Summer Internship Report
On
"HARDENING THE NETWORK "

(IT446 – Summer Internship - 2)

Prepared by

Siddharth Singh (20IT145)

Under the Supervision of

Assistant Professor

Nishat Shaikh

Submitted to

Charotar University of Science & Technology (CHARUSAT)
for the Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology (B.Tech.)
for Semester 7

Submitted at

SMT. KUNDANBEN DINSHA PATEL DEPARTMENT OF INFORMATION TECHNOLOGY

Chandubhai S. Patel Institute of Technology (CSPIT)

Faculty of Technology & Engineering (FTE), CHARUSAT

At: Changa,Dist: Anand, Pin: 388421.

July,2022

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

Accredited with Grade A by NAAC
Accredited with Grade A by KCG

CERTIFICATE

This is to certify that the report entitled "**HARDENING THE NETWORK** is a bonafied work carried out by **Siddharth Singh (20IT145)** under the guidance and supervision of **Assistant Professor Nishat Shaikh** for the subject **Summer Internship – I (IT346)** of 7th Semester of Bachelor of Technology in **Department of Information** at Chandubhai S. Patel Institute of Technology (CSPIT), Faculty of Technology & Engineering (FTE) – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate herself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred by the examiner(s).

Under the supervision of,

Sanket Suthar
Assistant Professor
Smt. Kundanben Dinsha Patel Department of Information Technology
CSPIT, FTE, CHARUSAT, Changa, Gujarat

Dr. Purvi Prajapati
Assistant Professor
Smt. Kundanben Dinsha Patel Department of Information Technology
CSPIT, FTE, CHARUSAT, Changa, Gujarat

Dr. (Prof.) Parth Shah
Head of Department(IT)
CHARUSAT, Changa, Gujarat

Vraj Gohil
Director
DevSquirrel Technologies Pvt.Ltd

ACKNOWLEDGEMENT

First I would like to thank Raj Bheda, Director, DevSquirrel Technologies Pvt. Ltd. And Vraj Gohil, Director, DevSquirrel Technologies Pvt. Ltd. For giving me the opportunity to do and internship within the organization. Also, I would like to thank Dr. Parth Shah and Sanket Suthar Sir in helping me to give permission for intern at Infosense Private Limited.

I am so grateful that I got this opportunity to showcase my skillset. It helped me to increase my experience of working with Azure technology in real Environment and live database and to improve my knowledge regarding Cybersecurity.

Finally, I would like to express my special thanks to my families and friends helping me in all aspects and appreciate me to spend my time in the work during my internship time.

Thanks,
Siddharth Singh (20IT145)
Chandubhai.S.Patel Institute of Technology

ABSTRACT

I have completed my internship at Infosense Private limited as an Cybersecurity Intern. Infosense Private Limited is a technology and information technology services firm that specializes in providing a wide range of solutions to clients in a variety of industries. Services such as software development, IT consulting, system integration, data analytics, cloud computing, cybersecurity, and digital transformation may be provided by the organization. Infosense Global may serve both domestic and foreign clients, and it may serve organizations of all sizes, from startups to huge corporations. This internship report provides a detailed overview of the steps to harden the internal and external network of Infosense Private Limited using Azure Kubernetes Service (AKS) for cloud-native app development and deployment. This report emphasizes the importance of securing both the internal and external network components to ensure the confidentiality, integrity, and availability of resources. This report covers various aspects of network hardening, including VNet configuration, NSGs, RBAC, Kubernetes RBAC, container security, TLS implementation, and monitoring for security insights.

Table of Contents

List of Figures

DESCRIPTION OF COMPANY

Infosense Private Limited is a technology and information technology services firm that specializes in providing a wide range of solutions to clients in a variety of industries. Services such as software development, IT consulting, system integration, data analytics, cloud computing, cybersecurity, and digital transformation may be provided by the organization. Infosense Global may serve both domestic and foreign clients, and it may serve organizations of all sizes, from startups to huge corporations. The company's focus might be on using emerging technology and industry best practices to provide clients with creative and scalable solutions. Its seasoned team of specialists, which includes software engineers, data scientists, cybersecurity experts, and project managers, collaborates to address the specific needs of each customer. Customer satisfaction, quality assurance, and on-time project delivery are most likely important to Infosense Global.

CHAPTER 1 INTRODUCTION

1.1 Overview of Cloud Native App Development and Deployment:

Cloud native application can be described as one that runs on cloud infrastructure and is built around design Cloud native is a term that is invoked often but seldom defined beyond saying ``we built it in the cloud'' as opposed to "on-prem". However, there is now an emerging consensus around key ideas and informal application design patterns that have been adopted and used in many successful cloud applications. In this introduction, we will describe these cloud native concepts and illustrate them with examples. We will also look at the technical trends that may give us an idea about the future of cloud applications. We begin by discussing the basic properties that many cloud native apps have in common. Once we have characterized them, we can then describe how these properties emerge from the technical design patterns. The most frequently cited properties of cloud native include the following.

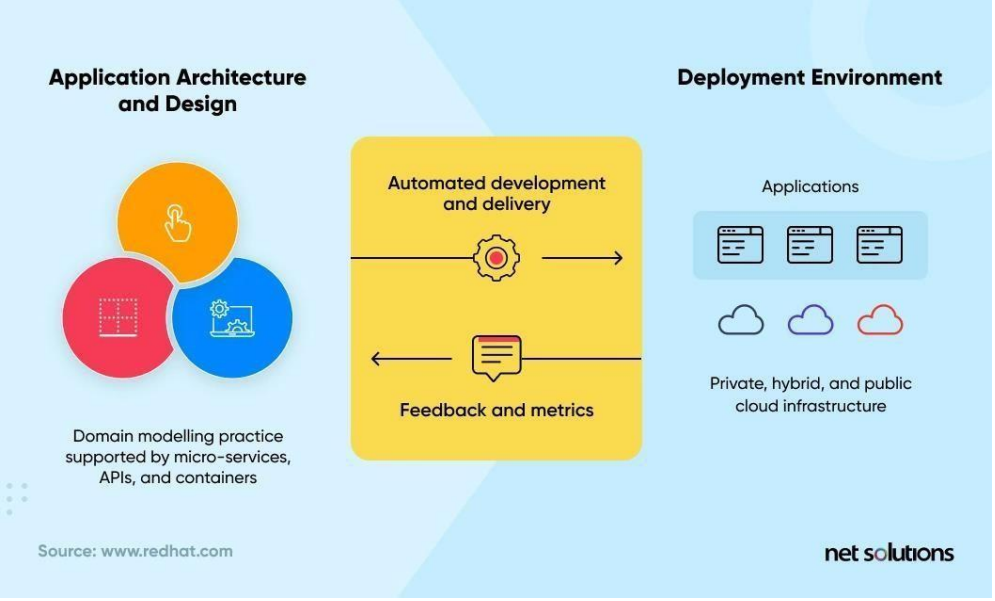


Figure 1.1 Development and deployment of cloud native applications

1. Cloud native applications often operate at global scale. While an ordinary website can be accessed anywhere that the Internet is unblocked, true global scale implies much more. It implies that the application's data and services are replicated in local data centers so that interaction latencies are minimized. It implies that the consistency models used are robust enough to give the user confidence in the integrity of the application.

2. Many cloud native applications must scale well with thousands of concurrent users. This isanother dimension of parallelism that is orthogonal to the horizontal scaling of data required for global-scale distribution and it requires careful attention to synchronization and consistency in distributed systems.

3. They are built on the assumption that infrastructure is fluid and failure is constant. This concept is the foundation for the original design of the Internet protocols, but applications that were built for execution on a single PC, mainframe or supercomputer assume the underlying OS and hardware are rock solid. Consequently, when these applications are portedto the cloud, they fail with the first hiccup in the datacenter or network. Even when the failure

rate for hardware or networks is extremely small, the law of large numbers guarantees that when you attempt global scale something is always broken or about to break.

4. Cloud native applications are designed so that upgrade and test occur seamlessly without disrupting production. While not every cloud native application is intended for use by a million concurrent users spread over the planet, most are designed for continuous operation. Monitoring critical instruments which must not be left unattended is one example. But all applications need to be upgraded, without interrupting normal operations, and the upgrades then need to be ested, so the architecture of the application must allow this.

5. Security and privacy are not an afterthoughts. As we shall see, many cloud native applications are built from many small component and these components must not hold sensitive credentials. Firewalls are not sufficient because access controls need to be managed at multiple levels in the application. Security must be part of the underlying application architecture.

1.2 Purpose of Internship:

Network hardening refers to the processes that minimize security gaps within a cybersecurity infrastructure. Network hardening standards help guide the processes used in optimizing network security across your organization's cybersecurity infrastructure.

Network hardening helps mitigate security risks related to:
Vulnerabilities in network configurations and devices,
Non-essential services running on your IT systems,
Beyond networks, hardening can be applied to any component within your infrastructure.

For example, the IT system components that hardening can secure include but are not limited to
Applications used to provide network or system access to end-users, such as:
1.Web applications
2.Mobile applications
3.Hardware used to host networks and software, such as:
4.Physical servers
5.Desktops
6.Mobile devices
7.Databases, such as those used to store: Sensitive user data, System files.

CHAPTER 2 NETWORK ARCHITECTURE AND AZURE KUBERNETES SERVICE INTEGRATION

2.1 Description of the Existing network Architecture:

The goal of network architecture is to ensure efficient and reliable communication between devices while optimizing performance, security, and scalability. It involves planning, designing, implementing, and managing the network infrastructure to meet the specific requirements and objectives of an organization.

Local Area Network (LAN): The Local Area Network is the internal network within the company's premises, connecting all devices like computers, servers, printers, and other networked devices. It is the foundation of the company's internal communication and data sharing.

Wide Area Network (WAN): The Wide Area Network connects different office locations or branches of the company. It allows for secure and efficient communication between geographically distributed sites and enables the centralization of certain resources and services.

Core Switches and Routers: Core switches and routers form the backbone of the network architecture. They provide high-speed connectivity between different parts of the network and handle the traffic efficiently. They are typically placed at the center of the network and facilitate communication between various switches and devices.

Distribution Switches: Distribution switches are used to connect the core switches to the access switches and other network devices. They help in segmenting the network into different broadcast domains and VLANs (Virtual LANs).

Access Switches: Access switches connect end-user devices like computers, IP phones, and printers to the network. They are usually placed at the edge of the network and provide connectivity to individual workstations.

Firewall: A firewall is a critical component of network security. It acts as a barrier between the company's internal network and the external internet, controlling incoming and outgoing traffic based on predefined security policies.

VPN (Virtual Private Network): VPNs are used to securely connect remote workers and branch offices to the company's network over the internet. This allows employees to access internal resources securely from outside the office.

Load Balancers: Load balancers distribute incoming network traffic across multiple servers to ensure efficient utilization of resources and high availability of services.

Intrusion Detection and Prevention Systems (IDPS): IDPS monitors network traffic for suspicious activities and potential security breaches. It helps in detecting and preventing various types of cyber threats.

Network Security Devices: Various security devices, such as proxy servers, content filters, and anti-malware appliances, are employed to enhance network security and protect against cyber threats.

Network Storage: Network-attached storage (NAS) and storage area networks (SAN) are used tostore and manage large volumes of data shared across the organization.

Cloud Integration: Companies often integrate cloud services into their network architecture toleverage the benefits of cloud computing, such as scalability, flexibility, and cost-effectiveness.

2.2 Integration of Azure Kubernetes Service with the Network

The integration of Azure Kubernetes Service (AKS) with the network is a crucial step in deploying and managing containerized applications in the Azure cloud environment. It involves connecting the AKS cluster to the existing on-premises network or other cloud services to enable seamless communication between containers and other resources. The integration ensures that AKS can access required resources and services while maintaining network security and data privacy.

Here's an overview of how Azure Kubernetes Service is integrated with the network:

Virtual Network (VNet) Configuration: When creating an AKS cluster, you can choose to deploy it within an existing Azure Virtual Network (VNet) or create a new one. Using an existing VNet allows you to integrate AKS with other Azure services and resources that reside in the same virtual network, ensuring they can communicate efficiently.

Subnet Allocation: Within the chosen VNet, you define subnets where the AKS master nodes and worker nodes will be placed. These subnets are dedicated to AKS and should have enough IP addresses to accommodate the desired number of nodes and pods.

Network Security Groups (NSGs): Network Security Groups can be applied to the AKS subnets to control inbound and outbound traffic. NSGs act as a firewall, allowing you to define rules to permit or deny specific types of network traffic, such as SSH, HTTP, or HTTPS. Properly configuring NSGs enhances the security of the AKS cluster.

Private AKS Cluster Deployment: To improve security, you can choose to deploy the AKS cluster in a private subnet. This configuration restricts direct access to the cluster from the internet, reducing the attack surface and ensuring the cluster is only accessible through authorized channels.DNS Integration: Integrating AKS with Azure DNS allows the cluster to register its internal services and pods using DNS names. This enables other services within the VNet to communicate with the AKS cluster using these DNS names.

Network Peering (Optional): If your AKS cluster needs to interact with other resources in different VNets, you can establish network peering between VNets. Network peering allows VNet resources to communicate with each other securely and

efficiently.

Private Endpoint (Optional): Azure Private Endpoint allows you to access AKS services privately from within your VNet. This ensures that traffic between the AKS cluster and other resources in the VNet remains within the Azure backbone network, reducing exposure to the public internet.

Load Balancer Integration:AKS automatically provisions an Azure Standard Load Balancer for the cluster. This load balancer handles network traffic distribution to the applications running in the AKS cluster. You can customize load balancer settings based on your application requirements.

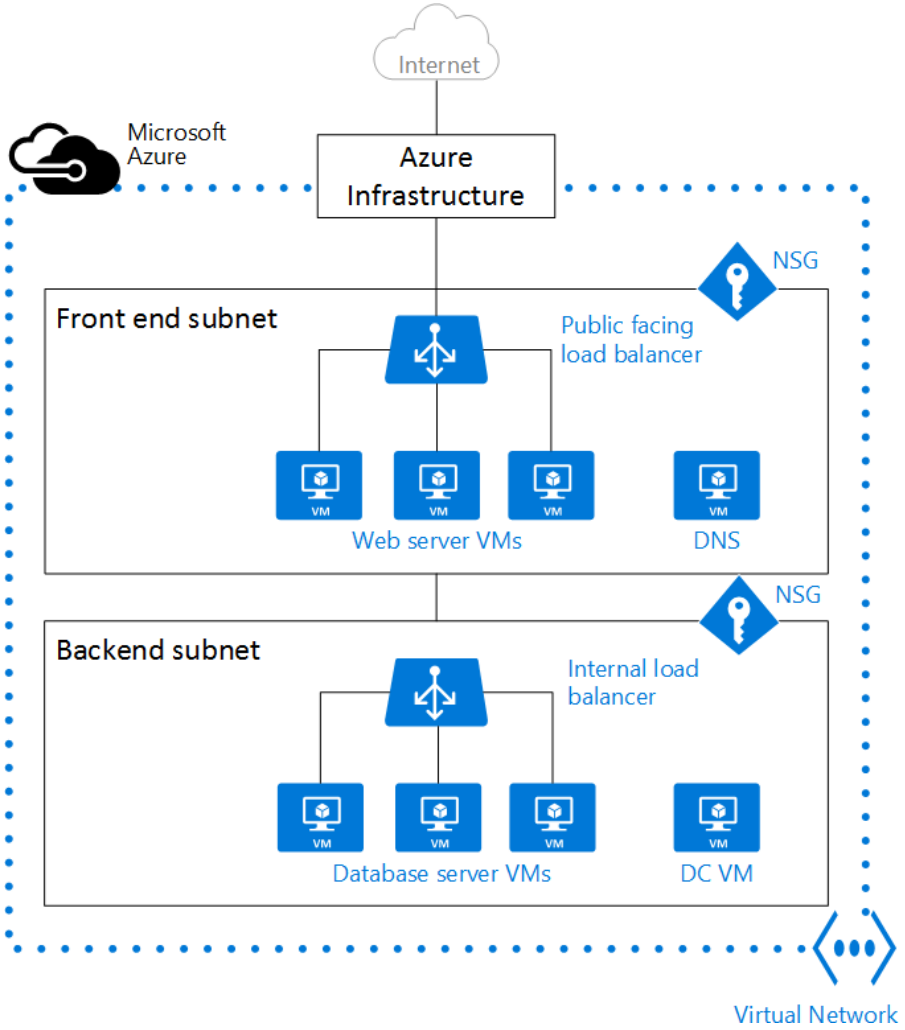
VPN Gateway and ExpressRoute (Optional):For hybrid network architectures, you can establish a site-to-site VPN connection or use Azure ExpressRoute to connect the on-premises network to the AKS cluster. This enables secure communication between resources in the on-premises network and the AKS cluster.

By integrating Azure Kubernetes Service with the network, organizations can efficiently deploy and manage containerized applications in a secure and scalable manner, leveraging the benefits of Azure's cloud-native ecosystem.

CHAPTER 3 INTERNAL NETWORK HARDENING

3.1 VIRTUAL NETWORK (VNet) AND SUBNET CONFIGURATION:

Creating a dedicated Virtual Network (VNet) for an Azure Kubernetes Service (AKS) cluster involves a series of steps in the Azure portal. The process typically includes creating a new VNetor using an existing one, defining subnets for master nodes, worker nodes, and system services, and configuring network security groups (NSGs) for access control. Below are the step-by-step instructions for creating a dedicated VNet for an AKS cluster:



3.1 Virtual Network Representation in Azure

Step 1: Sign in to the Azure Portal:

Log in to the Azure portal using your Azure account credentials.

Step 2: Create a New Resource Group (Optional):

If you want to organize your resources, you can create a new resource group or use an existingone to contain your AKS-related resources.

Step 3: Create a Virtual Network (VNet):

a. In the Azure portal, click on "+ Create a resource" on the left-hand side of the dashboard.

b. Search for "Virtual Network" and select "Virtual Network" from the search results.

c. Click on "Create" to begin the VNet creation process.

Step 4: Configure the Virtual Network (VNet):

a. In the "Basics" tab, provide a name for the VNet.

b. Choose the subscription and resource group (if applicable) where the VNet will be located.

c. Select the appropriate region for the VNet.

d. Specify the IP address range for the VNet. Ensure the IP range is large enough to accommodateall the subnets you plan to create.

Step 5: Define Subnets for Master Nodes, Worker Nodes, and System Services:

a. In the "Subnets" section of the VNet creation, click on "+ Add subnet."

b. Provide a name for the subnet, such as "aks-master-subnet."

c. Define the IP address range for the subnet. For example, 10.0.1.0/24.

d. Repeat the above steps to create subnets for worker nodes and system services, using appropriate names and IP address ranges.

Step 6: Network Security Groups (NSGs):

a. During the VNet creation process, you can optionally associate an NSG with each subnet. NSGsact as firewalls and control inbound and outbound traffic to the subnets.

b. You can create a new NSG or use an existing one. If creating a new NSG, define the necessaryinbound and outbound security rules based on your AKS cluster's requirements.

Step 7: Review and Create:

a. Review the settings for the VNet and subnets to ensure they are correct.

b. Click on "Create" to initiate the VNet creation process.

Step 8: Deploy AKS Cluster:

a. After the VNet is created, navigate to the AKS service in the Azure portal.

b. Click on "Add" to create a new AKS cluster.

c. In the "Basics" tab of the AKS creation, select the VNet you created earlier from the "VirtualNetwork" dropdown menu.

d. Choose the appropriate subnet for master nodes, worker nodes, and system services from theirrespective dropdown menus.

e. Complete the AKS cluster creation process by specifying other configuration settings such asnode count, node size, etc.

3.2 NETWORK SECURITY GROUP POLICY IMPLEMENTATION:

Network Security Groups (NSGs) play a crucial role in controlling inbound and outbound traffic to an Azure Kubernetes Service (AKS) cluster. They act as a distributed firewall and allow you to enforce network traffic filtering and access control at the subnet level, ensuring that only authorized traffic is allowed and malicious traffic is blocked. Here's how NSGs are used to control traffic for an AKS cluster:

Inbound Traffic Control:
NSGs allow you to define inbound security rules that control traffic coming into the AKS cluster's subnets from external sources or other subnets within the same Virtual Network (VNet).
By creating specific inbound rules, you can permit or deny traffic based on various parameters such as source IP addresses, source ports, destination IP addresses, destination ports, and protocols (e.g., TCP, UDP).

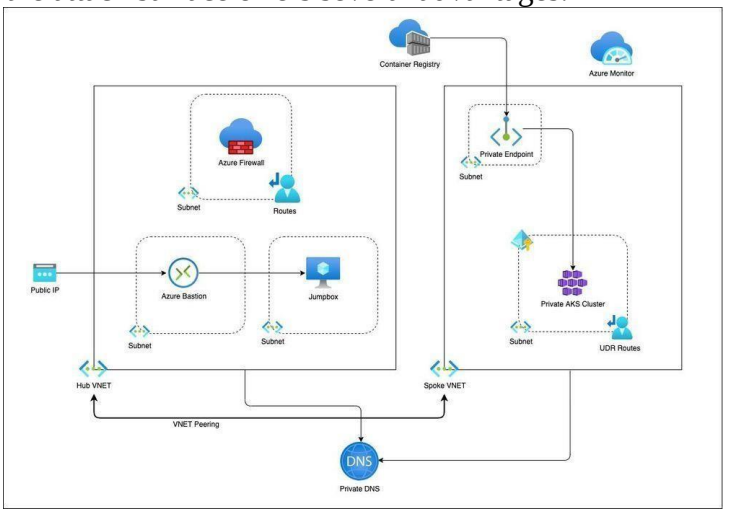
Outbound Traffic Control:
NSGs also allow you to define outbound security rules that control traffic leaving the AKS cluster's subnets and going to external destinations or other subnets within the VNet.
You can create outbound rules to restrict the types of traffic that the AKS nodes and services are allowed to initiate.

Default Rules:
NSGs come with default rules that allow outbound traffic and deny inbound traffic by default. These default rules ensure that the cluster's nodes and services can communicate with external resources while providing a level of protection against unauthorized external access.

Prioritization of Rules:
NSGs allow you to define rules in a priority order. When a packet matches a rule, NSGs stop evaluating subsequent rules. This allows you to implement more specific rules that take precedence over general rules.
For example, you might have a specific rule to allow SSH traffic from a trusted management IP address, and a general rule to block SSH traffic from other sources.

3.3 PRIVATE AKS CLUSTER DEVELOPMENT:

Deploying an Azure Kubernetes Service (AKS) cluster in a private subnet to limit direct access from the internet and reduce the attack surface offers several advantages.



3.3 Private AKS Cluster Architecture

Here are some benefits:

Improved Network Security: Placing the AKS cluster in a private subnet restricts direct internet access, making it less vulnerable to external threats. By isolating the cluster from the public internet, the attack surface is minimized, reducing the risk of unauthorized access and potential security breaches.

Data Privacy and Compliance: Hosting the AKS cluster in a private subnet enhances data privacy, ensuring that sensitive information and applications are shielded from public visibility. This approach can help organizations comply with regulatory requirements related to data protection and privacy.

Controlled Ingress and Egress Traffic: Deploying AKS in a private subnet allows organizations to exert granular control over incoming (ingress) and outgoing (egress) network traffic. Network Security Groups (NSGs) and other security measures can be applied to regulate communication, providing an additional layer of defense.

Enhanced Defense Against DDoS Attacks: Placing the AKS cluster in a private subnet can mitigate the risk of Distributed Denial of Service (DDoS) attacks. By isolating the cluster from the internet, it becomes less susceptible to volumetric attacks that aim to overwhelm network resources.

Secure Access Management: Access to the AKS cluster can be more tightly controlled when deployed in a private subnet. Administrators can use bastion hosts, VPNs, or ExpressRoute connections to establish secure access for authorized users. This helps prevent unauthorized access and strengthens the overall security posture.

Hybrid Cloud Integration: For organizations with hybrid cloud setups, deploying AKS in a private subnet allows for secure integration with on-premises resources. Using a site-to-site VPN or Azure ExpressRoute, AKS can communicate securely with private network resources, enabling a seamless and protected hybrid cloud environment.

3.4 ROLE BASED ACCESS CONTROL FOR AKS:

Role-Based Access Control (RBAC) for Azure Kubernetes Service (AKS) allows you to manage access to the AKS cluster and its resources based on predefined roles and permissions. RBAC provides a granular and secure approach to control who can perform specific actions within the AKS cluster. With RBAC, you can assign roles to users, groups, or service principals to determine what operations they are allowed to perform.

Roles in RBAC for AKS:

Cluster Roles: Cluster roles grant permissions for actions that apply to the entire AKS cluster, such as managing nodes, namespaces, and cluster-level resources. Examples of cluster roles include "ClusterAdmin," "ClusterReader," and "ClusterViewer."

Namespace Roles: Namespace roles grant permissions for actions that apply to a specific namespace within the AKS cluster. Namespace roles are used to control access to resources within a particular namespace, such as pods, services, and deployments. Examples of namespace roles include "NamespaceAdmin," "NamespaceEditor," and "NamespaceViewer."

Custom Roles: In addition to the built-in cluster and namespace roles, you can also create custom roles with specific permissions tailored to your organization's needs. Custom roles allow for fine-

grained control over access to resources within the AKS cluster.

Permissions in RBAC for AKS: Permissions define the actions that a user, group, or service principal is allowed to perform within the AKS cluster. Some examples of permissions include creating, reading, updating, and deleting resources like pods, services, and deployments.

Role Assignment: Role assignment involves associating a specific role with a user, group, or service principal. Role assignments can be done at the cluster level or the namespace level, depending on the desired level of access control.

Role Binding: Role binding is the process of associating a role with a user, group, or service principal within a specific namespace. This allows for targeted access control within specific namespaces in the AKS cluster.

Benefits of RBAC for AKS:

Security: RBAC ensures that only authorized users have access to the AKS cluster and its resources, reducing the risk of unauthorized access and data breaches.

Control and Compliance: RBAC allows organizations to enforce strict access controls and comply with industry regulations and security best practices.

Granular Access Control: RBAC provides fine-grained control over who can perform specific actions within the AKS cluster, allowing for more precise management of access permissions.

Simplified Management: RBAC streamlines the process of managing user access, as permissions can be assigned and revoked at a role level, rather than individually.

3.5 KUBERNETES RBAC FOR POD AND NAMESPACE ISOLATION:

Kubernetes Role-Based Access Control (RBAC) is an essential feature that allows administrators to define granular permissions for various Kubernetes resources, including pods and namespaces. Kubernetes RBAC helps enforce security measures by restricting access to resources based on predefined roles, ensuring that each user or service account has the least privilege necessary to perform their tasks. Here's how Kubernetes RBAC can be used for pod and namespace isolation:

Namespace Isolation:
In Kubernetes, namespaces are virtual clusters that allow you to logically isolate resources and applications. Each namespace acts as a separate environment within the cluster.Kubernetes RBAC can be used to create custom roles that grant specific permissions to users or service accounts within a particular namespace. This allows administrators to control access to resources at a namespace level. For example, you can create a custom role that permits a group of developers to manage pods, services, and deployments within a specific namespace, while restricting their access to other namespaces.

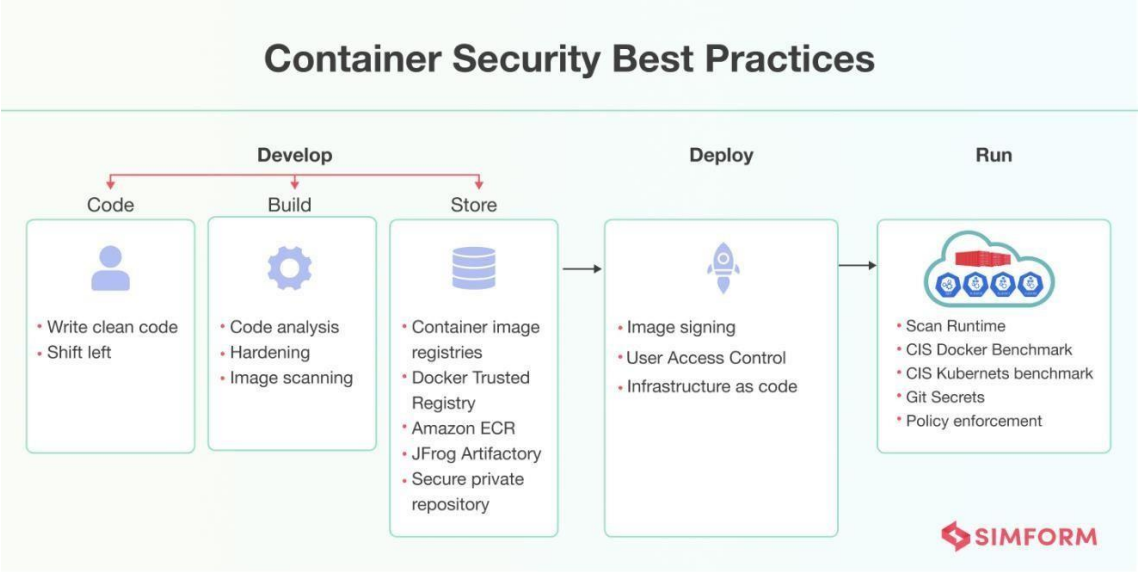
Pod Security Policies (PSPs) for Pod Isolation:
Pod Security Policies are a Kubernetes feature that helps enforce security policies on pods. PSPs define a set of rules that pods must comply with to be scheduled and run within a cluster.With PSPs, administrators can restrict privileged operations, enforce container resource limits, and block the use of specific security-sensitive capabilities within pods.By using PSPs, you can enforce pod isolation, ensuring that pods are deployed securely and adhere to your organization's security standards.

Service Accounts for Authentication:

Kubernetes Service Accounts provide an identity to a pod that allows it to interact with the Kubernetes API server and other resources within the cluster.RBAC can be used to define specific roles and permissions for service accounts, ensuring that pods have limited access to the cluster resources they need to function properly.By controlling the permissions of service accounts, you can enforce pod isolation and reduce the risk of unauthorized access to the cluster.

3.6 CONTAINER SECURITY BEST PRACTICES:

Container security is crucial to ensure the safety and integrity of applications and data within containerized environments. Here are some best container security practices to help mitigate risks and enhance overall security:



3.6 Container practices to increase Security

Use Trusted Base Images: Start with official, verified, and regularly updated base images from reputable sources. Avoid using images from untrusted or unknown registries, as they may contain security vulnerabilities.

Keep Containers Updated: Regularly update container images and underlying software components to patch known vulnerabilities and ensure the latest security fixes are applied.

Implement Image Scanning: Use container image scanning tools to check for known vulnerabilities, malware, and security issues in container images before deploying them in production.

Use Multi-Stage Builds: Employ multi-stage builds to separate the build environment from the production environment. This reduces the size and attack surface of the final container image.

Limit Container Privileges: Apply the principle of least privilege by restricting the capabilities and access rights of containers. Containers should run with the minimum necessary permissions to perform their tasks.

Container Runtime Security: Choose a secure container runtime and configure it to use appropriate security features like SELinux or AppArmor to enforce additional security policies.

Implement Network Segmentation: Use network segmentation to isolate containers based on their functions and sensitivity. This can be achieved using network policies or container network isolation mechanisms.

Secure Secrets Management: Store sensitive information like API keys and passwords securely using container orchestrator-provided solutions like Kubernetes Secrets or other secure storage mechanisms.

Enable Pod Security Policies (PSPs): Implement Pod Security Policies to enforce security best practices for pod creation, such as limiting privilege escalation and restricting the use of certain capabilities.

Monitor Container Activity: Employ monitoring and logging tools to track container activity and detect unusual behavior that might indicate a security breach.

Implement RBAC for Access Control: Utilize Role-Based Access Control (RBAC) to control access to container orchestrator resources and APIs. Define appropriate roles for users and service accounts to restrict unauthorized access.

Container Signing and Verification: Use container signing to verify the authenticity and integrity of container images before deployment.

Secure Host OS and Infrastructure: Ensure the host OS and infrastructure supporting containers are also appropriately hardened and regularly patched.

Encourage Code Review and Scanning: Conduct regular code reviews and use static code analysis tools to catch security flaws early in the development process.

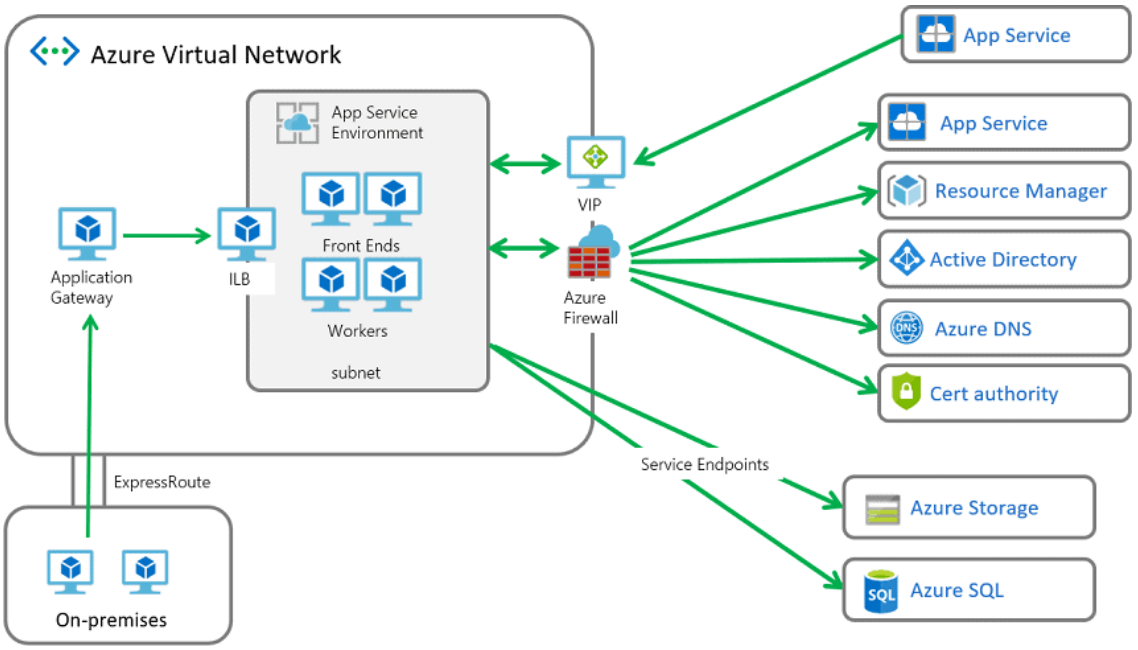
Regular Security Audits and Assessments: Conduct periodic security audits and assessments to identify potential vulnerabilities and weaknesses in the container environment.

Educate Development Teams: Train development teams on container security best practices and provide guidance for writing secure containerized applications.

CHAPTER 4 EXTERNAL HARDENING OF NETWORK

4.1 AZURE FIREWALL:

Azure Firewall can be used with Azure Kubernetes Service (AKS) to provide network security and control inbound and outbound traffic to the AKS cluster. Here are some key use cases for Azure Firewall with AKS:



4.1 Azure Firewall Applications

Cluster Egress Traffic Control: Azure Firewall can be deployed to control outbound (egress) traffic from the AKS cluster. This allows you to define rules and policies to restrict which destinations the cluster can access on the internet. For example, you can limit access to specific IP ranges, URLs, or ports, providing a secure egress filtering mechanism.

Load Balancer Inbound Traffic Filtering: When AKS services are exposed through Azure Load Balancer, Azure Firewall can be used to filter and control inbound traffic to these services. It acts as a centralized firewall for the load balancer, ensuring only authorized traffic is allowed to reach the AKS services.

Protecting Kubernetes API Server: Azure Firewall can be used to control access to the Kubernetes API server endpoint for the AKS cluster. By applying network rules, you can restrict access to the API server from specific IP ranges, such as your organization's corporate network or specific management systems.

Integration with Hub-and-Spoke Architecture: In hub-and-spoke network architectures, where AKS is deployed in a spoke VNet, Azure Firewall can be placed at the hub VNet to provide centralized security and traffic control for all connected spoke VNets, including the AKS cluster.

Ingress Traffic Control with Application Gateway: When AKS services are exposed through Azure Application Gateway, Azure Firewall can be used to inspect and control inbound traffic flowing through the gateway. This enables more sophisticated security features, such as URL

filtering, WAF (Web Application Firewall) capabilities, and SSL termination.

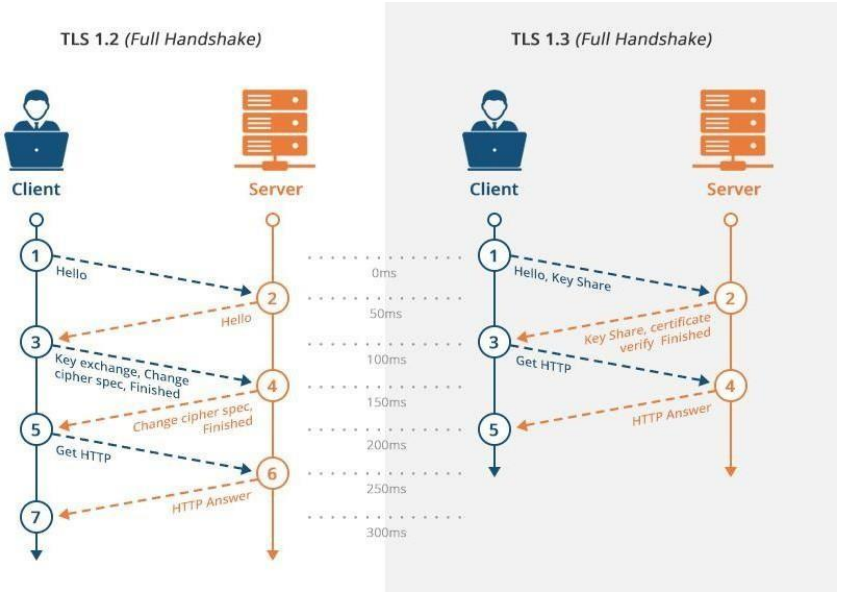
Hybrid Connectivity and VPN: If your AKS cluster requires hybrid connectivity with on-premises resources or other cloud environments, you can use Azure Firewall to secure and control traffic over VPN or ExpressRoute connections.

Logging and Monitoring: Azure Firewall provides logging capabilities that allow you to monitor and audit network traffic in and out of the AKS cluster. This helps with troubleshooting, security analysis, and compliance requirements.

Threat Intelligence Integration: Azure Firewall can integrate with threat intelligence feeds to block traffic from known malicious IP addresses or domains, adding an additional layer of security to your AKS cluster.

4.2 TRANSPORT LAYER SECURITY FOR SECURE COMMUNICATION:

Transport Layer Security (TLS) is a crucial technology for securing communication in Azure Kubernetes Service (AKS). It ensures that data transmitted between different components of the AKS cluster, as well as between AKS and external services, remains encrypted and secure. Here's how TLS is used for secure communication in AKS:



4.2 Transport Layer Security working

Secure Communication Between Pods:
Kubernetes supports TLS for securing communication between pods within the same cluster. When deploying applications in AKS, you can configure TLS for intra-cluster communication to encrypt data exchanged between pods.

TLS Termination at Ingress Controllers:
Ingress controllers like Azure Application Gateway or Azure Kubernetes Ingress Controller (AKS Ingress) can terminate TLS connections on behalf of the backend services.
TLS termination at the ingress controllers allows them to handle SSL/TLS certificates and encrypt/decrypt traffic between clients and the AKS cluster.

Secure Ingress Traffic:
In AKS, you can configure ingress resources with TLS settings to secure incoming traffic to your services.TLS can be enforced for specific hosts or paths, and you can specify the SSL/TLS

certificates to be used for the encryption.

Secure Communication with External Services:
When AKS services need to communicate with external services, TLS can be used to encrypt the traffic and ensure secure data exchange.
For example, when AKS pods interact with an external database or API, TLS can be enforced to protect sensitive data during transit.

Certificate Management:
To enable TLS in AKS, you need to manage SSL/TLS certificates.
You can use Kubernetes Secrets or Azure Key Vault to securely store and manage SSL/TLS certificates used for secure communication.

Self-Signed Certificates vs. CA-Signed Certificates: AKS allows you to use self-signed certificates or certificates signed by a trusted Certificate Authority (CA) for TLS.For production environments, it is recommended to use CA-signed certificates to ensure trust and security.

4.3 MONITORING AND LOGGING FOR SECUERITY INSIGHTS:

Monitoring and logging are critical components of ensuring security and gaining insights into the activities and performance of an Azure Kubernetes Service (AKS) cluster. By effectively monitoring and logging AKS, organizations can detect security threats, troubleshoot issues, and optimize cluster performance. Here's how monitoring and logging can be implemented for security insights in AKS:

4.3 Logging and Monitoring the data

Azure Monitor for Containers: Azure Monitor for Containers is a service that provides real-time monitoring of AKS clusters. It collects performance metrics, container logs, and infrastructure logs from AKS nodes and containers. With Azure Monitor for Container, you can gain insights into the resource utilization, performance, and health of the AKS cluster.

Container Insights: Container Insights is a feature of Azure Monitor for Containers that offers specialized monitoring for AKS clusters.t provides pre-built dashboards for AKS, enabling visualization of key metrics such as CPU usage, memory, and network traffic.Container Insights also offers insights into container performance, container restarts, and container-specific logs.

Kubernetes Events and Audit Logs: AKS generates Kubernetes events and audit logs, which provide information about cluster activities, including pod creations, updates, and deletions. These logs can be collected and analyzed to identify any suspicious or unauthorized activities within the cluster.

Centralized Logging with Azure Log Analytics: Azure Log Analytics can be used to centralize and store logs from multiple AKS clusters.
Log Analytics allows you to query and analyze the logs to identify security-related patterns and anomalies.

Alerting and Action Groups: Azure Monitor allows you to set up alerts based on predefined conditions or custom queries against log data.
When an alert is triggered, it can be sent to Action Groups, which can notify stakeholders or trigger automated actions.

Integrate with Azure Security Center: By integrating AKS with Azure Security Center, you can gain additional security insights and recommendations to enhance the overall security posture of the cluster.

Collecting Application Logs: In addition to infrastructure and container logs, it is essential to collect application-specific logs generated by the containers in AKS.Application logs can provide insights into application behavior and potential security-related issues.

Monitoring Kubernetes API Server Activity: Monitoring the Kubernetes API server's activity can help identify potential unauthorized access or unusual patterns of API requests.

4.4 REGULAR UPDATES AND PATCHES:

Regular updates and patches are crucial for hardening the network in an environment that uses Azure Kubernetes Service (AKS). Keeping the AKS cluster, underlying infrastructure, and connected resources up-to-date ensures that known vulnerabilities are mitigated, security improvements are implemented, and the overall security posture remains strong. Here are some best practices for regular updates and patches in an AKS environment:

4.4 Advantages of patching and updating

AKS Cluster Updates: AKS clusters are regularly updated by Microsoft to include the latest security patches, bug fixes, and new features.
Enable automatic cluster upgrades to ensure that your AKS cluster is always running the latest stable version of Kubernetes.

Node Updates: Regularly update the operating system and software packages running on the AKS nodes.Enable automatic OS upgrades for AKS nodes to ensure they are patched with the latest security updates.

Container Image Updates: Continuously monitor and update container images used in your AKS deployment to ensure they are based on the latest, patched versions.Implement an image scanning process to detect vulnerabilities in container images before deployment.

Kubernetes Components Updates: Monitor updates for Kubernetes components like the kubelet, kube-proxy, and CoreDNS, and apply the latest security patches to these components.

Network Security Group (NSG) Rules Review: Regularly review and update the NSG rules for the AKS cluster and its connected resources.
Remove any unnecessary open ports and restrict inbound and outbound traffic to the minimum required for the cluster to function.

Use Azure Policy and Azure Security Center: Leverage Azure Policy and Azure Security Center

to enforce compliance and security policies for AKS clusters.Set up policies to detect and remediate non-compliant configurations.

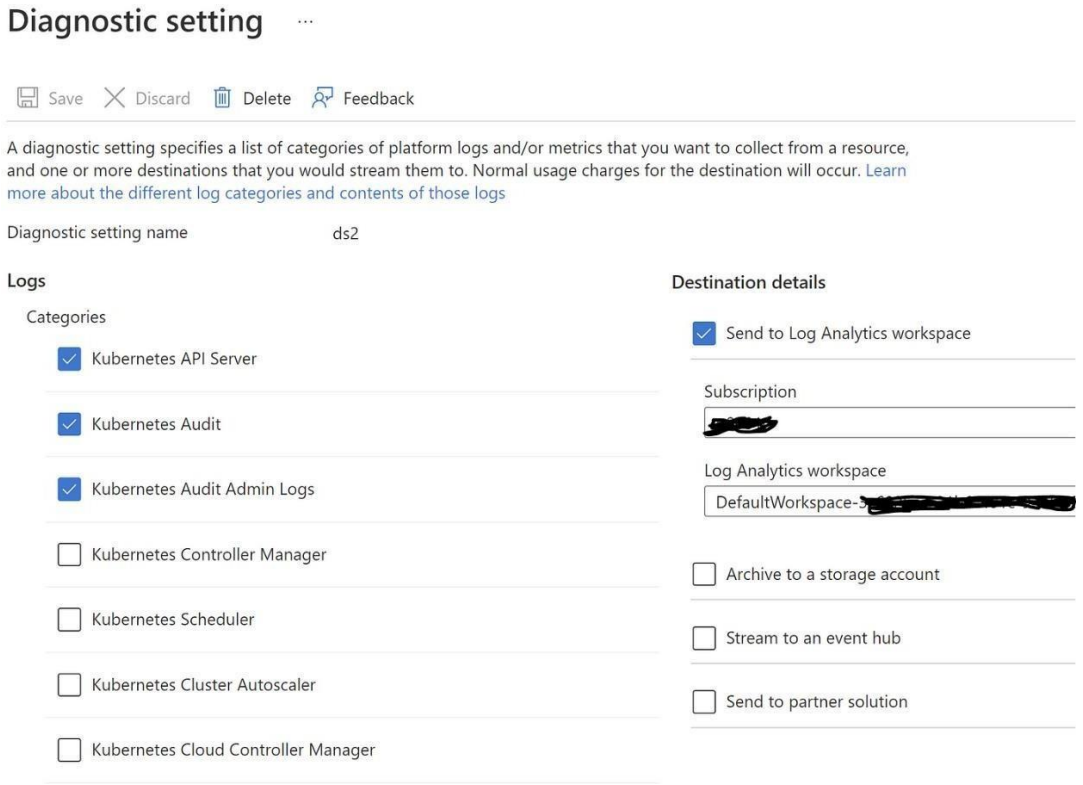
Regular Security Assessments: Perform regular security assessments and audits on the AKS environment to identify potential security weaknesses and gaps.Use tools like Azure Security Center to conduct continuous security assessments.

Backup and Disaster Recovery: Implement regular backups of critical AKS resources, including persistent volumes and configuration data.
Test disaster recovery scenarios to ensure that data and applications can be restored in the event of a catastrophic failure.

Educate and Train Staff: Educate your development and operations teams about the importance of updates and patches. Encourage a culture of security awareness and ensure that all team members understand their role in maintaining a secure AKS environment.

4.5 TESTING AND AUDITING:

Auditing and testing the network that uses Azure Kubernetes Service (AKS) is essential for several reasons to ensure the security, reliability, and performance of the AKS environment. Here are some key reasons why auditing and testing the network are necessary:



4.5 Microsoft Azure Sentinel Setting for auditing and testing the Network

Security Assurance: Auditing and testing the network help identify potential security vulnerabilities and weaknesses in the AKS cluster's network architecture. It allows organizations to implement appropriate security measures to protect against unauthorized access, data breaches, and other security threats.

Compliance Requirements: Many industries and regulatory standards mandate regular auditing and testing of networks to meet compliance requirements. Organizations using AKS may need to adhere to specific security and privacy standards, such as GDPR, HIPAA, or PCI DSS, and auditing helps demonstrate compliance.

Identifying Misconfigurations: Network auditing and testing can uncover misconfigurations in AKS network settings that could lead to unintended exposures or performance issues. Regular audits ensure that the network is properly configured and aligned with best practices.

Detecting Anomalies and Intrusions: Regular monitoring, auditing, and testing allow organizations to detect unusual activities or potential intrusions into the AKS cluster's network. Identifying anomalies early can help prevent security breaches and mitigate the impact of attacks.

Performance Optimization: Auditing and testing the network can help identify network bottlenecks, latency issues, or suboptimal configurations affecting AKS performance. By optimizing the network, organizations can enhance the responsiveness and reliability of their applications.

Redundancy and Resilience Testing: Auditing the network helps ensure that AKS has the necessary redundancy and resilience to handle potential failures. Organizations can perform network testing scenarios, such as simulating node or network failures, to validate the cluster's resilience.

Load Balancing and Traffic Management: Network testing helps evaluate the effectiveness of load balancers and traffic management mechanisms in distributing incoming requests to AKS services. This ensures high availability and efficient resource utilization.

Scaling Readiness: Regular network testing can reveal whether the AKS cluster is ready for scaling in response to increased demand. It helps ensure that the network architecture can accommodate growth without performance degradation.

Change Management Validation: Network audits validate that any changes made to the network configuration do not introduce vulnerabilities or disrupt existing services.

Preventing Downtime: Network testing can help identify potential causes of downtime or service disruptions, allowing proactive measures to prevent or minimize downtime impact.

CHAPTER 5 CONCLUSION

Through this internship project, I gained valuable hands-on experience in implementing security measures for a cloud-based container orchestration service like Azure Kubernetes Service. The project allowed me to understand the significance of network hardening and the importance of staying up-to-date with the latest security practices.o further enhance the security of the AKS cluster, I recommend conducting regular security audits and vulnerability assessments. Additionally, continuous monitoring and analysis of network traffic should be implemented to detect any anomalous behavior or potential security breaches.

REFERENCES

WEB REFERENCES

1.Official Microsoft Azure Kubernetes Service (AKS) Documentation:https://docs.microsoft.com/en-us/azure/aks/

2.Azure Security Center Documentation:https://docs.microsoft.com/en-us/azure/security-center/

3.National Institute of Standards and Technology (NIST) Special Publication 800-53:https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final

4.Center for Internet Security (CIS) Kubernetes Benchmark:https://www.cisecurity.org/benchmark/kubernetes/

5.Kubernetes.io Security Documentation:https://kubernetes.io/docs/concepts/security/

6.OWASP Kubernetes Security Project:https://owasp.org/www-project-kubernetes-security/

7.Microsoft Security Compliance Toolkit:https://www.microsoft.com/en-us/download/details.aspx?id=55319

8.Microsoft Cloud Adoption Framework for Azure - Security and Networking:https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/network_security

9.Nginx Ingress Controller Documentation:https://kubernetes.github.io/ingress-nginx/

10.Azure Network Security Best Practices:https://docs.microsoft.com/en-us/azure/security/fundamentals/network-best-practices

11.Azure Network Security Group (NSG) Documentation:https://docs.microsoft.com/en-us/azure/networking/network-security-groups/

12.Kubernetes Network Policies:https://kubernetes.io/docs/concepts/services-networking/network-policies/