Hi,

You will find below a report detailing my solution for the network engineering assignment.

The solution consists of two steps:

1. Building the network topology
2. Configuring the network

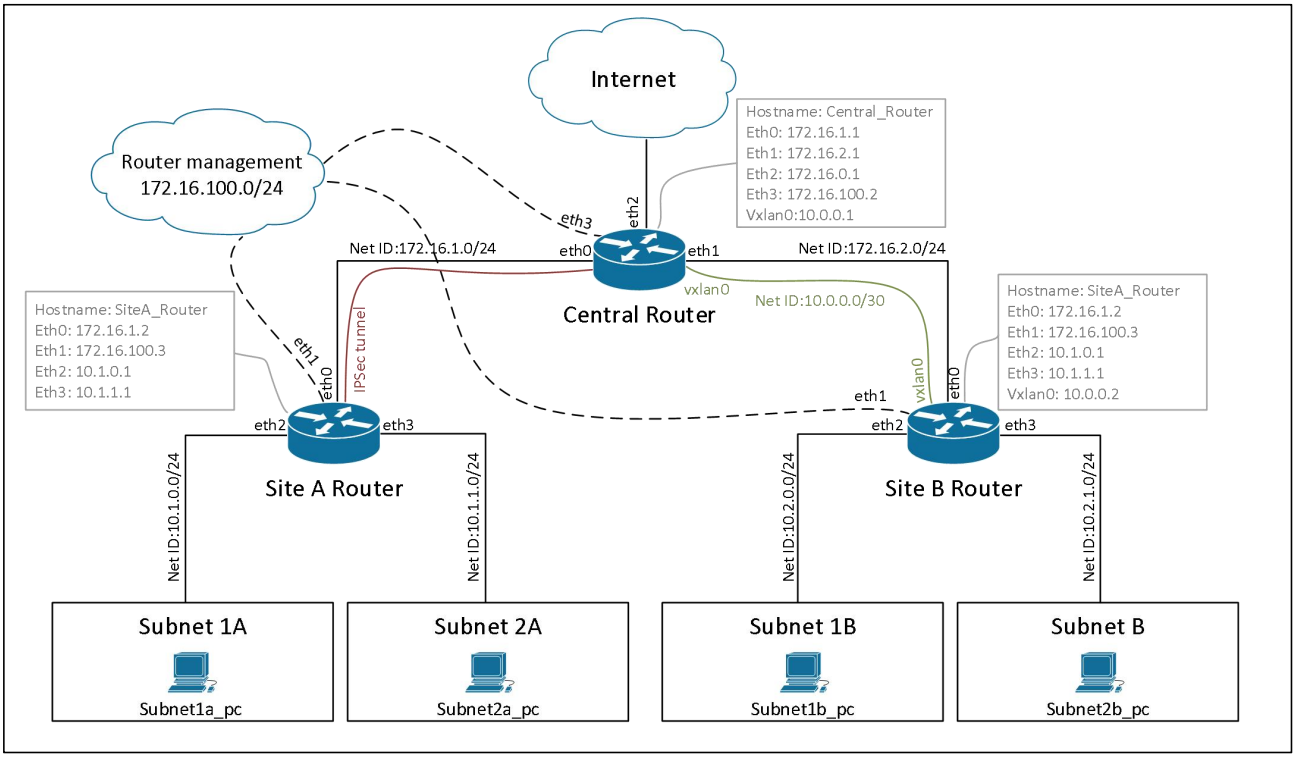
I wrote two different scripts - one using Ansible and the other using Shell to build the network topology. Either script can be used to create the network. For the second step, I wrote a Python script that connects and configures the VyOS routers using the VyOS API. I solely relied on my own code for configuring the routers and did not use any pre-existing components or libraries, such as vycontrol, that are available for configuring VyOS routers through the API.

I wrote a shell script named "main.sh" located in the root folder of assignment to run my solution. During the implementation of the network, I faced some challenges which I have documented in the "Challenges" section along with the solutions I employed to overcome them.

Meanwhile, I have designed and implemented the network using IPv4. If it needs to be done with IPv6 instead, please let me know. Also, please let me know if you need any information regarding the scripts I wrote for the assignment.

Best regards,
Ali Imani

■ Network topology



■ Docker network list

Docker network name	Driver name	Network ID	Gateway	Description
router_mng	bridge	172.16.100.0/24		To manage and configure routers
external	bridge	172.16.0.0/24	172.16.0.1	Internet
central_site_a	macvlan	172.16.1.0/24	172.16.1.1	
central_site_b	macvlan	172.16.2.0/24		
subnet1a	macvlan	10.1.0.0/24	10.1.0.1	
subnet2a	macvlan	10.1.1.0/24	10.1.1.1	
subnet1b	macvlan	10.2.0.0/24	10.2.0.1	
subnet2b	macvlan	10.2.1.0/24	10.2.1.1	

As the routers are connected through the macvlan network, it was not possible to connect and configure them through the API. To overcome this issue, I added an additional network called "router_mng." Although there are alternative solutions, such as using the bridge driver instead of the macvlan driver, I chose this solution to maintain isolation between the two routers.

■ Docker containers list

Docker container name	Docker image	Network name	Int name	IP address	Default gateway
central_router	VyOS	central_site_a	eth0	172.16.1.1	172.16.0.1
		central_site_b	eth1	172.16.2.1	
		external	eth2	172.16.0.1	
		manage_routers	eth3	172.16.100.2	
		vxlan	vxlan0	10.0.0.1	
site_a_router	VyOS	central_site_a	eth0	172.16.1.2	172.16.1.1
		subnet1a	eth1	172.16.100.3	
		subnet2a	eth2	10.1.0.1	
		manage_routers	eth3	10.1.1.1	
site_b_router	VyOS	central_site_b	eth0	172.16.2.2	10.0.0.1
		subnet1b	eth1	172.16.100.4	
		subnet2b	eth2	10.2.0.1	
		manage_routers	eth3	10.2.1.1	
		vxlan	vxlan0	10.0.02	
subnet1a_pc	Ubuntu	subnet1a	eth0	10.1.0.3	10.1.0.1
subnet2a_pc	Ubuntu	subnet2a	eth0	10.1.1.3	10.1.1.1
subnet1b_pc	Ubuntu	subnet1b	eth0	10.2.0.3	10.2.0.1
subnet2b_pc	Ubuntu	subnet2b	eth0	10.2.1.3	10.2.1.1

* The script attaches the Docker networks to the VyOS containers in the specified order. However, on two occasions, I figured out that the Ethernet numbering of each container was altered and did not match my configuration. If you encounter this issue, please disconnect the networks from the VyOS containers before proceeding to the second stage and manually reattach them in the correct order.

■ Routers information

Router Name	Username	Password	SSH service	Https service	Input interfaces
central_router	vyos	swisscom	Yes : port 22	Yes : port 443	all
site_a_router	vyos	swisscom	Yes : port 22	Yes : port 443	all
site_b_router	vyos	swisscom	Yes : port 22	Yes : port 443	all

■ Network Access list

Networks	external	central_site_a	central_site_b	subnet1a	subnet2a	subnet1b	subnet2b
external	✓	---	---	---	---	---	---
central_site_a	---	✓	✓	✓	✓	✓	✓
central_site_b	---	✓	✓	✓	✓	✓	✓
subnet1a	✓	✓	✓	✓	✓	✓	---
subnet2a	---	✓	✓	✓	✓	---	✓
subnet1b	✓	✓	✓	✓	---	✓	✓
subnet2b	---	✓	✓	---	✓	✓	✓

■ Routing table and policies

Router name	Network	Destination		Description
		Gateway	Output interface	
central_router	10.1.0.0/23	-	eth0 (IPsec tunnel)	Excluded network
	10.2.0.0/23	10.0.0.2	vxlan0	
	0.0.0.0/0	172.16.0.1	eth2	
site_a_router	10.2.0.0/23	-	eth0 (IPsec tunnel)	Excluded network
	0.0.0.0/0	172.16.1.1		
site_b_router	0.0.0.0/0	10.0.0.1	vxlan0	

■ Challenges

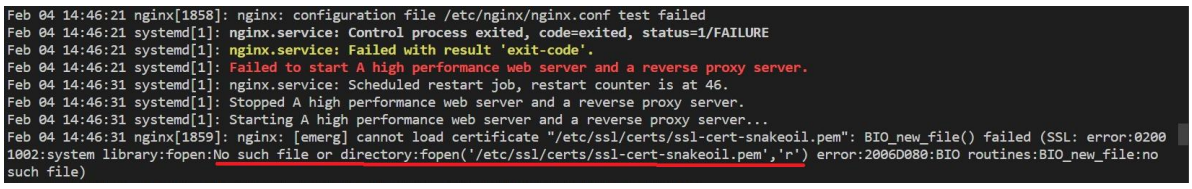
1- API service is available with VyOS default configuration

Due to the unavailability of the API service in the default configuration of VyOS, it is impossible to connect and configure the device via VyOS API. To address this, the script, after creating the router containers, injects a predefined config file into all routers. The config file can be found in the "config" folder.

2- The self-signed certificate was not found by the VyOS router

After enabling the API service, I encountered difficulty connecting to the router via the API. It took me around an hour to identify the issue, which was the absence of a self-signed certificate needed to enable the API service. This problem only occurred when VyOS was loaded into a docker container and I did not experience it when running VyOS on a virtual machine. The root cause could have been related to the Ubuntu operating system used in the VMware workstation. To resolve this issue, my script generates a self-signed certificate for each router after creation.

You can find the error displayed by the router in the image below.



3- The IPv6 is not enable on the docker containers with default settings.

To configure the VyOS interface, it is necessary to enable IPv6. This is done during the creation of the Docker network for all networks required to implement the network topology. However, since the VXLAN is created by the router and not Docker, I encountered an issue

with running the VXLAN configuration on the routers. To resolve this problem, the script enables IPv6 for the router containers after they have been created.

4- Docker DHCP service.

In our network topology, each router has its own designated default gateway. The DHCP service is enabled on the Docker networks, including macvlan and bridge networks. However, the default gateway specified by the Docker DHCP caused difficulties in implementing the topology. To address these issues, the script deletes the default gateway from the VyOS containers by injecting a shell command into them after creating the containers.