VIRTUAL NETWORK CONFIGURATION

- ANKITHA BM - 3BR23CD006
- SAMRUDDHI V MUNI - 3BR23CD084
- SANJANA RAGHU - 3BR23CD085
- SHAGUFTHA NIKHAR - 3BR23CD087
- SHREYASREE MATH - 3BR23CD091

***INTRODUCTION*:**

Virtual network configuration :

It refers to the process of setting up and managing a virtual network within a cloud or virtualized environment. A virtual network (VNet) allows devices, services, and applications to communicate with each other over a simulated network rather than a physical one. Virtual networks are typically used in cloud computing platforms like Microsoft Azure, AWS, or VMware.

PURPOSE:

The main purpose of virtual network configuration is to enable seamless communication between virtual resources (like virtual machines, containers, databases) while providing flexibility, scalability, and security. Virtual networks allow organizations to logically separate and manage workloads, reducing the need for physical networking hardware.

GOAL:

- **Isolation**: Segregate environments (e.g., development, testing, production) or workloads for better management and security.
- **Connectivity**: Facilitate communication between virtual resources, other networks, and external environments (like on-premises systems or the internet)
- **Scalability**: Easily scale the network to accommodate growing infrastructure demands without physical limitations.
- **Security**: Implement firewall rules, access control, and encryption to protect network traffic and prevent unauthorized access.
- **Cost Efficiency**: Reduce the need for physical network devices, lowering infrastructure costs and simplifying network management.

PROBLEM STATEMENT:

- CRUD: Network settings.
- Setup: Virtual networks (network_config).
- Setup and configure virtual networks.
- Update: Network settings (settings_id), and updating network settings as needed.

TECHNOLOGY USED:

- PYTHON(IDLE shell 3.12.4 version)
- PYTHON Dictionary

***SYSTEM DESIGN*:**

- NetworkSetting: A class representing individual network settings.
- NetworkManager: Manages all network settings and encompasses the CRUD operations.
- VirtualNetwork: Handles setting up and managing virtual networks.

ALGORITHM:

1.Start

2.Initialize network settings (example: a dictionary to hold networks).

Create Network:

1.If create_network(network_id, name, ip_range, status) is called:

-Check if network_id exists in network_settings.

~If exists, print "Network with ID already exists.“

~Otherwise, add the new network to network_settings and print confirmation.

Read Network:

1.If read_network(network_id) is called:

-Retrieve and print the network information for network_id.

~If network_id doesn't exist, print "Network with ID does not exist.“

Update Network:

1.If update_network(network_id, name, ip_range, status) is called:

-Check if network_id exists in network_settings.

~If exists, update the network fields (name, IP range, status) as provided, and print confirmation.

~If not, print "Network with ID does not exist."

Delete Network:

1.If delete_network(network_id) is called:

-Check if network_id exists in network_settings.

~If exists, delete the network from network_settings and print confirmation.

~If not, print "Network with ID does not exist.“

Setup Virtual Network:

1.If setup_virtual_network(network_id, name, ip_range, status) is called:

-Check if network_id exists in network_settings.

~If not, call create_network() and set up the new network.

~Otherwise, print that the network already exists.

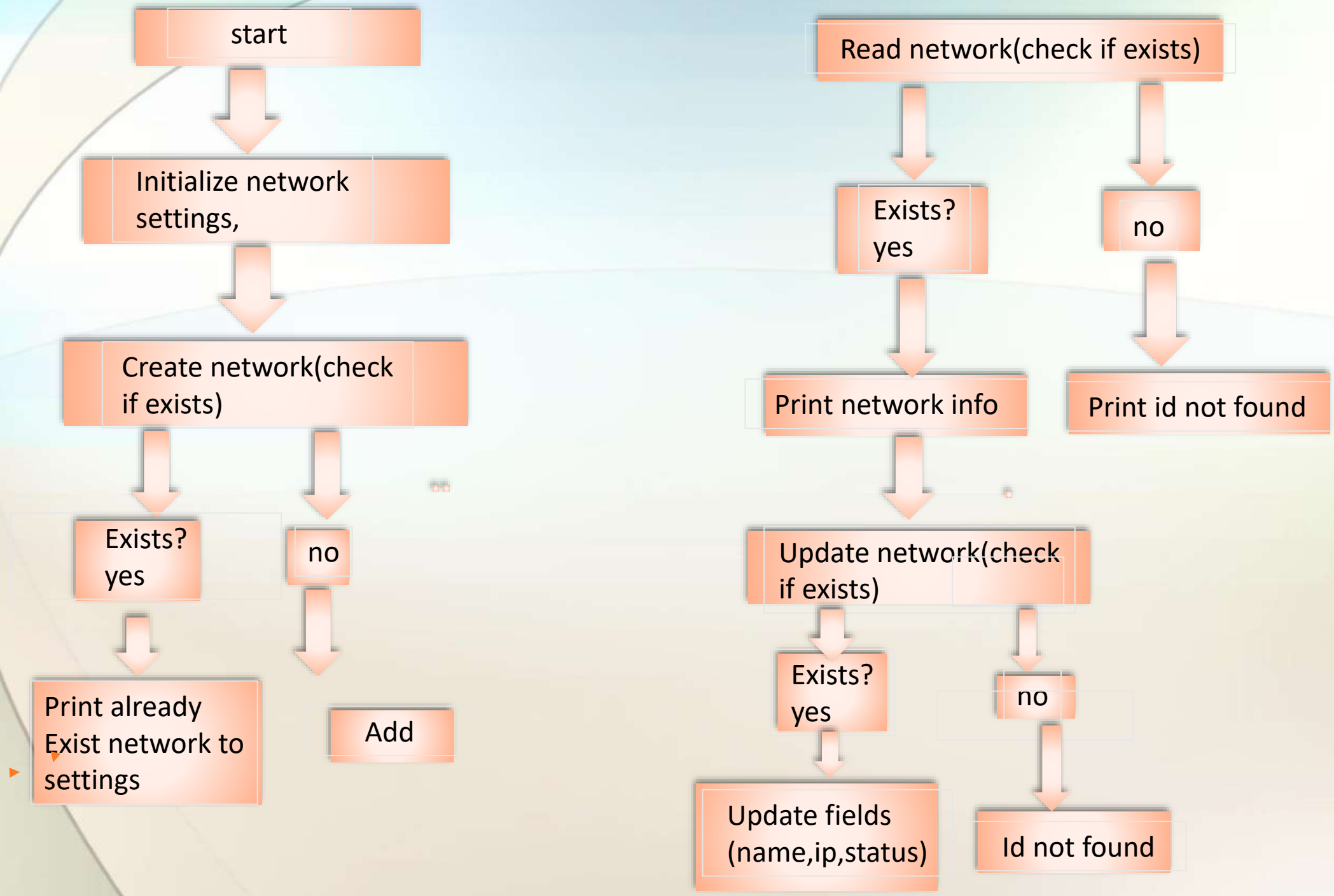
Update Network Settings:

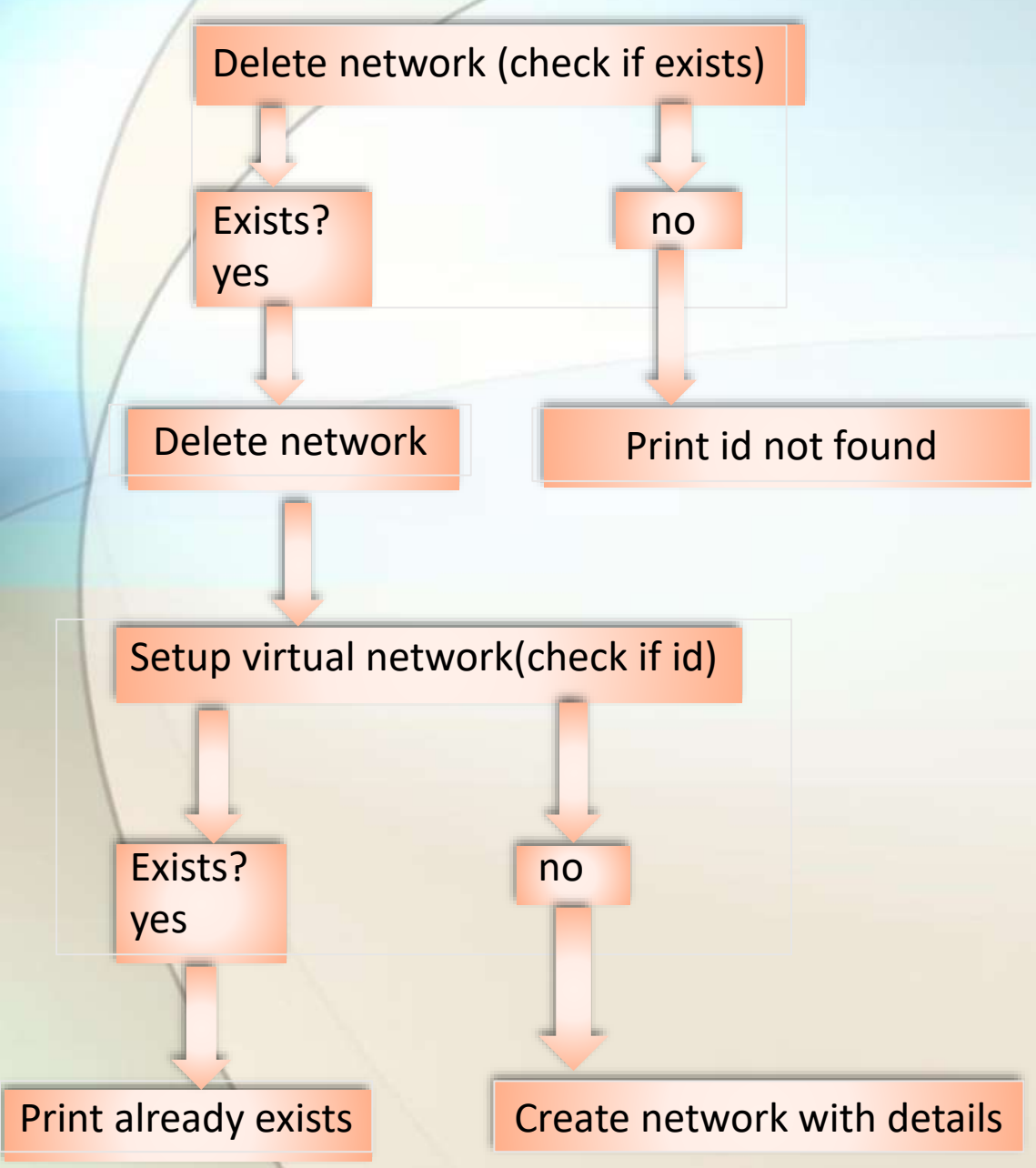
1.If update_network_settings(settings_id, new_config) is called:

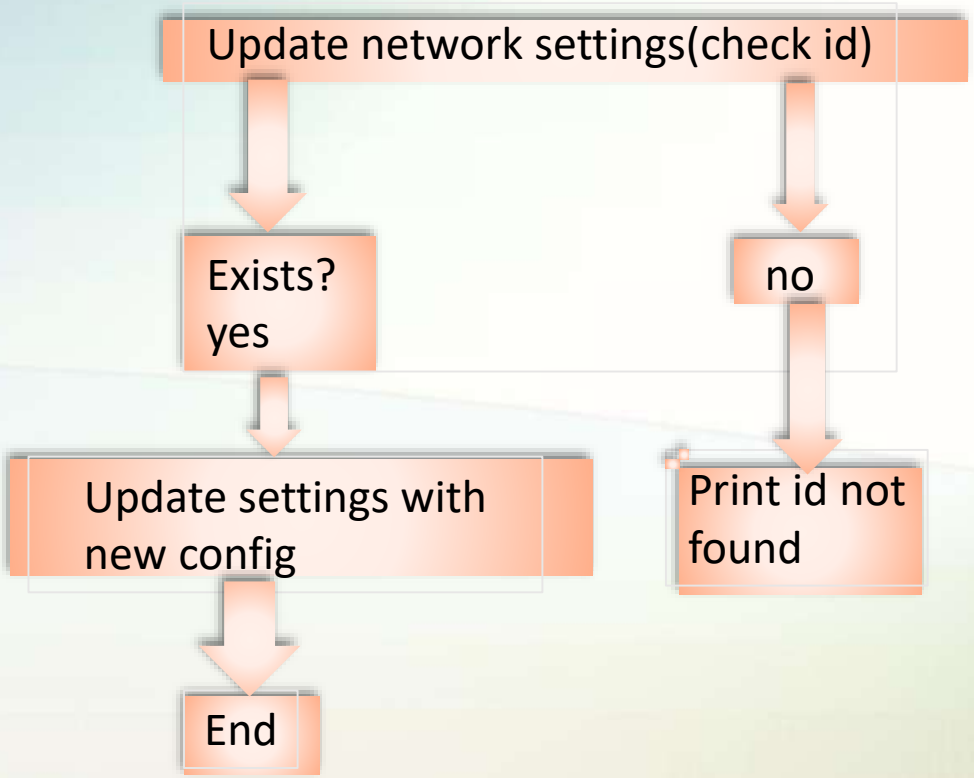
-Check if settings_id exists in network_settings.

~If exists, update the network settings with new_config and print confirmation.

~If not, print "Network with ID does not exist."End


start
Initialize network settings,
Create network(check if exists)
Exists? yes
no
Print already Exist network to settings
Add
Read network(check if exists)
Exists? yes
no
Print network info
Print id not found
Update network(check if exists)
Exists? yes
no
Update fields (name,ip,status)
Id not found


Delete network (check if exists)
Exists?
yes
no
Delete network
Print id not found
Setup virtual network(check if id)
Exists?
yes
no
Print already exists
Create network with details


Update network settings(check id)
Exists?
yes
no
Update settings with
new config
Print id not
found
End

NETWORK SETTING:

We'll use a Python dictionary to represent our network configurations. Each network will have a unique network_id and associated configuration

```
network_settings = {
 1: {'name': 'Network_1', 'ip_range': '192.168.1.0/24', 'status': 'active'},
  2: {'name': 'Network_2', 'ip_range': '192.168.2.0/24', 'status': 'inactive’}
}
```

CRUD OPERATIONS:

Create, Read, Update, and Delete operations for network settings.

1.Create a network configuration:

```
def create_network(network_id, name, ip_range, status):
        if network_id in network_settings:
                print(f"Network with ID {network_id} already exists.")
        else:
                network_settings[network_id] = {'name': name, 'ip_range': ip_range, 'status':
status}
                print(f"Network '{name}' created with ID {network_id}.”)
```

2.Read network configuration:

```
def read_network(network_id):
        network = network_settings.get(network_id)
        if network:
                print(f"Network {network_id}: {network}")
        else:
                print(f"Network with ID {network_id} does not exist.”)
```

3.Update network configuration:

```
def update_network(network_id, name=None, ip_range=None, status=None):
        if network_id in network_settings:
                if name:
                        network_settings[network_id]['name'] = name
        if ip_range:
                network_settings[network_id]['ip_range'] = ip_range
                if status:
                        network_settings[network_id]['status'] = status
print(f"Network {network_id} updated.")
        else:
                print(f"Network with ID {network_id} does not exist.")
```

4.Delete Network Configuration :

```
def delete_network(network_id):
        if network_id in network_settings:
                del network_settings[network_id]
                print(f"Network {network_id} deleted.")
        else:
                print(f"Network with ID {network_id} does not exist.")
```

Real-Life Example:

Virtual Network Configuration for IT Infrastructure Imagine an IT department within a medium to large organization that manages multiple virtual networks for various departments, such as finance, human resources, and development. The Virtual Network Configuration system helps the IT team efficiently configure, manage, and update virtual networks, ensuring that network resources are used optimally and securely.

1. CRUD: Network Settings The system provides capabilities to create, read, update, and delete network settings, allowing the IT team to maintain accurate and current configurations for each virtual network..

Example:

- Create: When a new department is formed, the IT team creates network settings for the finance department, specifying the IP address range, subnet mask, and other relevant configurations.
- Read: The IT staff can view existing network settings to ensure they align with the department's requirements. For instance, they check the settings for the Development Network to verify the allocated resources.
- Update: If the organization undergoes a merger, the IT team can update the network settings to accommodate new departments, changing the IP address range and adding VLAN configurations as necessary.
- Delete: When a project is completed, the corresponding virtual network settings are deleted from the system to avoid clutter and confusion.

Setup and configure Virtual Networks:

```
def setup_virtual_network(network_id, name, ip_range, status):    if
network_id not in network_settings:
        create_network(network_id, name, ip_range, status)
        print(f"Virtual network '{name}' is set up with IP
        range {ip_range} and status {status}.")
        else:
                print(f"Network with ID {network_id} already exists,
        consider updating it.")
```

2. setup_virtual_networks(network_config) The tool enables the IT team to set up and configure virtual networks based on specific requirements and configurations.

Example:

- The IT department needs to set up a new virtual network for a project team working on a sensitive application. They use the system to input the network configuration details, such as security protocols, bandwidth limits, and access permissions.
- The configuration is processed, and the virtual network is created, providing isolated resources for the project team while maintaining compliance with organizational policies

Update Network Settings Based on settings_id

Network configurations need to be updated dynamically based on certain network settings (like settings_id).

```
def update_network_settings(settings_id, new_config):
        if settings_id in network_settings:
        network_settings[settings_id].update(new_config)
        print(f"Network {settings_id} updated with new
        configuration: {new_config}")
        else:
                print(f"Network with ID {settings_id} does not exist.")
```

3. update_network_settings(settings_id) The system also allows for easy updates to existing network settings, ensuring that the configurations remain relevant and effective as needs change. Example:

- The company decides to implement stricter security measures. The IT team identifies the network settings for the HR Network and updates them to include enhanced encryption protocols and stricter access controls.
- The system logs the changes made, providing a historical record of modifications for compliance and auditing purposes.

FUTURE SCOPE:

- 1. AI-Driven Automation: Automate network setup, management, and self-healing using AI.
- 2. Enhanced Security: Implement context-aware security, zero trust, and AI-driven threat detection.
- 3. Advanced Monitoring: Real-time interactive maps, predictive analytics, and detailed traffic insights.
- 4. User-Friendly Interfaces: Drag-and-drop design, pre-built templates, and virtual testing environments.
- 5. Multi-Cloud Integration: Seamless networking across multiple cloud providers and edge computing support.
- 6. Scalability: Use NFV, dynamic bandwidth, and latency optimization. and so on….

CONCLUSION:

- Providing an organized and structured way to manage network settings across various virtual networks,ensuring that configurations are current and accurate.
- Enabling seamless setup of virtual networks tailored to specific departmental needs, allowing for flexibilityand isolation of resources.
- Facilitating quick and efficient updates to network settings in response to changing organizational requirements, maintaining security and compliance.
- It ensures that the network infrastructure is robust, secure, and adaptable, ultimately supporting business operations and enhancing productivity across.

REFERENCES:

https://www.sdxcentral.com/security/definitions/data-security-in-the-cloud-best-practices/virtual-network-security-works/

https://docs.oracle.com/cd/E19504-01/802-5753/6i9g71m52/index.html

https://docstore.mik.ua//manuals/hp-ux/en/T2767-90141/ch08s01.html

THANK YOU!