Test Project

IT Network System Administration

Module C – Infrastructure programmability and automation

Submitted by:
Thushjandan Ponnudurai CH
Jun Tian CN
Anthony Lebarbanchon FR
Hamed Kargarzadeh IR
John Leong SG
Janos Csoke HU
Gen Lee EE

WSC2022_TP39_MC_actual_en

Contents

Introduction to Test Project

The following is a list of sections or information that must be included in all Test Project proposals that are submitted to WorldSkills.

- Contents including list of all documents, drawings and photographs that make up the Test Project
- Introduction/overview
- Short description of project and tasks
- Instructions to the Competitor
- Equipment, machinery, installations and materials required to complete the Test Project
- Marking Scheme (incl. assessment criteria)

Introduction

This Test Project consists of the following documentation/files:

- WSC2022_TP39_EN_Mod_C.docx
- hosts (on HOST VM: /etc/ansible)
- users.csv (on HOST VM: /etc/ansible)
- .vault_pass (on HOST VM: /etc/ansible)
- customers.json (on HOST VM: /etc/ansible)
- api_cisco_devices.json (on HOST VM: /etc/ansible)
- WSCSE2022_Module_C_API.postman_collection.json (on DEV VMs home directory)

Accurate and up-to-date documentation has always been a challenge in IT. With multiple engineers working on the same systems, it is hard to keep track of who changed what. Applix Corporation decided to fix this problem and hired you to modernize, harden and extend their infrastructure.

Documentations

The following documentations installed to DEV VMs. You can use it with Zeal Docs.

- Ansible (version 2.10.8)
- Python 3 (version 3.9.2)
- Jinja (version 2.11.3)
- Flask (version 1.1.2)
- Django (version 2.2.7)
- Yang Models for Cisco CSR
- FastAPI (version 0.63.0)

Description of project and tasks

You will be migrating VMs to Infrastructure as Code (IaC) and simplify the process of creating new services. You have access to development VMs (DEV-LIN & DEV-WIN). These VMs can be used for developing and testing your work.

Login for all VMs and Devices:

Username Linux:	root / appadmin
Username Windows:	Administrator / appadmin
Username Cisco:	appadmin
Username CML:	admin
Password:	P@ssw0rd

All VMs and devices are connected to the management network (10.22.0.0/24) and have a statically configured IP address. All IP addresses in network table will not change for marking. The management network will be used for configuring the different hosts. You can login using username and password over SSH, HTTPS or WinRM.

You may install any additionally required packages and features on the VMs.

Instructions to the Competitor

Part 1. Linux

Use Ansible to configure the Linux hosts LIN[1-5] from HOST VM. There is a preconfigured hosts file located under /etc/ansible/hosts. **Do not change this file.** Before assessment all LIN[1-5] VMs will be reset to original state and the LIN[1-5] VMs will be randomly removed and added to different groups in the hosts file. For marking, all playbooks will be run in order using the command "ansible-playbook playbookname.yml" in the /data/ansible/linux directory. Variables like "hostname", "webport" and "webcolor" in /etc/ansible/hosts are subject to change for marking.

You can connect the Debian 11.3 BD ISOs 1-4 to the VMs.

General

- Create a directory /data/ansible/linux
  - All playbooks should be located at the root of this directory
  - You are free to create folders/files in this directory for running the playbooks
  - All tasks should have state "ok" or "skipped" after running the playbooks a second time
  - Make sure, ansible username and password are encrypted using the /etc/ansible/.vault_pass file and are automatically decrypted when running the playbooks

Hostname

- Create a playbook called 1-hostname.yml for configuring the hostname
  - All hosts should receive the hostname based on the "hostname" variable in /etc/ansible/hosts file

nftables

- Create a playbook called 2-nftables.yml for filtering incoming traffic on all LIN hosts.
  - Incoming packets should be dropped by default
  - Allow minimal traffic for the services to work
  - Configure a list named "trusted" which contains HOST VM and DEV VMs and use it in the firewall rules
  - Allow SSH and ICMP traffic from member of the "trusted" list to the LIN hosts
  - Make sure, that nftables persist across reboots

DNS

- Create a playbook called 3-dns-server.yml for configuring two or more DNS servers
- Install a DNS service on all hosts in the group “dns”
    - The first host in “dns” group should be master for the domain “applix.com”
    - All other hosts in this group should be slave DNS servers for domain “applix.com”
    - Make sure, all hosts in /etc/ansible/hosts have an A record for <hostname>.applix.com
    - intranet.applix.com should resolve to 10.22.0.51
- Create a playbook called 4-dns-client.yml
    - Make sure, that all LIN Servers and HOST use first DNS host as primary DNS server and all slave DNS servers as secondary nameserver
    - Configure DNS suffix “applix.com”

Web

- Create a playbook called 5-web-server.yml for configuring two or more web servers
- Install a web service on all hosts in the group “web”
    - The local website should listen on port “webport” variable in /etc/ansible/hosts
    - Display the following content with textcolor based on the “webcolor” variable in /etc/ansible/hosts file

      “Hello from <hostname> !”
    - Create a virtual host listening on port 8081 called “intranet.applix.com” displaying the following content

      “Welcome to the intranet of Applix”
      “This site was served by <hostname>”

High availability intranet

- Create a playbook called 6-ha-intranet.yml for configuring two or more HA servers
- Install keepalived and HAProxy on all hosts in the group “ha”
    - Configure 10.22.0.51 as floating IP and use the last host in “ha” group as VRRP master
    - If the master fails, the second last server in the ha group should take over and so on (priority in reverse order)
    - Use password authentication
    - Configure HAProxy to load balance “http://intranet.applix.com” between all available web servers using round robin
        - Add Header “x-haproxy-host” with the hostname of current HAProxy host

Users

- Create a playbook called 7-users.yml for importing users on all LIN hosts
    - Import the users from /etc/ansible/users.csv on all LIN hosts
    - Make sure, that password is not changed if there is already an existing user with same username and UID

File Share

- Create Ansible playbook called 8-nfs.yml to create NFS shares on host group "nfs".
    - Create a NFS share for each item in the variable "shares" in /etc/ansible/hosts. The variable "shares" is a list of items, where every item has the attributes "name" and "size".
    - Each share must use a new logical volume with a specified size.
    - Use directory /nfs followed by share name to mount the logical volume.
    - The share must be protected by limiting access only by HOST VM.
    - You can use any filesystem to format the logical volume.

DNS Backup

- Create a playbook 9-backup.yml for configuring DNS server backup. Configure backups on all hosts in the group "dns" and use hosts in the group "backup" as backup destination.
    - Configure a systemd service called backup_dns.service to backup DNS zones
    - Configure systemd timer called backup_dns.timer to run backup job every 5 minutes
    - Store backups at /backups/ folder with file format YYYY-MM-DD_hh-mm-ss_[hostname].tar.gz

Part 2. Windows

Use Ansible to configure the Windows hosts WIN[1-5] from HOST VM. There is a preconfigured hosts file located under /etc/ansible/hosts. **Do not change this file**. Before assessment all WIN[1-5] VMs will be reset to original state and the WIN[1-5] VMs will be randomly moved to different groups in the hosts file. For marking, all playbooks will be run in order using the command "ansible-playbook playbookname.yml" in the /data/ansible/windows directory. Variables like "hostname", "RootCAPriv" in /etc/ansible/host are subject to change.

General

- Create a directory /data/ansible/windows
    - All playbooks should be located at the root of this directory
    - You are free to create folders/files in this directory for running the playbooks
    - Make sure ansible username, password and certificate secrets are encrypted and automatically decrypted when running the playbooks

Hostname

- Create a playbook called 1-hostname.yml for configuring the hostname
    - All hosts should receive the hostname based on the "hostname" variable in /etc/ansible/hosts file

Root certificate

- Create a playbook called 2-cert.yml for configuring certificates
    - Create a root certificate with the following properties
        - Common Name = Applix Root CA
        - Organization Name = Applix Corporation
        - Country Code = KR
    - Distribute the root certificate to all Windows hosts as trusted root CA in computer store
        - Make sure that the appropriate Windows hosts based on the "RootCAPriv" variable in /etc/ansible/hosts file have the certificate in the personal computer store with the private key

Security and logging

- Create a playbook called 3-sec-log.yml for configuring security settings
    - Stop and disable the Remote Desktop Service on all Windows hosts
    - Create and scheduled task called LogUptime to append the current uptime to C:\uptime.txt in the format given below every 30 seconds
      XX days XX hours XX minutes XX seconds
      XX days XX hours XX minutes XX seconds
      XX days XX hours XX minutes XX seconds

Management tools

- Create a playbook called 4-tools.yml for installing Windows tools
    - Install the telnet client on servers with a GUI installation of Windows

Customer Deployments

- Create a playbook called 5-environment.yml for installing the customer environment
    - Configure all servers in group "dc" as DNS servers on all WIN hosts
    - Configure all servers in group "dc" as domain controller
        - Use customers.com as domain name
        - Use "P@ssw0rd" as safe password
    - Join all non-DC Windows hosts to the domain
    - Install IIS feature on all servers in group "iis"
- Create a playbook called 6-customers.yml for deploying customer web environments
    - For each customer in /etc/ansible/customers.json file
        - Create an OU based on "name" attribute
        - Create an AD user based on "username" and "password" attribute in this OU
        - Create a DNS entry for domain_prefix.customers.com pointing to a random IIS server and use domain_prefix as randomization seed
        - Create a virtual host listening on port 80 on the selected IIS server which displays the "message" attribute

Part 3. Network

Use Ansible to configure the Cisco routers RTR[1-8] from HOST VM. There is a preconfigured hosts file located under /etc/ansible/hosts. **Do not change this file**. Before assessment all RTR[1-8] routers will be wiped with bootstrap config. For marking, all playbooks will be run in order using the command "ansible-playbook playbookname.yml" in the /data/ansible/cisco directory. Variables like "bgp_as_internal", "bgp_as_external" and "loop_net" in /etc/ansible/hosts are subject to change.

General

- Create a directory /data/ansible/cisco
    - All playbooks should be located at the root of this directory
    - You are free to create folders/files in this directory for running the playbooks
    - All tasks should have state "ok" or "skipped" after running the playbooks a second time
    - Make sure ansible username, password and eBGP peering secret are encrypted and automatically decrypted when running the playbooks

Hostname

- Create a playbook called 1-hostname.yml for configuring the hostname
    - All routers should receive the hostname based on the "hostname" variable in /etc/ansible/hosts file
    - When the variable of hostname is not defined in /etc/ansible/hosts, the inventory name of the Cisco host should be used as hostname.
    - Only when the device's hostname is changed, running-config will be automatically saved as startup-config.

Security

- Create a playbook called 2-security.yml for configuring access to management interface
    - Allow SSH, ICMP and HTTPS Traffic only from HOST and DEV VMs to Gig1

Loopback interfaces

- Create a playbook called 3-loopback.yml for configuring the loopback interfaces
    - All routers should receive one IP address from the "loop_net" subnet in /etc/ansible/hosts
    - Use loopback Interface 0 and a prefix length of /32
    - Use description "Ansible - Routing Loopback Interface"
    - The first router in /etc/ansible/hosts should receive the first available IP address of "loop-net" and so on

Internal routing

- Create a playbook called 4-igp.yml for configuring the internal dynamic routing
    - You may use any modern IGP routing protocol
    - All internal and edge routers should participate in the routing process
    - Advertise loopback interface 0 and the LAN network (10.0.0.0/24)

Internal BGP routing

- Create a playbook called 5-ibgp-lan.yml for configuring the internal dynamic routing
    - All internal and edge routers should participate in full-mesh internal AS based on the "bgp_as_internal" variable in /etc/ansible/hosts file
    - Use loopback 0 for peering
    - Advertise loopback 0 and LAN network (10.0.0.0/24) on each router
- Create a playbook called 6-ibgp-wan.yml for configuring the internal dynamic routing of external routers
    - All external routers should participate in full-mesh internal AS based on the "bgp_as_external" variable in /etc/ansible/hosts file
    - Advertise loopback 1 and 99 on each router

eBGP peerings

- Create a playbook called 7-ebgp-peering.yml for configuring the eBGP peering
    - Create a full-mesh of eBGP peerings between the edge and external routers
    - Use password authentication with string "EBGP$Secret"
    - Make sure that eBGP peerings are load balanced

Connectivity test

- Make sure, that all internal and edge routers can ping loopback interfaces of the external routers

Config backup

- Create an empty git repository at /data/cisco/backup on the HOST VM
- Create a playbook called 8-backup.yml for backing up the running configuration and tracking changes
    - Name the files <RTR ID>.cfg in the backup folder
    - Create a git commit with the comment "Automated Ansible backup YYYY-mm-dd hh:mm:ss"
    - A commit should only be created when any config file has changed

Part 4. API

Create a Python API for querying data about the infrastructure. The API should be hosted on the VM **HOST** and all API files should be located under /data/api. You are free to use any modules or framework which are available on the Debian ISOs.

- The API should be listening to HTTPS on port 443 on the IP 10.22.0.50
    - Make sure, that your API is reachable under https://api.applix.com/ from HOST and DEV VMs
    - There should be no certificate warnings in Edge & Firefox browser on DEV VMs
- Create a systemd service called "applix-api"
    - The service should start on boot
    - The API should be controllable using the "start, restart, stop" systemd commands
- Endpoints general
    - Make sure, that the API returns information about the content format

- **Endpoint /customers/ (PUT)**
  - This endpoint is used for deploying new customers
  - This endpoint requires following parameters in the HTTP body as JSON
    - name = customer name
    - domain_prefix = prefix for domain
    - username = Active Directory username
    - password = Active Directory password
    - message = text displayed on website
  - This endpoint should add the new customer to the /etc/ansible/customers.json file
    - Make sure that name, domain_prefix and username are unique
    - If they are not, return a HTTP Forbidden error
  - After the customer has been added, the 6-customers.yml playbook should be run automatically
- **Endpoint /network/devices/stats (GET)**
  - This endpoint should return statistics of all active (up) interfaces from the cisco network devices in JSON format
  - Query the data from the routers using RESTCONF (use unicast packets for pkts-in/pkts-out)

```
{
        "devices": [{
                "hostname": "X",
                "management_ip": "XXX.XXX.XXX",
                "ios_version": "XX.XX",
        "interfaces": [
        {
          "name": "GigabitEthernet1",
          "mac": "XX:XX:XX:XX:XX:XX",
          "ip": "XXX.XXX.XXX.XXX",
          "pkts-in": "XX",
          "pkts-out": "XX"
        },
        {
          "name": "Loopback0",
          "mac": "XX:XX:XX:XX:XX:XX",
          "ip": "XXX.XXX.XXX.XXX",
          "pkts-in": "XX",
          "pkts-out": "XX"
        }, ....
      ]
        }, {
                "hostname": "X",
                "management_ip": "XXX.XXX.XXX.XXX",
                "ios_version": "X", ....
        }, .... ]
}
```

  - It should be possible to filter the output using query parameter management_ip=XXX.XXX.XXX.XXX
    - If no device has been found, the API should return HTTP not found error

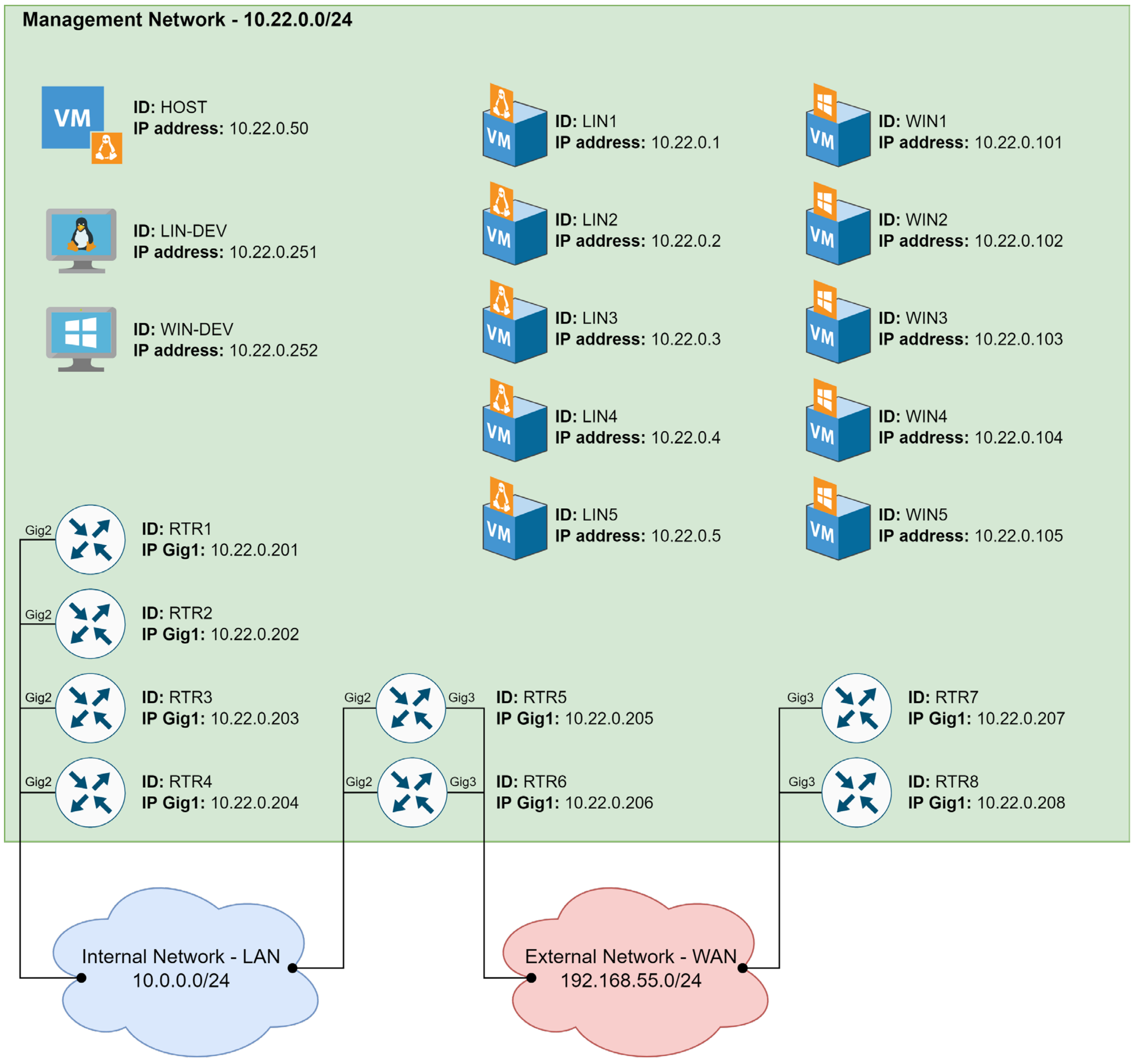
Equipment, machinery, installations, and materials required

Network table

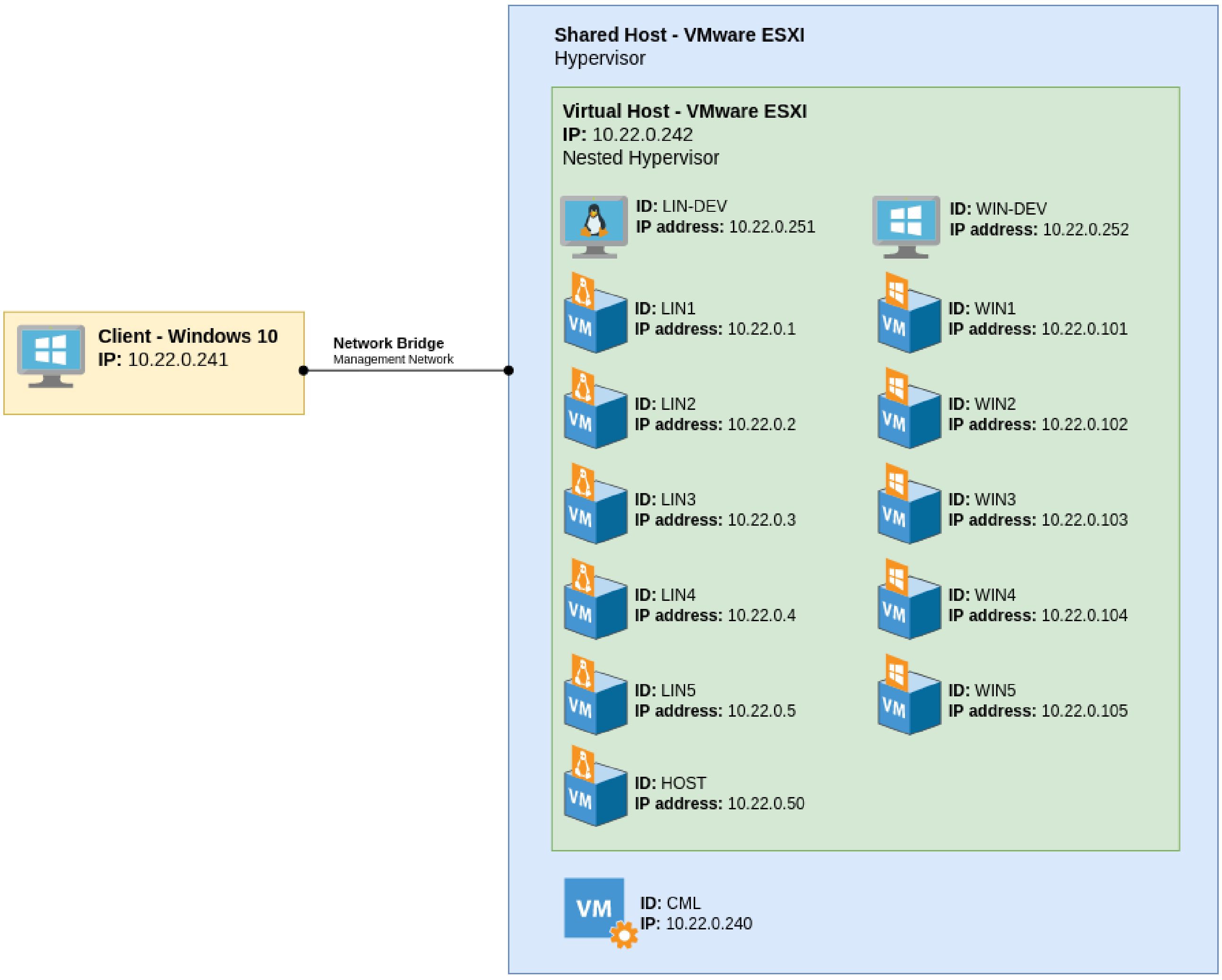
ID	IP	OS	DESCRIPTION
DEV-LIN	10.22.0.251	Debian 11.3 (GNOME)	Development VM with the following software installed: - Postman - Python3
DEV-WIN	10.22.0.252	Windows 10 Pro	Development VM with the following software installed: - Postman - Python3
HOST	10.22.0.50	Debian 11.3	Host Server
LIN1	10.22.0.1	Debian 11.3	Dynamic Config
LIN2	10.22.0.2	Debian 11.3	Dynamic Config
LIN3	10.22.0.3	Debian 11.3	Dynamic Config
LIN4	10.22.0.4	Debian 11.3	Dynamic Config
LIN5	10.22.0.5	Debian 11.3	Dynamic Config
WIN1	10.22.0.101	Windows Server 2019	Dynamic Config
WIN2	10.22.0.102	Windows Server 2019 (Core)	Dynamic Config
WIN3	10.22.0.103	Windows Server 2019 (Core)	Dynamic Config
WIN4	10.22.0.104	Windows Server 2019	Dynamic Config
WIN5	10.22.0.105	Windows Server 2019 (Core)	Dynamic Config
RTR1	Gig1 - 10.22.0.201 Gig2 - 10.0.0.1	Cisco CSR1000v	Internal - Dynamic Config
RTR2	Gig1 - 10.22.0.202 Gig2 - 10.0.0.2	Cisco CSR1000v	Internal - Dynamic Config
RTR3	Gig1 - 10.22.0.203 Gig2 - 10.0.0.3	Cisco CSR1000v	Internal - Dynamic Config
RTR4	Gig1 - 10.22.0.204 Gig2 - 10.0.0.4	Cisco CSR1000v	Internal - Dynamic Config

RTR5	Gig1 - 10.22.0.205 Gig2 - 10.0.0.5 Gig3 - 192.168.55.1	Cisco CSR1000v	Edge - Dynamic Config
RTR6	Gig1 - 10.22.0.206 Gig2 - 10.0.0.6 Gig3 - 192.168.55.2	Cisco CSR1000v	Edge - Dynamic Config
RTR7	Gig1 - 10.22.0.207 Gig3 - 192.168.55.3 Loop1 - 1.1.1.1 Loop99 - 99.99.99.99	Cisco CSR1000v	External - Dynamic Config
RTR8	Gig1 - 10.22.0.208 Gig3 - 192.168.55.4 Loop1 - 8.8.8.8 Loop99 - 99.99.99.99	Cisco CSR1000v	External - Dynamic Config
CML	10.22.0.240	Cisco Modelling Lab	Web Access

Network diagram (logical)



Network diagram (physical)


Shared Host - VMware ESXI
Hypervisor
Virtual Host - VMware ESXI
IP: 10.22.0.242
Nested Hypervisor
ID: LIN-DEV
IP address: 10.22.0.251
ID: WIN-DEV
IP address: 10.22.0.252
ID: LIN1
IP address: 10.22.0.1
ID: WIN1
IP address: 10.22.0.101
ID: LIN2
IP address: 10.22.0.2
ID: WIN2
IP address: 10.22.0.102
ID: LIN3
IP address: 10.22.0.3
ID: WIN3
IP address: 10.22.0.103
ID: LIN4
IP address: 10.22.0.4
ID: WIN4
IP address: 10.22.0.104
ID: LIN5
IP address: 10.22.0.5
ID: WIN5
IP address: 10.22.0.105
ID: HOST
IP address: 10.22.0.50
ID: CML
IP: 10.22.0.240
Client - Windows 10
IP: 10.22.0.241
Network Bridge
Management Network