TEST PROJECT
MODULE C

IT NETWORK SYSTEMS ADMINISTRATION

KELOMPOK INFORMATION AND COMMUNICATION TECHNOLOGY

LOMBA KOMPETENSI SISWA
SEKOLAH MENENGAH KEJURUAN
TINGKAT NASIONAL KE XXX
TAHUN 2022

Deskripsi

A small UMKM in Indonesia just procured a lot of servers to support their containerized online services. They are hiring for new engineers to help manage the servers. This is the sample of their technical test for recruitment. You will need to utilize many tools to configure, manage, and deploy the infrastructure.

Credentials

Ubuntu
username : competitor/root
password : Skills39

Preconfigure

Already installed : ansible, python3.8, docker, nginx

Network Programmability

Scripted Configuration

- Prepare all other servers to be able managed by the deployer server.
    - Configure IP Address of all other servers and routers on the network according to the addressing table.
    - Create username admin with password cisco privilege 15 in router
    - Enable restconf on router

```
router# configure terminal
router (config)# restconf
```

- Create Python scripts to configure router via restconf
    - Create a script in /home/competitor/restconf/ipaddress.py
        - When the script is executed, it will set int loopback8888 with ip address 8.8.8.8/32

API url :
GET https:://<iprouter>/restconf/data/Cisco-IOS-XE-native:native/interface/Loopback=100

Response json :

```
{
    "Cisco-IOS-XE-native:Loopback": {
        "name": 100,
        "ip": {
            "address":{
                "primary":{
                    "address": "1.1.1.1",
                    "mask": "255.255.255.255"
                }
            }
        }
    }
}
```

Automation tools

Service Automation with ansible

- Access **Deployer with user root** to do tasks in this section
- Grant Deployer for access ssh to fw with no password
- Ansible already installed on Deployer
- Create structure folder for ansible playbook in /root/ansible/

- Create ansible tasks
    - Install nginx in **FW** server
    - Configure nginx loadbalancer roundrobin to srv1 and srv2, use templates for configuration

```
/roles/nginx/templates/loadbalancer.conf
---
upstream backend {
  server srv1.artemis.local:8081;
  server srv2.artemis.local:8082;
}

server {
  listen 80;
  server_name www.artemis.local;
  location / {
    proxy_pass http://backend;
  }
}
```

Containerized Services

Local Registry

- Access **srv1.artemis.local** to do tasks in this section
- Container Name : **proteus-registry**
- Use image **registry:2.7.0** from docker hub
- Volume mounts : /mnt/registry to /var/lib/registry
- Expose container port **1234**
- Enable https by using local self-signed CA
    - Generate CA using openssl
    - Common name : **srv1.artemis.local**
    - Country : ID
    - Organizational : ITNSA-CA
- Enable basic authentication with following credentials:
    - Username: devops
    - Password : lksn2022
- Make sure it's accessible from **srv2.artemis.local** server

Dockerize Python Application

- Access **srv2.artemis.local** to do tasks in this section
- Pull image python:3.8-slim from docker hub
    - Push this image to **srv1.artemis.local** with the same name&tag.
- Pull image nginx:latest from docker hub
    - Push this images to **srv1.artemis.local** container with name nginx:base
- Pull image httpd:latest from docker hub
    - Push this images to **srv1.artemis.local** container with name httpd:base
- Install python3-pip and other required packages to run the code main.py
- Save the following code in directory "/opt/sample-web"
    - main.py

```
from flask import Flask
from flask import request
from flask import render_template

sample = Flask(__name__)
@sample.route("/")
def main():
  return render_template("index.html")

if __name__ == "__main__":
  sample.run(host="0.0.0.0", port=8080)
```

- index.html

```
<html>
<head>
      <title>Sample app</title>
</head>
<body>
      <h1>You are calling me from
{{request.remote_addr}}</h1> </body>
</html>
```

- Use the code Python Web Application in /opt/sample-web to create a Dockerfile
  - Use base image python:3.8-slim
  - Expose port 8080
  - Copy all code into workdir.
  - Use /opt/sample-web as workdir inside the container.
  - Save the Dockerfile in /opt/sample-web/Dockerfile
  - Build the Dockerfile into image with name **proteusweb:latest**
  - Push the image to the **srv1.artemis.local** with the same name and tag.

Running Containers

- Running webserver on container based on **srv1** and **srv2**
  - Get the images **httpd:base** from local registry and run it as container
    - Container name : web-httpd
    - Make it accessible via **http://{container_ip_address}**
    - Do not expose any port on the host.
    - Save the following code in directory /home/competitor/httpd
      - httpd.html

```
<html>
<head>
      <title>HTTPD website</title>
</head>
<body>
      <h1>Welcome to LKSN 2022</h1></body>
</html>
```

  - Get the images **nginx:base** from local registry and run it as container
    - Container name : web-nginx
    - Make it accessible via **http://{container_ip_address}**
    - Do not expose any port on the host.
    - Save the following code in directory /home/competitor/nginx
      - nginx.html

```
<html>
<head>
      <title>NGINX website</title>
```

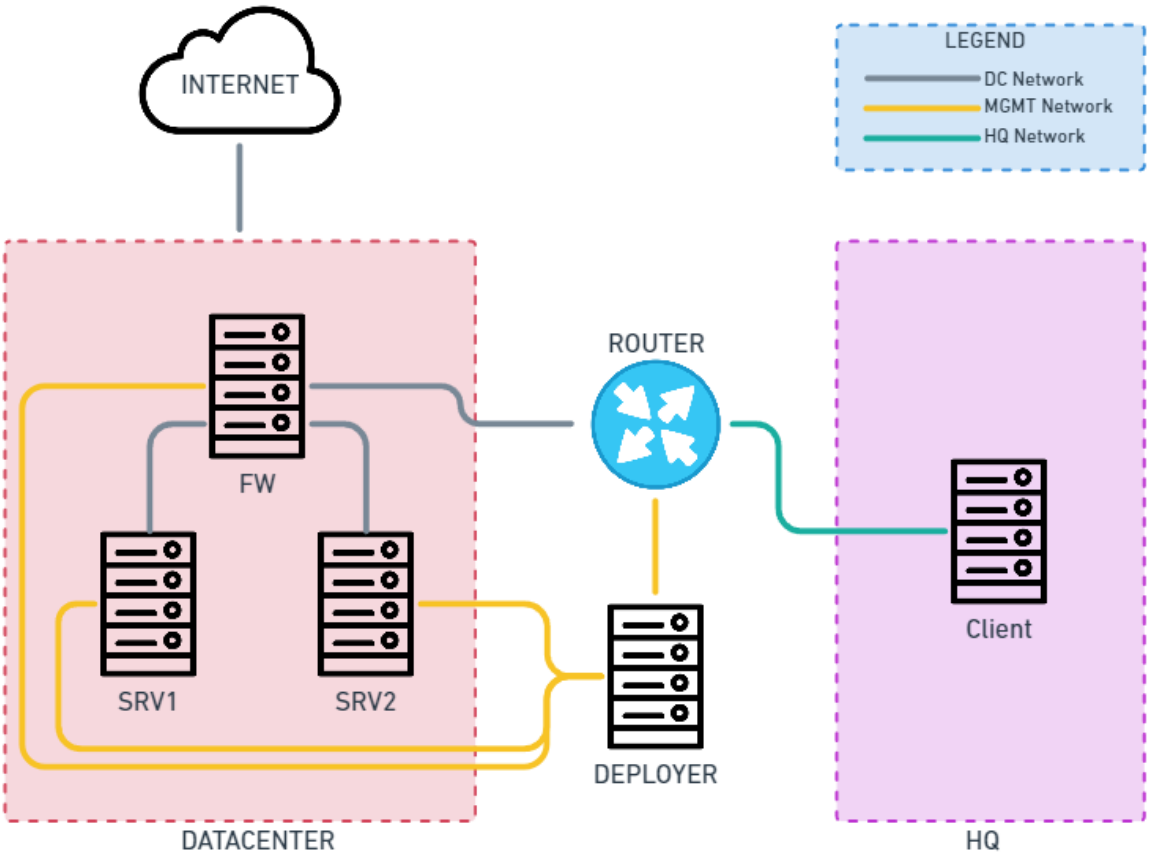
```
</head>
<body>
      <h1>Welcome to LKSN 2022</h1></body>
</html>
```

- Running simple python Application
    - On **srv1** get the image **proteusweb:latest** from local registry and run it as container
        - Container name : proteus-web
        - Expose via host port 8081
    - On **srv2** get the image **proteusweb:latest** from local registry and run it as container
        - Container name : proteus-web
        - Expose via host port 8082

Appendix

Device	IP Address	OS
router	172.16.10.1/24 (DC) 192.168.10.1/24 (HQ) 10.10.10.10/24 (MGMT)	csr
deployer	10.10.10.1/24 (MGMT) DHCP (inet)	Ubuntu 20.04
FW	172.16.10.2/24 (DC) DHCP (inet) 10.10.10.2/24 (MGMT)	Ubuntu 20.04
srv1	172.16.10.3/24 (DC) DHCP (inet) 10.10.10.3/24 (MGMT)	Ubuntu 20.04
srv2	172.16.10.4/24 (DC) DHCP (inet) 10.10.10.4/24 (MGMT)	Ubuntu 20.04
client	192.168.10.100/24 (HQ)	Ubuntu 20.04

Topology


INTERNET
LEGEND
DC Network
MGMT Network
HQ Network
ROUTER
FW
SRV1
SRV2
DEPLOYER
Client
DATACENTER
HQ