TecnoCampus
Escola Superior
Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

AUTOMATITZACIÓ DE XARXES

Memòria

ARNAU AMARGANT DOMENE
TUTOR: PERE BARBERAN AGUT

CURS ACADÈMIC
2020-2021

TecnoCampus
Mataró-Maresme

Abstract

The complexity and dimensions of the networks of many organizations today are increasing more and more, either because society is moving towards a more digital world or because organizations have more competition, more agile and automated methods are required for the management of these. The objectives of this project has been to research and implement different protocols and tools for the automation and monitoring of networks such as Ansible, RESTCONF and NETCONF.

Resum

La complexitat i dimensions de les xarxes d'avui en dia de moltes organitzacions està incrementant cada dia més, ja sigui perquè la societat va cap a un món més digital o perquè les organitzacions tenen més competència, es requereixen mètodes més àgils i automatitzats per la gestió d'aquestes. Els objectius d'aquest treball han estat investigar i implementar diferents protocols i eines per l'automatització i monitoratge de xarxes com Ansible, RESTCONF i NETCONF.

Resumen

La complejidad y dimensiones de las redes de hoy en día de muchas organizaciones está incrementando cada día más, ya sea porque la sociedad va hacia un mundo más digital o porque las organizaciones tienen más competencia, se requieren métodos más ágiles y automatizados para la gestión de estas. Los objetivos de este trabajo han sido investigar e implementar diferentes protocolos y herramientas para la automatización y monitorización de redes como Ansible, RESTCONF y NETCONF.

Índex

Índex de figures

Índex de taules

1. Objecte del projecte

L'objecte del present projecte és el d'aplicar i investigar els diferents mecanismes de programació i automatització de xarxes amb l'objectiu de dissenyar topologies de xarxa automatitzada i veure els avantatges que aquestes comporten respecte a xarxes no automatitzades.

En els darrers anys les empreses s'han vist cada cop més obligades a ser eficients i dinàmiques en totes les seves àrees, una d'elles és la d'administració de xarxes, amb un increment del nombre de hardware a les xarxes i augment de la seva complexitat es fa més complicat fer tasques de manteniment, configuracions i actualitzacions, és per això que s'està promovent l'automatització d'aquestes, ja que comporta beneficis com poden ser millorar l'eficiència, reduir errors humans i despeses operatives entre d'altres.

Finalment, el que es vol deixar clar amb aquest projecte és la raó principal per la qual s'hauria d'automatitzar una xarxa que seria aconseguir una xarxa més fiable que permeti l'escalabilitat i la reducció de costos, de personal i temps operacional serien objectius secundaris.

2. Estudi previ: context, antecedents i necessitats d'informació

Les primeres xarxes de comunicacions es van elaborar per realitzar les transmissions telefòniques i consistien en uns circuits on manualment es canviaven els canals de transmissió per connectar l'emissor i el receptor, com podem veure era una tasca lenta i inconvenient.

A principis dels anys 60 els ordinadors van començar a guanyar popularitat per la seva gran utilitat i es va començar a idear mecanismes per poder emetre informació entre diferents ordinadors, va ser quan Leonard Kleinrock va publicar el primer document sobre la teoria de "packet switching" [1] on explicava la viabilitat de fer servir paquets de dades en comptes de circuits per la comunicació entre ordinadors, per un altra banda dos equips d'investigadors més havien arribat a la mateixa conclusió independentment [2]. A partir d'aquí es va començar a desenvolupar una sèrie de protocols (TCP/IP, UDP) que faria possible la comunicació efectiva entre els ordinadors.

A mesura que van anar apareixent noves invencions com la tecnologia Ethernet i les xarxes van anar incrementant d'usuaris es va anar adaptant i incorporant nous protocols i conceptes a les xarxes, una d'elles va ser el "World Wide Web" un sistema d'informació on s'identifiquen documents i recursos, a més a més el mateix inventor Sir Timothy Berners-Lee va crear el primer buscador de la història.

Des d'aleshores internet s'ha estes arreu del món i més de la meitat de la població mundial es activa diàriament en aquesta tecnologia [3]. Això comporta que les xarxes s'hagin anat adaptant a aquest ús massiu d'usuaris augmentant la seva complexitat i les seves dimensions.

Durant l'última dècada les xarxes de les organitzacions són les que s'han vist més afectades i posades al límit per la tendència en l'increment d'aplicacions al cloud, IoT (Internet of things), centres de dades, estadístiques i analítiques, [4] això ha comportat que les xarxes s'hagin d'estendre per suportar aquestes càrregues. Com a

conseqüència l'administració de les xarxes s'ha tornat una tasca repetitiva i difícil d'escalar i gestionar.

A més a més un dels principals inconvenients de les xarxes són els errors humans, fins a un 80% de falles a les xarxes son per errors de configuracions, dimensionament incorrecte de les carregues de treball dels servidors, etc... [5].

Un dels altres problemes de les xarxes és que no es poden controlar des d'un sol punt centralitzat, s'ha de gestionar cada dispositiu independentment. Aquest es un dels problemes que es va trobar Martin Casado, un dels creadors del protocol OpenFlow, quan treballava pel govern dels Estats Units. OpenFlow es un protocol que permet desvincular el panell de control del dispositiu del panell de dades fent possible la configuració de les taules de flux i conseqüentment modificant com el dispositiu reenvia els paquets a la xarxa [6]. Es diu que aquest protocol ha sigut el catalitzador de la revolució de les xarxes definides per software i ha obert els ulls a les organitzacions i enginyers de xarxes de que es necessita un canvi en la manera en què es gestiona la xarxa, ja que no ha canviat en els últims 20 anys.

Però que es l'automatització de xarxes? **"Network programmability and automation** is about simplifying the tasks involved in configuring, managing and operating network equipment, network topologies, network services, and network connectivity**"** [7]**.**

L'automatització de xarxes substitueix majoritàriament el hardware pel software, en comptes de tenir routers o switchos tenim màquines virtuals que actuen com a tal, això disminueix el temps d'aprovisionar nous serveis eliminant la necessitat d'instal·lar, cablejar i aprovisionar el hardware a l'entorn.

Fer-ho amb software és tan fàcil com implementar una nova màquina virtual al servidor amb la possibilitat de clonar-la i poder fer còpies de seguretat.

Un dels inconvenients per això és que molts dels venedors tradicionals de hardware de xarxes limiten les seves aplicacions virtuals, ja que el seu model de negoci primari és la venda de hardware [7, p. 15], però això està canviant amb l'increment de la demanada d'aplicacions virtuals i la competència més forta.

Les xarxes definides per software et permeten interactuar amb els dispositius de la xarxa a través d'APIs, fent possible la interacció i recollida d'informació del dispositiu en temps real, fent més fàcil la feina dels enginyers de xarxes o desenvolupadors de software que poden crear scripts per testejar noves topologies, noves característiques de la xarxa o validar noves configuracions.

Un dels altres beneficis de les xarxes definides per software i el més comú a les xarxes automatitzades és la configuració dels dispositius, abans s'ha parlat que l'error més comú de les falles de xarxa són els errors humans, doncs bé, amb l'automatització de xarxes és possible fer scripts o "templates" i abans d'executar-lo realitzar testos contra el script per assegurar-nos que es correcte.

A més d'això, en el moment en què comencem a extreure informació sobre la xarxa en temps real es poden crear informes i arribar a conclusions d'una manera automàtica, amb la possibilitat de programar de què quan passi un esdeveniment la xarxa es reorganitzi automàticament.

NETCONF és un protocol que s'utilitza per a la comunicació entre client-servidor quan es fan crides amb l'API i encara que no és un protocol nou ha guanyat popularitat amb l'arribada de les xarxes definides per software, ja que més dispositius el suporten i s'han desenvolupat més funcionalitats. NETCONF utilitza una codificació en format XML i el seu funcionament és per transaccions indicant que si es realitzen vàries ordres de configuració a la vegada o totes es realitzen amb èxit o cap ordre és aplicada.

Un altre protocol de comunicació entre client-servidor és RESTCONF on la seva crida d'API és generada a partir de models de dades YANG suporta diferents tipus d'ordres com per exemple GET, PUT, PATCH, POST i DELETE fent-la molt útil per certes operacions a la xarxa. Aquest protocol suporta models de dades JSON i XML.

A mesura de què la necessitat de trobar maneres més eficients i ràpides per administrar i gestionar la xarxa ha incrementat, han aparegut diferents plataformes per fer-ho com és Ansible, que és un programa open-source per la gestió, configuració i desplegament d'aplicacions que permet la programació i visualització de tota la xarxa des d'un punt centralitzat.

Finalment, l'automatització de xarxes està sorgint aquests últims anys com el següent pas a fer per a les organitzacions si volen sobreviure en un món on cada vegada es depèn més de la xarxa. De fet es preveu que el valor del mercat d'automatització de xarxes el 2025 es multipliqui per tres [8].

3. Objectius i requeriments

La finalitat d'aquest projecte és el de construir una xarxa automatitzada per comprovar els beneficis que comporta i conèixer les diferents eines i mecanismes amb què es pot realitzar.
L'objectiu és aconseguir poder configurar i monitorar els diferents dispositius virtuals de la xarxa des d'un punt centralitzat mitjançant scripts programats en Python amb la utilització de diferents models i formats de dades (YAML, JSON, XML i YANG) i mitjançant l'eina d'automatització Ansible. En aquest projecte no es pretén fer la realització d'un producte.
S'entén per node qualsevol dispositiu connectat a la xarxa, poden haver-hi de dos tipus: els de comunicació de dades com pot ser un router, hub o switch i per altra banda els equips de dades terminals com pot ser un ordinador, telèfon o impressora.

Objectius principals:

- Dissenyar, automatitzar i monitorar una xarxa de 75 nodes per comprovar els avantatges que s'obtenen en temps operacional, temps d'implementacions, instal·lacions i gestió de diferents protocols i paquets.
- Conèixer i implementar diferents mecanismes i eines per automatitzar una xarxa com Ansible, RESTCONF i NETCONF.

Objectius secundaris:

- Construir una xarxa de tipus campus
- Exposar els beneficis que comporta l'automatització de la xarxa
- Conèixer i aplicar el procés d'implementació de configuració d'una xarxa mitjançant els formats de dades JSON, XML i YAML
- Conèixer el funcionament i aplicar els diferents models de dades YANG.
- Conèixer i utilitzar Cisco Modeling Labs per la implementació de la xarxa.

3.1 Definició de requeriments funcionals i tecnològics

Requeriments Funcionals:

- Permetre la configuració dels nodes mitjançant el protocol NETCONF i model de dades XML.
- Permetre l'extracció de dades dels nodes mitjançant el protocol NETCONF i model de dades XML.
- Permetre la configuració dels nodes mitjançant el protocol RESTCONF amb model de dades JSON.
- Permetre l'extracció de dades dels nodes mitjançant el protocol RESTCONF.
- Permetre la configuració dels dispositius de la xarxa mitjançant el programa Ansible.
- Instal·lar model de dades YANG automàticament en els dispositius amb Ansible.
- Extreure dades amb NETCONF i mostrar per pantalla en forma de taula.
- Extreure dades amb RESTCONF i mostrar per pantalla en forma de taula.
- Construir una topologia de tipus campus.
- Importar la topologia en el laboratori amb l'API del laboratori.
- Explicar el funcionament del protocol NETCONF.
- Explicar el funcionament del protocol RESTCONF.
- Explicar el funcionament de l'eina d'automatització Ansible.
- Explicar el funcionament dels models de dades YANG.

Requeriments Tecnològics:

S'utilitza la configuració de l'ordinador següent per recomanació de la plataforma Cisco Labs que és amb la que es realitza la xarxa automatitzada [9].

- Disposar d'un ordinador de 8gb de memòria RAM o superior.

- Disposar d'un ordinador amb processador de quatre nuclis.
- Disposar d'un ordinador amb memòria de disc dur de 16gb o superior.
- Disposar del programa Microsoft Word
- Disposar de connexió a internet.
- Disposar del llenguatge de programació Python.
- Disposar del programa de *software* Ansible
- Disposar del programa Postman.
- Disposar del programa Cisco AnyConnect Secure VPN.

4. Metodologia

S'han considerat diferents tipus de metodologies per la realització d'aquest projecte com per exemple Agile o Waterfall, però es creu que aquestes metodologies són més útils en desenvolupament de software s'ha decidit decantar-nos per la metodologia que ens proporciona l'empresa de xarxes Cisco Systems que es basa en sis etapes: Preparar, Planificar, Dissenyar, Implementar, Gestionar i Optimitzar [10].

Les estratègies de cerca d'informació és generalment cerca per internet amb paraules clau per trobar articles, informes, estudis i llibres relacionats amb el tema, per avaluar la informació cercada i saber si és legítima es recerca l'autor o l'organització per saber el seu perfil i es fa una avaluació de la seva experiència, coneixements o documentació.

Aquesta avaluació comporta posar en comú tota la informació cercada sobre l'autor o l'organització i analitzar-la comprovant el disseny de la pàgina web, si té informació de contacte, si la informació coincideix amb altres textos ja analitzats, la gramàtica i puntuació.

Així mateix es té en compte sota quin domini es publica la informació, es pensa que si el domini de nivell superior es .edu o .gov es pot confiar amb la informació en canvi si és .com s'avalua com s'explica anteriorment.

Aquest projecte es realitza en tres etapes definides que són el lliurament de l'Avantprojecte el 12 de febrer, lliurament de la memòria intermèdia el 22 d'abril i lliurament de la documentació final el 18 de juny.

5. Desenvolupament

5.1 Eines

5.1.1 Cisco Modeling Labs

El començament del desenvolupament del present projecte ha estat fer un estudi de les eines amb les que es realitzen les xarxes automatitzades i el seu funcionament.

Cisco Systems és una empresa de hardware de xarxes que disposa una plataforma de simulació de xarxes en línia que s'executa als seus servidors i on es poden dissenyar, testejar i configurar xarxes amb màquines virtuals que simulen el hardware de xarxa com poden ser routers, switchos, ordinadors, servidors, etc..., de manera gratuïta.

La plataforma s'anomena Devnet Sandbox Labs i és una eina on s'ha de reservar una franja horària per poder-hi fer ús. Aquesta eina disposa de diferents entorns virtuals que s'anomenen 'sandbox' especialitzats en diferents temes com seguretat, IoT, centres de dades, Cloud i xarxes.

Primerament s'han realitzat alguns dels laboratoris d'aprenentatge que ens proporciona Cisco basats en aquestes plataformes, són laboratoris on es treballa l'ús d'Ansible, que és una eina d'automatització i configuració de xarxes on es detalla més endavant, i tenint en compte l'aprenentatge d'aquests laboratoris s'ha decidit fer l'ús 'Multi-IOS Cisco Test Network Sandbox' que es un laboratori que ens proporciona un servidor que s'anomena VIRL(Virtual Internet Routing Labs) amb un conjunt d'imatges de diferents hardwares.

En aquest entorn s'han realitzat diferents topologies de testatge i laboratoris d'aprenentatge, no obstant això després de conèixer el seu funcionament i investigar, s'ha decidit canviar de plataforma del laboratori, ja que s'ha vist que hi ha la versió dos d'aquest mateix servidor VIRL, s'anomena Cisco Modeling Labs i és una plataforma totalment renovada que incorpora una API i amb una interfície d'usuari realitzada amb HTML5 on la interacció amb la xarxa de simulació és més convenient d'utilitzar.

Per fer la posada en marxa d'aquest laboratori és necessari la utilització del programa "AnyConnect Secure Mobility Client" [11], és un VPN que ens proporciona Cisco i es pot descarregar directament de la seva pàgina web.

Seguidament és reserva el laboratori que es vol fer-hi ús, a la pàgina web de Devnet Sanbox Labs de Cisco, un cop dintre es veuen tots els laboratoris disponibles, escollim Cisco Modeling Labs i cliquem a sobre.

Fig. 5.1.1.1 Laboratori de Cisco Modeling Labs.

Un cop dintre del laboratori es veu una definició de la topologia que ens proporciona aquest laboratori en especial i és al node *Cisco Modeling Labs* on es realitza la topologia. El node DevBox és una màquina virtual amb el sistema operatiu CentOs on es pot interactuar amb els nodes de la topologia. El següent pas és reservar el laboratori clicant al boté "Reservar":

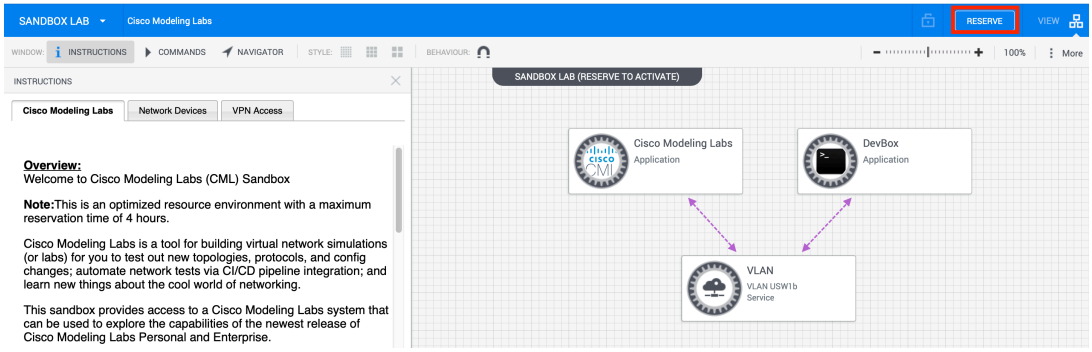

Fig. 5.1.1.2 Panell de control del laboratori.

El següent pas és escollir la durada (màxim 4h), i quin dia i hora es vol reservar, un cop seleccionat es fa clic a reservar.

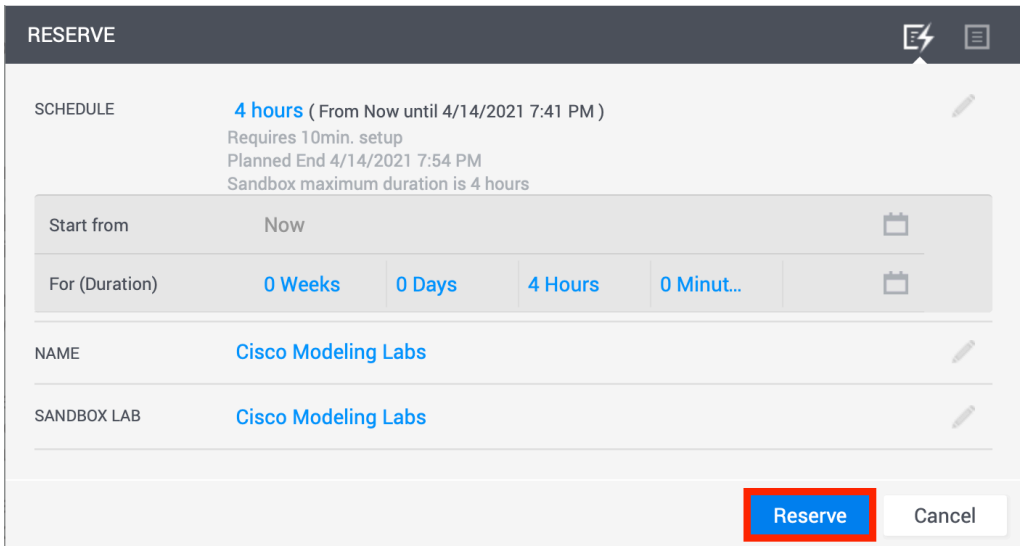

Fig. 5.1.1.3 Panell de reserva del laboratori.

Cisco ens envia un correu per avisar que estan preparant el laboratori i que triga uns 10 minuts.

Un cop el laboratori està llest ens envien un a correu amb les credencials i la "Lab Network Address" per connectar-nos amb el VPN.

Fig 5.1.1.4 Credencials i adreça del laboratori.

Obrim el programa AnyConnect (VPN), introduïm l'URL que ens ha donat Cisco per correu, introduïm la contrasenya i cliquem a "Connect". Ja es pot fer servir el laboratori.

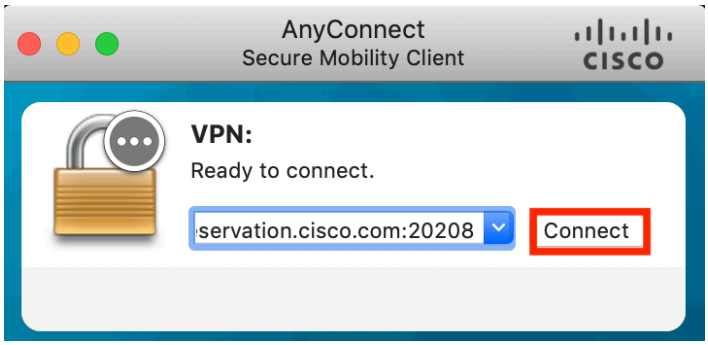

Fig. 5.1.1.5 Connexió del programa VPN de Cisco.

A la pàgina web del laboratori on s'ha reservat es detalla com connectar-nos al servidor de Cisco Modeling Labs que és on estan les màquines virtuals per fer la topologia. Obri'm una altra pestanya i ens connectem a l'URL "https://10.10.20.161" i introduïm les credencials.

Devbox
- **Address**: 10.10.20.50
- **Username**: developer
- **Password**: C1sco12345
- **SSH Port**: 22

Fig. 5.1.1.6 Credencials de l'ordinador Devbox del laboratori.

- **Cisco Modeling Labs Server**
  - **Address**: 10.10.20.161
  - **Username**: developer
  - **Password**: C1sco12345
  - **HTTPS Port for CML GUI/API**: 443
    Example Connection: https://10.10.20.161
  - **SSH Port for Console Connections**: 22

Fig. 5.1.1.7 Credencials del laboratori Cisco Modeling Labs.

Per fer la connexió a la DevBox es fa a través de SSH des del nostre terminal:

```
$ ssh developer@10.10.20.50
```

Introduïm la contrasenya.

Ja disposem de connexió a la Devbox i l'entorn preparat per l'automatització de xarxa on s'explica en els propers punts.

5.1.2 Ansible

En aquest projecte es fa l'ús de l'eina Ansible com a mitjà de configuració dels dispositius de la xarxa remotament a través d'un ordinador.

Ansible és una eina d'automatització de tasques on el seu us és molt convenient especialment on s'han d'acomplir tasques repetitives, quan tens molts dispositius a configurar, monitorar o instal·lar-hi aplicacions.

El funcionament d'Ansible consisteix a connectar-se a les màquines remotament a traves de SSH i executar tasques definides prèviament,
Ansible és *agentless* vol dir que no fa falta instal·lar cap programa a les maquines remotes, només al controlador.

Ansible fa ús de plantilles definides en YAML per la definició de les tasques de configuració que s'han de realitzar.

Ansible es constitueix de diferents fitxers, un d'ells és "ansible.cfg", en aquest fitxer es defineix la localització del nostre inventari de hosts dintre del projecte i variables de funcionament general d'Ansible, el següent fitxer és l'inventari, aquí es defineixen els dispositius que es vol controlar remotament definint la seva IP o ID. Dins de l'inventari es pot agrupar els dispositius en diferents grups depenent del tipus de dispositiu o funció. Un dels altres fitxers són els *playbooks* que són un conjunt de tasques que s'executen en els hosts definits.

Ansible té un repositori públic amb el nom de Ansible Galaxy [12] i és on estan els mòduls i col·leccions que es poden instal·lar. Els mòduls o col·leccions són formats de distribució pel contingut dels "playbooks" d'Ansible, serveixen per poder configurar dispositius varis o de diferents venedors. Els que s'han fet servir per aquest projecte són els de Cisco.Ios [13], Cisco.Nxos [14] i Cisco.Asa [15]. En resum, les col·leccions d'Ansible són *plugins* que et permeten definir una sintaxi en els *playbooks* per diferents dispositius de la xarxa.

5.1.3 Models i formats de Dades

En els següents apartats es fa una explicació dels formats i models de dades que s'han utilitzat per a les sol·licituds i respostes dels protocols que s'expliquen en els següents capítols.

Els formats de dades són estructures de dades definides perquè sigui a la vegada comprensible per les persones i els ordinadors. Els formats més populars i més comuns en programació son XML, JSON i YAML. Aquests tres diferents formats de dades tenen en comú que poden representar objectes, *arrays* o llistes i a més es defineixen en forma de clau i valor.

XML

XML (Extensible Markup Language) és un llenguatge de marcatge que moltes aplicacions fan servir per guardar, transferir o llegir dades. Es va dissenyar originalment per l'internet per això es ressembla tant a HTML, a més totes dues són llenguatges de marcatge.

XML fa servir etiquetes i elements que representen la funció de clau i valor.
Els espais en blanc no afecten el format del XML.

Com a exemple és té les següents dades estructurades en format XML, cada clau té una etiqueta d'obertura "<clau>" i de tancada "</clau>" entremig està el valor o valors d'aquesta etiqueta.

```
<interface>
    <name>eth1/1</name>
    <description>Core1 to Core2 interface</description>
    <enabled>true</enabled>
    <ipv4>
        <address>
            <ip>20.0.0.101</ip>
            <netmask>255.255.255.0</netmask>
        </address>
    </ipv4>
</interface>
```

L'etiqueta "interface" s'obre al principi i es tanca al final, això ens indica que els valors entre mig son els atributs de l'objecte "interface". Quan l'etiqueta obre i tanca i entremig només hi ha una *string* o numero vol dir que l'etiqueta és la clau i el numero o *string* el seu valor.

JSON

JSON (JavaScript Object Notation) similar a XML també el fan servir aplicacions per guardar, transferir o llegir dades. Fa servir la representació de clau i valor per assignar variables. Utilitza les {} per objectes i [] per llistes. Quan hi ha diverses variables dintre del mateix objecte se separa en comes.

En les següents dades es pot veure la sintaxi de JSON amb les mateixes variables de l'exemple de XML:

```
{
  "interface": {
    "name": "eth1/1",
    "description": "Core1 to Core2 interface",
    "enabled": "true",
    "ipv4": {
      "address": [
        {
          "ip": "20.0.0.101",
          "netmask": "255.255.255.0"
        }
      ]
    }
  }
}
```

YAML

YAML (YAML Ain't Markup Language) és un format de serialització de dades. Igual que JSON i XML també es fa servir per aplicacions on les dades es guarden o transfereixen. És el més fàcilment llegible per les persones i comprensible pels ordinadors. És definit per combinacions de llistes i mapatges. Aquestes llistes estan

fetes en format de parelles clau i valor que són habitualment anomenades "diccionaris". Opcionalment comencen per tres guions "---" i acaben amb tres punts "..."

```
---
interface:
  name: eth1/1
  description: Core1 to Core2 interface
  enabled: "true"
  ipv4:
    address:
      - ip: 20.0.0.101
        netmask: 255.255.255.0
```

YANG

YANG (Yet Another Next Generation) és un llenguatge de modelatge de dades per la definició de dades enviades per protocols de gestió de xarxes com NETCONF o RETCONF [16].

YANG defineix i descriu les característiques d'un dispositiu. Hi ha dos tipus de models, els que son estàndard per tota la indústria o els que són específic de cada venedor.

YANG utilitza contenidors per agrupar nodes relacionats i llistes per identificar nodes que estan guardats en seqüència.

En les següents dades es pot veure un model de dades YANG i un altre en format XML:

YANG:

```
module openconfig-interfaces {

    namespace "http://openconfig.net/yang/interfaces";

    container interfaces {
        description
            "Top level container for interfaces";
```

```
        list interface {
            key "name";

            description
            "The list of named interfaces on the device.";

            leaf name {
            type leafref {
                path "../config/name";
            }
            description
                "References the name of the interface";
            }
        }
    }
}
```

XML:

```
<interfaces xmlns="http://openconfig.net/yang/interfaces">
    <interface>
        <name>eth1/1</name>
    </interface>
</interfaces>
```

La relació que hi ha entre els noms dels dos formats és com RESTCONF i NETCONF saben el que li estem demanant que ens retorni del model de dades YANG que és on es defineixen les dades del dispositiu.

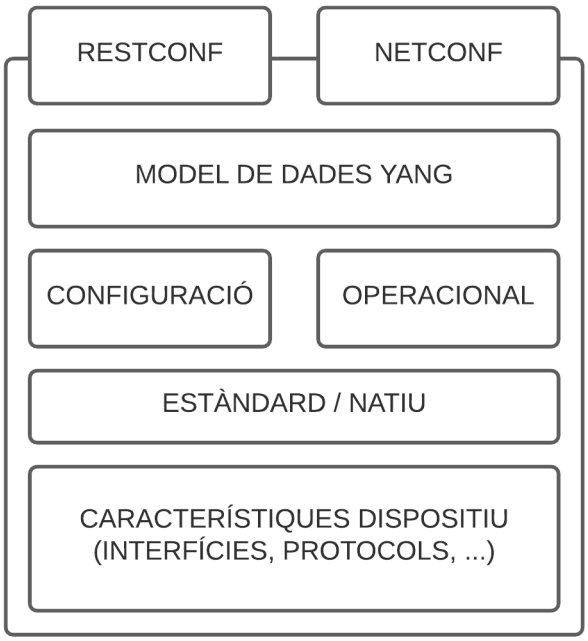

Fig. 5.1.3.1 Capes del model de dades YANG.

A la imatge de dalt es pot veure una representació d'un model de dades YANG, per una banda té la part de configuració i per l'altre l'operacional. A més es divideix en dades estàndards o d'un venedor específic. Per últim tenim les característiques del dispositiu com les interfícies o protocols varis.

YANG pot vindre preconfigurat en el dispositiu o es poden instal·lar.

5.1.4 RESTCONF

RESTCONF combina la simplicitat de HTTP amb la predictibilitat i potencial d'automatització d'una API basada en esquemes [17]. Proveeix d'una interfície programàtica per accedir a les dades definides en YANG d'un dispositiu que suporti aquest protocol.

RESTCONF suporta com a formats de dades JSON o XML. Les consultes que un client pot fer a un servidor RESTCONF son GET, POST, PUT, PATCH i DELETE.

A l'apartat d'automatització amb RESTCONF s'explica en més profunditat aquest protocol i el seu funcionament.

5.1.5 NETCONF

NETCONF és un protocol que s'ha desenvolupat i estandarditzat per l'organització Internet Engineering Task Force (IETF) el desembre del 2006 [18]. És un protocol que fa servir SSH com a transport i similar a RESTCONF permet fer operacions de configuració i monitoratge.

El juny del 2011 va ser revisat i actualitzat com a RFC6241 [19]. Hi tenim quatre components: el transport, els missatges, les operacions i el contingut. Ja s'ha parlat del transport amb SSH, les operacions i missatges és com NETCONF comunica i intercanvia dades amb el dispositiu. Es fa servir crides RPC (Rremote Procedure Call) pels dos casos i bàsicament RPC és un protocol que et permet sol·licitar un servei d'un programa localitzat en un altre ordinador. RPC és capaç de cridar processos en sistemes remots com si ho fes localment. Per últim tenim el contingut que són les dades que es vol extreure o configurar.

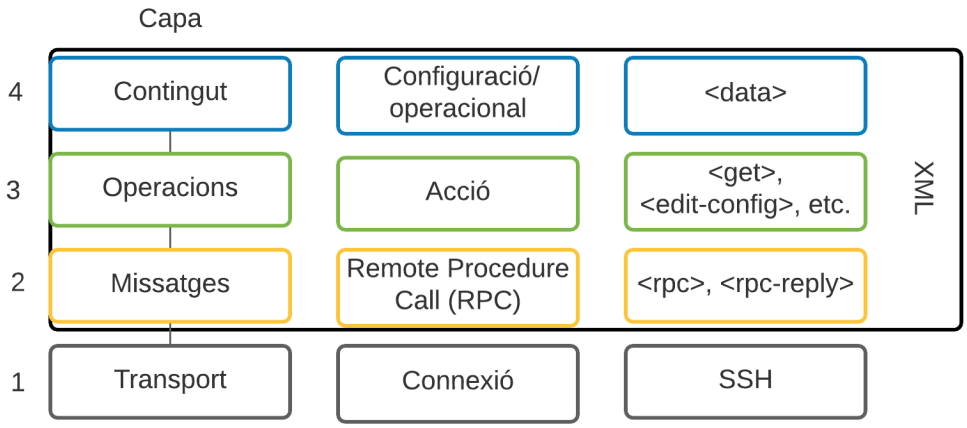

Fig. 5.1.5.1 Capes i informació del protocol NETCONF.

Exemple:

Per connectar-nos en un servidor NETCONF a través de SSH s'executa la següent comanda.

Iniciem sessió:

```
$ ssh cisco@10.10.20.177 -p 830 -s Netconf
User Access Verification
Password:
```

Escrivim la contrasenya.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
        <capability>urn:ietf:params:netconf:base:1.0</capability>
        <capability>urn:ietf:params:netconf:base:1.1</capability>
    </capabilities>
    <session-id>213998732</session-id>
</hello>
```

Ens envia un missatge de *hello* amb les capacitats que suporta aquest dispositiu. Les capacitats són els models de dades YANG que té instal·lat el dispositiu.

Al següent pas el client li ha de tornar el missatge de *hello* per iniciar la connexió:

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
        <capability>urn:ietf:params:netconf:base:1.0</capability>
    </capabilities>
</hello>
]]>]]>
```

En el punt d'automatització amb NETCONF d'aquest document s'explica com automatitzar aquestes sol·licituds amb una llibreria anomenada ncclient de Python.

5.2 Topologia

5.2.1 Definició

La topologia que s'ha utilitzat per construït la xarxa definida per software ha sigut una xarxa de tipus campus on tenim la capa del nucli, la capa de distribució i la capa d'accés, on s'ha implementat amb switchos.

Aquest disseny de topologia aporta una estructura en forma jeràrquica on cada capa té un rol en específic.

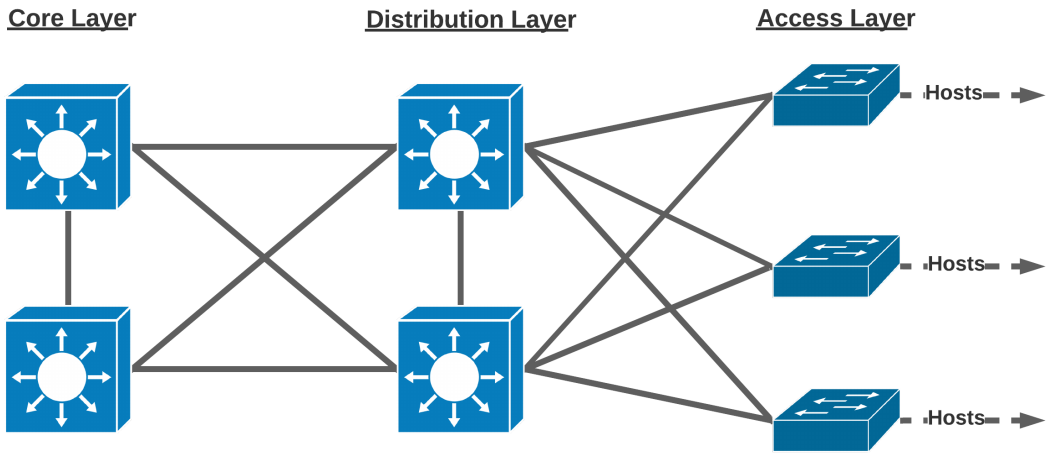

Fig. 5.2.1.1 Capes d'una xarxa campus.

La capa d'accés connecta els hosts (ordinadors, servidors, telèfons, impressores, etc...) cap a la capa de distribució i principalment és la responsable de donar accés a la xarxa als usuaris.

La capa de distribució agrega els nodes de la capa d'accés de la mateixa branca i proporciona una sortida *gateway* per els seus hosts, on aquesta reencamina els paquets cap a la capa del nucli o altres nodes d'accés.

La capa del nucli és un punt on s'agrega la capa de distribució i les diferents branques que pot tenir la xarxa, proporciona escalabilitat i alta disponibilitat. En el nucli menys

complexitat i quantitat de protocols és millor per tenir una alta rapidesa de reencaminament de paquets. [20]

S'han utilitzat dues versions diferents de switch. Una versió s'anomena IOSvL2 que és una implementació virtual del sistema operatiu
IOS (Internetwork Operating System), aquest sistema operatiu és un paquet integrat per diferents funcions d'encaminament, comunicacions i interconnexions de xarxes i telecomunicacions i a on es fa servir a molts dels dispositius de Cisco. [21] En el cas d'aquest switch només pot fer funcions de switch de capa 2 on no es pot fer encaminament entre VLANs.

L'altra versió de switch s'anomenen Nexus, aquests switchos són màquines virtuals que qualsevol empresa pot desplegar en el seu propi entorn virtual o a un núvol allotjat per un proveïdor.

El sistema operatiu NXOS (Nexus Operating System), està basat en Wind River Linux i disposa de les mateixes funcions que el sistema operatiu IOS, però a més a més disposa d'una *Bash-shell* i suporta els protocols NETCONF i RESTCONF entre altres funcionalitats. [22]

Els switch IOSvL2 s'han utilitzat per definir la capa d'accés, ja que només es necessita la implementació de VLANs i encaminament dels paquets. Els dispositius Nexus en ser switchos de capa 3 s'han utilitzat per la capa de distribució i nucli perquè tenen la funcionalitat d'encaminament entre VLANs (*InterVLAN Routing*) i encaminament d'IPs de capa 3 (*Layer 3 IP Routing*).

La diferència entre un switch de capa 2 i un de capa 3 és la funció d'encaminament. El switch de capa 2 treballa només amb adreces MAC en canvi el switch de capa 3 pot fer totes les funcionalitats d'un switch de capa 2, però a més a més té la funcionalitat de poder fer encaminaments d'IP dinàmics i estàtics.

Els diferents protocols i configuracions de la xarxa es defineixen a continuació:

- **Capa d'accés**: S'han creat dues *VLANs* per cada switch de capa 2, dos ports que van cap a dos ordinadors són d'accés *VLANs* i tenim quatre ports més en mode *trunk* que connecten amb la capa de distribució.

- **Capa de distribució:** En aquesta capa s'han configurat els ports que van cap a la capa d'accés en mode *trunk* i els ports que van cap a la capa del nucli s'ha configurat en mode capa 3 que són ports encaminats. També s'ha configurat el protocol HSRP (*Hot Standby Router Protocol*) com a *gateway* dels ordinadors i redundància.

- **Capa del nucli**: En aquesta capa s'han configurat tots els ports com ports encaminats, tant com els ports que van del nucli a distribució com els que van de nucli a nucli, s'han configurat en capa 3, ja que proporciona una convergència més ràpida que un port en capa 2 i en el cas de que un port de capa 3 caigui el reencaminament de paquets cap a una altra ruta es més ràpida [23].

- **Accés a internet:** La branca d'accés a internet s'ha configurat un *Firewall* ASAv amb NAT i un *router* CSV1000 com a xarxa de fora amb un ordinador.

- **Branca de servidor:** La branca de servidors conté els mateixos protocols que les branques d'usuari menys que la de servidors només té una VLAN per tota la branca.

5.2.2 Creació

La creació de la topologia s'ha fet a la plataforma Cisco Modeling Labs(CML) que s'ha explicat anteriorment. A continuació s'explica com s'ha construït la xarxa, quins dispositius s'han escollit i com s'han configurat aquests dispositius per a poder controlar-los remotament amb Ansible.

Per fer-ho primer s'ha construït la xarxa al CML, com que només es pot utilitzar durant quatre hores, es pot descarregar la topologia quan acabes en format YAML. A partir d'aquí pots automatitzar la construcció de la xarxa a través de l'API que ens proporciona el CML. A continuació s'explica el procés.

En el següent mètode *login()* fem una crida a l'adreça IP (10.10.20.161) del Laboratori CML mes l'URL de l'API que forma la variable *baseUrl*.

```
baseUrl = "https://10.10.20.161/api/v0"

def login():
    credencials = json.dumps({
        "username": "developer",
        "password": "C1sco12345"
    })
    headers = {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer <Bearer Token>'
    }
    token_auth = requests.request("POST", baseUrl + "/authenticate",
                                  headers=headers, data=credencials,
                                  verify=False)

    token = 'Bearer ' + token_auth.json()
    return token
```

Primerament hem importat les llibreries Requests [24] i JSON.
Requests és una llibreria HTTP que ens permet fer sol·licituds fàcilment a través d'HTTP.

Quan s'executa el mètode es crea la variable credencials on es guarda un objecte Python amb el nom i la contrasenya del Laboratori que posteriorment convertim a *string* de JSON a través del mètode *dumps()*, a continuació creem la variable *headers*

on especifiquem el contingut i el tipus d'autorització que es vol fer servir i finalment fem una crida *POST* que ens retorna un *token* per l'autenticació de les següents crides.

En el següent mètode *createTopologyFromFile(token)*, es passa el *token* com a paràmetre del mètode, s'obre l'arxiu *campusTopology.yaml* on el tenim al mateix directori que aquest programa de Python, assignem el *token* a la variable *Authorization* de la capçalera i finalment es fa la crida *POST* a la direcció *baseUrl* mes l'extensió */import* i el títol.

```
def createTopologyFromFile(token):

    with open('campusTopology.yaml', 'rb') as payload:

        headers = {'content-type': 'application/json',
                   'Authorization': token }

        response = requests.request("POST", baseUrl +
                                    "/import?title=net_automation",
                                    headers=headers, data=payload,
                                     verify=False)
```

Un cop executat el programa apareix la topologia a la nostra sessió del laboratori:

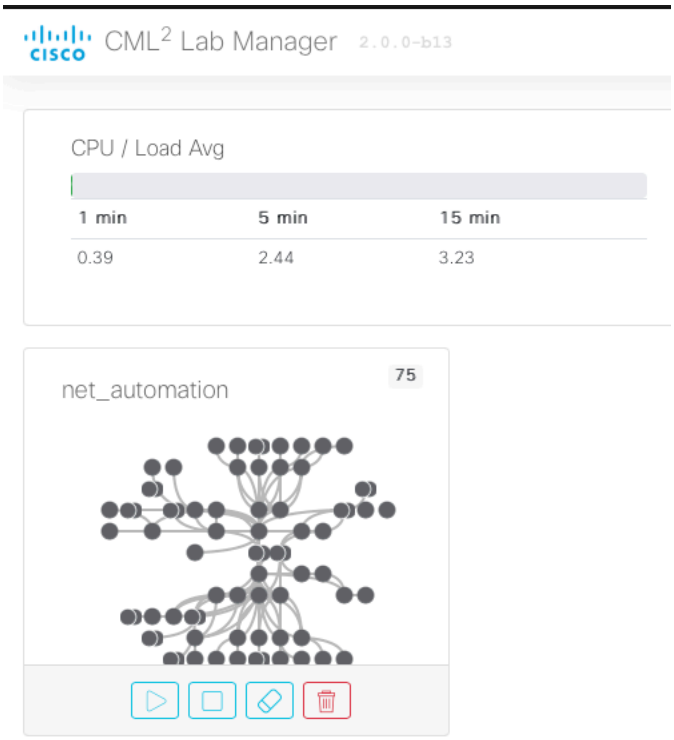

Fig. 5.2.2.1 La topologia al laboratori.

Es fa clic sobre de la topologia per entrar-hi.

Per temes de dimensions i claredat s'adjunta la topologia sense els ordinadors i sense els nodes *sanbox_backend1, sanbox_backend2* i *bridge_to_sandbox1,* ja que són nodes que es necessiten per fer una connexió externa i poder connectar-se a través de la DevBox. L'únic dispositiu amb permís de connexió a internet és la DevBox.

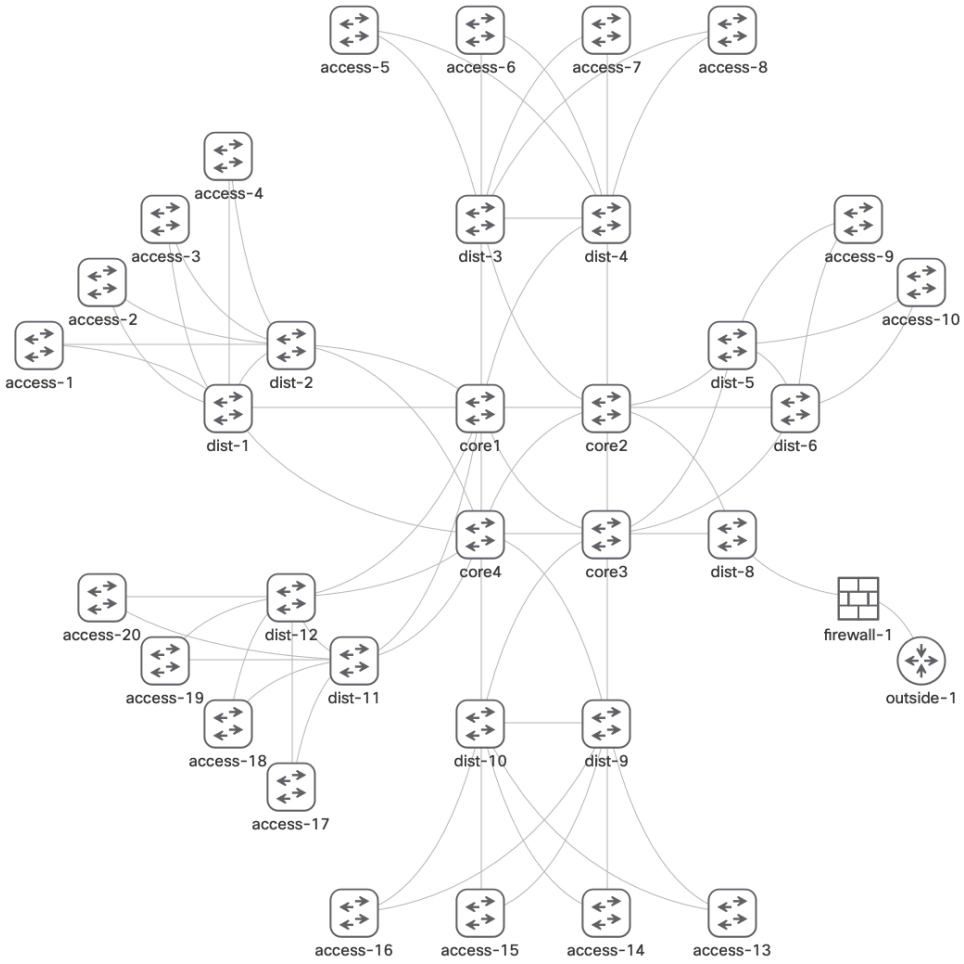

Fig. 5.2.2.2 Imatge de la topologia del projecte.

És una topologia tipus campus amb sis branques, quatre d'elles són per usuaris, una branca és pels servidors i un altre branca per la connexió a internet.

5.2.3 Configuració inicial

Per poder configurar dispositius remotament a través de SSH es necessita que aquest protocol estigui activat en els dispositius. A més a més, en el cas del laboratori necessitem configurar una IP a la interfície de *management* del dispositiu i aquesta IP ha d'estar a la mateixa xarxa que la DevBox per poder fer la connexió externa.

En la següent configuració del dispositiu IOSvL2 el que és fa primer es configurar la interfície de *management* que és la *GigabitEthernet0/0* amb una IP. Aquesta interfície va connectada en els *switchos* sanbox_backend1 o sandbox_backend2 i aquests dos nodes estan connectats en el bridge_to_sandbox1 que exposa la IP del switch a l'exterior per poder-se connectar. La IP d'aquesta interfície ha d'estar a la xarxa 10.10.20.0/24 en tots els dispositius.

IOSvL2:

```
access1(config)#interface GigabitEthernet 0/0
access1(config-if)#no switchport
access1(config-if)#ip address 10.10.20.212 255.255.255.0
access1(config-if)#no shutdown
access1(config-if)#exit
access1(config)#ip domain-name virl
access1(config)#crypto key generate rsa modulus 2048
The name for the keys will be: access1.virl

% The key modulus size is 2048 bits
% Generating 2048 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 14 seconds)

access1(config)#username cisco privilege 15 password cisco
access1(config)#line vty 0 4
access1(config-line)# login local
access1(config-line)# transport input ssh
access1(config-line)#end
access1#copy running-config startup-config
```

Quan s'ha configurat la IP es configura un nom de domini i unes claus que es necessiten per fer la connexió amb SSH. A continuació es configura cisco/cisco com a usuari i contrasenya i per últim la interfície virtual (vty 0 4) que s'utilitza per donar accés a la línia de comandes i es dóna accés al protocol SSH.

Nexus:

```
core1(config)# vrf context management
core1(config-vrf)# ip route 0.0.0.0/0 10.10.20.254
core1(config-vrf)# exit
core1(config)# interface mgmt0
core1(config-if)# vrf member management
core1(config-if)# ip address 10.10.20.177/24
core1(config-if)# end
core1# copy running-config startup-config
```

El switch Nexus ja té el protocol SSH activat per defecte i falta crear una instància vrf (Virtual Rotuing and Forwading) que crea una taula d'encaminaments separada per la interfície de *management* amb una ruta *gateway* i configurar la interfície de *management* amb la seva IP corresponent.

ASAv:

```
ciscoasa(config)# interface Management 0/0
ciscoasa(config-if)# nameif MGMT
ciscoasa(config-if)# ip address 10.10.20.214 255.255.255.0
ciscoasa(config-if)# no shutdown
ciscoasa(config-if)# exit
ciscoasa(config)# username cisco password cisco privilege 15
ciscoasa(config)# aaa authentication ssh console LOCAL
ciscoasa(config)# ssh 10.10.20.213 255.255.255.0 MGMT
ciscoasa(config)# ssh version 2
ciscoasa(config)# copy running-config startup-config
```

En el Firewall ASAv s'ha hagut de configurar la interfície de *management* amb una IP, usuari i contrasenya cisco/cisco, activar l'autentificació, permetre accés SSH a la interfície de MGMT i es força la versió 2 de SSH.

Hosts:

En els ordinadors i servidors s'ha configurat una IP estàtica i una ruta per defecte diferent per cadascun, directament en l'arxiu YAML que ens descarreguem del laboratori:

```
- id: n3
    label: desktop-0-VLAN2
    node_definition: desktop
```

```
    x: -1100
    y: -150
    configuration: |-
       ifconfig eth0 up 192.168.2.4 netmask 255.255.255.0
       route add default gw 192.168.2.1 dev eth0
    image_definition: desktop
```

Aquesta configuració s'executa a l'arrencar el dispositiu.

Aquestes són les configuracions inicials que tenen els dispositius de la topologia. Aquestes configuracions ja estan implementades a la topologia del repositori/carpeta del projecte.

5.3 Desplegament

Per fer el desplegament s'ha de tenir una sessió del laboratori Cisco Modeling Labs Sandbox activa i una connexió SSH des del terminal a l'ordinador Devbox del laboratori. En el punt 6.1.1 s'explica com fer-ho.

Des de la connexió del nostre terminal a la Devbox executem la següent comanda:

```
[developer@devbox ~]$ source <(curl
https://raw.githubusercontent.com/aamargant/net_automation_aamarga
nt/master/setup.sh)
```

Amb aquesta comanda el que es fa és extreure la informació de l'arxiu "setup.sh" del repositori del projecte amb CURL i executar-lo al mateix terminal amb la comanda *source*.

Tasques que realitza l'arxiu:

- Clonar el repositori del projecte
- Posicionament dintre del directori
- Crear un entorn virtual Python
- Activar l'entorn virtual
- Instal·lació d'Ansible i altres llibreries necessàries
- Instal·lar les col·leccions de Cisco per Ansible
- Descarregar localment un model de dades YANG
- Clonar el repositori OpenConfig YANG *data model*
- Execució de l'arxiu "topologySetUp.py" per importar la topologia amb la configuració inicial al laboratori.

Un cop finalitza es veu la topologia en el laboratori (https://10.10.20.161), si no hi és s'actualitza la pàgina i ha d'aparèixer.

5.3.1 Automatització amb Ansible

En aquest apartat s'explica com configurar amb Ansible els nodes del nucli, per configurar els nodes restants de la topologia és el mateix procés però amb altres *playbooks*.

Primer de tot s'han d'activar els nodes, per això es passa el ratolí per sobre del node i es clica el botó del *play*.

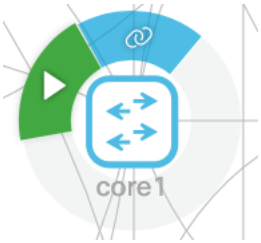

Fig. 5.3.1.1 Botó de *play* del node del laboratori.

Es fa el mateix amb els nodes core2, core3, core4 i sandbox_backend1, sandbox_backend2 i bridge_to_sandbox1 i s'espera a que s'activin amb la insígnia de color verd.

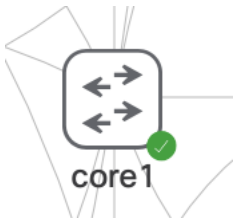

Fig. 5.3.1.2 Node del laboratori engegat.

Des del terminal on tenim la connexió a la Devbox es pot fer un `$ ls` per veure els arxius que hi ha. Es té un total de tretze *playbooks,* a continuació es defineix quins *playbooks* configuren cada node:

Playbooks	Nodes
access_leaf_1.yml	access-1-2-3-4
access_leaf_2.yml	access-5-6-7-8

access_leaf_3.yml	access-9-10
access_leaf_4.yml	access-13-14-15-16
access_leaf_5.yml	access-17-18-19-20
dist_spine_1.yml	dist-1-2
dist_spine_2.yml	dist-3-4
dist_spine_3.yml	dist-5-6
dist_spine_4.yml	dist-8
dist_spine_5.yml	dist-9-10
dist_spine_6.yml	dist-11-12
core.yml	core1-2-3-4
asav.yml	firewall-1

Taula 5.3.1.1 Taula de relació entre el fitxer YAML i els nodes que configura.

Per configurar el nucli amb Ansible s'executa la següent comanda:

```
$ ansible-playbook core.yml
```

Quan s'executa el *playbook* Ansible es connecta a través de SSH amb els switchos. El següent pas comprova quins són els *hosts* que ha de configurar i mira a la carpeta "group_vars" del repositori el fitxer YAML que té el mateix nom que el grup de *hosts* on està el dispositiu així sap que és un dispositiu amb un sistema operatiu nxos(Nexus) i fa servir la col·lecció *cisco.nxos* per entendre la sintaxis del *playbook*:

Playbook core.yml	**Arxiu /group_vars/core.yml**

`---`
`- name: Configure Core Group`
`  hosts: core`
`  tasks:`
`. . .`
	`ansible_network_os: nxos`

Taula 5.3.1.2 Taula de relació entre el *playbook* i el fitxer de grups

Quan la connexió és exitosa apareix el següent:

```
TASK [Gathering Facts] ***************************************************
ok: [10.10.20.179]
ok: [10.10.20.180]
ok: [10.10.20.178]
```

```
ok: [10.10.20.177]
```

Aquestes són les IP de la interfície de *management* dels switchos del nucli.
Un cop es connecta comença a executar les tasques de l'arxiu YAML. Primer de tot configura tasques que s'han de realitzar en els quatre switchos del nucli.

A l'esquerra la tasca del *playbook* i a la dreta la sortida de la tasca realitzada en el terminal.

- Activació dels protocols RESTCONF i NETCONF:

```
- name: Enable RESTCONF
  nxos_feature:
    feature: restconf
    state: enabled
```

```
TASK [Enable RESTCONF] **
changed: [10.10.20.179]
changed: [10.10.20.178]
changed: [10.10.20.177]
changed: [10.10.20.180]
```

```
- name: Enable NETCONF
  nxos_feature:
    feature: netconf
    state: enabled
```

```
TASK [Enable NETCONF] **
changed: [10.10.20.179]
changed: [10.10.20.178]
changed: [10.10.20.177]
changed: [10.10.20.180]
```

Quan la tasca modifica el dispositiu, per pantalla surt "changed".

- Activació del terminal del switch:

```
- name: Enable Bash-Shell
  nxos_feature:
    feature: bash-shell
    state: enabled
```

```
TASK [Enable Bash-Shell] **
changed: [10.10.20.179]
changed: [10.10.20.178]
changed: [10.10.20.177]
changed: [10.10.20.180]
```

- Activació de l'API del switch que ens permet tenir una connexió HTTPS i poder-nos connectar a la IP del switch i disposar d'una interfície grafia web per fer proves amb NETCONF i RESTCONF:

```
- name: Enable NXAPI
  nxos_feature:
    feature: nxapi
    state: enabled

- name: Set ports NXAPI
  cisco.nxos.nxos_nxapi:
    enable_http: true
    enable_https: true
    http_port: 80
    https_port: 443
```

```
TASK [Enable NXAPI] **
changed: [10.10.20.180]
changed: [10.10.20.179]
changed: [10.10.20.178]
changed: [10.10.20.177]

TASK [Set ports NXAPI] **
changed: [10.10.20.180]
changed: [10.10.20.178]
changed: [10.10.20.179]
changed: [10.10.20.177]
```

- Es copia el model de dades YANG que s'ha descarregat en executar l'arxiu setup.sh des de la Devbox al *bootflash* del switch:

```
- name: initiate file copy from device
  cisco.nxos.nxos_file_copy:
    file_pull: true
    remote_file: /home/developer/net_automation_aamargant/mtx-
                 openconfig-all-1.0.0.0-9.2.3.lib32_n9000.rpm
    remote_scp_server: 10.10.20.50
    remote_scp_server_user: developer
    remote_scp_server_password: C1sco12345
    vrf: management
```

```
TASK [initiate file copy from device] ************************************
changed: [10.10.20.177]
changed: [10.10.20.178]
changed: [10.10.20.180]
changed: [10.10.20.179]
```

- S'instal·la el model de dades YANG en el switch:

```
- name: Install OpenConfig rpm package
  cisco.nxos.nxos_rpm:
    pkg: mtx-openconfig-all-1.0.0.0-9.2.3.lib32_n9000.rpm
```

```
TASK [Install OpenConfig rpm package] ************************************
changed: [10.10.20.179]
```

```
changed: [10.10.20.180]
changed: [10.10.20.178]
changed: [10.10.20.177]
```

En acabar les tasques de tot el grup del nucli comença a fer les tasques individuals per cada nucli com la configuració d'IP i interfícies.

Quan acaba el *playbook,* Ansible mostra un resum dels canvis o falles que hi ha hagut:

```
PLAY RECAP *********************************************************************
10.10.20.177 : ok=13   changed=11   unreachable=0    failed=0    skipped=0
10.10.20.178 : ok=13   changed=11   unreachable=0    failed=0    skipped=0
10.10.20.179 : ok=13   changed=11   unreachable=0    failed=0    skipped=0
10.10.20.180 : ok=13   changed=11   unreachable=0    failed=0    skipped=0
```

Per cada *host* s'ha realitzat un total d'onze canvis i les tretze tasquen han sigut exitoses.

Per configurar el node firewall-1 amb Ansible primerament s'ha d'activa amb "enabled" a la línia de comandes i introduir l'usuari i contrasenya cisco/cisco.

5.3.2 Automatització amb NETCONF

En aquest apartat s'explica com es realitza l'automatització amb NETCONF, les tasques que es realitzen són les d'extracció de dades dels dispositius i les estadístiques.

Un cop s'ha configurat el dispositiu amb Ansible es pot realitzar l'extracció de dades del dispositiu, ja que s'ha d'activar el protocol NETCONF perquè funcioni i la instal·lació del model de dades YANG en el switch.

S'ha creat un fitxer Python anomenat nodes.py amb les dades de cada node de la topologia com el següent:

```
core1 = {
            "name": "Core 1",
            "address": "10.10.20.177",
            "port": 830,
            "username": "cisco",
            "password": "cisco"
        }
```

Per fer la connexió NETCONF amb el dispositiu s'ha utilitzat una llibreria anomenada ncclient [25], que ens permet programar les sol·licituds amb NETCONF.

```
from ncclient import manager
def get_request(xml_filter):
    with manager.connect(host=node["address"], port=node["port"],
        username= node["username"], password=node["password"],
        hostkey_verify=False) as device:

        netconf_reply = device.get(('subtree', xml_filter))
```

La llibreria te una variable anomenada Manager per connectar-nos amb el dispositiu a través del mètode connect(), on introduïm la IP, el port i les credencials del node corresponent, aquest mètode guarda la configuració i connexió com a variable "device". El següent pas amb el mètode *get()* es passa com a variable el filtre XML de la informació que es vol demanar al dispositiu:

```
<interfaces xmlns="http://openconfig.net/yang/interfaces">
    <interface>
        <name>eth1/1</name>
    </interface>
</interfaces>
```

Per saber l'estructura que s'ha de crear per fer el filtre XML, s'ha de direccionar al repositori Openconfig prèviament clonat i anar al directori d'on es vol fer servir els models de dades YANG, en aquest cas es volen modificar les interfícies, aleshores anem al model de dades d'interfície del repositori:

```
$ cd public/release/models/interfaces
```

En aquest directori es tenen tots els models de dades YANG sobre les interfícies d'un dispositiu, per saber l'estructura del model de dades YANG s'executa la següent comanda:

```
$ pyang -f tree openconfig-interfaces.yang
```

Es mostra l'estructura del model:

```
module: openconfig-interfaces
  +--rw interfaces
     +--rw interface* [name]
        +--rw name             -> ../config/name
        +--rw config
        |  +--rw name?            string
        |  +--rw type             identityref
        |  +--rw mtu?             uint16
        |  +--rw loopback-mode?   boolean
        |  +--rw description?     string
        |  +--rw enabled?         boolean
        +--ro state
        |  +--ro name?            string
        |  +--ro type             identityref
        |  +--ro mtu?             uint16
        |  +--ro loopback-mode?   boolean
        |  +--ro description?     string
        |  +--ro enabled?         boolean
        |  +--ro ifindex?         uint32
        |  +--ro admin-status     enumeration
        |  +--ro oper-status      enumeration
        |  +--ro last-change?     oc-types:timeticks64
        |  +--ro logical?         boolean
        |  +--ro management?      boolean
        |  +--ro cpu?             boolean
        |  +--ro counters
        |     +--ro in-octets?             oc-yang:counter64
        |     +--ro in-pkts?               oc-yang:counter64
        |     +--ro in-unicast-pkts?       oc-yang:counter64
        |     +--ro in-broadcast-pkts?     oc-yang:counter64
```

```
        |      +--ro in-multicast-pkts?     oc-yang:counter64
        |      +--ro in-discards?           oc-yang:counter64
        |      +--ro in-errors?             oc-yang:counter64
        |      +--ro in-unknown-protos?     oc-yang:counter64
        |      +--ro in-fcs-errors?         oc-yang:counter64
        |      +--ro out-octets?            oc-yang:counter64
        |      +--ro out-pkts?              oc-yang:counter64
        |      +--ro out-unicast-pkts?      oc-yang:counter64
        |      +--ro out-broadcast-pkts?    oc-yang:counter64
        |      +--ro out-multicast-pkts?    oc-yang:counter64
        |      +--ro out-discards?          oc-yang:counter64
        |      +--ro out-errors?            oc-yang:counter64
        |      +--ro carrier-transitions?   oc-yang:counter64
        |      +--ro last-clear?            oc-types:timeticks64
```

En el filtre que s'ha passat amb el programa es demana la interfície amb *name* eth1/1 que està dintre de *interface* i aquesta dintre de *interfaces*.

Del model de dades YANG les variables amb "rw" són de lectura i configuració i les variables "ro" només de lectura.

Per configurar per exemple la descripció de la interfície s'ha de canviar la crida que fem a la variable "device" que hem creat anteriorment amb la connexió NETCONF pel mètode *edit_config():*

```python
def get_request(xml_filter):
    with manager.connect(host="10.10.20.177", port="830",
        username="cisco", password="cisco",
        hostkey_verify=False) as device:

        netconf_reply = device.edit_config(xml_filter, target = 'running')
```

En aquest mètode es passa el filtre i el dispositiu objectiu que és el mateix dispositiu que estem connectats.

El filtre també canvia a aquest:

```
<config>
    <interfaces xmlns="http://openconfig.net/yang/interfaces">
                <interface>
                <name>eth1/1</name>
                    <config>
                        <description>new description</description>
                    </config>
                </interface>
    </interfaces>
</config>
```

S'abraça tot amb l'etiqueta "config" i es defineix el nom de la interfície que es vol configurar i la configuració que es vol fer.

Per comprovar-ho anem al directori de *netrest/* i s'executa l'arxiu netconf_interface_info.py de la següent manera:

```
$ python netconf_interface_info.py core1
```

El resultat és que s'imprimeix una taula amb la interfície eth1/1 amb la nova descripció:

```
Core 1 (52:54:00:00:6A:48)
+-------------+---------------------------+-----------+-------------+
| Interface   | Description               | Enabled   | IP / Mode   |
+=============+===========================+===========+=============+
| eth1/1      | new description           | true      | 20.0.0.101  |
+-------------+---------------------------+-----------+-------------+
| eth1/2      | Core1 to Core4 interface  | true      | 20.0.0.97   |
+-------------+---------------------------+-----------+-------------+
| eth1/3      | Core1 to Core3 interface  | true      | 20.0.0.113  |
+-------------+---------------------------+-----------+-------------+
| eth1/4      | Core1 to Dist3 interface  | true      | 20.0.0.22   |
+-------------+---------------------------+-----------+-------------+
| eth1/5      | Core1 to Dist4 interface  | true      | 20.0.0.30   |
+-------------+---------------------------+-----------+-------------+
| eth1/6      | Core1 to Dist2 interface  | true      | 20.0.0.10   |
+-------------+---------------------------+-----------+-------------+
| eth1/7      | Core1 to Dist1 interface  | true      | 20.0.0.2    |
+-------------+---------------------------+-----------+-------------+
| eth1/8      | Core1 to Dist12 interface | true      | 20.0.0.90   |
+-------------+---------------------------+-----------+-------------+
| eth1/9      | Core1 to Dist11 interface | true      | 20.0.0.82   |
+-------------+---------------------------+-----------+-------------+
```

La taula s'ha retallat per motius de dimensions, però aquesta taula, a més a més especifica els paquets *broadcast*, *unicast* i *multicast* d'entrada i sortida de cada interfície del node.

Per imprimir les taules dels altres nodes s'escriu el nom del node al costat del fitxer python i executar-ho.

Els noms i grups disponibles es poden veure executant el fitxer sense cap nom de node i surt una taula amb els dispositius i grups de dispositius disponible:

```
$ python netconf_interface_info.py
```

```
+------------+-------+-------------+-------------+-------------+
| Groups     | core  | dist_spine_1 | dist_spine_2 | dist_spine_3 |
+------------+-------+-------------+-------------+-------------+
| Inidivudal | core1 | core2        | core3        | core4        |
+------------+-------+-------------+-------------+-------------+
```

5.3.3 Automatització amb RESTCONF

En aquest apartat s'explica com es realitza l'automatització amb RESTCONF, les tasques que es realitzen són les d'extracció de dades del dispositiu i un exemple de configuració.

Per utilitzar RESTCONF en els dispositius de la topologia han d'estar configurat per Ansible, ja que s'ha de tenir el protocol RESTCONF activat i els models de dades YANG.

En aquesta automatització també es fa servir l'arxiu nodes.py anomenat a l'apartat anterior.

```
def get_request():
  url = "https://" + node["address"] + "/restconf/data/Cisco-NX-OS-
        device:System/ipv4-items/inst-items/dom-items/Dom-list"
  payload=""
  headers = {
    'Content-Type': 'application/yang.data+json',
    'Accept': 'application/yang.data+xml',
  }

  response = requests.request("GET", url, auth=('cisco', 'cisco'),
                              headers=headers, data=payload,  verify=False)
```

Primer de tot s'importa la llibreria *request*, aquesta llibreria es la mateixa que s'ha utilitzat per fer el programa de python que importa la topologia en el laboratori explicat en l'apartat 6.2.2.

Seguidament en el mètode *get_request()* es crea la variable "url" amb la IP del node que es vol connectar més l'adreça del model de dades YANG d'on es vol treure la informació. A la variable "headers" s'especifica el tipus de contingut, en aquest cas s'interactua amb un model de dades YANG i en el cas que es passi un filtre de configuració aquest ha de ser en JSON, també s'especifica a la variable "Accept" amb quin format de dades el client vol rebre la resposta, en aquest cas XML.

Per finalitzar s'executa la sol·licitud amb el mètode "requets()" i es passa com a paràmetres el tipus de sol·licitud que es vol fer, en aquest cas és un GET, l'autenticació i les variables *headers* i *data*.

Amb aquesta sol·licitud el que es demana són les interfícies i la taula d'encaminament d'IP dels nodes. Tota la informació que es rep s'analitza i s'insereix en una taula

La sol·licitud s'executa amb la següent comanda:

```
$ python restconf_ip_routing_table.py core2
```

```
RESTCONF ip routing table:

Core 2
+------------------+--------------+-----------------+
| IP               | Next Hop     | Via Interface   |
+==================+==============+=================+
| 192.168.30.0/24  | 20.0.0.101/32 | eth1/1         |
+------------------+--------------+-----------------+
| 192.168.30.0/24  | 20.0.0.117/32 | eth1/2         |
+------------------+--------------+-----------------+
| 209.165.201.0/24 | 20.0.0.106/32 | eth1/3         |
+------------------+--------------+-----------------+
| 209.165.201.0/24 | 20.0.0.61/32  | eth1/9         |
+------------------+--------------+-----------------+
| 192.168.26.0/24  | 20.0.0.101/32 | eth1/1         |
+------------------+--------------+-----------------+
| 192.168.26.0/24  | 20.0.0.117/32 | eth1/2         |
+------------------+--------------+-----------------+
| 192.168.12.0/24  | 20.0.0.17/32  | eth1/4         |
+------------------+--------------+-----------------+
| 192.168.12.0/24  | 20.0.0.25/32  | eth1/5         |
+------------------+--------------+-----------------+
| 192.168.4.0/24   | 20.0.0.101/32 | eth1/1         |
+------------------+--------------+-----------------+
```

Aquesta taula es fa amb la llibreria de python *tabulate* [26]. La taula s'ha retallat per motius de dimensions.

Per realitzar una configuració amb RSTCONF i JSON per exemple afegir rutes d'encaminament en el switch es fa de la següent manera:

```
def get_request():
  url = "https://" + node["address"] + "/restconf/data/Cisco-NX-OS-
        device:System"

  payload= open("ip_routing_conf.json").read()
  headers = {
    'Content-Type': 'application/yang.data+json',
    'Accept': 'application/yang.data+json',
  }

  response = requests.request("PATCH", url, auth=('cisco', 'cisco'),
headers=headers, data=payload,  verify=False)
```

Es crea la variable "url" amb la IP del node corresponent. S'assigna el filtre JSON a la variable "payload" i es crea la variable "headers" especificant que el contingut que es transfereix és en format JSON i la resposta es vol que també sigui amb JSON. Per finalitzar s'executa la sol·licitud en mode PATCH i s'afegeixen les variables a la sol·licitud.

Per saber com realitzar el filtre JSON s'ha de comprovar el model de dades YANG i la seva estructura, en aquest cas es vol afegir rutes d'encaminament en el switch. Per aquest exemple es fa servir el model de dades YANG de Cisco Nexus:

Es direcciona a la carpeta dels models de dades del Cisco Nexus:

```
$ cd yang/vendor/cisco/nx/9.2-3
```

S'executa la següent comanda:

```
$ pyang -f tree Cisco-NX-OS-device.yang
```

```
module: Cisco-NX-OS-device
  +--rw System
      +--rw ipv4-items
      |   |  +--rw inst-items
      |   |     +--rw dom-items
      |   |     |  +--rw Dom-list* [name]
      |   |     |     +--rw rt-items
      |   |     |     |  +--rw Route-list* [prefix]
      |   |     |     |     +--rw prefix       address_Ip
      |   |     |     |     +--rw nh-items
```

```
      |   |     |     |     |  +--rw Nexthop-list* [nhIf nhAddr nhVrf]
      |   |     |     |     |     +--rw rtname?    string
      |   |     |     |     |     +--rw descr?     naming_Descr
      |   |     |     |     |     +--rw nhIf       nw_IfId
      |   |     |     |     |     +--rw nhAddr     address_Ip
      |   |     |     |     |     +--rw nhVrf      l3_VrfName
      |   |     |     |     |     +--rw rwEncap?   string
      |   |     |     |     |     +--ro operSt?    ip_OperSt
```

Per motius de dimensions el model de dades s'ha retallat.

Les variables entre claudàtors són paràmetres d'entrada.

Per afegir rutes IP en el switch s'ha d'arribar fins a la llista "Route-list" i es passa un paràmetre anomenat "prefix" que és la IP que es vol encaminar. Seguidament s'arriba fins a la llista "Nexthop-list" on s'han de passar obligatòriament tres paràmetres la interfície de sortida, la IP de la pròxima interfície i el nom de la variable VRF que s'assigna com a "default".

El filtre amb JSON queda de la següent manera:

```
{
  "ipv4-items": {
    "inst-items": {
      "dom-items": {
        "Dom-list": {
          "name": "default",
          "rt-items": {
            "Route-list": {
              "prefix": "1.1.1.1",
              "nh-items": {
                "Nexthop-list": [
                  {
                    "nhIf": "eth1/1",
                    "nhAddr": "2.2.2.2",
                    "nhVrf": "default"
                  },
                  {
                    "nhIf": "eth1/2",
                    "nhAddr": "3.3.3.3",
                    "nhVrf": "default"
                  }
                ]
              }
            }
          }
```

```
        }
      }
    }
  }
}
```

A la sol·licitud que s'ha fet en el mètode "get_request()" es demana la informació dintre de "System" i amb el JSON es filtra el que es vol dintre de "System" fins a arribar a les variables que es vol configurar.

S'executa el programa i es comprova que s'han afegit les variables:

```
$ python restconf_ip_routing_table.py core2
```

```
RESTCONF ip routing table:

Core 2
+------------------+--------------+-----------------+
| IP               | Next Hop     | Via Interface   |
+==================+==============+=================+
| 192.168.30.0/24  | 20.0.0.101/32 | eth1/1          |
+------------------+--------------+-----------------+
| 192.168.30.0/24  | 20.0.0.117/32 | eth1/2          |
+------------------+--------------+-----------------+
| 1.1.1.1          | 2.2.2.2      | eth1/1          |
+------------------+--------------+-----------------+
| 1.1.1.1          | 3.3.3.3      | eth1/2          |
+------------------+--------------+-----------------+
| 209.165.201.0/24 | 20.0.0.106/32 | eth1/3          |
+------------------+--------------+-----------------+
| 209.165.201.0/24 | 20.0.0.61/32  | eth1/9          |
+------------------+--------------+-----------------+
| 192.168.26.0/24  | 20.0.0.101/32 | eth1/1          |
+------------------+--------------+-----------------+
| 192.168.26.0/24  | 20.0.0.117/32 | eth1/2          |
+------------------+--------------+-----------------+
| 192.168.12.0/24  | 20.0.0.17/32  | eth1/4          |
+------------------+--------------+-----------------+
```

6. Anàlisi de resultats

En el present projecte s'han definit les eines que s'han utilitzat per a l'automatització de xarxes; reservar el laboratori per virtualitzar xarxes de Cisco, els diferents formats que s'utilitzen, que són els models de dades YANG i com s'utilitzen, l'eina Ansible per automatitzacions i els protocols RESTCONG i NETCONF. També s'ha explicat les xarxes tipus campus i les diferents capes que interactuen.

Totes aquestes eines i funcionalitats ajuden a que les xarxes amb les quals s'està treballant siguin més autònomes, més escalables, es produeixi menys errors i aquests siguin més fàcilment solucionables i per últim que la xarxa sigui més visible i fiable.

7. Conclusions

Les xarxes definides per software i les eines d'automatització de xarxa són un canvi cap a la millora en diferencia de les xarxes tradicionals, des de tenir una xarxa més visible a la completa automatització de la configuració de la xarxa. L'automatització de xarxes és essencial on es busqui escalabilitat i optimització. Amb la realització d'aquest projecte s'han comprovat els beneficis que comporta l'automatització d'aquestes i els diferents protocols per realitzar-ho, s'ha explicat la posada en marxa i el procés de configuració dels diferents mecanismes d'automatització i extracció de dades com RESTCONF, NETCONF, Ansible i YANG.

8. Possibles ampliacions

Per a possibles ampliacions d'aquest projecte són la implementació de pyATS i Genie a la xarxa. La plataforma pyATS és un ecosistema que s'especialitza en l'automatització de testos reutilitzables i basats en dades i Genie és un programa que incorpora esdeveniments de *triggers* i verificació de característiques de la xarxa amb la utilització de models en format YAML. [27]

9. Bibliografia

[1] L. Kleinrock, «Information Flow in Large Communication Nets,» RLE Quarterly Progress Report, 1961.

[2] B. Leiner, «A Brief History of the Internet,» *Internet Society,* 2003.

[3] J. Clement, «Worldwide digital population as of October 2020,» November 2020. [En línia]. Available: https://www.statista.com/statistics/617136/digital-population-worldwide/. [Últim accés: January 2021].

[4] Cisco, «Cisco,» cisco, september 2020. [En línia]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. [Últim accés: january 2021].

[5] L. DiDio, «ITIC 2017 - 2018 Global Server Hardware, Server OS Reliability Survey,» Information Technology Intelligence Consulting, 2018.

[6] A. Telesis, «Allied Telesis,» Allied Telesis, 2017. [En línia]. Available: https://www.alliedtelesis.com/sites/default/files/openflow_feature_overview_guide_reve.pdf.

[7] S. S. L. M. O. Jason Edelman, Network Programmability and Automation: Skills for the Next-Generation Network Engineer, O'Reilly, 2018.

[8] M. Intelligence, «Research and Markets,» August 2020. [En línia]. Available: https://www.researchandmarkets.com/reports/4591229/network-automation-market-size-growth-trends.

[9] Cisco, «System Requirements,» [En línia]. Available: https://developer.cisco.com/docs/modeling-labs/#!system-requirements.

[10] C. Systems, «Cisco's PPDIOO Network Cycle,» 2011. [En línia]. Available: https://www.ciscopress.com/articles/article.asp?p=1697888.

[11] Cisco, «Cisco,» [En línia]. Available: https://www.cisco.com/c/en/us/support/security/anyconnect-secure-mobility-client-v4-x/model.html#~tab-documents.

[12] A. Galaxy, «galaxy.ansible.com,» [En línia]. Available: https://galaxy.ansible.com/home.

[13] Ansible, «ansible,» Ansible, [En línia]. Available: https://docs.ansible.com/ansible/latest/collections/cisco/ios/?extIdCarryOver=true&sc_cid=701f2000001OH7YAAW.

[14] Ansible, «Ansible,» [En línia]. Available: https://docs.ansible.com/ansible/latest/collections/cisco/nxos/?extIdCarryOver=true&sc_cid=701f2000001OH7YAAW.

[15] Ansible, «Ansible,» [En línia]. Available: https://docs.ansible.com/ansible/latest/collections/cisco/asa/index.html.

[16] Wikipedia, «wikipedia,» [En línia]. Available: https://en.wikipedia.org/wiki/YANG.

[17] ietf, «datatracker.ietf,» [En línia]. Available: https://datatracker.ietf.org/doc/html/rfc8040#section-1.3.

[18] IETF, «datatracker.ietf,» ietf, [En línia]. Available: https://datatracker.ietf.org/doc/html/rfc4741.

[19] IETF, «datatrcker.ietf,» [En línia]. Available: https://datatracker.ietf.org/doc/html/rfc6241#section-1.2.

[20] Cisco, «ciscolive,» Cisco, 9 June 2019. [En línia]. Available: https://www.ciscolive.com/c/dam/r/ciscolive/us/docs/2019/pdf/BRKCRS-2031.pdf.

[21] Wkipedia, «Wikipedia,» Cisco, [En línia]. Available: https://en.wikipedia.org/wiki/Cisco_IOS.

[22] Cisco, «cisco,» Cisco, [En línia]. Available: https://www.cisco.com/c/en/us/products/ios-nx-os-software/nx-os/index.html.

[23] Cisco, «ciscolive.com,» [En línia]. Available: https://www.ciscolive.com/c/dam/r/ciscolive/us/docs/2019/pdf/BRKCRS-2031.pdf.

[24] Python, «pypi.org,» [En línia]. Available: https://pypi.org/project/requests/.

[25] S. Bhushan. [En línia]. Available: https://github.com/ncclient/ncclient.

[26] python, «pypi.org,» [En línia]. Available: https://pypi.org/project/tabulate/.

[27] Cisco, «ciscolive,» [En línia]. Available: https://www.ciscolive.com/c/dam/r/ciscolive/emea/docs/2018/pdf/DEVNET-1480.pdf.