux/ubuntu focal/stable amd64 containerd.io amd64 1.6.25-1 [28.7 MB]
Fetched 28.7 MB in 3s (8,752 kB/s)
(Reading database ... 68921 files and directories currently installed.)
Preparing to unpack .../containerd.io_1.6.25-1_amd64.deb ...
Unpacking containerd.io (1.6.25-1) over (1.6.24-1) ...
Setting up containerd.io (1.6.25-1) ...
Processing triggers for man-db (2.9.1-1) ...
kube@node9k8s:~$ containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
kube@node9k8s:~$ sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml
kube@node9k8s:~$ sudo systemctl restart containerd
kube@node9k8s:~$ sudo systemctl enable containerd
kube@node9k8s:~$ |
```

Figure 5.3.16. Containerd configure

Add Apt Repository for Kubernetes

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/kubernetes-xenial.gpg

$ sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

```
kube@node9k8s:~$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/kubernetes-xenial
.gpg
File '/etc/apt/trusted.gpg.d/kubernetes-xenial.gpg' exists. Overwrite? (y/N) y
kube@node9k8s:~$ sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
Hit:1 http://vn.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://vn.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://vn.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://vn.archive.ubuntu.com/ubuntu focal-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu focal InRelease
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8,993 B]
Fetched 8,993 B in 2s (4,843 B/s)
Reading package lists... Done
```

Figure 5.3.17. Add apt repository

Install Kubectl, Kubeadm and Kubelet

apt update

apt install -y kubelet kubeadm kubectl

apt-mark hold kubelet kubeadm kubectl

```
kube@node9k8s:~$ sudo apt install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools ebtables kubernetes-cni socat
Suggested packages:
  nftables
The following NEW packages will be installed:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 91 not upgraded.
Need to get 87.1 MB of archives.
After this operation, 336 MB of additional disk space will be used.
Get:2 http://vn.archive.ubuntu.com/ubuntu focal/main amd64 conntrack amd64 1:1.4.5-2 [30.3 kB]
Get:4 http://vn.archive.ubuntu.com/ubuntu focal/main amd64 ebtables amd64 2.0.11-3build1 [80.3 kB]
Get:5 http://vn.archive.ubuntu.com/ubuntu focal/main amd64 socat amd64 1.7.3.3-2 [323 kB]
Get:1 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 cri-tools amd64 1.26.0-00 [18.9 MB]
Get:3 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubernetes-cni amd64 1.2.0-00 [27.6 MB]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubelet amd64 1.28.2-00 [19.5 MB]
Get:7 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubectl amd64 1.28.2-00 [10.3 MB]
Get:8 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubeadm amd64 1.28.2-00 [10.3 MB]
Fetched 87.1 MB in 8s (11.2 MB/s)
```

Figure 5.3.18. Install Kubectl, Kubeadm and Kubelet

Initialize Kubernetes Cluster with Kubeadm

Run the following Kubeadm command on the master node only.

sudo kubeadm init --control-plane-endpoint=192.168.50.10 --upload-certs --apiserver-advertise-address=192.168.50.3

In master node 1, do the following commands

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of the control-plane node running the following command on each as root:

  kubeadm join 192.168.50.10:6443 --token yvyoqq.nzd14lzefhdejvfw \
        --discovery-token-ca-cert-hash sha256:3fa10bbe233bf939dd3e43f8995ebed81217424a69e5ef2b9fa4caf9d55412b8 \
        --control-plane --certificate-key 44f7759789071d86ac1b7869fa9fc871a129bdd66397af3657befb14c5ff356e
```
Figure 5.3.19. Initialize Kubernetes Cluster

In master node 2 do a command as below

```
kubeadm join 192.168.50.10:6443 --token yvyoqq.nzd14lzefhdejvfw \
        --discovery-token-ca-cert-hash sha256:3fa10bbe233bf939dd3e43f8995ebed81217424a69e5ef2b9fa4caf9d55412b8 \
        --control-plane --certificate-key 44f7759789071d86ac1b7869fa9fc871a129bdd66397af3657befb14c5ff356e
```

Figure 5.3.20. Join master node 2 to Kubernetes Cluster

Join Worker Nodes to the Cluster. After that do these commands:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
kubeadm join 192.168.50.10:6443 --token yvyoqq.nzd14lzefhdejvfw \
        --discovery-token-ca-cert-hash sha256:3fa10bbe233bf939dd3e43f8995ebed81217424a69e5ef2b9fa4caf9d55412b8
```

Figure 5.3.21. Join worker node to Kubernetes Cluster

5.3.2.4. Set up kubectl

Download kubectl.exe by the command: curl.exe -LO
https://dl.k8s.io/release/v1.28.3/bin/windows/amd64/kubectl.exe

Set kubectl as environment variable

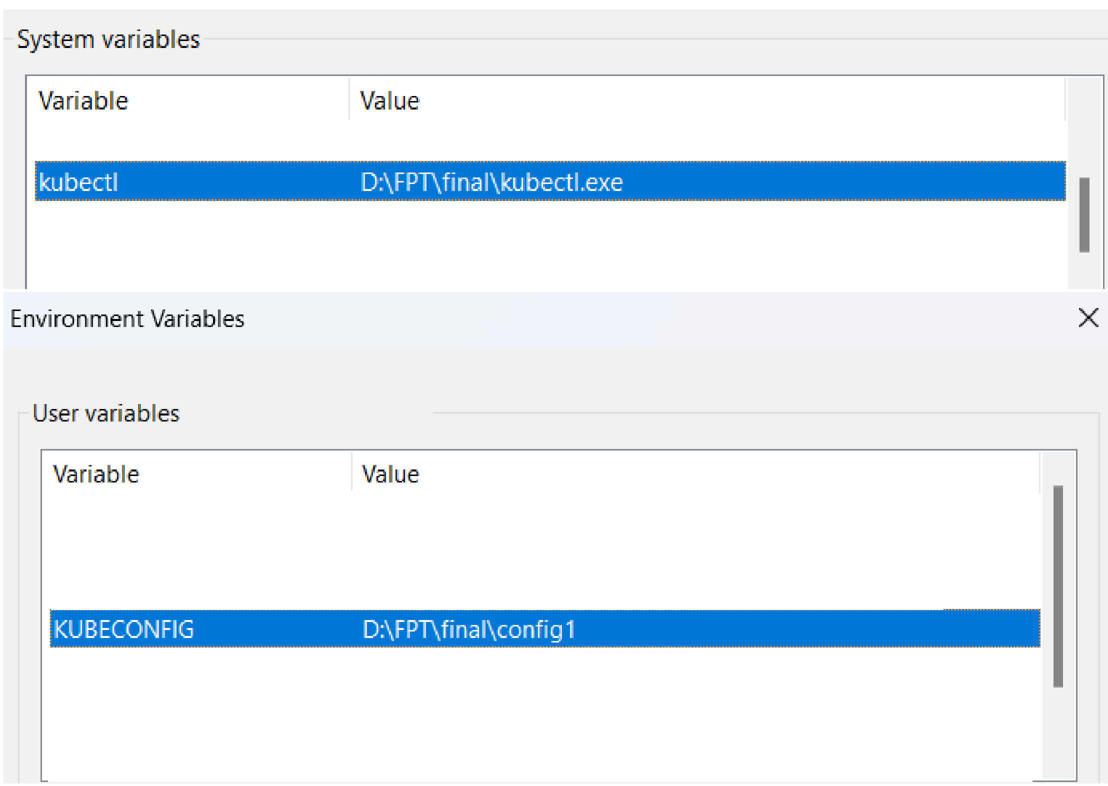
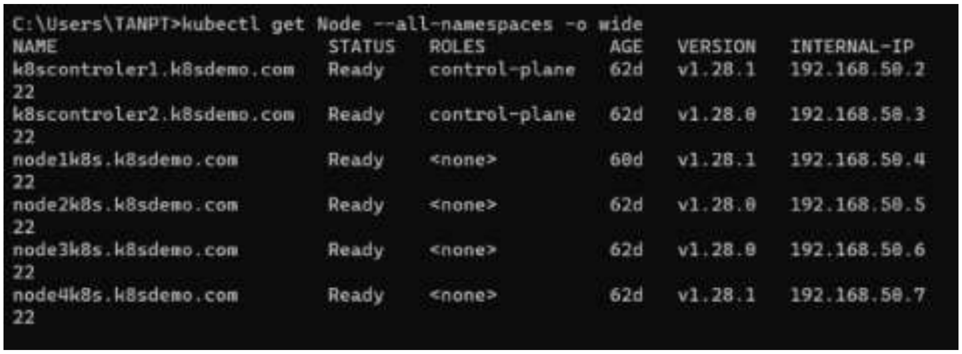


Figure 5.3.22. Set Kubectl as environment variable

Test kubectl on windows



```
C:\Users\TANPT>kubectl get Node --all-namespaces -o wide
NAME                        STATUS   ROLES           AGE   VERSION   INTERNAL-IP
k8scontroler1.k8sdemo.com   Ready    control-plane   62d   v1.28.1   192.168.50.2
22
k8scontroler2.k8sdemo.com   Ready    control-plane   62d   v1.28.0   192.168.50.3
22
node1k8s.k8sdemo.com        Ready    <none>          60d   v1.28.1   192.168.50.4
22
node2k8s.k8sdemo.com        Ready    <none>          62d   v1.28.0   192.168.50.5
22
node3k8s.k8sdemo.com        Ready    <none>          62d   v1.28.0   192.168.50.6
22
node4k8s.k8sdemo.com        Ready    <none>          62d   v1.28.1   192.168.50.7
22
```

Figure 5.3.23. Test Kubectl

5.3.2.5. Install K9s and np-viewer

Download k9s at: https://github.com/derailed/k9s/releases

Extract downloaded folder and copy k9s.exe to C:\Windows\system32

Start k9s by cmd by type “k9s” after that k9s is running

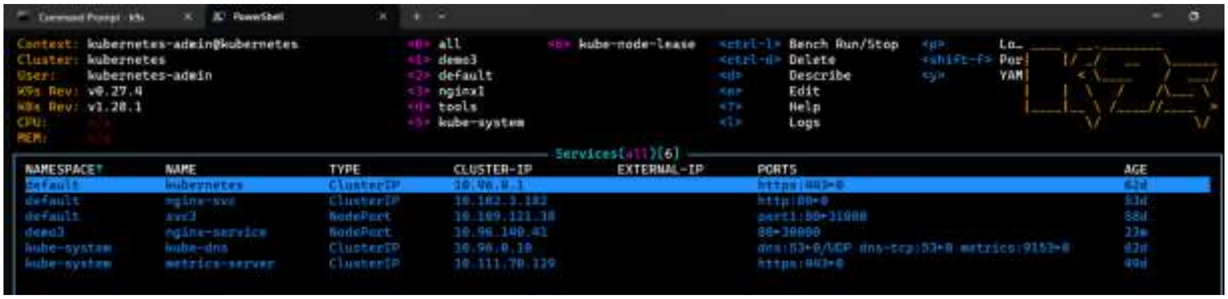
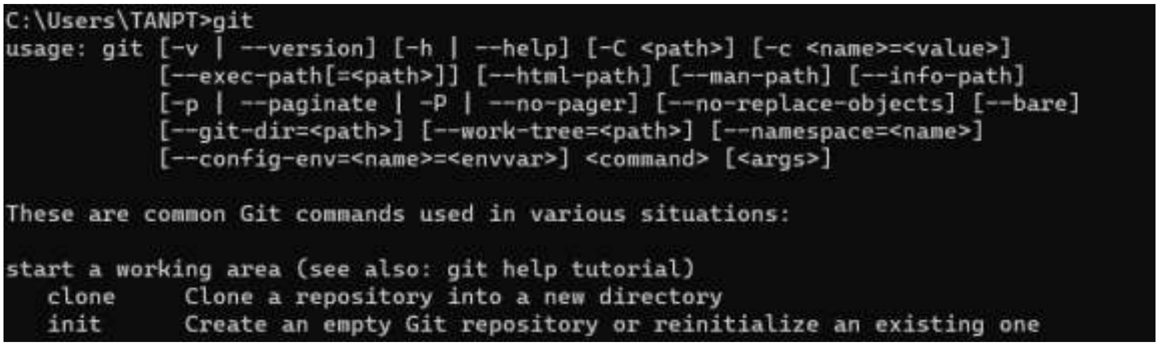


Figure 5.3.24. K9s dashboard

Install np-viewer

Install Git for Windows at: https://gitforwindows.org/



```
C:\Users\TANPT>git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--config-env=<name>=<envvar>] <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one
```

Figure 5.3.25. Git

Download krew.exe from the [Releases](#) page to a directory

Figure 5.3.26. Krew release

Change name of file to krew.exe and launch a command prompt with administrator privileges (since the installation requires use of symbolic links) and navigate to krew directory and run the command: ".\krew install krew"

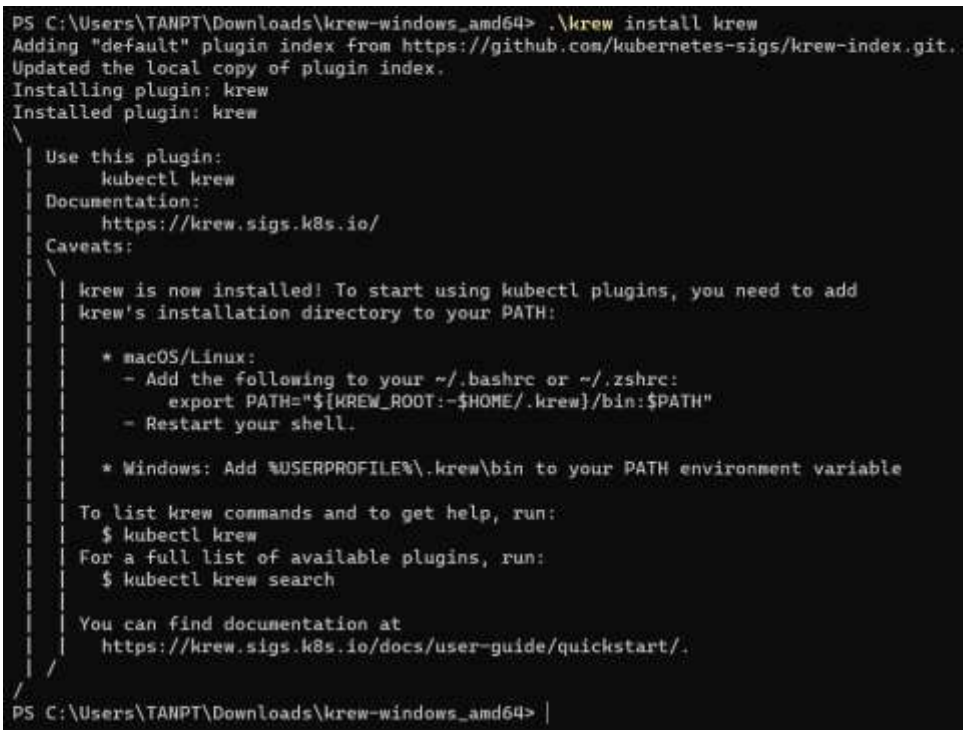


Figure 5.3.27. Start krew installation

Add the *"%USERPROFILE%\.krew\bin"* directory to PATH environment variable

Run *"kubectl krew"* to check the installation.

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.22631.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>kubectl krew
krew is the kubectl plugin manager.
You can invoke krew through kubectl: "kubectl krew [command]..."

Usage:
  kubectl krew [command]

Available Commands:
  help        Help about any command
  index       Manage custom plugin indexes
  info        Show information about an available plugin
  install     Install kubectl plugins
```

Figure 5.3.28. Check krew installation

Install np-viewer with krew

```
C:\Windows\System32>kubectl krew install np-viewer
Updated the local copy of plugin index.
Installing plugin: np-viewer
Installed plugin: np-viewer
\
 | Use this plugin:
 |      kubectl np-viewer
 | Documentation:
 |      https://github.com/runoncloud/kubectl-np-viewer
 | Caveats:
 | \
 |  | In order to get the plugin running you will need permissions to GET
 |  | Network Policies in at least one Namespace.
 | /
/
WARNING: You installed plugin "np-viewer" from the krew-index plugin repository.
   These plugins are not audited for security by the Krew maintainers.
   Run them at your own risk.

C:\Windows\System32>
```

Figure 5.3.29. Install np-viewer

5.3.2.6. Install Prometheus

Using helm to install prometheus on Cluster with chart prometheus-community by this command: "*helm install prometheus promethesus-community/prometheus –f value-capstone.yaml –namespace=Prometheus*"

Figure 5.3.30. Download promethesus

Configure file value and upgrade Prometheus with our configuration

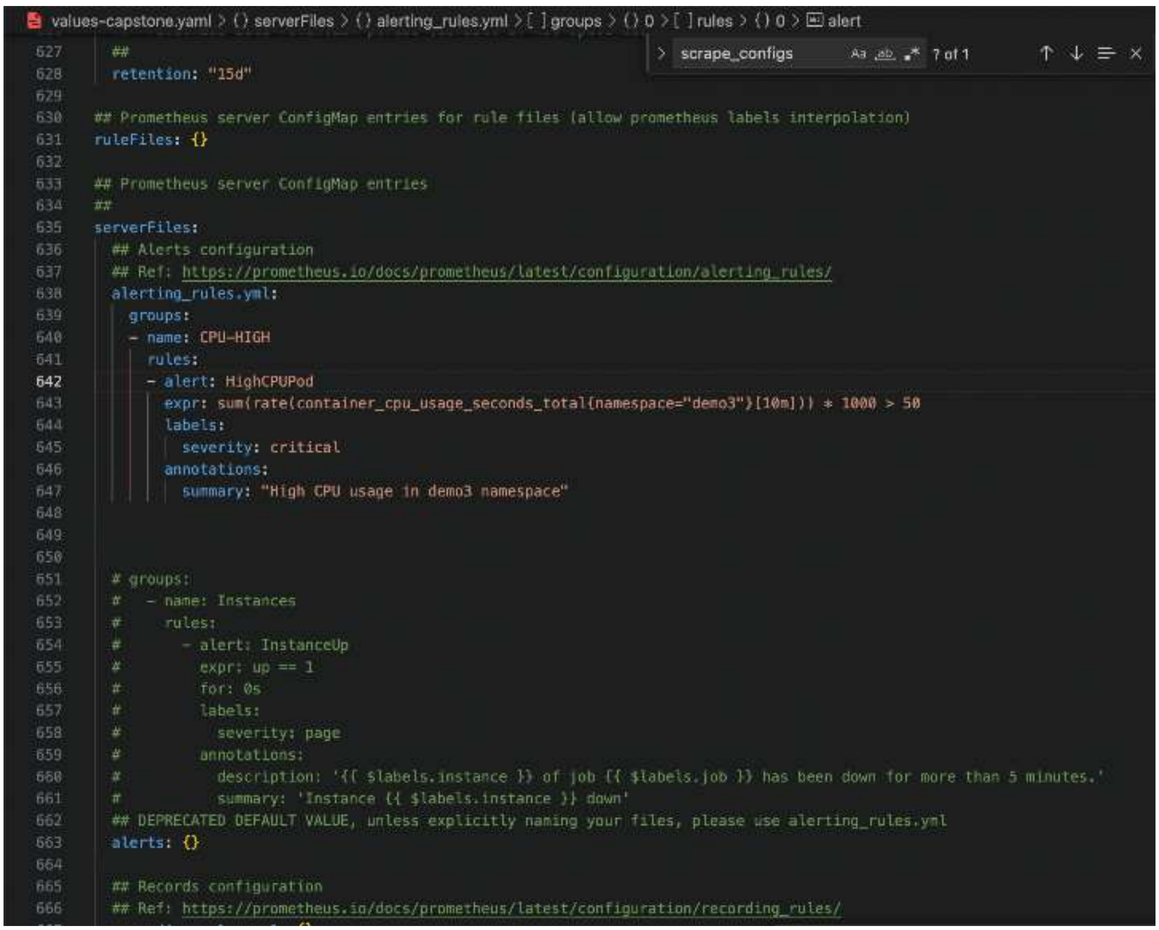


Figure 5.3.31. Prometheus configuration

After that upgrade helm chart with command "*helm upgrade –install prometheus-community/prometheus --create-namespace –namespace prometheus –values values-capstone.yaml*". And the result

```
NAME: prometheus
LAST DEPLOYED: Wed Nov 22 17:20:42 2023
NAMESPACE: prometheus
STATUS: deployed
REVISION: 28
TEST SUITE: None
NOTES:
The Prometheus server can be accessed via port 80 on the following DNS name from within your cluster:
prometheus-server.prometheus.svc.cluster.local


Get the Prometheus server URL by running these commands in the same shell:
  export POD_NAME=$(kubectl get pods --namespace prometheus -l "app.kubernetes.io/name=prometheus,app.kubernetes.io/instanc
="{.items[0].metadata.name}")
  kubectl --namespace prometheus port-forward $POD_NAME 9090
```

Figure 5.3.32. Upgrade helm chart

After install Prometheus we will install Alert Manager (will push alert to telegram) with this configuration

```yaml
config:
  global:
    resolve_timeout: 5m
  route:
    group_by: ['alertname', 'namespace', 'pod']
    group_wait: 10s
    group_interval: 10s
    repeat_interval: 5m
    receiver: 'alert-capstone-data'
    routes:
      - receiver: 'alert-capstone-data'
        match_re:
          severity: ^(warning|critical)$
        continue: true

  receivers:
    - name: 'alert-capstone-data'
      telegram_configs:
      - send_resolved: true
        http_config:
          follow_redirects: true
        api_url: "https://api.telegram.org"
        bot_token: "6706796834:AAGolyG-rkVdUC6N6sq7YLwA4GVC2154VAE"
        chat_id: -1002015547121
        message: '{{ template "telegram.default.message" . }}'      EQ-0623, 4 months ago •
        parse_mode: HTML
                                                                     Code langu
  templates:
```

Figure 5.3.33. Install Alert Manager

Set a namespace for Alert manager same as Prometheus by the command: "helm upgrade –install alertmanager Prometheus-community/alertmanager –create-namespace –namespace Prometheus –value values_alert_manager.yaml"

After that we will have prometheus and alert manager in same namespace

Figure 5.3.34. Prometheus namespace

5.3.3. Demonstration

5.3.3.1. Case 1: Isolate a Node in Kubernetes Cluster

Purpose

Node isolation is used to repair devices without affecting services running in the system. When Applications are created and moved across Nodes in the system, they are also automatically configured with the necessary parameters desired by the administrator (nodes will have their own IP range or private IP address). Policy for that node, when an application is moved from one Node to another, those configurations are still guaranteed.

Model

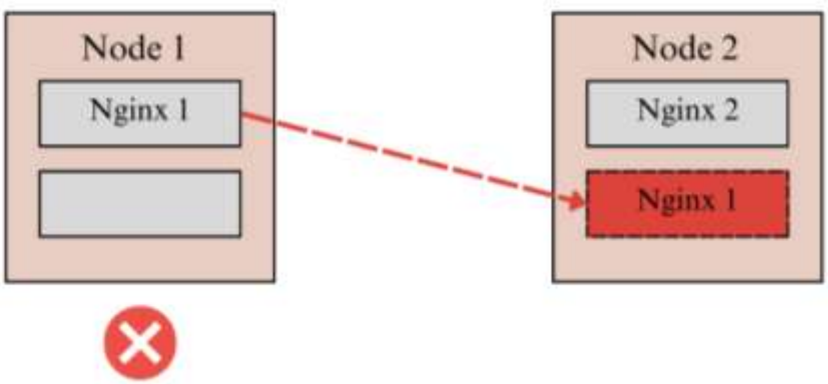


Figure 5.3.35. Case 1 model

Step by step

Start k9s and check the node status

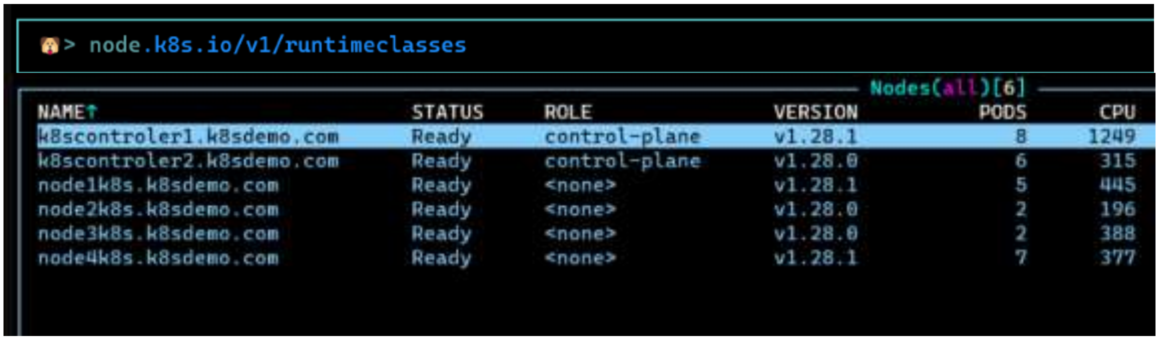


Figure 5.3.36. Check node status with k9s

Access Node 4 (node4k8s.k8sdemo.com) to see list of pods are running by Enter

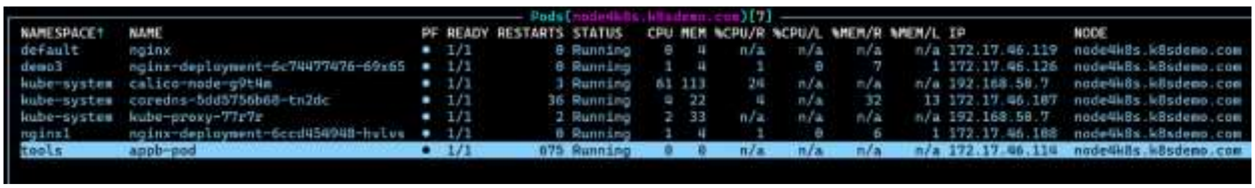


Figure 5.3.37. List of pods in Node 4

This node has 7 pods running, then we begin to isolate this node. Click “ESC” to return to the node page list. Navigate to Node 4 and type “C” to start the isolation process and confirm with “OK”.

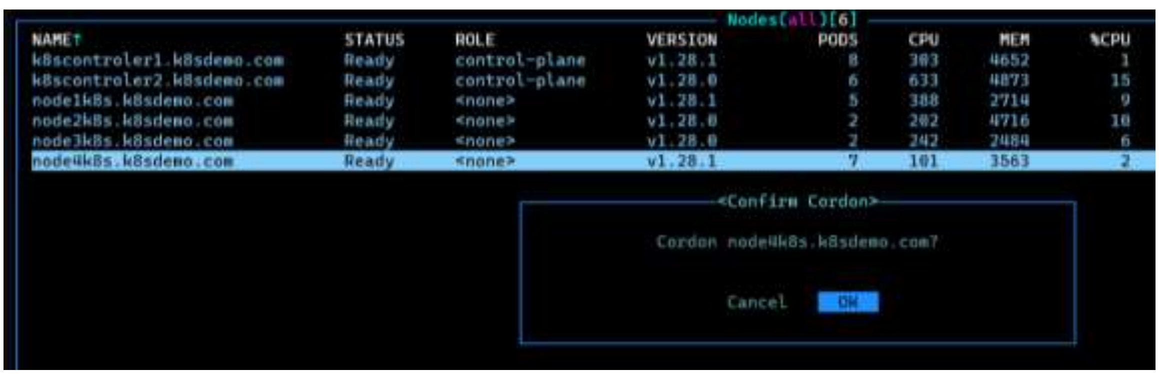


Figure 5.3.38. Start isolate Node

The node is being isolate

Figure 5.3.39. Node is being isolate

We will test the isolation by creating 8 more Nginx pods

Figure 5.3.40. Create more pods

The creation was successful, but no pods were created inside Node 4 except the existing Nginx. So, we have successfully isolated Node 4. Then we need to push the pods in Node 4 out for maintenance

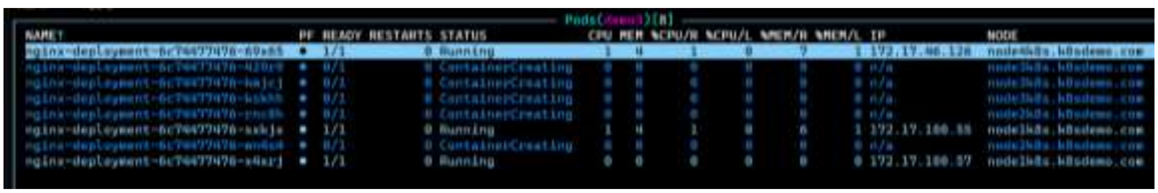


Figure 5.3.41. New pods were not created inside Node 4

Back to the Node page list. Type “R” to execute the push command and select option as below, then “OK”

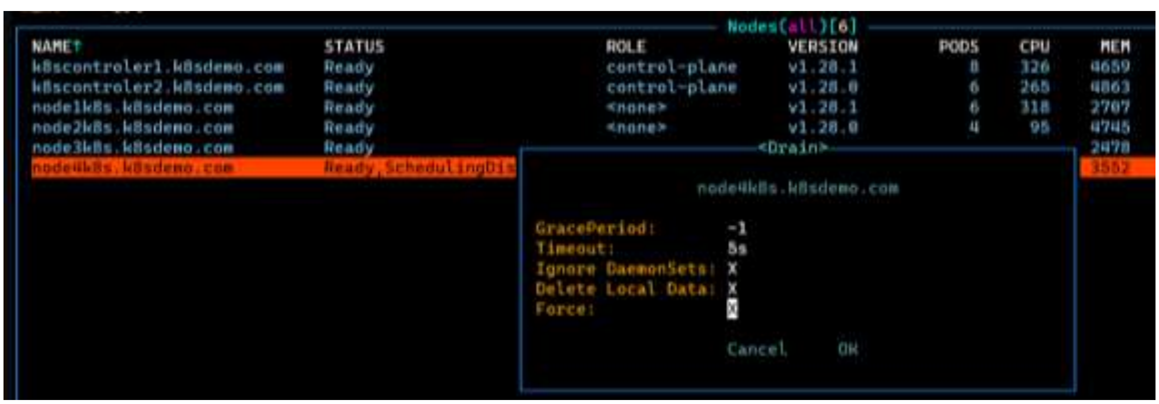


Figure 5.3.42. Drain Node 4

After confirming, these pods will be moved out of Node 4

Go to the pod page list, we will see the existing Nginx pod in Node4 are moved to Node1

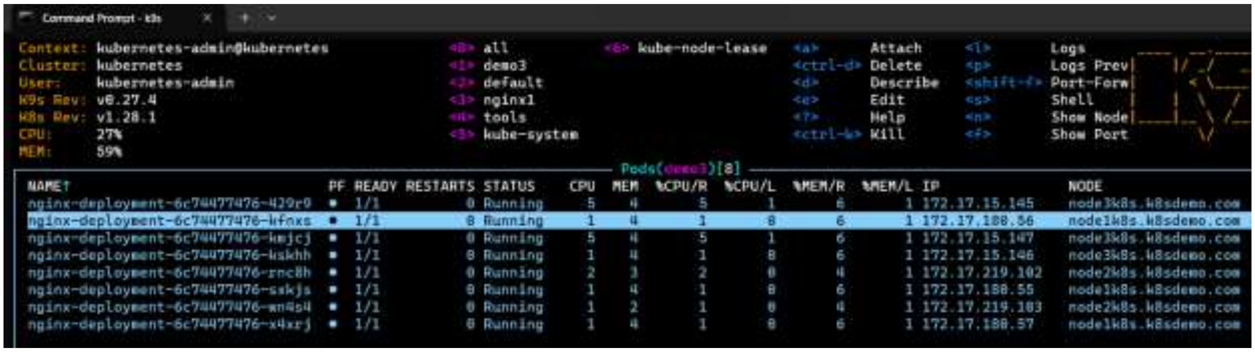


Figure 5.3.43. Drain Node 4 successfully

5.3.3.2. Case 2: Implement security policy for the network

Purpose

This demo demonstrates the advantages of SDN in improving system security by centralized management through the Master Node (Control node). Administrators can easily deploy and control communication and policies between pods, services in the system and machines in the system.

Model

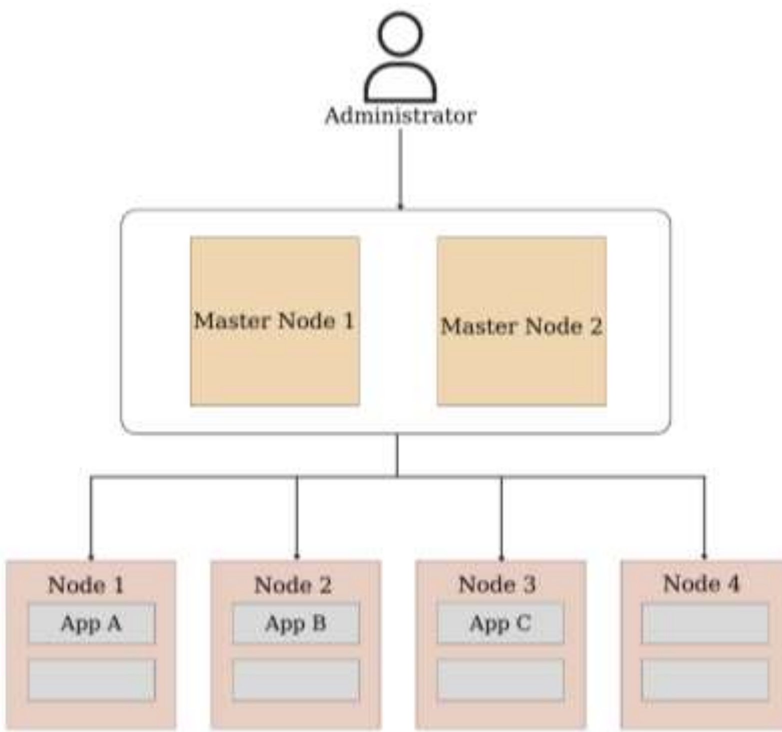


Figure 5.3.44. Case 2 model

Use case in this demo

Environment separation: In an environment with multiple development, staging, and production environments running in the same cluster, you may want to prevent teams in the development environment from communicating with teams in the production environment. Therefore, we use simple commands to test the communication between applications.

Network Management tool user guide

In this demo, we have a network management tool made by us. This tool has a menu interface with simple steps that can apply a network policy to the target pods, applications...

When you have a policy that want to apply to cluster, it can be made by our management tool with several step:

Step 1: Select namespace by option 2

Step 2: Choose a namespace

Step 3: Choose a pod in selected namespace with option 3

Step 4: Choose a policy template and apply it with the option “Execute”. Another option which is “Export to a YAML file with the name of your choices”. In this option, your policy is not applied automatically to the cluster. Besides that, you can view your policy in the selected namespace by option “View network menu”

Step by step

Create a YAML file as follows named “*demo-networkpolicy.yaml*” to create 3 app pods containing testing tools such as ping

```
apiVersion: v1
kind: Namespace
metadata:
  name: tools
---
apiVersion: v1
kind: Pod
metadata:
  name: appa-pod
  namespace: tools
  labels:
    app: AppA
spec:
  containers:
  - name: alpine
    image: alpine:latest
    command: ["/bin/sh"]
    args: ["-c", "sleep 3600"]
---
apiVersion: v1
kind: Pod
metadata:
  name: appb-pod
  namespace: tools
  labels:
    app: AppB
spec:
  containers:
  - name: alpine
    image: alpine:latest
    command: ["/bin/sh"]
    args: ["-c", "sleep 3600"]
---
apiVersion: v1
kind: Pod
metadata:
  name: appc-pod
  namespace: tools
  labels:
    app: AppC
spec:
  containers:
  - name: alpine
    image: alpine:latest
    command: ["/bin/sh"]
    args: ["-c", "sleep 3600"]
---
```

We used the YAML file above to initiate three pods. Go to CMD and type the command "*kubectl apply –f .\demo-networkpolicy.yaml*".

```
PS C:\Users\TANPT\OneDrive - Đại học FPT- FPT University\FALL23\SDN\configure\YAML FILE> kubectl apply -f .\demo-networkpolicy.yaml
Warning: resource namespaces/tools is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply sh
ould only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically
.
namespace/tools configured
pod/appa-pod created
pod/appb-pod created
pod/appc-pod created
PS C:\Users\TANPT\OneDrive - Đại học FPT- FPT University\FALL23\SDN\configure\YAML FILE>
```

Figure 5.3.45. Three applications are created

After running, go to K9s to check pods status

```
                                          Pods(tools)[3]
NAME↑      PF  READY  RESTARTS STATUS     CPU  MEM  %CPU/R  %CPU/L  %MEM/R  %MEM/L IP             NODE
appa-pod   ●   1/1           0 Running    0    0    n/a     n/a     n/a     n/a 172.17.15.148  node3k8s.k8sdemo.com
appb-pod   ●   1/1Δ          0 RunningΔ   0    0    n/a     n/a     n/a     n/a 172.17.180.60  node1k8s.k8sdemo.com
appc-pod   ●   1/1           0 Running    0    0    n/a     n/a     n/a     n/a 172.17.15.150  node3k8s.k8sdemo.com
```

Figure 5.3.46. Check pods status

Access the CLI of app A (appa-pod) to perform a ping to app B (appb-pod). Select app A and type "S" to access the Shell. After that, execute ping command and it was successful

At here, we have two options for create a network policy for this application:

- **Create network policy by network management tool**

```
===== Main MENU =====
1. View Network Menu
2. Display the Namespace list and select Namespace
3. Select Pod in Namespace
4. Exit

Select options: 2
List of Namespaces:
1. default
2. demo3
3. kube-node-lease
4. kube-public
5. kube-system
6. nginx1
7. prometheus
8. tools
Select Namespace (enter the corresponding number): 8

===== Main MENU =====
1. View Network Menu
2. Display the Namespace list and select Namespace
3. Select Pod in Namespace
4. Exit

Select options: 3
List of Labels :
1. {'app': 'AppA'}
2. {'app': 'AppB'}
3. {'app': 'AppC'}
4. {'app': 'web'}
Select Pod (enter the corresponding number): 2
```

```
Menu Network Policy:
01. Deny_all_traffic_to_an_application
02. Allow_traffic_to_an_application_from_all_namespaces
03. Deny_all_none_whitelisted_traffic_to_a_name_space
04. Deny_all_traffic_from_other_namespace
05. Deny_all_traffic_from_app_to_app
Select Policy for seleted Label: 5
Select the Label you want to deny all traffic to AppB:
1. {'app': 'AppA'}
2. {'app': 'AppB'}
3. {'app': 'AppC'}
4. {'app': 'web'}
Select Pod (enter the corresponding number): 1
1. Execute
2. Export to a yaml file with a name of your choice
Select an option (1 or 2): 1
networkpolicy.networking.k8s.io/deny-from-appa-to-appb created


===== Main MENU =====
1. View Network Menu
2. Display the Namespace list and select Namespace
3. Select Pod in Namespace
4. Exit

Select options: 4
Thanks for using this Tools!!
```

Figure 5.3.47. Steps to apply policy that deny all traffic from app to app with our tool

- **Create network policy by manual**

Create a YAML file contain network policy named "*policy-blockping.yaml*" to block the pings from app A to app B

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-appa-to-appb
  namespace: tools
spec:
  podSelector:
    matchLabels:
      app: AppB
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchExpressions:
          - {key: app, operator: NotIn, values: ["AppA"]}
```

Apply to the system and check again

Figure 5.3.48. Apply network policy

Access the Shell of app A and ping to app B to test the policy. There is 100% packet loss, so we successfully block ping traffic from A to B. But to ensure the policy works well we execute several steps further, ping from app B to app A and ping from app C to app B. And as expected, ping is still working.

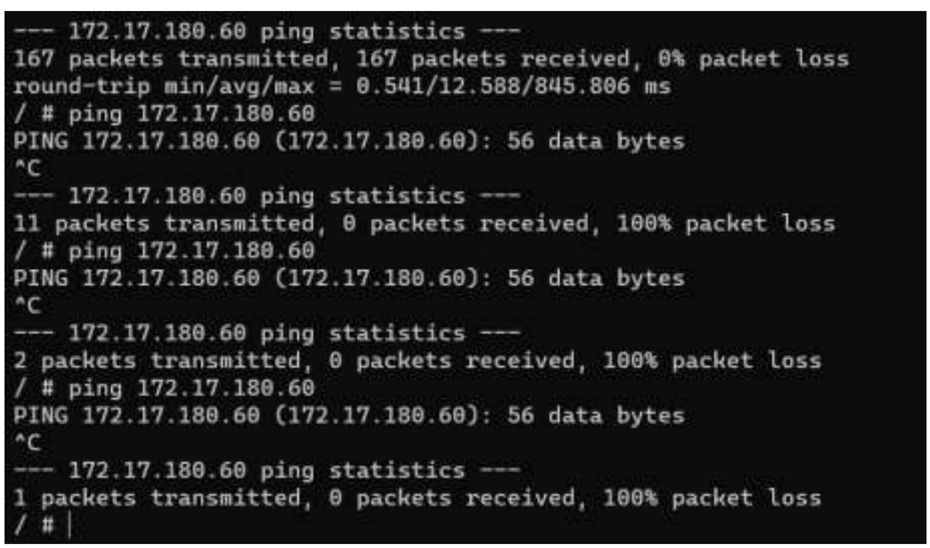


Figure 5.3.49. Complete packet loss from app A to app B

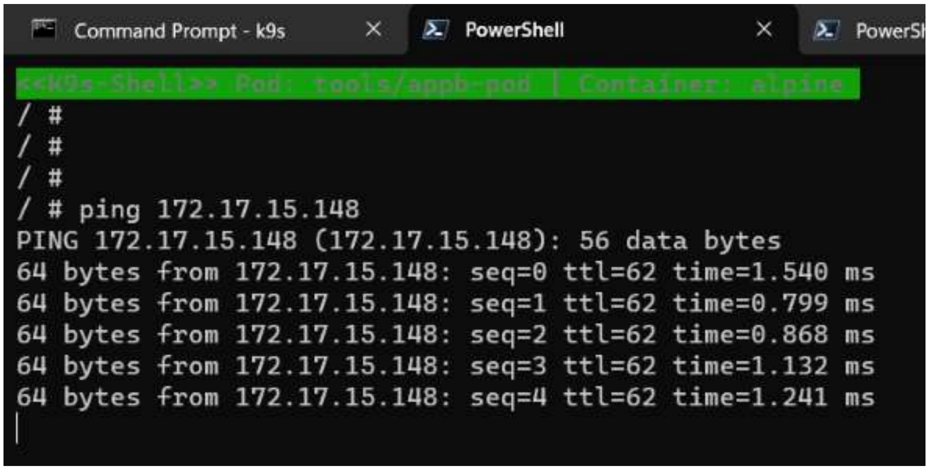


Figure 5.3.50. App B can communicate with app A

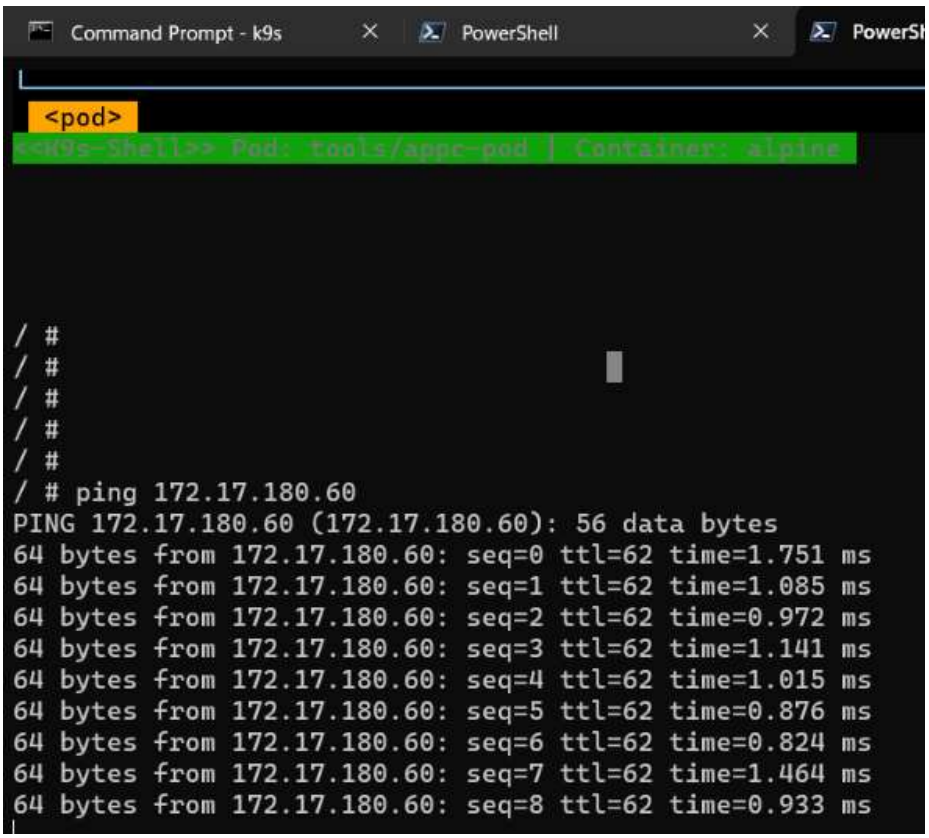


Figure 5.3.51. App C can communicate with app B

Other policy templates

a. Deny_all_traffic_to_an_application

Purpose

This Network policy will drop all traffic to pods of an application, selected using Pod Selectors.

Step by step

Start a web service on tools namespace by command: “kubectl run web --namespace=tools --image=nginx --labels="app=web" --expose --port=80”

```
PS C:\Users\[redacted]> kubectl run web --namespace=tools --image=nginx --labels="app=web" --expose --port=80
service/web created
pod/web created
PS C:\Users\[redacted]>
```

Figure 5.3.52. Create an application named web

Create 01_Deny_all_traffic_to_an_application.py file

```python
with open("namespace.json", "r") as config_file:
    config = json.load(config_file)

namespace = config["namespace"]
selected_label = config.get("label", {}).get("app", "")


network_policy = {
    "kind": "NetworkPolicy",
    "apiVersion": "networking.k8s.io/v1",
    "metadata": {
        "name": f"{selected_label.lower()}-deny-all-policy",
        "namespace": namespace
    },
    "spec": {
        "podSelector": {
            "matchLabels": {
                "app": selected_label  # Use the label in the pod selector
            }
        },
        "policyTypes": ["Ingress"],  # Specify the policy type
        "ingress": []
    }
}
```

Figure 5.3.53. Code of Deny_all_traffic_to_an_application.py file

Apply file 01_Deny_all_traffic_to_an_application.py for namespace=tools and app=web with these steps

```
Select options: 2
List of Namespaces:
1. default
2. demo3
3. kube-node-lease
4. kube-public
5. kube-system
6. nginx1
7. prometheus
8. tools
Select Namespace (enter the corresponding number): 8

===== Main MENU =====
1. View Network Menu
2. Display the Namespace list and select Namespace
3. Select Pod in Namespace
4. Exit

Select options: 3
List of Labels:
1. {'app': 'AppA'}
2. {'app': 'AppB'}
3. {'app': 'AppC'}
4. {'app': 'web'}
Select Pod (enter the corresponding number): 4
Menu Network Policy:
01. Deny_all_traffic_to_an_application
02. Limit_traffic_to_an_application
03. Deny_all_none_whitelisted_traffic_to_a_name_space
04. Deny_all_traffic_from_other_namespace
05. Deny_all_traffic_from_app_to_app
Select Policy for: 1
1. Execute
2. Export to a YAML file with a name of your choice
Select an option (1 or 2): 1
networkpolicy.networking.k8s.io/web-deny-all-policy created
```

Figure 5.3.54. Set rule number 1 with Network management tool

Let us check to see if this file has been applied successfully

```
PS C:\Users\PC> kubectl exec -i -t --namespace=nginx1 test-duynam -- sh
/ # wget -qO- --timeout=2 http://web.tools
wget: download timed out
/ #
```

Figure 5.3.55. Query web service

b. Deny_all_traffic_from_other_namespaces

Purpose

This Network policy will deny all the traffic from other namespaces while allowing all the traffic coming from the same namespace the pod deployed to.

Step by step

Create file python: 04_Deny_all_traffic_from_other_namespaces.py

```
import yaml
import subprocess
import kubernetes
import json
```

```
with open("namespace.json", "r") as config_file:
    config = json.load(config_file)
namespace = config["namespace"]


network_policy = {
        "kind": "NetworkPolicy",
        "apiVersion": "networking.k8s.io/v1",
        "metadata": {
            "namespace": namespace,
            "name": "deny-from-other-namespace-to-namespace-" +
namespace
        },
        "spec": {
            "podSelector": {
                "matchLabels": {}
            },
            "ingress": [
                {
                    "from": [
                        {
                            "podSelector": {}
                        }
                    ]
                }
            ]
        }
    }

def apply_kubernetes_yaml(yaml_file_path):
    try:
        # The command you would normally type in the terminal
        cmd = ['kubectl', 'apply', '-f', yaml_file_path]

        # Execute the command
        result = subprocess.run(cmd, check=True, capture_output=True,
text=True)

        # Print the output from the command
        print(result.stdout)

    except subprocess.CalledProcessError as e:
        # If the command failed, it will raise this exception
        print("Error applying YAML:", e.stderr)
    except Exception as e:
        # Catch-all for any other exceptions
```

```
        print("An error occurred:", str(e))


while True:
        print("1. Execute")
        print("2. Export to a yaml file with a name of your choice")
        choice = input("Select an option (1 or 2): ")

        if choice == "1":
            # Limit_traffic_to_an_application_yaml =
yaml.dump(network_policy, default_flow_style=False)
            # with open("Limit_traffic_to_an_application_yaml", "w")
as temp_file:
            #
temp_file.write(Limit_traffic_to_an_application_yaml)
            #
apply_kubernetes_yaml('Limit_traffic_to_an_application_yaml')
            yaml_string = yaml.dump(network_policy,
default_flow_style=False)
            new_yaml_filename = f"deny-all-traffic-from-other-
namespaces.yaml"

            with open(new_yaml_filename, "w") as temp_file:
                temp_file.write(yaml_string)

            apply_kubernetes_yaml(new_yaml_filename)

            break
        elif choice == "2":
            filename = input("Enter the file name you want to save
(for example, data(.yaml)): ")
            with open(filename, 'w') as file:
                yaml.dump(network_policy, file)
            print(f"Saved to {filename}.yaml!")
            break
        else:
            print("Invalid selection. Please select again.")
```

Start a web service in namespace nginx1 by the command “*kubectl run web --namespace=nginx1 --image=nginx --labels="app=web" --expose --port=80*”

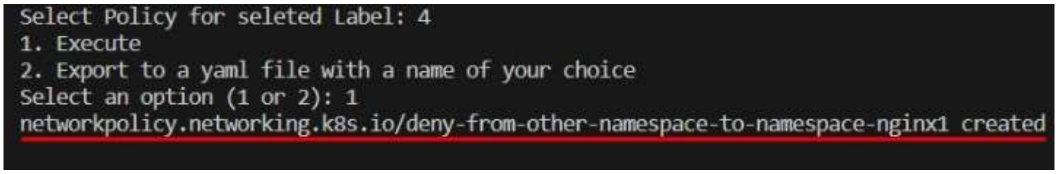
Query this web service from the tools namespace

```
PS C:\Users\[redacted] kubectl run test-tidiay --namespace=tools --rm -i -t --image=alpine -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- --timeout=2 http://web.nginx1
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

Figure 5.3.56. Query web service from different namespace

Apply file “04_Deny_all_traffic_from_other_namespaces.py for namespace=nginx1” to restrict the traffic with [a few steps]

```
===== Main MENU =====
1. View Network Menu
2. Display the Namespace list and select Namespace
3. Select Pod in Namespace
4. Exit

Select options: 2
List of Namespaces:
1. default
2. demo3
3. kube-node-lease
4. kube-public
5. kube-system
6. nginx1
7. prometheus
8. tools
Select Namespace (enter the corresponding number): 6

===== Main MENU =====
1. View Network Menu
2. Display the Namespace list and select Namespace
3. Select Pod in Namespace
4. Exit

Select options: 3
List of Labels :
1. {'app': 'nginx', 'pod-template-hash': '6ccd454948'}
2. {'app': 'web'}
Select Pod (enter the corresponding number): 2
Menu Network Policy:
01. Deny_all_traffic_to_an_application
02. Allow_traffic_to_an_application_from_all_namespaces
03. Deny_all_none_whitelisted_traffic_to_a_name_space
04. Deny_all_traffic_from_other_namespace
05. Deny_all_traffic_from_app_to_app
```



```
Select Policy for seleted Label: 4
1. Execute
2. Export to a yaml file with a name of your choice
Select an option (1 or 2): 1
networkpolicy.networking.k8s.io/deny-from-other-namespace-to-namespace-nginx1 created
```

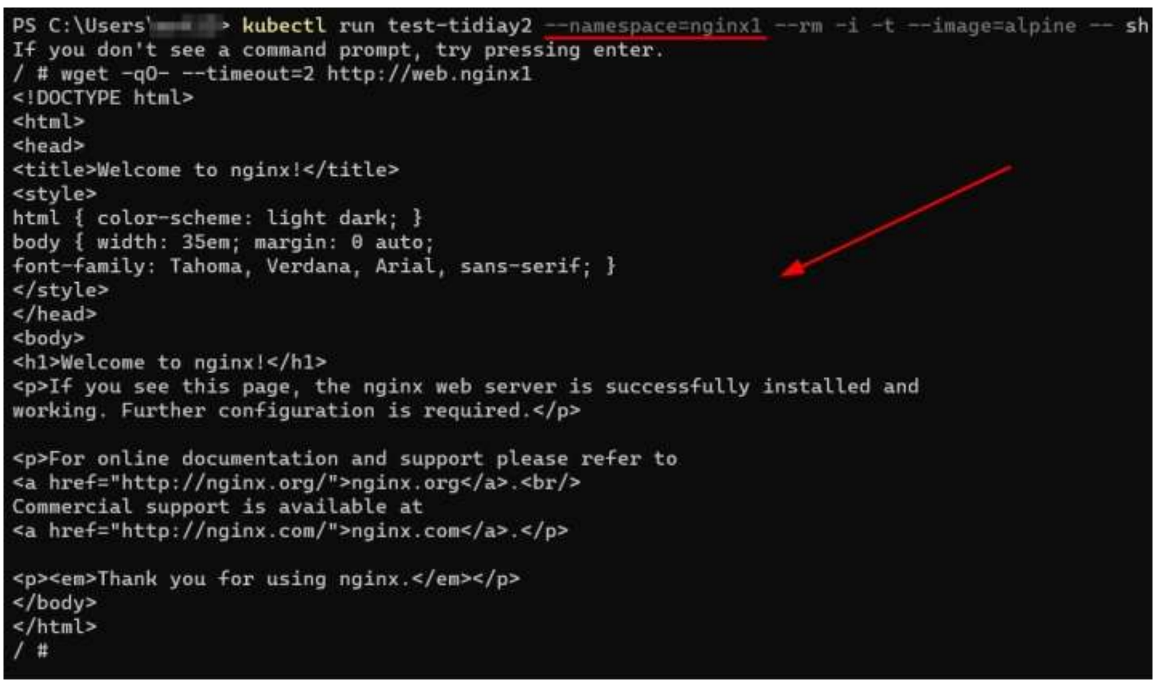
Figure 5.3.57. Apply rule number 4

Check the rule from query this service from namespace “tools” as a result it is fail

```
/ # wget -qO- --timeout=2 http://web.nginx1
wget: download timed out
/ #
```

Figure 5.3.58. Check rule status

Test again with another app in namespace nginx1 to see if this policy works



```
PS C:\Users\[redacted]> kubectl run test-tidiay2 --namespace=nginx1 --rm -i -t --image=alpine -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- --timeout=2 http://web.nginx1
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

Figure 5.3.59. Query web service in the same namespace

c. Allow_traffic_to_an_application_from_all_namespaces

Purpose

This Network policy will allow traffic from all pods in all namespaces to a particular application.

Step by step

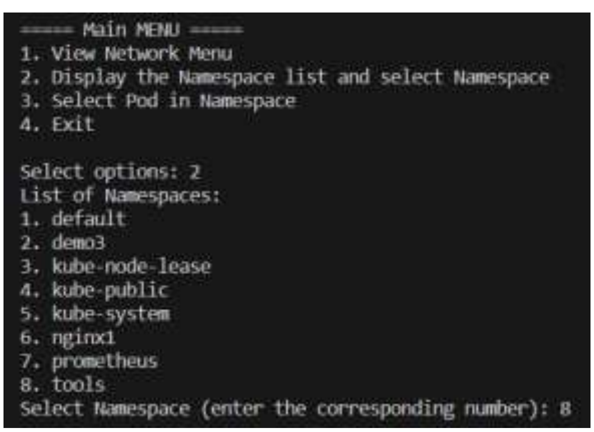
Create 02_Allow_traffic_to_an_application_from_all_namespaces.py template file written by python

```
import yaml
```

```python
import subprocess
import kubernetes
import json

with open("namespace.json", "r") as config_file:
    namespace_config = json.load(config_file)
    namespace = namespace_config["namespace"]
    selected_label = namespace_config.get("label", {}).get("app", "")

network_policy = {
    "kind": "NetworkPolicy",
    "apiVersion": "networking.k8s.io/v1",
    "metadata": {
        "namespace": namespace,
        "name": selected_label.lower() + "-allow-all-namespaces"
    },
    "spec": {
        "podSelector": {
            "matchLabels": {
                "app": selected_label,
            },
        },
        "ingress": [
            {
                "from": [
                    {"namespaceSelector": {}}
                ]
            }
        ]
    }
}

def apply_kubernetes_yaml(yaml_file_path):
    try:
        # The command you would normally type in the terminal
        cmd = ['kubectl', 'apply', '-f', yaml_file_path]

        # Execute the command
        result   =   subprocess.run(cmd,   check=True,   capture_output=True,
text=True)

        # Print the output from the command
        print(result.stdout)

    except subprocess.CalledProcessError as e:
        # If the command failed, it will raise this exception
        print("Error applying YAML:", e.stderr)
    except Exception as e:
        # Catch-all for any other exceptions
```

```python
        print("An error occurred:", str(e))

while True:
        print("1. Execute")
        print("2. Export to a yaml file with a name of your choice")
        choice = input("Select an option (1 or 2): ")

        if choice == "1":
            yaml_string = yaml.dump(network_policy,
default_flow_style=False)
            new_yaml_filename = f"Allow-traffic-to-an-application-from-
all-namespaces.yaml"

            with open(new_yaml_filename, "w") as temp_file:
                temp_file.write(yaml_string)

            apply_kubernetes_yaml(new_yaml_filename)

            break
        elif choice == "2":
            filename = input("Enter the file name you want to save (for
example, data(.yaml)): ")
            with open(filename, 'w') as file:
                yaml.dump(network_policy, file)
            print(f"Saved to {filename}.yaml!")
            break
        else:
            print("Invalid selection. Please select again.")
```

We restrict namespace “tools” with the rule number 4



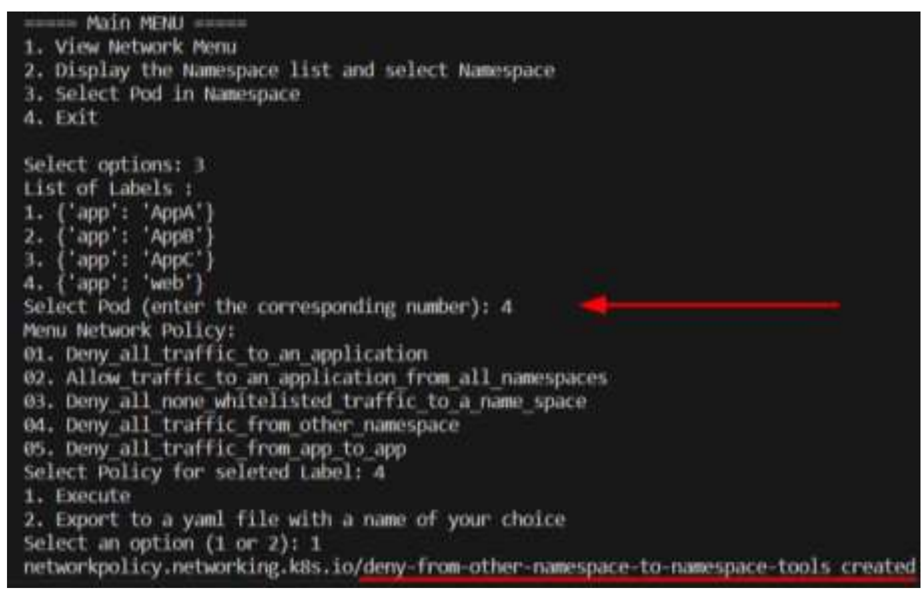


Figure 5.3.60. Apply rule number 4 for namespace “tools”

Apply template policy created before for app named “web” by our tool [with these steps]

```
===== Main MENU =====
1. View Network Menu
2. Display the Namespace list and select Namespace
3. Select Pod in Namespace
4. Exit

Select options: 3
List of Labels :
1. {'app': 'AppA'}
2. {'app': 'AppB'}
3. {'app': 'AppC'}
4. {'app': 'web'}
Select Pod (enter the corresponding number): 4
Menu Network Policy:
01. Deny_all_traffic_to_an_application
02. Allow_traffic_to_an_application_from_all_namespaces
03. Deny_all_none_whitelisted_traffic_to_a_name_space
04. Deny_all_traffic_from_other_namespace
05. Deny_all_traffic_from_app_to_app
Select Policy for seleted Label: 2
1. Execute
2. Export to a yaml file with a name of your choice
Select an option (1 or 2): 1
networkpolicy.networking.k8s.io/web-allow-all-namespaces created
```

Figure 5.3.61. Apply rule number 2 for app web

Check to see if this file has been applied successfully

```
/ # wget -qO- --timeout=2 http://web.tools
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

Figure 5.3.62. Query app web successfully from outside namespace “tools”

d. Deny_all_none_whitelisted_traffic_to_a_name_space

Purpose

This is a fundamental policy, blocking all cross-pod networking other than the ones whitelisted via the other Network Policies you deploy.

Step by step

Create 03_Deny_all_none_whitelisted_traffic_to_a_name_space.py file

```python
with open("namespace.json", "r") as config_file:
    config = json.load(config_file)

namespace = config["namespace"]
selected_label = config.get("label", {}).get("app", "")

network_policy = {
    "kind": "NetworkPolicy",
    "apiVersion": "networking.k8s.io/v1",
    "metadata": {
        "name": f"{selected_label.lower()}-deny-all-none-whitelisted",
        "namespace": namespace
    },
    "spec": {
        "podSelector": {},
        "policyTypes": ["Ingress"],
        "ingress": [
            {
                "from": [
                    {
                        "podSelector": {
                            "matchLabels": {
                                # Add labels for whitelisted pods
                                "whitelist-label-key": "whitelist-label-value"
                            }
                        }
                    }
                ]
            }
        ]
    }
}
```

Figure 5.3.63. Deny_all_none_whitelisted_traffic_to_a_name_space.py file

Apply file 03_Deny_all_none_whitelisted_traffic_to_a_name_space.py for namespace tools and app web [with these steps]

```
Select options: 2
List of Namespaces:
1. default
2. demo3
3. kube-node-lease
4. kube-public
5. kube-system
6. nginx1
7. prometheus
8. tools
Select Namespace (enter the corresponding number): 8

===== Main MENU =====
1. View Network Menu
2. Display the Namespace list and select Namespace
3. Select Pod in Namespace
4. Exit

Select options: 3
List of Labels:
1. {'app': 'AppA'}
2. {'app': 'AppB'}
3. {'app': 'AppC'}
4. {'app': 'web'}
Select Pod (enter the corresponding number): 4
Menu Network Policy:
01. Deny_all_traffic_to_an_application
02. Limit_traffic_to_an_application
03. Deny_all_none_whitelisted_traffic_to_a_name_space
04. Deny_all_traffic_from_other_namespace
05. Deny_all_traffic_from_app_to_app
Select Policy for: 3
1. Execute
2. Export to a yaml file with a name of your choice
Select an option (1 or 2): 1
networkpolicy.networking.k8s.io/web-deny-all-none-whitelisted created
```

Figure 5.3.64. Apply rule number 3 for app web

5.3.3.3. Case 3: Maintain system availability

Purpose

One of the key goals of Software-Defined Networking (SDN) is to improve system performance by minimizing downtime and ensuring high availability for the entire system. SDN provides the ability to automatically direct network traffic to the appropriate servers, with Scale Up and Scale Down features creating load balancing between servers and system flexibility.

Model

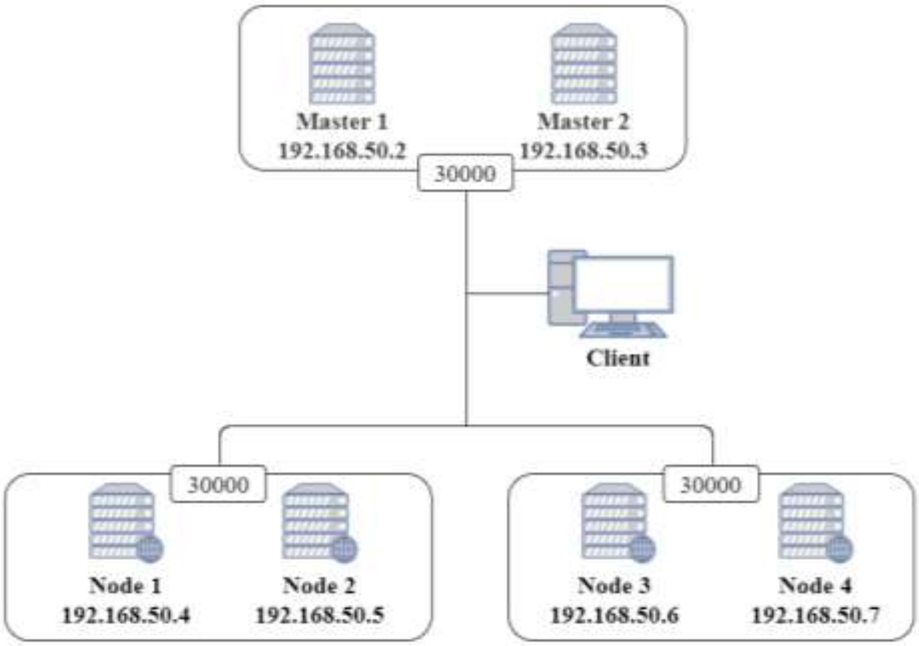


Figure 5.3.65. Case 3 model

Step by step

Create a YAML file as below to create a Nginx pod. Then apply it to the system and check pod status

```
apiVersion: v1
kind: Namespace
metadata:
  name: demo3
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: demo3
spec:
  minReadySeconds: 10
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
        resources:
          limits:
            cpu: "0.5"
            memory: "256Mi"
          requests:
            cpu: "0.1"
            memory: "64Mi"
        readinessProbe:
          httpGet:
            path: /
            port: 80
          initialDelaySeconds: 15
          periodSeconds: 5
          failureThreshold: 3
```

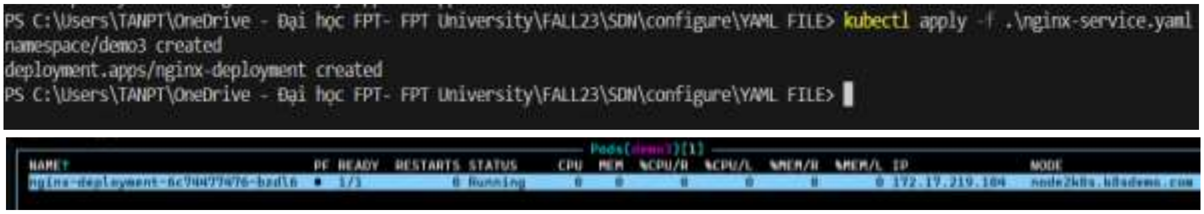


Figure 5.3.66. Create Nginx pod

The Nginx pod has been created in Node 2.

We configure “NodePort” service to Nginx as below and apply to the system

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
  namespace: demo3
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30000
  type: NodePort
```

Service has been created

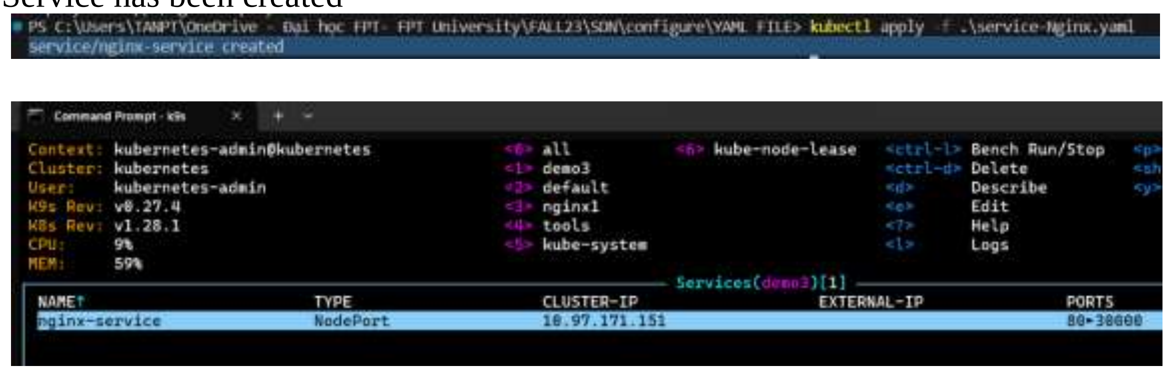


Figure 5.3.67. Create service for Nginx pod by YAML file

Press Enter to see the information of this service

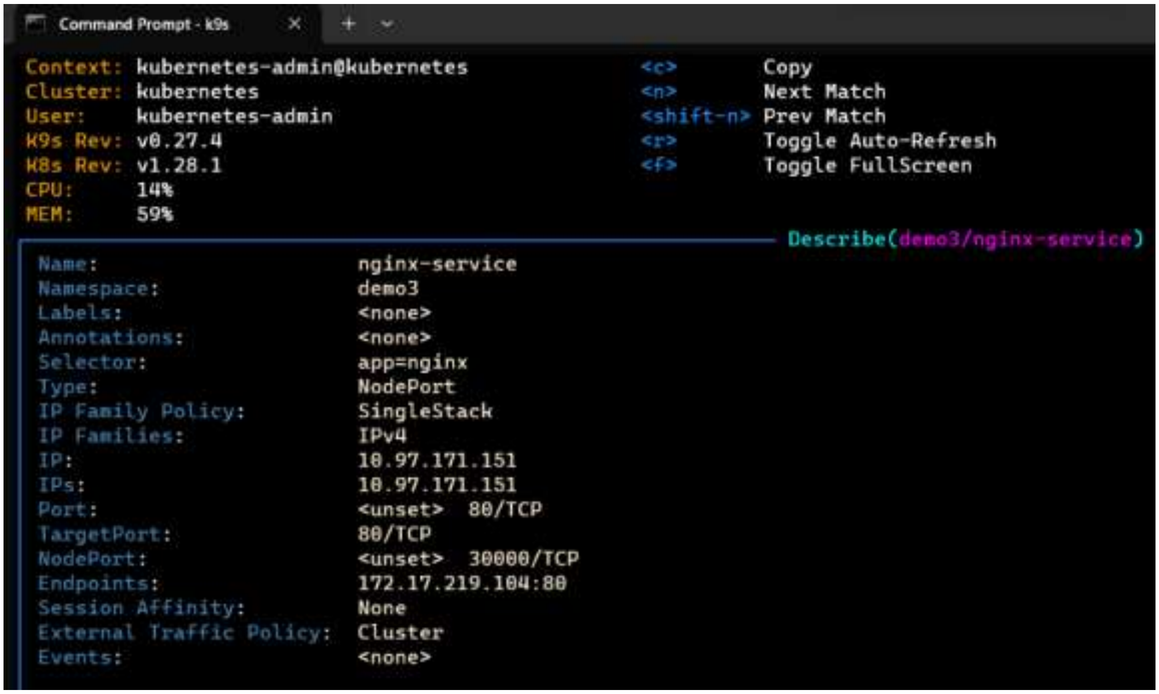


Figure 5.3.68. Information of Nginx service

We can access this Nginx pod from any Node in the system thanks to this service.

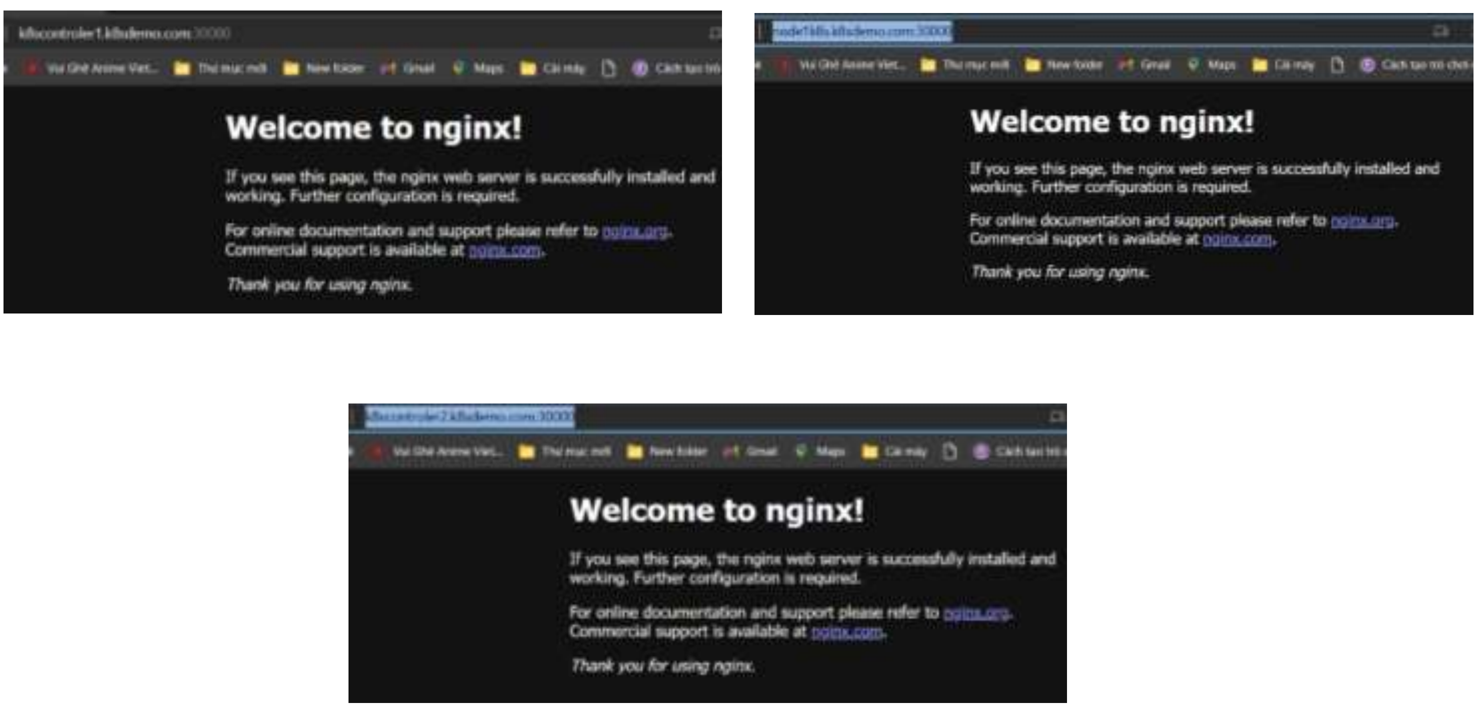


Figure 5.3.69. Access Nginx service

Create additional HPA using YAML file as below so that the system can automatically expand according to needs. We need the system to always have 2 apps running and a maximum of 8 apps.

apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-hpa
  namespace: demo3
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
minReplicas: 2
maxReplicas: 8
metrics:
  - type: Resource
    resource:
      name: cpu
      target :
        type: Utilization
averageUtilization: 50

Apply successfully

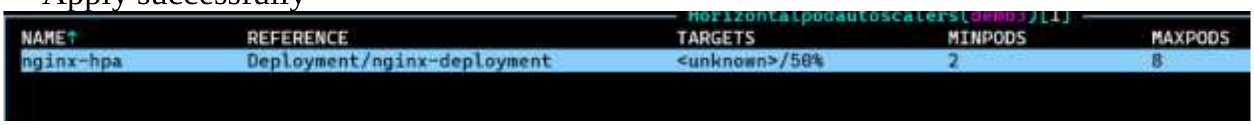


Figure 5.3.70. Apply HPA

Check the number of pods to verify that they are created according to the minimum requirement of 2

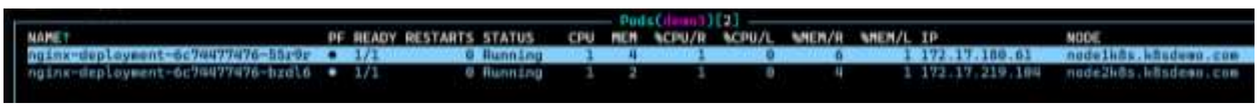


Figure 5.3.71. Minimum number of pods

Check the service information again to see the difference

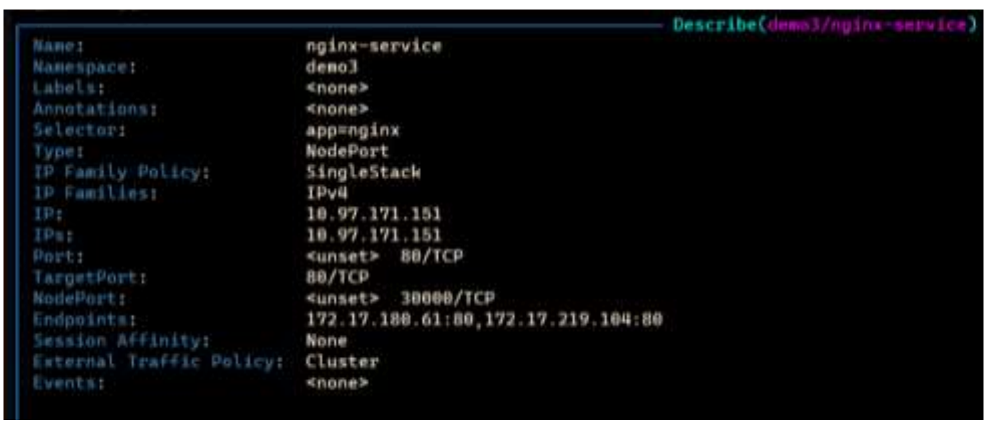


Figure 5.3.72. Information of Nginx service after applying HPA

Scenario 1: We will perform an Internet outage on Node 1

Figure 5.3.73. Internet outage on Node 1

Access Node 1 to view the system's automated check status

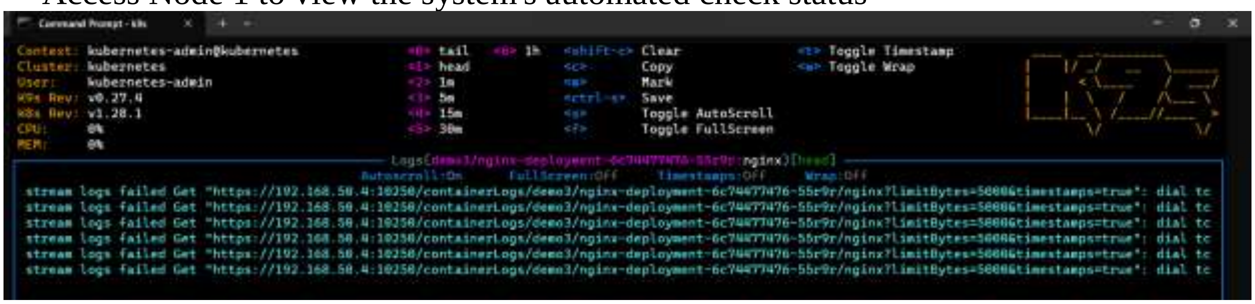
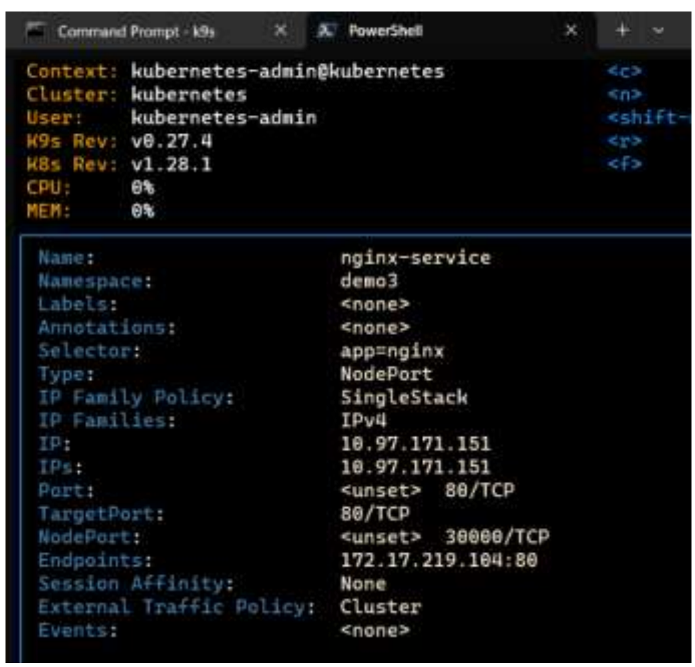


Figure 5.3.74. Status of Node 1

Go to the service description page after Node 1 checks its status more than 3 times. We see that IP Node 1 was automatically deleted but Node 1 is still in the system



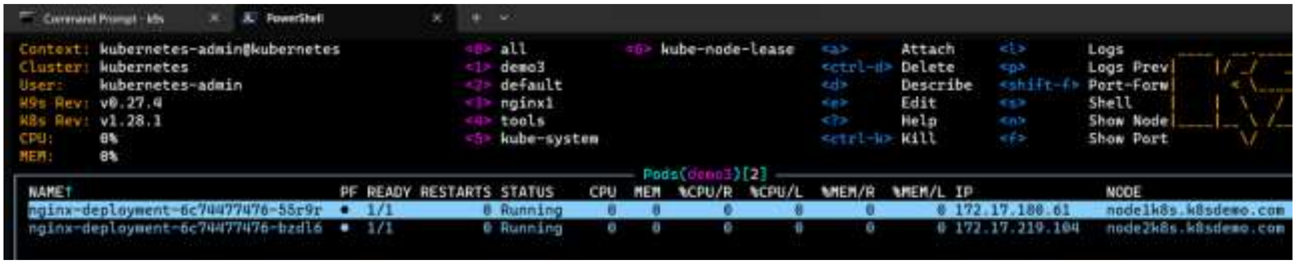


Figure 5.3.75. Node 1 state in the system after self-check more than 3 times

Nginx on Node 1 was automatically deleted after 5 minutes Node 1 crashed and the system created a new Node to replace it.

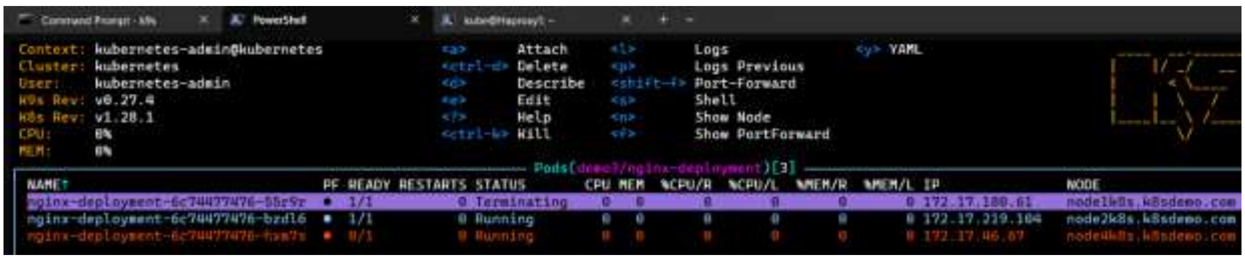


Figure 5.3.76. Node 1 was deleted after 5 minutes

Scenario 2: We will perform a Denial of Services (DoS) using ab tool

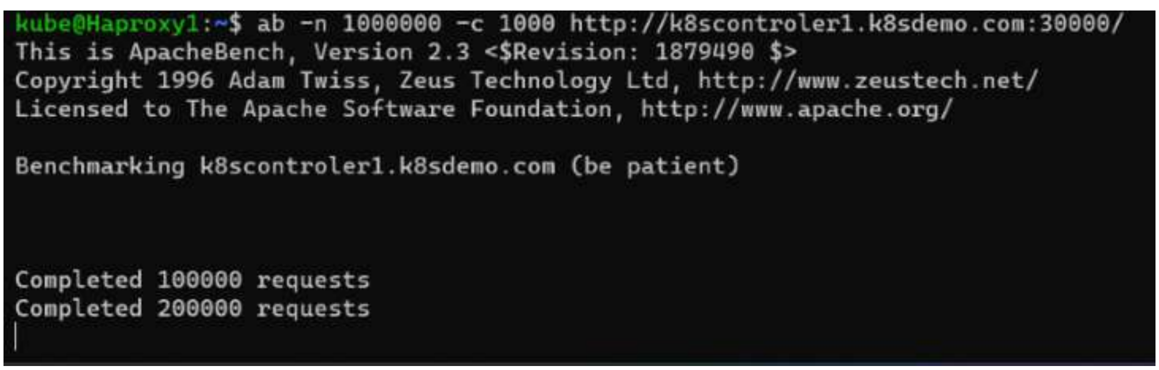


Figure 5.3.77. DoS the system by Apache Bend mark

The system notices unusual traffic so it pushes the Nginx Pod number up to accommodate that traffic

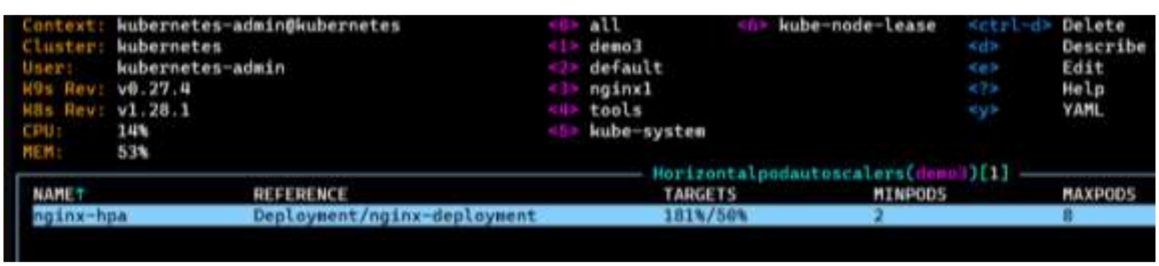


Figure 5.3.78. Number of Nginx pods before perform DoS

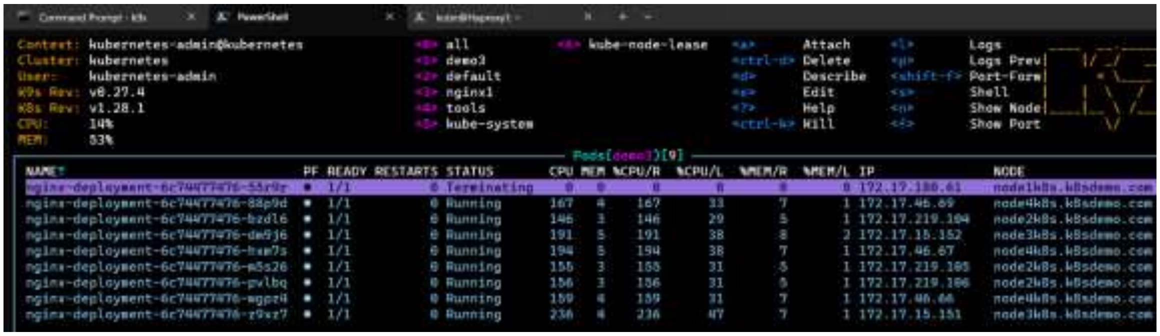


Figure 5.3.79. Number of Nginx pods after perform DoS

And the service description page also changes in the “Endpoints” field

Figure 5.3.80. Nginx information after perform DoS

When the CPU percentage increases more than 50% due to DoS, we will get a high CPU alert

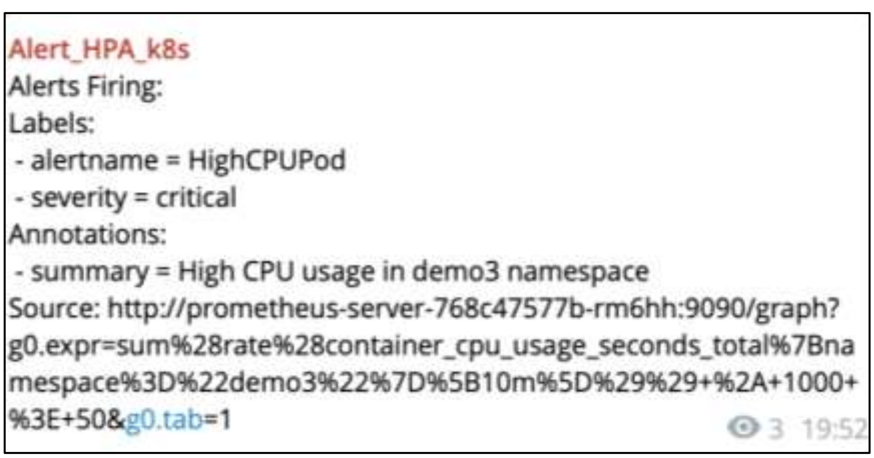


Figure 5.3.81. Alert High CPU pod

There is one more thing we do in this demo, which is to use it with the Haproxy system to enhance the availability and flexibility of the system. This ensures that service is still provided even if a failure occurs in one part of the system. Helps adapt to changes in system needs or size.

We do an additional configure for 2 server HAProxy

```
frontend nginx
  bind 192.168.50.10:80
  mode tcp
  option tcplog
  default_backend nginx-nodes
backend nginx-nodes
 mode tcp
 option tcp-check
 balance leastconn
 option redispatch
 server node1s node1k8s.k8sdemo.com:30000 check
 server node2s node2k8s.k8sdemo.com:30000 check
#server node2s node2k8s.k8sdemo.com:31319 check
 server node3s node3k8s.k8sdemo.com:30000 check
 server node4s node4k8s.k8sdemo.com:30000 check
 server control1 k8scontroler1.k8sdemo.com:30000 check
 server control2 k8scontroler2.k8sdemo.com:30000 check
```

Figure 5.3.82. Additional configuration for HAProxy

After that, restart the HAProxy by the command "*sudo Systemctl restart haprox*" and user can access by the IP 192.168.50.10 (VIP)

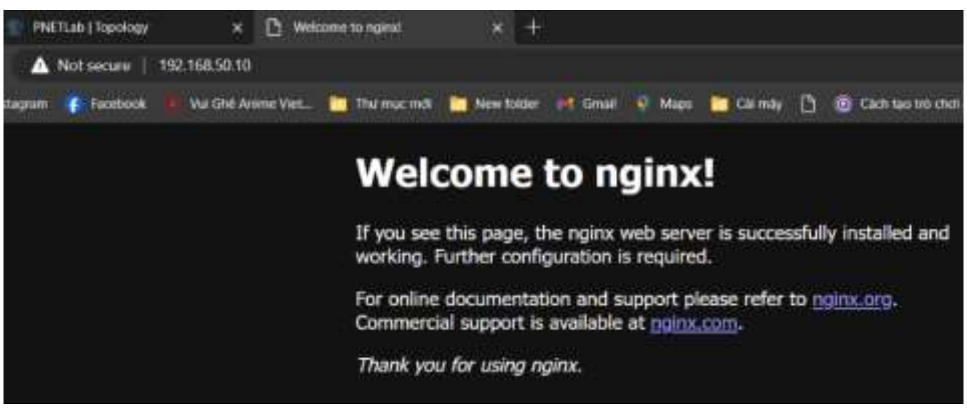


Figure 5.3.83. Nginx web access by VIP

We can see the statistics report of HAProxy

Figure 5.3.84. Status report of HAProxy

CHAPTER 6: VALIDATING DOCUMENTATION

6.1. Repeat Risk Assessment Process

6.1.1. Check and Add for a New Critical Asset Appeared

Software-Defined Networking, as a dynamic and programmable infrastructure, can often witness the addition of new assets. An asset is any resource, service, or component that is a part of the network, contributing to its functionality and value. An asset becomes 'critical' when its confidentiality, integrity, or availability has a major influence on the network's overall operation.

When a new critical asset is added to the SDN:

- **Identification:**

Start by identifying the new asset. It may include new nodes, applications, or even a new software-defined service that has been introduced into the network.

- **Assessment:**

Evaluate the criticality of the asset based on factors like its role in the network, the sensitivity of the data it handles, and its importance to the network's overall functionality.

- **Update Inventory:**

Update your asset inventory to include the new asset. This inventory should provide a comprehensive view of all assets on your SDN, including their properties and how they interact with other components on the network.

- **Risk Assessment:**

Conduct a risk assessment for the new critical asset. This involves identifying potential threats and vulnerabilities that could affect the asset, analyzing these risks for their potential impact and likelihood, and then prioritizing them based on these factors.

- **Mitigation Strategy:**

Develop and implement strategies to mitigate the identified risks. This might involve measures like improving the asset's security protocols, integrating it into the network's monitoring systems, or training personnel on how to handle the asset securely.

- **Documentation:**

Document the entire process and update existing documentation to include information about the new critical asset and the risks associated with it. This not only helps to maintain an accurate record of your network's security posture but also provides a resource for future reference during subsequent risk assessments.

6.1.2. Check for a Change of IT Environment

- **Regular Monitoring**

Keep an eye on your IT environment for any changes. Monitoring tools can help detect alterations in software, hardware, configurations, network traffic, and user behavior.

- **Change Identification**

Once a change is detected, identify its nature and scope. For instance, a change could be an update to an existing application, the introduction of new devices into the network, or a modification in network configurations.

- **Impact Analysis**

Analyze the potential impact of the identified change on the network's security and functionality. Some changes might expose the network to new vulnerabilities or even enhance its security.

- **Risk Assessment**

Conduct a risk assessment for the change, following the same process as for new critical assets. This includes identifying potential threats and vulnerabilities, analyzing their likelihood and potential impact, and then prioritizing them.

- **Mitigation Strategy**

Based on the risk assessment, update your mitigation strategies to address the risks associated with the change. This could involve updating security protocols, altering configurations, or deploying additional security tools.

- **Documentation**

Document the change, its impact, the risk assessment process, and the updated mitigation strategies. This documentation not only serves as a record of changes in your IT environment but also aids in future risk assessments and audits.

6.2. Risk Analysis

6.2.1. Qualitative Analysis

- **User Experience**

Investigate the experiences and perceptions of users who interact with SDN systems. This can involve collecting qualitative data through interviews, focus groups, or surveys to understand their satisfaction, challenges, and suggestions for improvement. Explore topics such as ease of use, performance, reliability, and the impact of SDN on their daily tasks.

- **Network Administrator Perspectives**

Examine the viewpoints of network administrators or IT professionals responsible for managing SDN deployments. Gain insights into their experiences with configuration, monitoring, troubleshooting, and overall network management. Explore their perceptions of the benefits and challenges associated with SDN, such as network programmability, automation, security, and scalability.

- **Organizational Impact**

Analyze the organizational impact of implementing SDN. This can involve exploring how SDN affects business processes, collaboration among different teams, or the overall network architecture. Investigate the changes in roles and responsibilities, the challenges faced during the transition, and the benefits achieved after adopting SDN.

- **Performance Evaluation**

Assess the SDN model's performance in network latency, throughput, scalability, and reliability. Use qualitative methods such as capturing network traces or conducting experiments to understand the performance characteristics under different scenarios. Gain insights into the effectiveness of traffic engineering, load balancing, or quality of service (QoS) mechanisms provided by the SDN model.

- **Security and Privacy Considerations**

Investigate the perceptions and concerns related to security and privacy in SDN environments. Explore how network administrators and users perceive the security features and capabilities of SDN, as well as potential vulnerabilities or risks associated with the technology. Analyze their attitudes towards data privacy, access control, and the ability to enforce security policies in an SDN context.

6.2.2. Quantitative Analysis

- **Network Performance Metrics**

Measure and analyze performance metrics such as network latency, throughput, packet loss, and jitter in an SDN environment. Use tools like network monitoring software or SDN controllers to collect relevant data and perform statistical analysis to assess the performance of the network.

- **Traffic Engineering and Load Balancing**

Quantitatively evaluate the effectiveness of traffic engineering and load balancing techniques in SDN. Collect data on network traffic patterns, flows, and utilization. Analyze the distribution of traffic across different paths, the efficiency of load balancing algorithms, and the impact on network performance and resource utilization.

- **Resource Allocation and Scalability**

Quantify the ability of an SDN model to allocate network resources dynamically and efficiently. Measure resource utilization, such as CPU, memory, and bandwidth, under different traffic loads. Assess the scalability of the SDN solution by analyzing the performance impact as the network grows in terms of the number of switches, controllers, or connected devices.

- **Security Analysis**

Quantitatively analyze the security aspects of SDN, such as network intrusion detection, anomaly detection, or DoS (Denial of Service) mitigation. Collect relevant security-related data, such as network traffic logs, and apply statistical techniques to identify patterns, detect anomalies, or evaluate the effectiveness of security mechanisms.

- **Energy Efficiency:** Quantify the energy consumption and efficiency of SDN deployments. Measure power consumption at different network components, such as switches or controllers, and analyze the impact of SDN on energy savings. Compare energy efficiency metrics between traditional network architectures and SDN to assess the potential benefits of adopting SDN in terms of reduced power consumption.
- **QoS (Quality of Service) Evaluation:** Quantitatively assess the QoS provided by SDN in terms of network performance and user experience. Collect data on latency, jitter, and packet loss for distinct types of traffic or service classes. Analyze the adherence to defined QoS policies, the impact of SDN on meeting QoS requirements, and the effectiveness of QoS mechanisms implemented in the SDN model.

6.2.3. Provable Risk Mitigation

- **Risk Assessment**

Conduct a comprehensive risk assessment to identify and evaluate potential risks associated with SDN deployment. This involves analyzing the impact and likelihood of various risks, such as security vulnerabilities, performance bottlenecks, or configuration errors. By conducting a thorough risk assessment, you can proactively address and mitigate the identified risks.

- **Security Measures**

Implement robust security measures in the SDN environment to mitigate security risks. This can include features such as access controls, authentication mechanisms, encryption, intrusion detection and prevention systems (IDPS), and security monitoring tools. Regularly assess the effectiveness of these security measures through audits, penetration testing, and vulnerability assessments.

- **Redundancy and Resilience**

Implement redundancy and resilience mechanisms in the SDN infrastructure to mitigate the risks of single points of failure. This can involve deploying redundant controllers, switches, or network paths to ensure continuous network operation even in the event of failures. Validate the effectiveness of these mechanisms through testing and monitoring.

- **Traffic Engineering and QoS Mechanisms**

Utilize SDN's traffic engineering capabilities to optimize network resource utilization, load balancing, and quality of service (QoS) provisioning. By effectively managing traffic flows and prioritizing critical applications or services, you can mitigate risks associated with performance degradation or service disruptions. Monitor and measure network performance to ensure the desired QoS levels are achieved.

- **Change Management and Configuration Control**

Establish robust change management and configuration control processes for SDN deployments. Implement version control, configuration baselining, and change tracking mechanisms to mitigate risks introduced by unauthorized or erroneous changes. Regularly review and validate configuration changes to ensure adherence to policies and best practices.

- **Monitoring and Analytics**

Deploy monitoring and analytics tools to continuously monitor the SDN environment and detect anomalies or potential risks. Utilize network telemetry, flow data, and log analysis to identify security incidents, performance bottlenecks, or abnormal behavior. Proactively address the identified risks based on the insights gained from monitoring and analytics.

6.3. Advantages and Disadvantage of Solutions

6.3.1. Advantages

SDN has completely changed the network world by driving several advantages over the traditional network:

- Efficient Resource Management: SDN offers the flexibility and efficiency needed for effective network resource management.
- Flexibility and Automation: SDN enables flexible network configurations and automates deployment and management processes.
- Integrated Network Services: Seamless integration of network services such as security system, load balancing, and monitoring.

In our project, we gave an overview of the SDN concept and how it can adapt to the business environment to bring benefits and improve the quality of work performance. Additionally, the advantages make SDN a different approach to network management

6.3.2. Disadvantages

Beside the advantages we notice above, SDN still have some drawback that should be focus when deployment it:

- Deployment Costs: Implementing SDN infrastructure may involve a substantial upfront investment.
- Steep Learning Curve: Learning and implementing SDN can require considerable time and resources from administrators.

In this project, we still do not have the opportunity to develop in a physical environment, everything is virtual. Additionally, our self-grown network management tool has few policy templates and an unattractive interface.

6.4. Future Plan

In the future, we want to study deep dive into SDN concept that can help us develop a specific system that adapt to our plan:

- SDN for Enterprise Networks: Providing comprehensive network solutions to optimize performance and costs for businesses.
- Integration with Emerging Networks: Combining SDN with new networks to leverage high speeds and extensive connectivity, particularly in applications like future cloud services.

REFERENCES

1	*Software-Defined Networking: Challenges and research opportunities for Future Internet* Authors: *Akram Hakiri, Aniruddha Gokhale, Pascal Berthou, Douglas C. Schmidt, Gayraud Thierry*
2	*A comprehensive review on SDN architecture, applications and major benefits of SDN* Authors: *V. Thirupathi, Ch. Sandeep, S. Naresh Kumar, P. Pramod Kumar*
3	*Security Issues in Software Defined Networking (SDN): Risks, Challenges and Potential Solutions* Authors: *Maham Iqbal, Farwa Iqbal, Fatima Mohsin, Dr. Muhammad Rizwan, Dr. Fahad Ahmad*
4	*Kubernetes: Documentation*
5	*Understanding the Security Implications of Kubernetes Networking*
6	*Cisco Annual Internet Report (2018–2023) White Paper*
7	*How to Install Kubernetes Cluster on Ubuntu 22.04*
8	*Kubernetes Network Policy Recipes*
9	*Kubernetes Network Policy Tutorial*
10	*Kubernetes Client Libraries*
11	*Kubernetes Python Client*
12	*What is Software-Defined Networking?*