

GRADO EN INGENIERÍA TELEMÁTICA



VNIVERSITAT
DE VALÈNCIA

TRABAJO DE FIN DE GRADO

**AUTOMATIZACIÓN DE CONFIGURACIÓN DE REDES
MASIVAS A TRAVÉS DE ANSIBLE**

AUTOR:
ALIN MIHAI CRETU BRINZA

TUTORÍA:
MIGUEL GARCÍA PINEDA
ANTONIO MARI ROMERO

JULIO, 2018

GRADO EN INGENIERÍA TELEMÁTICA

TRABAJO DE FIN DE GRADO

AUTOMATIZACIÓN DE CONFIGURACIÓN DE REDES MASIVAS A TRAVÉS DE ANSIBLE

AUTOR:
ALIN MIHAI CRETU BRINZA

TUTORÍA:
MIGUEL GARCÍA PINEDA
ANTONIO MARI ROMERO

TRIBUNAL:

PRESIDENTE/PRESIDENTA: VOCAL 1:

VOCAL 2: FECHA DE DEFENSA:

CALIFICACIÓN:

Agradecimientos

Después de seis meses intensos de trabajo, escribo este apartado de agradecimientos para finalizar mi Trabajo Final de Grado. Ha sido un período de aprendizaje intenso, no solo a nivel técnico sino también a nivel personal. La realización de este proyecto ha tenido un gran impacto en mí y es por eso que me gustaría agradecer a todas aquellas personas que me han ayudado y apoyado durante este proceso.

Académicamente me gustaría agradecer a todos los profesores que me han apoyado y ayudado durante toda la carrera y a todos mis compañeros con los cuales he pasado buenos momentos.

En particular, me gustaría agradecer a Miguel García Pineda, por ofrecerme la posibilidad de realizar este trabajo, también por la amistad y todo el apoyo ofrecido durante la realización del Trabajo Final de Grado (TFG).

Agradecer a Toni Marí Romero, por ofrecer a la universidad la posibilidad de colaborar con una gran empresa como Red Hat. Por la ayuda prestada en momentos de bloqueo, por enseñarme nuevas tecnologías y por estar siempre pendiente de mí.

A Costel y Iulia, mis padres, por estar en todo momento a mi lado y ofrecerme siempre lo mejor, ayudandome en todo lo que he necesitado y apoyandome hasta el final poniendome una buena cara o diciendome unas bonitas palabras.

Y cómo no, quiero hacer mención especial a mi novia, Toñi. Gracias a sus consejos empece la carrera universitaria y ahora mismo la estoy finalizando. Le tengo que agradecer también por la paciencia que ha tenido en los momentos más intensos y toda la ayuda que me ha dado todo este tiempo, además de las magníficas ideas que me ha aportado.

1. Resumen

Hoy en día hay una gran cantidad de datos e información circulando por la red, y las empresas tienen la necesidad de almacenar toda esa información. Sin embargo, debido a que la cantidad de datos es muy elevada se necesita automatizar el proceso de almacenamiento y gestión de los datos.

Empezando por las pequeñas empresas hasta las multinacionales, todas ellas necesitan dispositivos de red como por ejemplo: switches, routers, servidores para aplicaciones, servidores de bases de datos. Todo ello para poder tener comunicación fuera y dentro de la empresa y para poder gestionar todo lo mencionado anteriormente acerca del manejo de los datos, lo que hace que aumenten considerablemente la cantidad de estos dispositivos.

Por lo tanto, para que un administrador de sistemas pueda garantizar el perfecto funcionamiento y una rápida resolución de los problemas, la automatización de las tareas en la actualidad, es una necesidad más que una comodidad.

Volviendo unos años atrás, para gestionar y configurar varios dispositivos como por ejemplo Routers, había que conectarse uno por uno y modificar o añadir nuevas configuraciones, lo cual no era nada eficiente. Hecho que muchas empresas a día de hoy siguen haciendo, perdiendo mucho tiempo en llegar al dispositivo en concreto en caso de emergencia.

Así, el presente TFG tiene como propósito explotar el potencial que tienen las herramientas de automatización de redes y así poder introducir estas nuevas tecnologías en el día a día de desarrolladores y administradores de sistemas, para que la utilización de estas herramientas sea más frecuente.

Por un lado, se hará una comparativa entre las distintas herramientas disponibles y para la realización de todas las pruebas junto con la documentación de las mismas se ha escogido Ansible, una herramienta creada por Michael DeHann y que fue adquirida por Red Hat en 2015 y donde más adelante veremos detalladamente los motivos de su elección.

Se replicará, mediante el simulador gráfico de red GNS3, una topología basada en el CCNA: Routing and Switching Essentials de CISCO, que con la ayuda de una simple configuración inicial, se procederá a la automatización de todas las configuraciones, necesarias para el correcto funcionamiento de toda la red, mediante la herramienta Ansible. Además, se implementará la topología diseñada en dispositivos físicos para ver una aplicación real de esta herramienta de automatización.

Por tanto, se proporcionarán los conocimientos necesarios para que un administrador de sistemas o un desarrollador pueda replicar los mismos pasos y también poder aplicar dichos conocimientos a una red física.

Este TFG se ha realizado con la ayuda de Toni Marí, empleado de Red Hat, ayudando durante el periodo de desarrollo con los problemas técnicos sobre Ansible.

Palabras clave

Automatización, Redes, Ansible, Cisco, Routing, Switching, VLAN, Protocolos de enrutamiento, Python, Linux, AWX, Ansible Tower.

Abstract

Nowadays there is a large amount of data and information going through the network, and companies need to store all that information. However, due to the fact that the amount of data is so high, it is necessary to automate the process of storing and managing data.

All, from small businesses to multinationals, need network devices such as switches, routers, application servers, database servers, etc., in order to enable communication inside and outside the company and to be able to manage all the above mentioned about the handling of data, makes the number of these devices increase considerably.

Therefore, so that a system administrator can guarantee perfect operation and rapid problem solving, automation of tasks today is a necessity rather than a convenience.

Going back some years, in order to manage and setup multiple devices such as routers, you had to connect each one by one and modify or add new configurations, which was not efficient. Something that many companies still do today, wasting a lot of time in reaching a particular device in case of emergency.

Thus, the purpose of the present TFG is to exploit the potential of network automation tools in order to introduce these new technologies into the daily lives of developers and system administrators and to make the use of these tools more frequent.

On the one hand, a comparison will be made between the different automation tools available. For all tests and their documentation, a tool created by Michael DeHann and purchased by glsRedHat in 2015, has been chosen, Ansible. We will look at the reasons for his choice further on.

The graphical network simulator GNS3 will be used to replicate a topology based on the CCNA: Routing and Switching Essentials by CISCO. Which, with the help of a simple initial configuration, all configurations necessary for the correct operation of the entire network will be automated using the Ansible tool.

In addition, the topology designed will be implemented in physical devices, to see the real application of this automation tool.

Therefore, the necessary knowledge will be provided so that system administrator or a developer will be able to replicate the same steps and also be able to apply this knowledge to a physical network.

This TFG was done in collaboration with Toni Marí, an employee of Red Hat, helping throughout the development period with the technical problems on Ansible.

Keywords

Automation, Networking, Ansible, Cisco, Routing, Switching, VLAN, Routing Protocols, Python, Linux, AWX, Ansible Tower.

Índice

| | |
|---|-----------|
| 1. Resumen | 7 |
| Lista de figuras | 11 |
| Lista de tablas | 13 |
| Listado | 13 |
| Lista de acrónimos | 15 |
| 2. Introducción | 17 |
| 2.1. Automatización | 17 |
| 2.2. Automatización de redes | 17 |
| 2.3. Metodología de resolución | 18 |
| 2.4. Organización de la memoria | 19 |
| 3. Motivación y objetivos | 21 |
| 3.1. Motivación | 21 |
| 3.2. Objetivos | 23 |
| 4. Estado del arte | 25 |
| 4.1. Automatización de las redes | 25 |
| 4.2. OpenSource | 25 |
| 4.3. Herramientas de automatización | 26 |
| 4.3.1. Ventajas y Desventajas | 27 |
| 4.4. Emuladores de red | 29 |
| 4.4.1. Ventajas y Desventajas | 29 |
| 5. Especificaciones | 31 |
| 5.1. Análisis de requisitos | 31 |
| 5.2. Estimación temporal | 32 |
| 5.3. Estimación de los costes | 38 |
| 5.3.1. Costes de hardware | 38 |
| 5.3.2. Costes de software | 39 |
| 5.3.3. Costes de personal | 39 |
| 5.3.4. Costes totales | 40 |
| 5.3.5. Viabilidad | 40 |
| 5.4. Análisis de riesgos | 40 |
| 6. Desarrollo del Proyecto | 47 |
| 6.1. Herramientas utilizadas | 47 |
| 6.1.1. Ansible | 47 |
| 6.1.1.1. Arquitectura de Ansible | 48 |
| 6.1.1.2. Instalación Ansible | 58 |

| | | |
|---------------------|---|------------|
| 6.1.2. | Ansible Tower | 60 |
| 6.1.3. | AWX | 61 |
| 6.1.4. | GNS3 | 62 |
| 6.1.4.1. | Instalación GNS3 | 64 |
| 6.1.5. | VMWare Workstation | 66 |
| 6.1.6. | Docker | 66 |
| 6.2. | Fase de análisis | 67 |
| 6.2.1. | Casos de Uso | 67 |
| 6.2.2. | Formatos expandidos | 68 |
| 6.3. | Fase de Diseño | 71 |
| 6.3.1. | Diseño de la topología | 71 |
| 6.3.2. | Estructura del sistema | 73 |
| 6.3.2.1. | Automatización mediante Ansible por Interfaz de Línea de Comandos (CLI) | 73 |
| 6.4. | Fase de implementación | 75 |
| 6.4.1. | Importación del proyecto de Ansible | 78 |
| 6.4.2. | Implementación para utilizar Ansible desde CLI | 80 |
| 6.4.2.1. | Host Inventory | 81 |
| 6.4.2.2. | ansible.cfg | 81 |
| 6.4.2.3. | Roles | 85 |
| 6.4.3. | Implementación para utilizar Ansible desde AWX | 101 |
| 7. | Pruebas y resultados | 107 |
| 7.1. | Pruebas unitarias | 107 |
| 7.2. | Pruebas de integración | 109 |
| 7.2.1. | Pruebas desde la máquina Admin | 110 |
| 7.2.2. | Pruebas desde AWX | 112 |
| 7.3. | Pruebas del sistema | 117 |
| 8. | Conclusiones y Trabajo Futuro | 121 |
| 8.1. | Conclusiones | 121 |
| 8.2. | Trabajo Futuro | 122 |
| Glosario | | 123 |
| Bibliografía | | 125 |

Índice de figuras

| | | |
|-----|---|----|
| 1. | Comparativa entre la CLI de Telnet y SSH | 18 |
| 2. | Cantidad de tiempo dedicada. | 22 |
| 3. | Tareas y estimación de tiempos | 35 |
| 4. | Tareas, hitos y fechas de inicio y fin. | 36 |
| 5. | Tareas, hitos y fechas de inicio y fin. | 37 |
| 6. | Lenguajes de programación más usados | 48 |
| 7. | Herramientas de automatización más usadas. | 48 |
| 8. | Arquitectura Ansible. [22] | 49 |
| 9. | Ejemplo de los directorios de Ansible | 52 |
| 10. | Ejemplo de la estructura de un rol | 56 |
| 11. | Panel de control Ansible Tower | 60 |
| 12. | Panel de control AWX | 62 |
| 13. | Interfaz gráfica GNS3 | 62 |
| 14. | Topología de red compleja | 63 |
| 15. | Componentes adicionales | 64 |
| 16. | Directorio de instalacion. | 64 |
| 17. | Instalacion GNS3 en MacOS | 65 |
| 18. | Casos de uso | 68 |
| 19. | Casos de uso - Lanzar Playbook desde CLI | 69 |
| 20. | Casos de uso - Lanzar Playbook desde AWX | 69 |
| 21. | Casos de uso - Crear archivos de configuracion | 70 |
| 22. | Casos de uso - Iniciar funcionamiento de la red | 70 |
| 23. | Casos de uso - Detener funcionamiento de la red | 71 |
| 24. | Diseño de la topologia en GNS3 | 72 |
| 25. | Diseño de la topologia en GNS3 | 73 |
| 26. | Estructura del proyecto desde CLI | 74 |
| 27. | Estructura del proyecto con AWX | 75 |
| 28. | Máquina virtual GNS3 VM | 76 |
| 29. | Software GNS3 | 76 |
| 30. | Interfaz interna para el servidor GNS3 | 77 |
| 31. | Interfaz para tener acceso a internet | 77 |
| 32. | Escenario de GNS3 con la topologia | 78 |
| 33. | GitLab | 79 |
| 34. | GitHub | 80 |
| 35. | Proyecto clonado. | 83 |
| 36. | Permisos modificados. | 84 |
| 37. | Error en la conexión. | 84 |
| 38. | Estructura de la carpeta roles. | 85 |
| 39. | Estructura del rol Backup. | 86 |
| 40. | Estructura del rol config_sw_este. | 87 |
| 41. | Estructura del rol config_sw_oeste. | 90 |
| 42. | Estructura del rol config_r_oeste. | 93 |
| 43. | Estructura del rol config_r_norte. | 97 |

| | | |
|-----|---|-----|
| 44. | Estructura del rol config_r_este. | 99 |
| 45. | Crear usuario en AWX. | 102 |
| 46. | Crear credenciales para los dispositivos. | 102 |
| 47. | Crear inventario. | 103 |
| 48. | Crear máquinas en inventario. | 103 |
| 49. | Importar proyecto de GitHub. | 104 |
| 50. | Comprobación de la conexión con Sw-este. | 105 |
| 51. | Dispositivos físicos. | 105 |
| 52. | Error. | 108 |
| 53. | Correcto. | 108 |
| 54. | Comprobación de la configuración. | 109 |
| 55. | Sw-Este. | 110 |
| 56. | Sw-Oeste. | 111 |
| 57. | R-Oeste. | 111 |
| 58. | R-Norte. | 112 |
| 59. | Backup. | 112 |
| 60. | Resultado de la ejecución. | 112 |
| 61. | Inventario completo. | 113 |
| 62. | Creación de la plantilla de trabajo. | 113 |
| 63. | Ejecución de la plantilla de trabajo. | 114 |
| 64. | Plantilla ejecutándose. | 114 |
| 65. | Ejecución del rol config_sw_este. | 114 |
| 66. | Ejecución del rol config_sw_oeste. | 115 |
| 67. | Ejecución de la plantilla para configurar dispositivos físicos. | 115 |
| 68. | Ejecución de la plantilla para configurar dispositivos físicos. | 116 |
| 69. | Ejecución de la plantilla para configurar dispositivos físicos. | 116 |
| 70. | Ejecución de la plantilla para configurar dispositivos físicos. | 117 |
| 71. | Ejecución de la plantilla para configurar dispositivos físicos. | 117 |
| 72. | Prueba de conexión. | 118 |
| 73. | Prueba de conexión. | 118 |
| 74. | Prueba de conexión. | 118 |
| 75. | Prueba de conexión. | 118 |
| 76. | Prueba de conexión. | 119 |

Índice de tablas

| | | |
|-----|---|----|
| 1. | Comparacion de las herramientas de automatizacion | 27 |
| 2. | Comparacion de emuladores de red | 29 |
| 3. | Requisitos Funcionales | 31 |
| 4. | Requisitos Funcionales | 32 |
| 5. | Requisitos no Funcionales | 32 |
| 6. | Costes de software | 39 |
| 7. | Costes de personal | 39 |
| 8. | Costes totales | 40 |
| 9. | Riesgos | 41 |
| 10. | Mitigacion y contingencias | 44 |

Listado

| | | |
|-----|---|----|
| 1. | Ejemplo inventory file | 49 |
| 2. | Ejemplo Playbook | 51 |
| 3. | Ejemplo fichero de variables | 53 |
| 4. | Ejemplo incluir variables | 53 |
| 5. | Ejemplo incluir variables | 54 |
| 6. | Ejemplo salida de datos | 54 |
| 7. | Código ejecución playbook | 55 |
| 8. | Ejemplo colocación de variables | 55 |
| 9. | Ejemplo fichero de variables | 55 |
| 10. | Código para crear un rol | 56 |
| 11. | Ejecución de Playbook | 58 |
| 12. | Ejemplo código Ad-Hoc | 58 |
| 13. | Ejemplo códigos Ad-Hoc | 58 |
| 14. | Ejemplo instalación Ansible en MacOs | 59 |
| 15. | Ejemplo instalación Ansible en MacOs | 59 |
| 16. | Ejemplo instalación Ansible en Ubuntu | 59 |
| 17. | Repositorio para Debian | 59 |
| 18. | Ejemplo instalación Ansible en Debian | 59 |
| 19. | Ejemplo instalación Ansible en CentOS | 60 |
| 20. | Web con información sobre AWX | 61 |
| 21. | Ejemplo instalación GNS3 en Ubuntu | 65 |
| 22. | Página web oficial GNS3 | 65 |
| 23. | Subir un proyecto a un repositorio. | 78 |
| 24. | Clonar el proyecto. | 79 |
| 25. | Crear usuarios en Admin | 80 |
| 26. | Comprobación | 80 |
| 27. | Host Inventory | 81 |
| 28. | /etc/hosts | 81 |
| 29. | ansible.cfg | 82 |
| 30. | /etc/network/interfaces | 82 |

| | | |
|-----|--|-----|
| 31. | /etc/sudoers | 83 |
| 32. | Clonar proyecto | 83 |
| 33. | Permisos para el usuario técnico. | 83 |
| 34. | Rutas por defecto. | 84 |
| 35. | Rutas por defecto. | 85 |
| 36. | sh_run.yml | 86 |
| 37. | create_vlan.yml. | 87 |
| 38. | config_vlan.yml. | 87 |
| 39. | create_vlan.yml. | 89 |
| 40. | config_vlan.yml. | 90 |
| 41. | group_vars/switch/main.yml. | 92 |
| 42. | config_dhcp.yml. | 93 |
| 43. | config_interfaces.yml. | 94 |
| 44. | config_ospf.yml. | 95 |
| 45. | config_subinterfaces.yml. | 95 |
| 46. | vars/main.yml. | 96 |
| 47. | config_interfaces.yml. | 97 |
| 48. | config_ospf.yml. | 98 |
| 49. | vars/main.yml. | 98 |
| 50. | site.yml. | 100 |
| 51. | URL's de los repositorios. | 101 |
| 52. | Instalacion de AWX. | 101 |
| 53. | Comprobar Playbook. | 107 |
| 54. | Comprobar Playbooks. | 109 |
| 55. | Comprobar Playbook. | 110 |
| 56. | ansible localhost -m setup | 127 |
| 57. | Configuración básica para sw-este. | 128 |
| 58. | Rutas por defecto. | 129 |
| 59. | Rutas por defecto. | 130 |
| 60. | Rutas por defecto. | 130 |
| 61. | Rutas por defecto. | 131 |

Índice de acrónimos

API REST

Application Programming Interface REpresentational State Transfer

CLI

Interfaz de Línea de Comandos

DHCP

Dynaminc Host Configuration Protocol

DSL

Lenguaje específico de dominio

IDC

International Data Corporation

IOS

Internetwork Operating System

OSD

Open Source Definition

OSPF

Open Shortest Path First

SDN

Software Defined Networking

SSH

Secure Shell

Telnet

Telecommunication Network

TFG

Trabajo Final de Grado

TIC

Tecnología de la Información y la Comunicacióñ

UML

lenguaje unificado de modelado

2. Introducción

2.1. Automatización

Cuando hablamos de automatización y el uso de controles automáticos que garantizan la precisión y la calidad, todo el mundo piensa de manera negativa y que va a suponer el fin de los puestos de trabajo. Pero no es del todo cierto, y esta manera de ver las cosas ha generado desde siempre un revuelo, aunque nunca se haya logrado una automatización perfecta.

Volviendo a tiempos anteriores, se habla de automatización desde la prehistoria, pero en la década de 1940 en la Ford Motor Company, es donde se aplicó este término como tal. En los años 50 y 60, el matemático Norbert Wiener, dio la voz de alarma de que a través de la cibernetica y la introducción de los ordenadores el empleo iba tener un perjudicial desenlace. Pero la automatización ha tenido un proceso mucho más lento de lo que se esperaba.

La automatización puede definirse como una tecnología que se ocupa de realizar un proceso mediante comandos programados combinados con un control automático de retroalimentación para asegurar la correcta ejecución de las instrucciones. [1]

En la actualidad, el uso de la automatización depende cada vez más de los ordenadores y de los dispositivos relacionados con las telecomunicaciones. Por lo tanto, gracias a estos avances podemos llegar a un nivel de capacidad y rendimiento muy superior a la de los seres humanos para realizar las mismas actividades. Todo ello lleva a una mayor complejidad y sofisticación a la hora de automatizar los sistemas.

2.2. Automatización de redes

Marin Casado, es un científico de la computación que mientras se doctoraba en la Universidad de Stanford, trabajaba en el protocolo OpenFlow [2] que sirvió para impulsar el movimiento Software Defined Networking (SDN) [3]. A raíz de esto y mientras trabajaba para el gobierno nacional de EEUU, Casado tuvo la oportunidad de manejar multitud de ordenadores según sus necesidades. Después de todo, observó que la configuración de los dispositivos de red mediante la CLI era muy ineficiente.

La forma en la que se gestionaban las redes no había cambiado nunca en más de 20 años. La única mejora fue la migración de Telecommunication Network (Telnet) a Secure Shell (SSH). Aunque parezca que no haya cambiado nada a simple vista como podemos observar en la Figura 1, la gran mejora fue que ahora SSH usa técnicas de cifrado que hacen que la información que viaja por el medio de comunicación no pueda ser leída.

Para poder sacar el máximo rendimiento al protocolo SSH, han ido apareciendo herramientas de automatización de redes tales como: Ansible, Chef, Puppet, SaltStack o CFEngine para poder gestionar una red o unos dispositivos de red desde un único punto de control.

La finalidad de este proyecto es, por tanto, realizar un estudio exhaustivo, tanto teórico como práctico sobre la herramienta de automatización Ansible [4]. Concretamente, veremos como poder usar esta herramienta para automatizar la configuración de una red local y los beneficios que supondría a la hora de configurar dispositivos de red de forma masiva.

Para ello, vamos a usar una red propuesta por CISCO, para una sesión de Troubleshooting sobre Routing and Switching.

Además, para destacar y explicar adecuadamente todos los conceptos necesarios para utilizar Ansible, se va a crear una guía a modo de manual. Por tanto, además de los conceptos teóricos, cualquier persona va a ser capaz de aprender a manejar y utilizar esta herramienta para hacer pruebas o con fines profesionales.

En el Capítulo 4 se va a detallar en profundidad el motivo de haber elegido Ansible y GNS3 [5], así como una comparativa entre las distintas herramientas de automatización.

PROBLEM: NETWORK AGILITY
Not Much has Changed in the Last 20 Years

| 1994 | 2014 |
|--|---|
| <pre>Router> enable Router# configure terminal Router(config)# enable secret cisco Router(config)# ip route 0.0.0.0 0.0.0.0 20.2.2.3 Router(config)# interface ethernet0 Router(config-if)# ip address 10.1.1.1 255.0.0.0 Router(config-if)# no shutdown Router(config-if)# exit Router(config)# interface serial0 Router(config-if)# ip address 20.2.2.2 255.0.0.0 Router(config-if)# no shutdown Router(config-if)# exit Router(config)# router rip Router(config-router)# network 10.0.0.0 Router(config-router)# network 20.0.0.0 Router(config-router)# exit Router# copy running-config startup-config Router# disable Router></pre> <p>Terminal Protocol: Telnet</p> | <pre>Router> enable Router# configure terminal Router(config)# enable secret cisco Router(config)# ip route 0.0.0.0 0.0.0.0 20.2.2.3 Router(config)# interface ethernet0 Router(config-if)# ip address 10.1.1.1 255.0.0.0 Router(config-if)# no shutdown Router(config-if)# exit Router(config)# interface serial0 Router(config-if)# ip address 20.2.2.2 255.0.0.0 Router(config-if)# no shutdown Router(config-if)# exit Router(config)# router rip Router(config-router)# network 10.0.0.0 Router(config-router)# network 20.0.0.0 Router(config-router)# exit Router# copy running-config startup-config Router# disable Router></pre> <p>Terminal Protocol: SSH</p> |

Figura 1: Comparativa entre la CLI de Telnet y SSH
[6]

2.3. Metodología de resolución

Si se desea alcanzar de manera exitosa el objetivo de este trabajo final de grado se debe poseer buenos conocimientos sobre las tecnologías descritas anteriormente que se obtendrán siguiendo un orden de trabajo e investigación como el que se va a ver a continuación.

- El primer paso consiste en hacer el estudio del estado del arte sobre la automatización de redes y sus herramientas.
- Tras haber entendido los conceptos del primer punto, se procederá a detallar extensamente el funcionamiento de la herramienta escogida para este proyecto, Ansible, y se argumentará su elección como solución necesaria al aumento exponencial de configuraciones que hay que realizar cada día sobre los equipos que nos conectan y lo que lo hace destacar del resto de herramientas de automatización.

- Por otra parte, del mismo modo que en el paso anterior, se explicará y detallará el motivo de haber escogido el simulador de red (GNS3). así como su funcionamiento y ventajas.
- Con el fin de agrupar todos los conocimientos aprendidos se creará una guía a modo de manual para saber como crear una red con GNS3, instalar Ansible y que se ha instalado correctamente.
- Por lo tanto, teniendo todo claro se avanzará en el desarrollo del proyecto, generando la red propuesta por CISCO y tras ello se empezará a utilizar Ansible para poder automatizar todas las configuraciones.
- Finalmente, se comprobará que la red funciona correctamente y que si añadimos más dispositivos a la misma, podemos reutilizar todas las automatizaciones creadas anteriormente para replicar las configuraciones.
- Para entender y asegurar que todo lo creado funciona correctamente se implementará Ansible en dispositivos físicos.

2.4. Organización de la memoria

La memoria de este TFG se ha estructurado y clasificado separando los apartados principales por capítulos. A continuación, veremos un breve resumen de cada capítulo.

1-Resumen

En el capítulo 1 podemos encontrar un breve resumen de lo que se va a realizar en este trabajo final de grado, así como una pequeña introducción al tema.

2-Introducción

En el capítulo 2 se explica brevemente el origen de la automatización y como ha ido evolucionando hasta el punto en el que la automatización de redes y el uso de sus herramientas es primordial en el día a día.

3-Motivación y objetivos

En el capítulo 3 se presentan algunos motivos para introducir la automatización de redes juntos con las herramientas de automatización en el ámbito profesional y también se detallan los objetivos principales de este trabajo final de grado.

4-Estado del arte

El capítulo 4 trata de explicar a los lectores en el funcionamiento de la herramienta de automatización, las distintas tecnologías que existen y los motivos de la elección de esta.

4-Especificaciones

En el capítulo 5 se realizará un análisis de los requisitos, de los riesgos y también se realizará una estimación temporal y de costes.

6-Desarrollo del Proyecto

El capítulo 6 contendrá las tres fases por las cuales debe pasar un proyecto. La fase de análisis, de diseño y la fase de implementación. Es uno de los capítulos más importantes ya que es donde se desarrollará todo lo especificado en los requisitos.

7-Pruebas y Resultados

En el capítulo 7 se realizarán las pruebas necesarias para asegurar que todo lo implementado en el capítulo anterior funciona correctamente y además se recogerán los resultados obtenidos.

8-Conclusiones y Trabajo Futuro

Por último, en el capítulo 8 se expondrán las conclusiones obtenidas tras haber utilizado Ansible y las ideas para desarrollar en trabajos futuros.

3. Motivación y objetivos

3.1. Motivación

Existen muchos campos donde se puede aplicar la automatización, y uno de ellos es en el campo de las redes y está pensada como una medida para hacer las cosas más rápido. Aunque la automatización de los procesos lleve a una reducción en los tiempos de implementación y agilice los cambios de configuraciones, gran parte de las empresas siguen utilizando los métodos tradicionales para hacer las cosas. Por este motivo, hay que cambiar el modo de pensar de los administradores de sistemas o desarrolladores, para que empiecen a plantearse la automatización como una mejora y no como un proceso negativo y sin sentido. Hoy, más que nunca, la manera en la que los departamentos de la Tecnología de la Información y la Comunicación (TIC) gestionan las redes puede marcar la diferencia entre una empresa exitosa o una ineficiente. Esto es debido a que cada día aumenta exponencialmente las operaciones de negocio a través de la red.

La mayoría de los trabajadores de TIC intentan solucionar los problemas con cambios simples y únicos, lo que hace que una red no solo sea más difícil de mantener y gestionar, sino también de automatizar. Se debe tratar la automatización de la red como una tarea secundaria o un añadido extra, pero desde el primer día en el que se estructura una red, para que después sea todo mucho más fluido.

A día de hoy, todavía hay muchos departamentos de TIC que no incluyen esta útil forma de gestionar una red, lo que hace que se exija un tiempo considerable del personal. Pero si desde el comienzo del montaje lógico de una red, como hemos mencionado anteriormente, se introdujese la automatización, el gran uso de las aplicaciones, de los servidores, del almacenamiento y las configuraciones de las redes no necesitaría una carga tan grande de trabajo del personal de TIC.

Un estudio de International Data Corporation (IDC) muestra como este trabajo se ha convertido en una actividad tediosa. Se llega a una conclusión de que los responsables de TIC gastan un 76,8 % del tiempo en mantener los entornos existentes y que menos de la cuarta parte (23,2 %) del tiempo lo dedican a actividades que pudiesen dar un valor añadido a la empresa, ver en la Figura 2 [7]

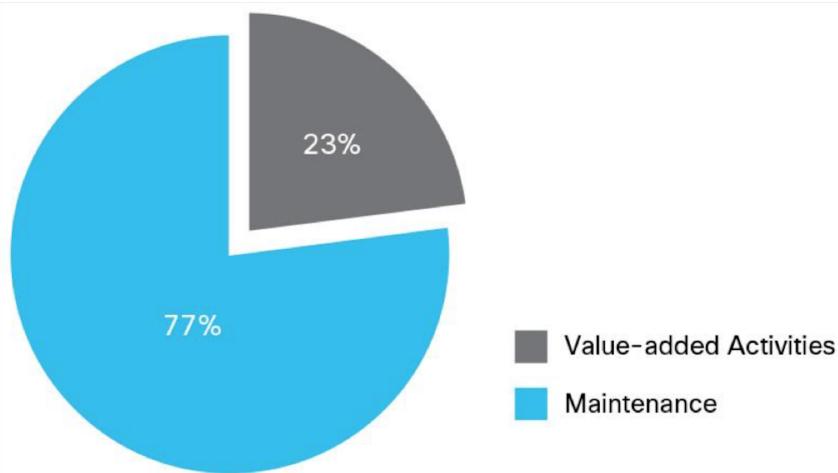


Figura 2: Cantidad de tiempo dedicada.

Como con números se ve todo más claro, el siguiente ejemplo hará que entendamos más el por qué debemos introducir ya la automatización. Por ejemplo, si necesitamos actualizar la imagen Internetwork Operating System (IOS) de un router CISCO. Esto implica muchas tareas como cargar la imagen en el router, comprobar su validez, modificar la variable de arranque. Si un ingeniero tarda media hora en realizar todas estas tareas sobre un solo router, está muy bien, pero si hubiesen mil routers no sería factible, ya que un ingeniero estaría 30.000 minutos para configurar todos los routers. Gracias a las herramientas de automatización como en nuestro caso Ansible podemos dedicar solamente 30 minutos para preparar toda la configuración necesaria y mediante Ansible lanzar todas las tareas en paralelo. [6]

Grandes compañías como Cisco [8], Hewlett Packard Enterprise [9], VMware [10], Oracle [11], Red Hat [12] y otras muchas más, están utilizando ya infraestructuras convergentes.

La infraestructura convergente es una solución definida por hardware diseñada para superar las limitaciones e ineficiencias de la estructura de silos independientes del almacenamiento y la computación de la TIC tradicional. Para minimizar los problemas de compatibilidad y simplificar la gestión, las soluciones convergentes unen la computación, la red, el almacenamiento, los sistemas de administración y el software en un paquete preconfigurado que funciona y se gestiona como un único sistema convergente que acelera la obtención de beneficios. [9]

Estas infraestructuras harán que los departamentos de TIC en un futuro se ofrezcan como servicios, con modelos de cloud privada, para proporcionar velocidad, flexibilidad, seguridad e innovación a empresas modernas. Algunos objetivos de las TIC como servicio serían: crecimiento del negocio sin ampliar personal TIC, visibilidad mundial como resultado a la agilidad y proporcionar capacidad para ampliar el tamaño, el alcance y poder escalar los servicios prestados a los clientes.

Dado que las redes tradicionales utilizan mecanismos manuales para realizar las configuraciones como la CLI, hace que esto sea inviable en el tipo de red mencionada en el párrafo anterior. Esta sería otra gran motivación para inculcar las herramientas de automatización en el día a día.

En la actualidad, grandes empresas como Apple, Juniper, Grainger, WeightWatchers, SaveMart, NASA, Red Hat, etc. usan este tipo de herramientas. De modo que cualquier empresa que tenga una gran cantidad de dispositivos de red puede y debería utilizar la automatización para mejorar y seguir creciendo.

3.2. Objetivos

La principal idea de este proyecto es introducir tanto a profesionales como a estudiantes en la automatización de redes, así como realizar una demostración práctica, mediante Ansible, de cómo aplicar estas herramientas. Entre los objetivos a cumplir en el TFG se destacan:

- Comprender el impacto que tiene la automatización de redes en los entornos profesionales o particulares.
- Realizar una comparativa entre las distintas herramientas de automatización disponibles a día de hoy y los motivos de haber escogido Ansible para este trabajo.
- Detallar en profundidad el funcionamiento de Ansible desde lo más básico hasta el nivel presentado en este TFG y también se explicará el funcionamiento del simulador de red GNS3.
- Montar la red y poner en práctica lo aprendido en los apartados anteriores y por último realizar todas las pruebas para comprobar el correcto funcionamiento de la red.
- Para la realización de las pruebas se implementará en virtual la red diseñada y también en dispositivos físicos.

4. Estado del arte

En este capítulo se analizarán las herramientas y emuladores que pueden ser útiles para desarrollar este proyecto. Una vez realizada la comparación se expondrán los motivos de haber elegido estas.

Primero, se hablará de la automatización de las redes en el día a día. Segundo, se realizará una explicación del OpenSource [13]. Después, se realizará la comparación entre las distintas herramientas de automatización y de los emuladores de redes. Finalmente se expondrán las desventajas y ventajas de las herramientas elegidas.

Este es uno de los capítulos más importantes del TFG, donde se conocerá el estado actual de la tecnología usada en este proyecto.

4.1. Automatización de las redes

Como es sabido, el número de dispositivos en una red está aumentando constantemente. Los métodos tradicionales utilizados para la configuración de los equipos de red requieren mucho tiempo, teniendo en cuenta también los conocimientos técnicos específicos del proveedor. La automatización de la red es una solución para el ahorro de gastos operacionales, mejorando no sólo el tiempo empleado en la configuración de los dispositivos de red, sino también la eficiencia del mantenimiento de la red a través de procedimientos que son más fáciles de seguir e implementar a gran escala.

Resumiendo, los motivos más importantes para la introducción de la automatización de las redes son: la reducción del tiempo de configuración de los equipos y la facilidad de mantenimiento, además de aumentar la productividad, evitando manualidades y tareas repetitivas para enfocar los esfuerzos en aportar mayor valor al negocio. También mejora la seguridad de la red al eliminar el factor humano y aumenta la estabilidad de la misma. Este método representa el futuro de las redes, permitiendo la gestión de un mayor número de dispositivos de forma unitaria.

Gracias a la automatización de las redes se puede llegar a construir una cultura DevOps que permita eliminar divisiones dentro de la organización. Aunque últimamente se está hablando de NetDevOps, que se centra en la cultura, la colaboración, las herramientas y la automatización y se extiende a los arquitectos y operadores de redes. [14]

4.2. OpenSource

En este apartado se hablará sobre OpenSource [13] Este término, es el software cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de autor, son publicados bajo una licencia de código abierto o forman parte del dominio público. En las licencias compatibles con la Open Source Definition (OSD) [15] el propietario de los derechos de autor permite a los usuarios utilizar, cambiar y redistribuir el software, a cualquiera, para cualquier propósito, ya sea en su forma modificada o en su forma original.

Hoy en día, mucha gente prefiere software OpenSource por mucho motivos como por ejemplo:

- Tener el control total del código.
- Empezar a practicar.
- Para una mayor seguridad.
- Conseguir una estabilidad.

En consecuencia, esta forma de trabajo se está expandiendo y se está utilizando cada vez más. A continuación, se procederá a analizar las distintas herramientas de automatización disponibles.

4.3. Herramientas de automatización

En la actualidad existen una gran cantidad de herramientas de automatización ya que hay grandes necesidades en las empresas. Existen herramientas con una curva de aprendizaje más lenta y otras más rápida, lo que puede ser fundamental a la hora de elegir.

Antes de ver la comparación, se expondrán las principales diferencias, entre las herramientas que se van a comparar.

- *Basado en agentes - Sin agentes*: Algunas herramientas requieren que un agente (un software) se ejecute en el sistema o dispositivo que se está administrando. Poniendo como ejemplo una automatización de red, esto podría ser un problema, ya que no todos los sistemas operativos de red admiten agentes activos en el propio dispositivo.
- *Centralizado - Descentralizado*: Las herramientas que requieren un agente, necesitan también de un servidor maestro centralizado.
- *Protocolo personalizado - Protocolo basado en estándares*: Algunas herramientas tienen un protocolo personalizado, en la mayoría de los casos está unido con las arquitecturas basadas en agentes. Otras herramientas que trabajan sin agentes utilizan SSH como protocolo de transporte.
- *Lenguaje específico de dominio (DSL) - Formatos de datos basados en estándares*: Un DSL es un lenguaje diseñado especialmente para un dominio o herramienta en concreto. Pero otras herramientas aprovechan YAML ya que se considera un lenguaje de propósito general usando en este entorno.
- *Lenguajes*: Estas herramientas ofrecen la posibilidad de extender su funcionalidad mediante lenguajes de scripting de alto nivel. Algunas herramientas pueden utilizar Ruby aunque es bastante más costoso de aprender, otras utilizan Python.
- *Push - Pull*: Algunas herramientas y las que se va a utilizar en este proyecto, operan en el modo *push*, es decir, la información es proporcionada desde un único punto hacia los servidores o sistemas a configurar, mientras que otros utilizan el método *pull*, que consiste en hacer peticiones constantes.

Tabla 1: Comparacion de las herramientas de automatizacion

| Herramientas | Agente | Pull | Push | Lenguaje | Dispositivos |
|--|--------|------|------|----------|----------------------------|
|  ANSIBLE | ✗ | ✓ | ✓ | Python | hosts, dispositivos de red |
|  CFEngine | ✓ | ✓ | ✗ | C | hosts |
|  CHEF™ | ✓ | ✓ | ✗ | Ruby | hosts, dispositivos de red |
|  puppet labs | ✓ | ✓ | ✗ | Ruby | hosts, dispositivos de red |
|  SALTSTACK | ✓ | ✓ | ✓ | Python | hosts, dispositivos de red |

En la Tabla 1 se pueden observar las distintas herramientas disponibles a día de hoy y sus características.

Para la realización de este proyecto se ha escogido Ansible como herramienta de automatización. Aunque Ansible sea una de las herramientas de automatización más recientes está ganando mucho terreno. Su versatilidad le permite su aplicación en gran variedad de dispositivos. Tiene como socios a grandes proveedores de dispositivos de red como por ejemplo CISCO y Juniper Networks. Tiene una gran comunidad detrás trabajando en la creación de módulos, resolución de problemas y mejoras para la herramienta, por lo que se considera muy activa. También dispone de una opción de pago, Ansible Tower que se explicará en la sección de desarrollo del proyecto.

Con lo cual a continuación se expondrán las ventajas y desventajas y los motivos de su elección.

4.3.1. Ventajas y Desventajas

Se puede considerar Ansible una herramienta muy potente y con muchas ventajas. Aunque sea más novedosa que muchas otras del mercado, está ganando terreno a pasos gigantes. Lo que ha ayudado a tomar una decisión sobre que herramienta de automatización escoger han sido las ventajas y desventajas.

Ventajas

- *Simple y fácil de usar:* Esta es una gran ventaja que ayuda a muchos administradores a elegir Ansible para automatizar sus redes. Tiene una documentación clara y fácil de seguir, su curva de aprendizaje es corta en el dominio del tiempo y como las tareas se ejecutan de un modo secuencial ayuda a la resolución de problemas.
- *Escrito en Python:* El uso de Python en su implementación ayuda mucho a la hora de poner en marcha Ansible porque la mayoría de máquinas disponen de Python. Otras herramientas usan otros lenguajes más complejos como por ejemplo Ruby que resultan ser más difíciles. Los módulos de Ansible pueden ser desarrollados en cualquier lenguaje siempre y cuando la salida tenga formato JSON.
- *Ansible Galaxy:* Existe un repositorio en la nube desde el cual se pueden descargar roles que otras personas hayan implementado y así poder reutilizarlos, además de subir tus propios roles para que otras personas los puedan usar acelerando significativamente el tiempo de implementación.
- *YAML:* Este es el formato que se utiliza para implementar todos los Playbooks y demás archivos de configuración y encaja mejor con el formato JSON que usa Ansible para la salida de los datos, también facilita la gestión de la configuración y la automatización porque es fácil de interpretar, soporta comentarios y tiene la posibilidad de hacer referencias a otros elementos mediante las anclas (`!``).
- *Agentless:* La herramienta no requiere el uso de agentes desplegados en las máquinas remotas para poder administrarlas como sucede en otras herramientas. Al no tener un software instalado y ejecutándose en el destino que queremos automatizar significa que el rendimiento es mucho mayor y el mantenimiento menor.
- *SSH:* Las conexiones con los dispositivos a automatizar se realizan a través de SSH, por lo tanto se asegura la seguridad de los datos ya que se mandan cifrados, los dispositivos de red tienen implementado este protocolo, de modo que activándolo es suficiente para tener conexión entre las máquinas.
- *Push:* Como se ha explicado, Ansible trabaja sin agente y está gran ventaja lleva a otra ya que cuando un administrador quiere realizar la automatización de máquinas, solamente tiene que mandar la configuración al destino (Push). Aunque Ansible también dispone de una opción que se llama *ansible-pull*, que permite realizar una copia Ansible en los nodos remotos.
- *Soporte para Windows:* A día de hoy Ansible se puede utilizar para administrar máquinas de Windows, pero a través de PowerShell, lo cual hace que desde una máquina Linux se pueda administrar tanto equipos con Windows, Linux e incluso dispositivos de red como en este TFG.

Desventajas

- *Falta de interfaz gráfica:* Desde el inicio Ansible se ha manejado desde la CLI. Los desarrolladores de esta herramienta crearon una herramienta con interfaz gráfica llamada Ansible Tower, que posteriormente se convirtió en AWX del cual se hablará más adelante. AWX proporciona una pequeña interfaz para el uso de Ansible pero no es suficiente comparado con todo lo que se puede realizar usando la CLI.

- *Control de versiones:* Ansible no sirve para control de versiones, para ello se necesita utilizar otras herramientas como podría ser Git.

4.4. Emuladores de red

Antes de la existencia de la virtualización, los administradores de redes y los estudiantes, para poder realizar pruebas tenían que construir sus propios laboratorios con hardware físico o alquilar un rack durante un periodo de tiempo. Ninguna de las dos opciones mencionadas era adecuada ya que eran muy costosas y limitaban mucho el diseño de las redes. Posteriormente aparecieron programas de simulación de software como por ejemplo RouterSim, donde simplemente se simulaban comandos de CISCO. Por su parte CISCO ofrece un producto llamado Virtual Internet Routing Lab que es muy parecido a los emuladores de hoy en día, pero su principal problema es que requiere una cuota anual, limita el número de objetos que se pueden utilizar y solo utiliza sistemas operativos Cisco simulados por lo tanto estos dos no se van a tener en cuenta.

Dado que hay una gran cantidad de emuladores y con distintas características y limitaciones a continuación se va a realizar una comparación entre todos los emuladores de red disponibles en el mercado.

Tabla 2: Comparacion de emuladores de red

| Emuladores | Lenguaje | Sistema Operativo | GUI | Escalabilidad |
|-------------------|-----------------|--------------------------|------------|----------------------|
| GNS3 | — | Windows, Linux, MacOS | Disponible | Grande |
| OMNET++ | C++ | Windows, Linux, MacOS | Disponible | Media |
| Packet Tracer | — | Windows, Linux, MacOS | Disponible | Media |
| NS2 | C++, OCTL | Windows, Linux, MacOS | Limitado | Limitado |
| Jsim | Java | Windows, Linux, MacOS | Disponible | Media |

Dado que en este proyecto se está simulando un laboratorio de CCNA de CISCO, el emulador elegido es GNS3 ya que permite virtualizar los dispositivos reales de CISCO cosa que los otros no lo permitían.

Además de permitir emular los dispositivos reales de CISCO tiene las siguientes ventajas y desventajas.

4.4.1. Ventajas y Desventajas

Ventajas

- *Emulación de dispositivos:* Además de emular los sistemas operativos de CISCO, los programadores de GNS3 están trabajando día a día para integrar nuevos sistemas como por ejemplo Juniper, HP.

- *Nuevas tecnologías*: GNS3 tiene un enfoque más tecnológico. Se centra en las tecnologías SDN, en la seguridad de la red, en la tecnología de procesamiento en la nube y la capacidad de trabajar en equipo en proyectos.
- *Libertad para crear redes*: Dado que su escalabilidad es ilimitada, ofrece la posibilidad de crear los laboratorios necesarios para realizar prácticas para certificaciones o para crear prácticas educativas.
- *Conectividad*: Además de la conectividad virtual entre los dispositivos de GNS3, este emulador permite la conexión a Internet a través de una interfaz virtual que se conecta directamente a la interfaz de red física del ordenador.
- *Análisis la red*: GNS3 ofrece la posibilidad de analizar la red que se está implementando. Esto se puede conseguir con un programa que captura paquetes y los analiza llamado WireShark. [16]

Desventajas

- *IOS CISCO*: Una gran desventaja de GNS3 es que para poder utilizar los dispositivos de CISCO virtualizados, debe ser el usuario quien proporcione las imágenes de los sistemas operativos de los dispositivos de CISCO. Para ello se debe descargar de la página oficial. [17]
- *Gran uso de los recursos del PC*: El uso de la CPU del ordenador que está ejecutando GNS3 se ve afectado ya que incrementa considerablemente. Pero no es GNS3 el que realmente consume los recursos sino Dynamips, el programa interno de GNS3 que emula las imágenes de CISCO.
- *Las interfaces están siempre levantadas*: En una red física real, los enlaces punto a punto dependen del estado de las dos interfaces y si un extremo está desconectado el otro extremo se desconecta automáticamente. En el caso de GNS3 si un extremo tiene la interfaz deshabilitada la otra interfaz sigue habilitada pudiendo causar problemas en escenarios de enrutamiento.

Vistas las ventajas y desventajas de GNS3 se puede decir con seguridad que este emulador es el ideal para la realización de este TFG, dado que se utilizarán dispositivos CISCO.

5. Especificaciones

Concluido el capítulo donde se han estudiado todas las tecnologías y herramientas necesarias para la realización de este TFG, es necesario describir los requisitos a cumplir, tanto funcionales como no funcionales, siendo estos el resultado de la investigación previa acerca del proyecto y de las reuniones con el tutor. Además, una planificación estimada de costes y temporal es fundamental para llevar a cabo un TFG exitoso, por lo tanto, se incluirá en esta sección. Para finalizar se realizará también un análisis de riesgos para localizar los posibles problemas que puedan surgir durante la realización del proyecto, así como un plan de mitigación para esos riesgos. [18]

5.1. Análisis de requisitos

En este apartado se especificarán los requisitos que debe cumplir el TFG. Para realizar una planificación estimada del tiempo y de los costes se deben separar los requisitos en dos grupos: los funcionales que se muestran en la Tabla 3, 4 y los no funcionales en la Tabla 5.

Tabla 3: Requisitos Funcionales

| Ref. | Descripción del requisito |
|-----------|---|
| RF1 | Entender la necesidad de la automatización |
| Prioridad | |
| alta | Estudio exhaustivo sobre Ansible |
| Ref. | Descripción del requisito |
| RF2 | Estudio exhaustivo sobre las herramientas adicionales |
| Prioridad | |
| alta | |
| Ref. | Descripción del requisito |
| RF3 | |
| Prioridad | |
| alta | |

Tabla 4: Requisitos Funcionales

| Ref. | Descripción del requisito |
|------------------|--|
| RF4 | Elaborar un manual con las instalaciones de las herramientas media |
| Prioridad | |
| alta | Creación de la topología de red |
| Ref. | Descripción del requisito |
| RF5 | Implementación de la automatización mediante Ansible |
| Prioridad | |
| alta | Realizar pruebas para ver el correcto funcionamiento |
| Ref. | Descripción del requisito |
| RF6 | |
| Prioridad | |
| alta | |
| Ref. | Descripción del requisito |
| RF7 | |
| Prioridad | |
| alta | |

Tabla 5: Requisitos no Funcionales

| Ref. | Descripción del requisito |
|------------------|--|
| RNF1 | Utilizar GNS3 para crear la topología y virtualizar los dispositivos de Cisco alta |
| Prioridad | |
| alta | Utilizar VMware para crear el servidor de GNS3 |
| Ref. | Descripción del requisito |
| RNF2 | |
| Prioridad | |
| alta | |
| Ref. | Descripción del requisito |
| RNF3 | Utilizar Github para control de versiones alta |
| Prioridad | |
| alta | Utilizar Docker para ejecutar AWX media |
| Ref. | Descripción del requisito |
| RNF4 | |
| Prioridad | |
| media | Crear los Playbook para la automatización alta |
| Ref. | Descripción del requisito |
| RNF5 | |
| Prioridad | |
| alta | |

5.2. Estimación temporal

Una vez identificados los requisitos del sistema, se debe realizar una planificación estimada de los tiempos debido a las exigencias del mundo actual en el que los proyectos deben ser terminados en un periodo de tiempo acotado. Por lo tanto, se puede estimar

una fecha para acabar el proyecto, pero siempre considerando un margen de error. Para ello se aplicarán los conocimientos adquiridos en la asignatura Gestión de Proyectos. [18]

Para estimar los tiempos de todas las fases que contiene este proyecto se ha utilizado la técnica de *juicio de expertos*. Esta técnica consiste en realizar unas estimaciones por expertos que en este caso tienen unos conocimientos muy avanzados en el tema de las redes y la automatización.

En la Figura 3 se muestran las tareas a realizar en este TFG. A continuación, se definirán las principales tareas del proyecto:

- **Documentación:** En la primera fase del proyecto se realiza una búsqueda extensa de documentación acerca de herramientas de automatización, del software necesario para el desarrollo del proyecto, así como cursos para entender los lenguajes de programación usados en este TFG.
- **Análisis:** El siguiente paso sería la captación de requisitos, así como realizar un análisis de la estructura del proyecto y de la topología elegida para ser montada en GNS3.
- **Diseño:** La tercera tarea de esta planificación consistiría en diseñar la topología en GNS3 que ya se ha analizado anteriormente y diseñar la estructura que tendrá internamente Ansible para realizar todas las automatizaciones.
- **Implementación:** En esta fase se procederá a la implementación de todos los archivos básicos de configuración necesarios para el funcionamiento de los dispositivos, así como la creación de los archivos que se utilizarán más adelante para realizar la automatización.
- **Pruebas:** Llegados a la fase de pruebas, solamente queda ejecutar todo lo creado en la fase de implementación y posteriormente realizar la comprobación de que todo ha funcionado correctamente y hay conectividad entre los distintos dispositivos de red.
- **Redacción de la memoria:** Aunque la redacción de la memoria aparezca como una última fase del proyecto, realmente se han ido redactando apartados y recogiendo información, pero la finalización de la misma se ha realizado en esta última fase.

Para conocer si el proyecto avanza adecuadamente hay que definir una serie de hitos, que ayudan al proyectista a llevar un control y poder así avanzar sin errores.

Los hitos para este proyecto son los siguientes:

- **Aceptación del análisis:** Este hito abarca todos los objetivos y requisitos que tiene que cumplir el proyecto para ser finalizado correctamente.
- **Aceptación del diseño:** Ya que el diseño y la implementación están muy unidos, se va a agrupar en este hito estos dos grandes bloques de Diseño e Implementación y se presentará al finalizar la implementación.
- **Aceptación de las pruebas:** Una vez finalizado y aceptado la fase de diseño e implementación se procederá a realizar todas las pruebas pertinentes para comprobar que todo funciona correctamente.
- **Aceptación de la memoria final:** Es el último hito a entregar y es donde se dará el visto bueno a todo el trabajo realizado y a su documentación.

Para una correcta estimación de los tiempos, se ha consultado al profesor de la ETSE encargado de ser el tutor, y a la persona que representa la empresa Red Hat, ya que los dos son expertos en el tema de la automatización y de las redes y se ha realizado una estimación de los tiempos de cada tarea.

- **Miguel García Pineda:** Profesor de la Universidad de Valencia, licenciado en Ing. de Telecomunicación y Master en Tecnologías, Sistemas y Redes de Comunicaciones. Experto en el área de la Ingeniería Telemática y más concretamente en dos campos: las redes de sensores, IoT y las comunicaciones inalámbricas y por otro lado la distribución de contenido multimedia sobre redes IP.
- **Toni Marí Romero:** Graduado en Ing. Electrónica de Telecomunicación y Master en Tecnologías, Sistemas y Redes de Comunicaciones ahora es Senior Platform Consultant en Red Hat. Experto en sistemas Linux y en la comunidad OpenSource.

En la figura siguiente se muestran las tareas junto con las estimaciones de los expertos, en días. El peso de la estimación del tutor es del 50 % y de la persona de la empresa externa del 50 %.

| Tarea | Descripción | Estimación 1 | Estimación 2 | Media (días) |
|-----------|---|--------------|--------------|--------------|
| T1 | [-] Documentación | 21 | 22 | 21 |
| T1.1 | Búsqueda de herramientas de automatización | 2 | 2 | 2 |
| T1.2 | Documentación sobre Ansible | 3 | 5 | 4 |
| T1.3 | Documentación sobre topologías de red | 4 | 3 | 3 |
| T1.4 | Estudio del funcionamiento de Ansible | 3 | 5 | 4 |
| T1.5 | Estudio del funcionamiento de GNS3 | 3 | 3 | 3 |
| T1.6 | Estudio de la topología de red | 4 | 2 | 3 |
| T1.7 | Estudio del lenguaje de programación Python | 2 | 2 | 2 |
| T2 | [-] Análisis del proyecto | 12 | 7 | 10 |
| T2.1 | Análisis de requisitos | 3 | 2 | 3 |
| T2.2 | Análisis de la estructura del proyecto | 4 | 3 | 3 |
| T2.3 | Análisis de la topología | 5 | 2 | 4 |
| T3 | [-] Diseño del proyecto | 17 | 18 | 18 |
| T3.1 | Diseño de la topología en GNS3 | 9 | 8 | 9 |
| T3.2 | Diseño de la estructura de Ansible | 8 | 10 | 9 |
| T4 | [-] Implementación | 39 | 41 | 40 |
| T4.1 | Implementación de los archivos de configuración | 11 | 10 | 10 |
| T4.2 | Implementación de las configuraciones básicas | 12 | 12 | 12 |
| T4.3 | Implementación de los Playbooks de Ansible | 16 | 19 | 18 |
| T5 | [-] Pruebas | 20 | 23 | 22 |
| T5.1 | Conexión de los dispositivos de red | 8 | 7 | 8 |
| T5.2 | Ejecución de configuraciones básicas | 4 | 5 | 5 |
| T5.3 | Ejecución de los Playbooks específicos | 4 | 7 | 5 |
| T5.4 | Prueba de conectividad entre los dispositivos | 4 | 4 | 4 |
| T6 | Redacción de la memoria | 13 | 13 | 13 |
| T6.1 | Redacción de la memoria | 13 | 13 | 13 |
| | TOTAL DÍAS | 122 | 124 | 124 |

Figura 3: Tareas y estimación de tiempos

Una vez recogida toda la información de los tiempos se puede observar que según la estimación, el proyecto acabaría en 124 días, considerando días laborables todos, excepto sábados, domingos y días festivos de la Comunidad. De este modo, contando 20 días al mes, el proyecto acabaría en 6 meses y 4 días aproximadamente.

Este tipo de proyectos se deben realizar de manera individual para que el proyectista demuestre que es capaz de aplicar los conocimientos aprendidos y a desarrollar proyectos de investigación, por lo tanto, hay que tener en cuenta que ninguna tarea ha podido ser realizada en paralelo.

En la Figura 4 se muestra la planificación de los tiempos con los hitos incluidos y las fechas de inicio y fin:

| Tarea | Descripción | Duración | Inicio | Finalizar | Predecesor |
|---------------|---|----------|----------|-----------|------------|
| T1 | ■ Documentación | 21d | 02/01/18 | 30/01/18 | |
| T1.1 | Búsqueda de herramientas de automatización | 2d | 02/01/18 | 03/01/18 | |
| T1.2 | Documentación sobre Ansible | 4d | 04/01/18 | 09/01/18 | 2 |
| T1.3 | Documentación sobre topologías de red | 3d | 10/01/18 | 12/01/18 | 3 |
| T1.4 | Estudio del funcionamiento de Ansible | 4d | 15/01/18 | 18/01/18 | 4 |
| T1.5 | Estudio del funcionamiento de GNS3 | 3d | 19/01/18 | 23/01/18 | 5 |
| T1.6 | Estudio de la topología de red | 3d | 24/01/18 | 26/01/18 | 6 |
| T1.7 | Estudio del lenguaje de programación Python | 2d | 29/01/18 | 30/01/18 | 7 |
| T2 | ■ Análisis del proyecto | 10d | 31/01/18 | 13/02/18 | |
| T2.1 | Análisis de requisitos | 3d | 31/01/18 | 02/02/18 | 8 |
| T2.2 | Análisis de la estructura del proyecto | 3d | 05/02/18 | 07/02/18 | 10 |
| T2.3 | Análisis de la topología | 4d | 08/02/18 | 13/02/18 | 11 |
| HITO 1 | Aceptación del análisis | | | | |
| T3 | ■ Diseño del proyecto | 18d | 14/02/18 | 09/03/18 | |
| T3.1 | Diseño de la topología en GNS3 | 9d | 14/02/18 | 26/02/18 | 12 |
| T3.2 | Diseño de la estructura de Ansible | 9d | 27/02/18 | 09/03/18 | 15 |
| T4 | ■ Implementación | 40d | 12/03/18 | 04/05/18 | |
| T4.1 | Implementación de los archivos de configuración | 10d | 12/03/18 | 23/03/18 | 16 |
| T4.2 | Implementación de las configuraciones básicas | 12d | 26/03/18 | 10/04/18 | 18 |
| T4.3 | Implementación de los Playbooks de Ansible | 18d | 11/04/18 | 04/05/18 | 19 |
| HITO 2 | Aceptación del diseño | | | | |
| T5 | ■ Pruebas | 22d | 07/05/18 | 05/06/18 | |
| T5.1 | Conexión de los dispositivos de red | 8d | 07/05/18 | 16/05/18 | 20 |
| T5.2 | Ejecución de configuraciones básicas | 5d | 17/05/18 | 23/05/18 | 23 |
| T5.3 | Ejecución de los Playbooks específicos | 5d | 24/05/18 | 30/05/18 | 24 |
| T5.4 | Prueba de conectividad entre los dispositivos | 4d | 31/05/18 | 05/06/18 | 25 |
| HITO 3 | Aceptación de las pruebas | | | | |
| T6 | Redacción de la memoria | 13d | 06/06/18 | 22/06/18 | |
| T6.1 | Redacción de la memoria | 13d | 06/06/18 | 22/06/18 | 26 |
| HITO 4 | Aceptación de la memoria final | | | | |

Figura 4: Tareas, hitos y fechas de inicio y fin.

Para finalizar con el apartado de la planificación temporal en la Figura 5 se puede observar el diagrama de Gantt con todas las actividades o tareas del proyecto y los tiempos necesarios para terminar cada una.

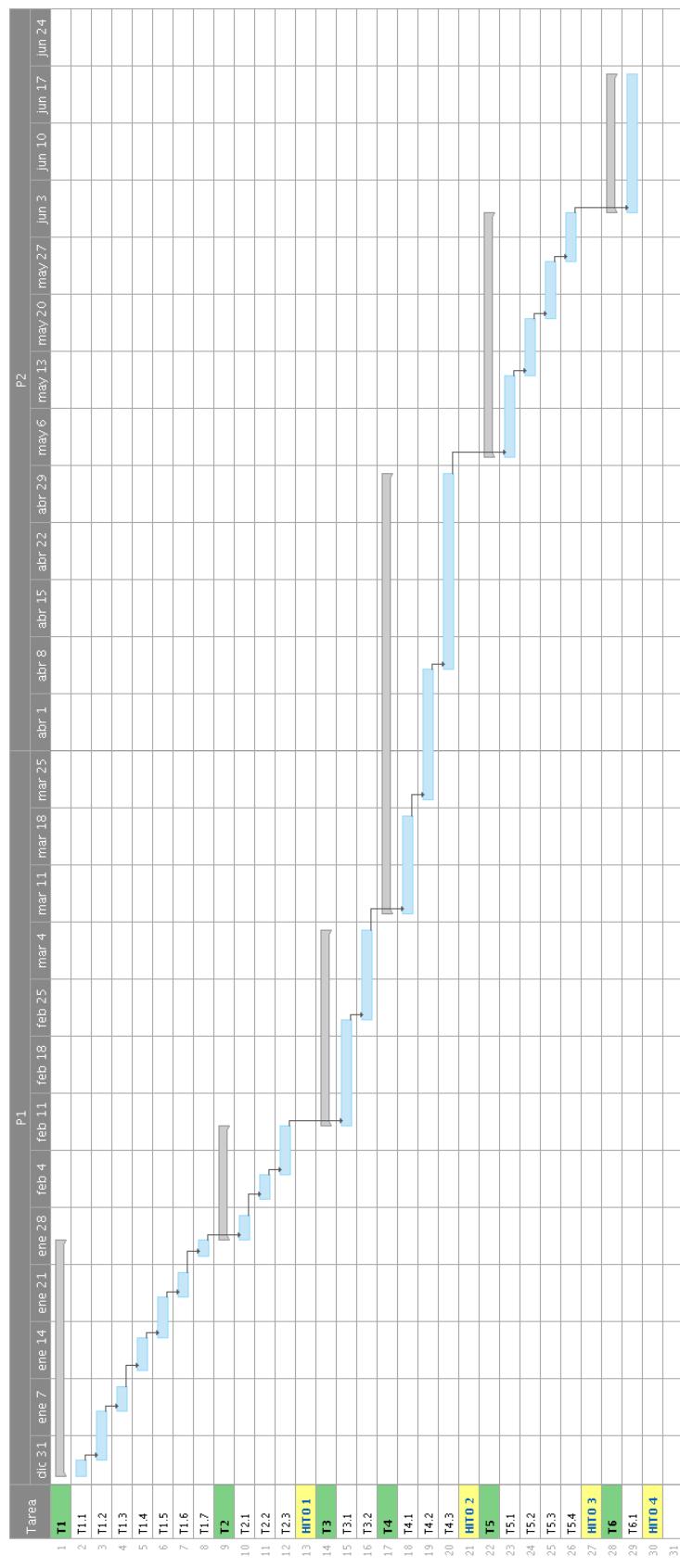


Figura 5: Tareas, hitos y fechas de inicio y fin.

5.3. Estimación de los costes

Este apartado tiene una gran importancia ya que hacer un estudio de los costes puede servir para conocer de antemano la viabilidad del proyecto y si la empresa podría soportar el gasto. Se detallarán los costes del hardware, software y de recursos humanos.

5.3.1. Costes de hardware

Para este proyecto se necesitan los siguientes elementos:

- Un ordenador para instalar GNS3, Ansible y poder ejecutar todos los Playbooks. El modelo es un MacBook Pro con las siguientes especificaciones:
 - Procesador 2,7 GHz Intel Core I5
 - 8 Gb de memoria Ram
 - 1 TB de almacenamiento
 - Pantalla de 13 pulgadas

Este ordenador tiene 1 año de uso, y según la vida media de los portátiles, se podría obtener un buen rendimiento durante 4 años más. Por lo tanto, la amortización anual de este portátil será de 280€/año. Dado que la estimación temporal del proyecto es de 6 meses y 4 días, los costes se calculan de la siguiente forma:

$$\frac{280\text{€}}{12\text{meses}} \cdot 6,4 = 147\text{€} \quad (1)$$

- Dos ordenadores de gama media, para realizar las pruebas finales.
Los dos ordenadores serán de la marca Asus de una gama media con un coste de 500€ y para este tipo de ordenadores se estima un periodo de amortización de 3 años por lo tanto la amortización anual será de 166€/año. Para calcular su coste se utiliza la misma fórmula que en el caso anterior:

$$\frac{166\text{€}}{12\text{meses}} \cdot 6,4 = 88,53\text{€} \quad (2)$$

- Se necesitan 2 Switches Cisco de capa 2.
Los dos Switches Cisco de capa 2 tienen un coste de 384,37€/unidad y se puede obtener un rendimiento durante 10 años. De modo que la amortización anual para estos dos dispositivos será de 76,87€/año. El coste de estos dos switches para la realización del proyecto será:

$$\frac{76,87\text{€}}{12\text{meses}} \cdot 6,4 = 40,99\text{€} \quad (3)$$

- Además se necesita también 2 Routers Cisco 7200.
Los dos Routers Cisco modelo 7200 tienen un coste de 600€/unidad. El rendimiento de estos dispositivos es muy alto y puede llegar a los 15 años. Por lo tanto, la amortización anual será de 80€/año y el coste para este proyecto será:

$$\frac{80\text{€}}{12\text{meses}} \cdot 6,4 = 42,66\text{€} \quad (4)$$

De modo que los costes de hardware para este TFG tendrán un valor de 319,18€.

5.3.2. Costes de software

Los costes del software que se va a utilizar en este proyecto están desglosados en la Tabla 6. En el TFG se utiliza software de código libre u OpenSource y también de pago. El periodo de amortización de estos productos suele ser de 6 años, lo que equivale a un 10 % de amortización para la realización de este proyecto.

Tabla 6: Costes de software

| Software | Descripción | Precio | Amortización | Coste |
|-------------------------|--------------------------------------|--------|--------------|---------------|
| Ansible | Herramienta de automatización | 0€ | 10 % | 0€ |
| AWX | Ansible con interfaz gráfica | 0€ | 10 % | 0€ |
| GNS3 | Emulador de redes y dispositivos | 0€ | 10 % | 0€ |
| VMWare | Virtualizador de Sistemas Operativos | 88,95€ | 10 % | 8,89€ |
| Visual Paradigm | Organizador de proyectos | 349€ | 10 % | 34,9€ |
| Microsoft Office | Herramienta de ofimática | 120€ | 10 % | 12€ |
| SmartSheet | Planificador de proyectos | 0€ | 10 % | 0€ |
| Docker | Virtualizador de Sistemas Operativos | 0€ | 10 % | 0€ |
| Sublime Text | Herramienta para programación | 0€ | 10 % | 0€ |
| GitHub | Gestor de proyectos | 0€ | 10 % | 0€ |
| TOTAL + 21 % IVA | | | | 55,79€ |

5.3.3. Costes de personal

Para la realización de este proyecto sería necesario dos perfiles profesionales, un jefe de proyecto y un administrador de sistemas. Puesto que el TFG se realiza de manera individual el coste del personal se estima según la duración del proyecto por el juicio de expertos realizado anteriormente y según el salario de un único Ingeniero que está definido en el boletín del estado. El salario base de un Ingeniero según el Boletín del Estado con fecha de 2017 es de 1253,16€/mes. [19]

Por lo tanto, el coste de recursos humanos se puede ver en la Tabla 7:

Tabla 7: Costes de personal

| Personal | Salario (mes) | Cantidad de trabajo | Duración (meses) | Coste |
|-------------------------|---------------|---------------------|------------------|-----------|
| Ingeniero en Telemática | 1253,16€ | 100 % | 6,3 | 7.894,90€ |

5.3.4. Costes totales

Para poder definir los costes totales se le debe añadir los costes indirectos que equivalen a un 20 % de los costes directos y que se pueden ver en la Tabla 8:

Tabla 8: Costes totales

| Concepto | Costes Directos | Costes Indirectos | Coste Total |
|--------------|-----------------|-------------------|------------------|
| Hardware | 319,18€ | 63,83€ | 383,01€ |
| Software | 55,79€ | 11,15€ | 66,94€ |
| Personal | 7894,90€ | 1.578,98€ | 9.473,88€ |
| TOTAL | | | 9.923,83€ |

El coste más elevado según las estimaciones para este proyecto, es el de recursos humanos y por lo tanto el presupuesto total necesario sería de **9923,83€**.

5.3.5. Viabilidad

Los costes económicos han sido desglosados en tres apartados, hardware, software y recursos humanos. Pese a realizar la estimación de costes de recursos humanos teniendo en cuenta a un Ingeniero, es el estudiante el que realiza todo el trabajo, por tanto, los costes de personal para este proyecto no supondrán ningún problema ya que como se ha dicho, es el estudiante el que realizará este proyecto.

Una parte de los dispositivos hardware, como los routers, switches y los dos ordenadores de gama media han sido suministrados por el tutor, dado que habían existencias en los distintos laboratorios. Esto ayuda a que los costes de hardware para este proyecto sean también nulos.

Por último, los costes de software son también nulos porque la mayoría de productos son de OpenSource y como el TFG se está realizando en un entorno educativo, los productos que tienen un coste acaban teniendo coste igual a 0€ por la existencia de licencias de estudiantes.

Finalmente, se considera que el proyecto es viable ya que el coste económico es nulo, el proyectista es capaz de desarrollar toda la tecnología necesaria y además no existe ninguna ley que pueda ser incumplida durante la realización de este proyecto.

5.4. Análisis de riesgos

Un factor muy importante a la hora de realizar la planificación es tener en cuenta los riesgos que existen y que pueden aparecer durante el desarrollo del proyecto. Realizar un análisis de los riesgos es una buena práctica antes de empezar a desarrollar cualquier proyecto, de modo que en la Tabla 9 se recogen los riesgos más probables de ocurrir durante este TFG. Para cada riesgo se asignará una probabilidad y el impacto que tendría y con estas dos variables se calcularía cuál de ellos sería máximo.

Tabla 9: Riesgos

| Cod | Riesgo | Probabilidad | Impacto | RIESGO |
|----------|--|--------------|---------|------------|
| A | Elaboración de la planificación | | Máx. | 3,5 |
| A.1. | Las definiciones de la planificación, de los recursos y del producto han sido impuestas por el cliente o un directivo superior, y no están equilibradas. | 40 % | 5 | 2,0 |
| A.2. | No se puede construir un producto de tal envergadura en el tiempo asignado. | 30 % | 4 | 1,2 |
| A.3. | Un retraso en una tarea produce retrasos en cascada en las tareas dependientes. | 50 % | 6 | 3,0 |
| A.4. | Las áreas desconocidas del producto llevan más tiempo del esperado en el diseño y en la implementación. | 70 % | 5 | 3,5 |
| B | Organización y Gestión | | Máx. | 4,2 |
| B.1. | El tiempo diario del que se dispone no es suficiente | 60 % | 7 | 4,2 |
| B.2. | El ciclo de revisión/decisión de la directiva es más lento de lo esperado. | 30 % | 8 | 2,4 |
| B.3. | La planificación es demasiado mala para ajustarse a la velocidad de desarrollo deseada. | 30 % | 6 | 1,8 |
| B.4. | Los planes del proyecto se abandonan por la presión, llevando al caos y a un desarrollo ineficiente. | 15 % | 8 | 1,2 |
| C | Ambiente/Lugar de Desarrollo | | Máx. | 4,0 |
| C.1. | Los espacios están sobreutilizados, son ruidosos o distraen. | 20 % | 6 | 1,2 |
| C.2. | Las herramientas de desarrollo no están disponibles en el momento deseado. | 40 % | 7 | 2,8 |
| C.3. | La curva de aprendizaje para la nueva herramienta de desarrollo es más larga de lo esperado. | 50 % | 8 | 4,0 |
| D | Requisitos | | Máx. | 4,0 |
| D.1. | Los requisitos se han adaptado, pero continúan cambiando. | 40 % | 8 | 3,2 |
| D.2. | Los requisitos no se han definido correctamente. | 30 % | 6 | 1,8 |
| D.3. | Se añaden requisitos extra. | 50 % | 8 | 4,0 |

| Cod | Riesgo | Proba- bilidad | Impacto | RIESGO |
|------|---|-------------------|---------|--------|
| E | Proyecto | | Máx. | |
| E.1. | Los módulos propensos a tener errores necesitan más trabajo de comprobación, diseño e implementación. | 60 % | 5 | 3,0 |
| E.2. | El desarrollo de funciones innecesarias alarga la planificación. | 30 % | 4 | 1,2 |
| E.3. | Los problemas de compatibilidad con el sistema existente necesitan un trabajo extra de comprobación, diseño e implementación. | 50 % | 5 | 2,5 |
| E.4. | El requisito de trabajar con varios sistemas operativos o entornos desconocidos necesita más tiempo del esperado. | 20 % | 4 | 0,8 |
| F | Personal | | Máx. | 4,2 |
| F.1. | La falta de motivación y de moral reduce la productividad. | 40 % | 6 | 2,4 |
| F.2. | La falta de la especialización necesaria aumenta los defectos y la necesidad de repetir el trabajo. | 45 % | 8 | 3,6 |
| F.3. | El personal necesita un tiempo extra para acostumbrarse a trabajar con herramientas o entornos nuevos. | 60 % | 7 | 4,2 |
| F.4. | El personal necesita un tiempo extra para aprender un lenguaje de programación nuevo. | 30 % | 6 | 1,8 |
| G | Diseño e Implementación | | Máx. | 2,8 |
| G.1. | Un diseño demasiado sencillo no cubre las cuestiones principales, con lo que hay que volver a diseñar e implementar. | 20 % | 5 | 1 |
| G.2. | Un diseño demasiado complejo exige tener en cuenta complicaciones innecesarias e improductivas en la implementación. | 40 % | 7 | 2,8 |
| G.3. | Un mal diseño implica volver a diseñar e implementar. | 10 % | 3 | 0,3 |

| Cod | Riesgo | Probabilidad | Impacto | RIESGO |
|------|--|--------------|---------|--------|
| H | Proceso | | Máx. | |
| H.1. | Un control de calidad inadecuado hace que los problemas de calidad que afectan a la planificación se conozcan tarde. | 60 % | 6 | 3,6 |
| H.2. | La falta de rigor (ignorar los fundamentos y estándares del desarrollo de software) conduce a un consumo de tiempo innecesario. | 10 % | 7 | 0,7 |
| H.3. | El exceso de rigor (aferramiento burocrático a las políticas y estándares de software) lleva a gastar más tiempo en gestión del necesario. | 15 % | 6 | 0,9 |
| H.4. | La creación de informes de estado a nivel de directiva lleva más tiempo al desarrollador de lo esperado. | 20 % | 5 | 1 |

Si se llegasen a producir estos riesgos se darían muchas dificultades para finalizar correctamente el proyecto, para ello en la Tabla 10 se han creado unas medidas de mitigación y contingencia que podrían suponer un gran ahorro de tiempo y dinero.

Tabla 10: Mitigación y contingencias

| Cod | Riesgo 1 | Valor |
|--------------------------------|---|-------|
| A.4. | Las áreas desconocidas del producto llevan más tiempo del esperado en el diseño y en la implementación. | 3,5 |
| MEDIDAS DE MITIGACIÓN | | |
| 1 | Buscar proyectos que puedan servir de referencia. | |
| 2 | Realizar un cursos de formación sobre la herramienta. | |
| MEDIDAS DE CONTINGENCIA | | |
| 1 | Contactar con el tutor de la universidad o de la empresa. | |
| 2 | Dedicar más tiempo a la búsqueda de documentación. | |
| Cod | Riesgo 2 | Valor |
| B.1. | El tiempo diario del que se dispone no es suficiente. | 4,2 |
| MEDIDAS DE MITIGACIÓN | | |
| 1 | Buena organización del día a día. | |
| 2 | Aprovechar al máximo los tiempos. | |
| MEDIDAS DE CONTINGENCIA | | |
| 1 | Pedir ayuda si hay alguna complicación. | |
| 2 | Dedicación intensiva al proyecto. | |
| Cod | Riesgo 3 | Valor |
| C.3. | La curva de aprendizaje para la nueva herramienta de desarrollo es más larga de lo esperado. | 4 |
| MEDIDAS DE MITIGACIÓN | | |
| 1 | Contactar con el soporte técnico de la herramienta. | |
| 2 | Aprovechar al máximo los tiempos. | |
| MEDIDAS DE CONTINGENCIA | | |
| 1 | Realizar cursos específicos. | |
| 2 | Tutorías con el responsable de la empresa. | |
| Cod | Riesgo 4 | Valor |
| D.3. | Se añaden requisitos extra. | 4 |
| MEDIDAS DE MITIGACIÓN | | |
| 1 | Los requisitos deben estar bien definidos al comienzo. | |
| 2 | Simplificar la finalidad del proyecto para cuadrar tiempos. | |
| MEDIDAS DE CONTINGENCIA | | |
| 1 | Los requisitos deben ser redefinidos con mucha claridad. | |
| 2 | Los requisitos nuevos deben ser viables para el tiempo restante. | |

| Cod | Riesgo 5 | Valor |
|--------------------------------|--|-------|
| E.1. | Los módulos propensos a tener errores necesitan más trabajo de comprobación, diseño e implementación. | 3 |
| MEDIDAS DE MITIGACIÓN | | |
| 1 | Revisar todos los archivos de configuración antes de ejecutar. | |
| 2 | Desarrollar los módulos estructurados y bien detallados. | |
| MEDIDAS DE CONTINGENCIA | | |
| 1 | Realizar todas las pruebas pertinentes para asegurar su correcto funcionamiento. | |
| 2 | Simplificar dichos módulos. | |
| Cod | Riesgo 6 | Valor |
| F.3. | El personal necesita un tiempo extra para acostumbrarse a trabajar con herramientas o entornos nuevos. | 4,2 |
| MEDIDAS DE MITIGACIÓN | | |
| 1 | Consultar foros y páginas de soporte sobre la herramienta. | |
| 2 | Lectura completa de la documentación. | |
| MEDIDAS DE CONTINGENCIA | | |
| 1 | Asistir a charlas de gente experta en el tema. | |
| 2 | Dedicar más horas a practicar de las definidas. | |
| Cod | Riesgo 7 | Valor |
| G.2. | Un diseño demasiado complejo exige tener en cuenta complicaciones innecesarias e improductivas en la implementación. | 2,8 |
| MEDIDAS DE MITIGACIÓN | | |
| 1 | Comprobar la viabilidad técnica y económica del proyecto. | |
| 2 | Asegurar todos los recursos hardware y software necesarios, antes de empezar. | |
| MEDIDAS DE CONTINGENCIA | | |
| 1 | Pedir ayuda ante la presencia de cualquier complicación. | |
| 2 | Intentar simplificar el diseño cumpliendo los mismos requisitos. | |
| Cod | Riesgo 8 | Valor |
| H.1. | Un control de calidad inadecuado hace que los problemas de calidad que afectan a la planificación se conozcan tarde. | 3,6 |
| MEDIDAS DE MITIGACIÓN | | |
| 1 | Acudir a todas las citas con los tutores. | |
| 2 | Realización de pruebas para ver si cumple los requisitos. | |
| MEDIDAS DE CONTINGENCIA | | |
| 1 | Redefinir con exactitud las soluciones. | |
| 2 | Ampliar plazos o realizar horas extra. | |

6. Desarrollo del Proyecto

Una vez establecidos los requisitos y estimada la planificación del proyecto, se procederá al desarrollo del mismo. Todos los proyectos de ingeniería tienen que pasar por tres fases: fase de análisis, fase de diseño y fase de implementación. [20]

Antes de entrar en los detalles de cada fase se van a detallar en profundidad las distintas herramientas que se van a utilizar en este TFG.

6.1. Herramientas utilizadas

6.1.1. Ansible

Ansible es una herramienta de OpenSource patrocinada por Red Hat, para la gestión, implementación y orquestación de la configuración de TIC. Ansible posee el único lenguaje de automatización que se puede utilizar en todos los equipos de TIC. Esta herramienta proporciona soluciones para todo tipo de empresas y para automatizar todas las aplicaciones, desde servidores en la nube, hasta contenedores.

Está diseñada para que sea única, segura y muy confiable. Su curva de aprendizaje es rápida ayudando a los administradores, desarrolladores e incluso técnicos de la TIC. Aunque es una herramienta con la cual se puede crear cosas muy complejas, tiene una gran facilidad para interpretar el código, lo cual es accesible para todos los niveles.

“Ansible is quite fun to use right away. As soon as you write five lines of code it works. With SSH and Ansible I can send commands to 500 servers without having even used the servers before.” [4]

Mark Mass, Unix/Linux Systems Administrator
BINCKBANK

Resumiendo, Ansible es útil, como se ha visto en la sección de estado del arte, para los siguientes aspectos:

- *Gestión de la configuración:* Se puede recuperar o modificar configuraciones en varios dispositivos que están en un inventario, que se va a especificar más adelante.
- *Despliegue de aplicaciones:* Muchas veces se necesita desplegar o instalar aplicaciones en servidores, aplicar parches específicos e incluso configurar las aplicaciones instaladas, Ansible puede realizar todas las tareas mencionadas.
- *Automatización de tareas:* Las tareas se pueden crear para unos dispositivos en concreto o distintas tareas para distintos dispositivos y todo ello se puede programar para que se ejecute de forma periódica o cuando el administrador lo crea conveniente.
- *Orquestación de TIC:* Para entender este concepto, se puede decir que si necesitamos actualizar ciertos Routers o Switches, Ansible es capaz de realizar los pasos secuenciales para aislar el Router en particular, subir la configuración, aplicar la configuración y pasar al siguiente dispositivo basándose en variables que recibe de la tarea anterior.

Antes de hablar sobre su Arquitectura, en las Figuras 6 y 7 se puede observar los resultados de una encuesta realizada por *Network to Code*, llamada *NetDevOps Survey*, donde recoge las respuestas de 307 ingenieros de redes sobre el uso de las herramientas de automatización. Se puede apreciar que Python es el lenguaje más usado y Ansible es la herramienta de automatización más usada. [21]

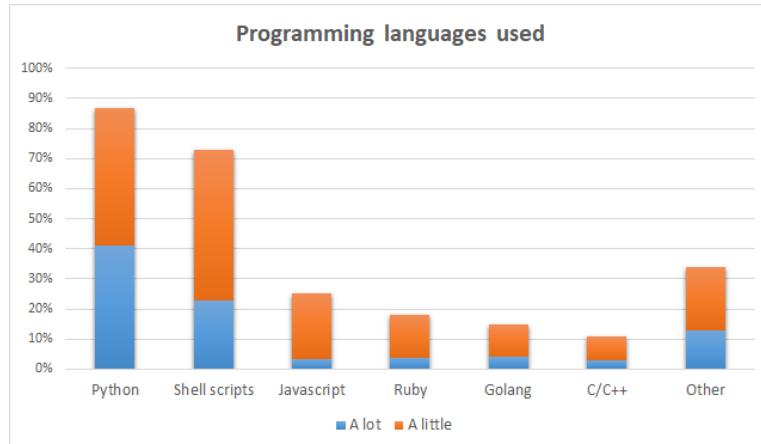


Figura 6: Lenguajes de programación más usados

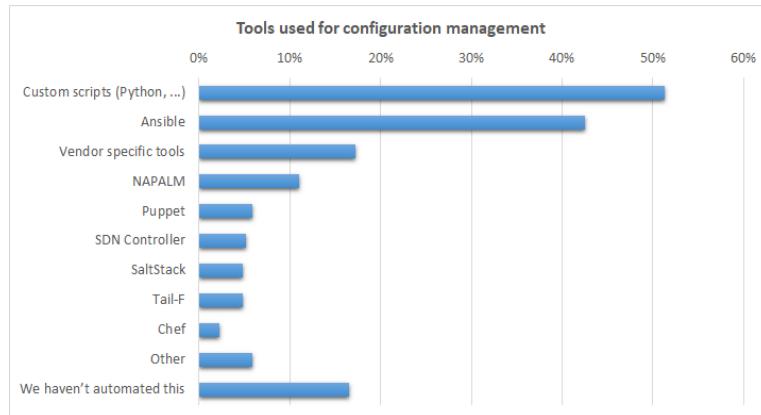


Figura 7: Herramientas de automatización más usadas.

6.1.1.1 Arquitectura de Ansible

Como se puede ver en la Figura 8, la idea es tener un único centro de control o varios desde donde se pueden lanzar los comandos a los distintos dispositivos remotos, así como ejecutar instrucciones secuenciadas a través de playbooks.

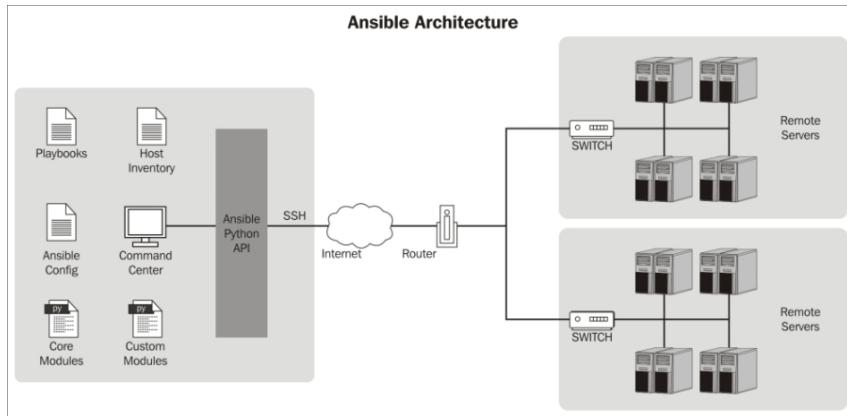


Figura 8: Arquitectura Ansible. [22]

Ansible funciona sin agentes, lo cual ayuda a no consumir recursos en los equipos gestionados. El usuario encargado de la administración de los dispositivos, puede generar unas credenciales y subirlas a los dispositivos para así tener un control total y una mayor seguridad a la hora de gestionar los equipos.

A continuación, se va a explicar los distintos componentes que forman Ansible:

Host Inventory

Este archivo es uno de los dos necesarios para empezar a automatizar dispositivos de red. En este archivo se determina los equipos remotos o locales en los que se ejecutarán las tareas programadas por el administrador.

Así sería un ejemplo básico de un archivo de inventario:

```
10.1.100.10
10.5.10.10
router-norte
```

Listado 1: Ejemplo inventory file

Esto es lo que contendría el archivo y es muy simple. Solamente tiene tres líneas, como se puede observar. Además, puede contener tanto direcciones IP como nombres de hosts (siempre y cuando esos nombres de hosts sean conocidos y definidos en la máquina). El inventario de máquinas es muy flexible y se pueden añadir más estructura y datos a medida que aumentan las tareas con Ansible. También cabe la posibilidad de crear varios inventarios si se trabaja en una red muy grande o dinámica. Conforme se vaya avanzando en el proyecto, se verá como crear archivos de inventario más complejos.

Ansible Config

Este archivo de configuración de Ansible utiliza un formato INI para almacenar todos los datos relacionados con la configuración. Este archivo tiene la gran ventaja de poder sobreescribir el que viene por defecto con la instalación y crear uno nuevo según las necesidades del administrador que vaya a realizar la automatización. Cuando se ejecuta Ansible, lo primero que busca es este archivo y siguiendo un orden:

- 1º *ANSIBLE_CONFIG*: Lo primero que mira nuestra herramienta de automatización es la variable de entorno que apunta al fichero de configuración.
- 2º *./ansible.cfg*: En segundo lugar, comprobaría si existe el fichero en el directorio desde donde se está ejecutando Ansible. En este TFG se va a utilizar esta opción.
- 3º *root/.ansible.cfg*: El siguiente directorio, donde podría localizarse el archivo, sería en el directorio raíz del usuario.
- 4º */etc/ansible/ansible.cfg*: Por último, comprobaría si el archivo se encontraría en el directorio donde se instala Ansible si se instalaría con un gestor de paquetes.

Cabe destacar que si la instalación se realiza a través del repositorio de GitHub, se puede encontrar dicho archivo de configuración en el mismo directorio donde se clonó el repositorio.

El fichero de configuración “ansible.cfg” tiene una gran variedad de parámetros para configurar, pero en la mayoría de los casos no hay necesidad de inicializarlos todos, por lo tanto, se van a mostrar los más importantes y necesarios:

- *inventory*: Este parámetro indica la dirección del archivo de inventario, del que se ha hablado anteriormente.
Ejemplo: inventory = /root/Desktop/Training/nombre_archivo
- *forks*: Mediante este parámetro, se puede modificar el número de procesos en paralelo que se generen.
Ejemplo: forks = 5
- *sudo_user*: Para poder usar ciertos comandos en la máquina remota o incluso local hay que especificar un usuario por defecto, esto se puede hacer inicializando este parámetro.
Ejemplo: sudo_user = root
- *host_key_checking*: Si se desea que Ansible no haga comprobaciones de la clave SSH se debe establecer como FALSE este parámetro, ya que por defecto está como TRUE.
Ejemplo: host_key_checking = FALSE
- *timeout*: Este es el valor predeterminado para el tiempo de espera de los intentos de conexión SSH y se mide en segundo.
Ejemplo: timeout = 60

Playbook

Estos ficheros llamados Playbook son unos de los más importantes de Ansible, ya que le indica que tareas debe ejecutar. Tienen un formato muy amigable y muy fácil de interpretar, son simples archivos YAML con una lista de tareas dentro y cada tarea está enlazada a un fragmento de código. Los fragmentos de código no están escritos en los Playbooks, sino que están agrupados en módulos, que son con los que interactúan los Playbooks y pueden ser escritos en cualquier lenguaje de programación pero con la condición de que la salida tenga un formato JSON. A continuación, se puede ver un ejemplo de un Playbook con distintas tareas, donde en la primera tarea se crearían dos Vlan y en la segunda tarea se mostraría un resumen de las Vlan:

```
#Create Vlan 21,23#
---
- name: Create Vlan 21 and 23
hosts: sw-oeste
connection: local

tasks:
- name: create VLAN
ios_config:
authorize: yes
auth_pass: ansible
lines:
- name {{ item.name }}
parents:
- vlan "{{ item.id }}"

with_items:
- { id: "{{ id_vlan21 }}", name: "{{ name_vlan21 }}" }
- { id: "{{ id_vlan23 }}", name: "{{ name_vlan23 }}" }

- name: show vlan
ios_command:
authorize: yes
commands:
- show vlan brief
register: sh_vlan

- debug: var=sh_vlan.stdout_lines
```

Listado 2: Ejemplo Playbook

Dentro de los Playbooks como se ha mostrado en el código anterior, puede haber múltiples tareas, que Ansible ejecutaría de forma secuencial. Por otro lado, la salida de la ejecución de estos Playbooks puede servir como variables para otras tareas, por lo tanto, se puede ver que todo es muy reutilizable y muy fácil de manejar. En los próximos capítulos se mostrarán ejemplos para que se pueda comprender mejor el funcionamiento de los Playbooks.

En la Figura 9, se muestra la estructura completa de Ansible en forma de árbol:

```
[eduroamnat56-180:Ejemplo Cretu]$ tree
.
├── README.md
├── ansible.cfg
├── group_vars
│   ├── router
│   │   └── main.yml
│   └── switch
│       └── main.yml
├── hosts
└── roles
    ├── Backup
    │   ├── README.md
    │   ├── defaults
    │   │   └── main.yml
    │   ├── handlers
    │   │   └── main.yml
    │   ├── meta
    │   │   └── main.yml
    │   ├── tasks
    │   │   ├── show_run.retry
    │   │   └── show_run.yml
    │   ├── tests
    │   │   ├── inventory
    │   │   └── test.yml
    │   └── vars
    │       └── main.yml
    ├── config_r_este
    │   ├── README.md
    │   ├── defaults
    │   │   └── main.yml
    │   ├── handlers
    │   │   └── main.yml
    │   ├── meta
    │   │   └── main.yml
    │   ├── tasks
    │   │   └── main.yml
    │   ├── tests
    │   │   ├── inventory
    │   │   └── test.yml
    │   └── vars
    │       └── main.yml
└── site.yml
```

Figura 9: Ejemplo de los directorios de Ansible

Viendo la imagen anterior, se puede identificar los Playbooks como los archivos con extensión .yml que están dentro de “*tasks*”.

A continuación, se van a explicar los dos últimos componentes que forman esta estructura de Ansible. No son necesarios para su funcionamiento, pero sí que son muy útiles para la organización de la automatización.

Variables

Las variables, como en todos los lenguajes de programación, sirven para declarar o almacenar valores que puedan ser usados más adelante en los Playbooks. Como se ha comentado anteriormente, las salidas de las tareas pueden ser usadas como variables.

Primero hay que crear y nombrar las variables correctamente y para ello hay que seguir unas pautas. Todos los nombres de variables tienen que empezar con una letra y después se pueden añadir números, letras y guiones bajos si se desea.

Ansible, a la hora de buscar variables realiza un escaneo de la parte más baja del árbol de directorios hasta la más alta.

Ejemplos de variables válidas:

- nombre_servicio
- servidor2
- Protocolo
- usuario_remoto2

Ejemplos de variables no válidas:

- nombre servicio
- 2servidor
- Protocolo.salida
- usuario-remoto2

Lo siguiente es tener claro donde ubicar las variables dependiendo del uso o de la complejidad de la red que se pretenda automatizar.

1º *Incluidas desde fichero* : A la hora de incluir las variables de un fichero, se debe tener en cuenta que esto anulará cualquier otra variable definida en diferente nivel jerárquico, excepto las definidas en la línea de comando cuando se ejecuta el Playbook. Esta modalidad es una de las más usadas ya que se dispone de un fichero externo al Playbook donde se pueden modificar las variables sin necesidad de modificar la tarea.

Un ejemplo sería tener un fichero llamado “router.yml” que contenga lo siguiente:

```
---
# vars file for config_r_norte

## Ip for Serial 6/0 ##
desc_s6_0: Connection to r-oeste
id_s6_0: Serial 6/0
ip_r_norte_s6_0: 172.16.1.2
mask_r_norte_s6_0: 255.255.255.252
```

Listado 3: Ejemplo fichero de variables

Y para poder utilizar este fichero con variables, en el Playbook se debe incluir con el siguiente parámetro:

```
---
- name: Enable Serial 6/0
hosts: r-norte
connection: local

tasks:
- include: /vars/router.yml    ## Parametro necesario
- name: Enable Serial 6/0
...
...
```

Listado 4: Ejemplo incluir variables

2º *Dentro del Playbook* : Las variables definidas dentro de los Playbooks sobrescriben otras variables establecidas en otro fichero. Se crean igual que en los ficheros externos, definiendo un nombre para la variable y su valor.

Un ejemplo sería el siguiente:

```
---
- name: Enable Serial 6/0
hosts: r-norte
connection: local
vars:
- desc_interfaz: "Connection to r-oeste"
- interfaz_serial6_0: "S6/0"
tasks:
- name: Enable Serial 6/0
...
...
```

Listado 5: Ejemplo incluir variables

3º *Group_vars* : Otra opción a la hora de configurar las variables es crearlas en un archivo global. Con esto se consigue reutilizar ciertas variables para no tenerlas repetidas en el directorio. El formato de este fichero es el mismo que en los anteriores que se han explicado. Mediante la opción *vars_files*: dentro del Playbook se puede especificar el fichero con las variables.

4º *La salida de un tarea* : Ansible dispone de una opción llamada *gather_facts* con la cual recoge datos sobre la máquina remota y los almacena en variables con nombres ya registrados por el propio Ansible

Las siguientes líneas muestran un ejemplo de cómo obtener datos sobre una máquina remota:

```
$ ansible 192.168.33.10 -i inventory -m setup
192.168.33.10 | success >> {
"ansible_facts": {
"ansible_all_ipv4_addresses": [
"10.0.2.15",
"192.168.33.10"
],
"ansible_all_ipv6_addresses": [
"fe80::a00:27ff:fed9:399e",
"fe80::a00:27ff:fe72:5c55"
],
"ansible_architecture": "x86_64",
-----
"ansible_distribution_major_version": "6",
"ansible_distribution_release": "Final",
"ansible_distribution_version": "6.4",
```

```

"ansible_domain": "localdomain"
-----
"ansible_swapfree_mb": 2559,
"ansible_swaptotal_mb": 2559,
"ansible_system": "Linux",
"ansible_system_vendor": "innotek GmbH",
"ansible_user_id": "vagrant",
"ansible_userspace_architecture": "x86_64",
"ansible_userspace_bits": "64",

```

Listado 6: Ejemplo salida de datos

- 5º *Variables en la línea de comandos* : Pasando una variable por la línea de comandos es una manera de facilitar a un usuario la posibilidad de añadir o modificar un valor de una variable. Usando esta opción se sobrescriben todas las variables que hayan sido definidas anteriormente.

Este modo de definición de variables se suele usar mucho a la hora de definir los hosts a los que van dirigidas las tareas. A continuación, se puede ver un ejemplo de su uso.

Este sería el comando para ejecutar Ansible:

```

ansible-playbook -i hosts playbooks/interface.yml --
extra-vars "host_name=r-norte"

```

Listado 7: Código ejecución playbook

Y este fragmento de Playbook serviría para ver donde se sustituirá la variable.

```

---
- name: Enable Serial 6/0
hosts: {{ host_name }}
connection: local

tasks:
- name: Enable Serial 6/0
...
...

```

Listado 8: Ejemplo colocación de variables

- 6º *Variables en el fichero de inventario* : Durante la automatización de una red, cabe la posibilidad que en algún momento se necesite crear ciertas variables para un host específico. Por lo tanto, para definir una variable de este modo se podría añadir a la hora de crear las máquinas en el fichero de inventario.

Un ejemplo sería el siguiente:

```

10.1.100.10
10.5.10.10

```

```
router-norte int_name=S6/0 desc_int=Connectio to r-
oeste
```

Listado 9: Ejemplo fichero de variables

Esta manera de declarar variables no es muy usada, de modo que en este TFG tampoco se va a usar.

Roles

Para finalizar con la estructura de Ansible, se va a explicar el último componente, los roles.

Las redes que proporcionan servicios a las empresas tienen un tamaño bastante grande y para poder llevar un correcto mantenimiento lo más importante es la organización. A la hora de automatizar las redes, los roles tienen un papel muy importante ya que permite organizar y estandarizar además de poder reutilizar dichos roles. Esto es posible gracias a que Ansible tiene una gran comunidad detrás que con la ayuda de un comando que se verá a continuación, cualquier persona será capaz de crear un rol con el nombre deseado. El comando a usar es el siguiente:

```
ansible-galaxy init role_name
```

Listado 10: Código para crear un rol

Una vez ejecutado el comando, en el directorio se crea una carpeta con el rol indicado. La estructura de dicho rol se puede observar en la Figura 10

```
eduroamnat56-180:Backup Cretu$ tree
.
├── README.md
├── defaults
│   └── main.yml
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── tasks
│   ├── show_run.retry
│   └── show_run.yml
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Figura 10: Ejemplo de la estructura de un rol

A continuación, se va a explicar el funcionamiento de cada subcarpeta que hay dentro del rol:

- *README.md*: Este documento suele estar en todos los proyectos ya que es donde se debe explicar detalladamente el funcionamiento del proyecto para que cualquier persona sepa cómo utilizar dicho código.
- *defaults*: Este directorio contiene archivos con variables por defecto para el rol.

- *handlers*: Estos ficheros se ejecutan cuando hay una notificación en el Playbook. Mediante el comando “`notify`”, Ansible ejecuta los handlers cuando se ha producido un cambio en el sistema. Un ejemplo sería que cuando se modifica el fichero de configuración de apache, siempre es conveniente reiniciar dicho servicio, por lo tanto, para no crear una nueva tarea dentro del Playbook se usan estos handlers.
- *meta*: En estos ficheros es donde se debe proporcionar una descripción acerca de las plataformas soportadas por tú proyecto, la versión mínima requerida para Ansible, dependencias y etiquetas.
- *tasks*: En esta subcarpeta es donde se crearán los Playbooks necesario para el rol.
- *tests*: Son archivos idénticos a los Playbooks pero que sirven para realizar pruebas antes de ejecutar las tareas sobre la máquina remota. Probar los roles es una muy buena práctica que todo el mundo debe realizar.
- *vars*: Esta es otra subcarpeta donde poder almacenar variables para el rol.

Una vez alcanzado el final de la estructura de Ansible se va a mostrar las dos formas de ejecutar esta herramienta.

Antes de ejecutar Ansible, como se ha mencionado al inicio de este apartado se necesita tener creado el fichero `ansible.cfg` y el inventario con los dispositivos remotos.

ansible-playbook

Está modalidad se ha explicado anteriormente, pero en esta sección se detallará en profundidad. Aunque dispone de una enorme variedad de opciones para añadir se van a exponer los más importantes y los más usados.

- *-k*: Esta opción hace que la máquina remota pida la contraseña del usuario que va a ejecutar las tareas. Es útil para asegurar que estamos realizando los cambios con el usuario correcto.
- *-u*: Se puede especificar el usuario con el que acceder al dispositivo a automatizar.
- *-c*: Esta opción permite especificar el tipo de conexión a realizar con las máquinas.
- *-e*: Si un administrador de red necesita pasar un valor a una variable por la línea de comandos mediante esta opción puede añadir dicho valor a una variable.
- *-f*: Hay situaciones en las que se desea que no se ejecuten todos los procesos en paralelo, por lo tanto, mediante esta opción se consigue indicar cuantos procesos en paralelo se desea ejecutar.
- *-i*: Si la persona encargada de la automatización ha decidido modificar el inventario de máquinas remotas con esta opción podemos especificar la ruta y el nombre del fichero nuevo.
- *-v*: Esta opción se utilizar para obtener una salida detallada de la ejecución, es muy útil para ver fallos en los procesos.

Un ejemplo de cómo sería una ejecución utilizando las opciones mencionadas:

```
ansible-playbook -i hosts playbook.yml -c local -v -f 3 -e host=router -k -u ansible
```

Listado 11: Ejecución de Playbook

comandos Ad-Hoc

Este tipo de comando son útiles para realizar pruebas rápidas sobre máquinas remotas y que no se pretende guardar después. Un uso muy común podría ser por ejemplo apagar toda una red de un laboratorio por un periodo vacacional. Se podría escribir una simple línea que haga lo deseado sin necesidad de crear un Playbook para ello, por ejemplo

```
ansible laboratorio -a "/sbin/reboot"
```

Listado 12: Ejemplo código Ad-Hoc

En los comandos Ad-Hoc también hay opciones que se pueden añadir al comando. Los más importantes son:

- **-m**: Mediante esta opción se indica qué módulo va a ser usado por el comando.
- **-a**: Ciertos módulos necesitan un argumento para poder funcionar y con esta opción se lo añadimos.
- **-B**: Si se desea modificar el tiempo de ejecución mediante esta opción se puede conseguir. Las unidades de tiempo que tiene esta opción son segundos.

También se pueden usar opciones de las mencionadas en el apartado 6.1.1.1. A continuación, se verá un ejemplo de cómo copiar un archivo de local a una máquina remota y de cómo instalar un paquete:

```
ansible -m copy -a "src=/etc/hosts dest=/tmp/hosts"  
ansible -m yum -a "name=git state=present"
```

Listado 13: Ejemplo códigos Ad-Hoc

6.1.1.2 Instalación Ansible

Como se ha visto en el apartado de ventajas, Ansible no necesita un agente en las máquinas remotas. En consecuencia, para instalar Ansible no hay necesidad de crear una base de datos y no se necesita arrancar el servicio. Basta con instalarlo en una máquina, que podría ser un portátil, y desde este punto central gestionar toda la infraestructura remota. Dado que no instala ningún software ni deja nada ejecutando en los dispositivos remotos, no va a haber ningún problema a la hora de actualizar la versión de Ansible. Un requisito muy importante para poder instalar Ansible es tener una versión de Python superior a 2.6 o 2.7. Otro dato importante es que Windows no puede ser la máquina que gestione la automatización. Para finalizar, se explicará como instalar Ansible en un dispositivo MacOs, Ubuntu, Debian y CentOS: [23]

MacOs

El modo preferido para instalar Ansible en MacOs es mediando *pip*. *Pip* es un gestor de paquetes de Python.

```
sudo easy_install pip ## Instalar pip  
sudo pip install ansible ## Instalar Ansible
```

Listado 14: Ejemplo instalación Ansible en MacOs

Si en algún momento hay necesidad de instalar la última versión de desarrollo se realizaría de la siguiente manera:

```
pip install git+https://github.com/ansible/ansible.git@devel
```

Listado 15: Ejemplo instalación Ansible en MacOs

Ubuntu

Para instalar Ansible en Ubuntu se necesita añadir un repositorio y tras ello instalar la herramienta y esto se consigue de esta forma:

```
sudo apt-get update  
sudo apt-get install software-properties-common  
sudo apt-add-repository ppa:ansible/ansible  
sudo apt-get update  
sudo apt-get install ansible
```

Listado 16: Ejemplo instalación Ansible en Ubuntu

Debian

Los usuarios de Debian podrían utilizar el mismo repositorio de Ubuntu y para ello lo primero es introducir la siguiente línea en el archivo */etc/apt/sources.list*:

```
deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main
```

Listado 17: Repositorio para Debian

A continuación en la CLI ejecutaríamos:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93  
C4A3FD7BB9C367  
sudo apt-get update  
sudo apt-get install ansible
```

Listado 18: Ejemplo instalación Ansible en Debian

CentOS

```
| sudo yum install ansible
```

Listado 19: Ejemplo instalación Ansible en CentOS

6.1.2. Ansible Tower

Está es la distribución comercial recomendada para el contexto empresarial. Dispone de una interfaz web que permite centralizar y controlar las infraestructuras. Incluye un panel de control, planificación de trabajos y gestión de inventarios, facilitando y permitiendo a desarrolladores crear pruebas y aplicar los roles.

En la Figura 11 se puede observar un ejemplo del panel de control de Ansible Tower:

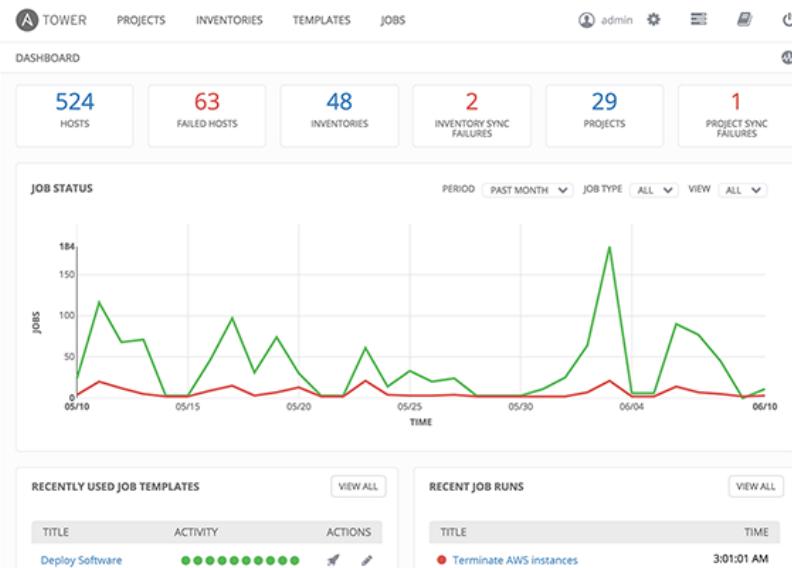


Figura 11: Panel de control Ansible Tower

El panel de control es muy útil por el hecho de que se puede seguir en tiempo real la ejecución de los Playbooks. Esto facilita el seguimiento de las tareas y permite ver las máquinas que han fallado o han tenido éxito. También permite ejecutar distintos Playbooks con inventarios independientes, usuarios diferentes o incluso si utilizan distintas credenciales. Ansible Tower monitoriza y registra toda la actividad de forma segura. Se puede saber quién, cómo y dónde se ejecutaron los Playbooks.

“Ansible Tower allows us to easily streamline the delivery of applications and services to both OpenStack and Amazon Clouds in a cost effective, simple, and secure manner.” [4]



Dispone de una opción para poder programar las tareas a ejecutar y mediante las notificaciones, que se pueden integrar en la plataforma, se puede saber en cualquier momento que está sucediendo con la automatización de la red.

Como se ha explicado, Ansible Tower es una herramienta comercial con un potencial enorme. Grandes empresas como Juniper o NASA pueden certificar que su uso es muy productivo y además dispone de una Application Programming Interface REpresentational State Transfer (API REST) para aquellas personas que necesiten personalizar las funciones de Tower e integrarlas en paneles de control personalizados.

6.1.3. AWX

Una opción gratuita a Tower es AWX. AWX nos proporciona una interfaz gráfica basada en web, API REST y con funcionalidades construidas sobre Ansible. Su funcionamiento es similar al de Ansible Tower solo que este se puede instalar gratuitamente siguiendo los pasos que se pueden encontrar en Github:

```
https://github.com/ansible/awx
```

Listado 20: Web con información sobre AWX

Esta herramienta es OpenSource por lo tanto hay un gran comunidad detrás resolviendo siempre dudas y problemas que le surge a la gente. Durante el TFG se realizarán pruebas con Ansible a través de CLI y también mediante AWX. En la Figura 12 se puede apreciar su parecido con Tower.

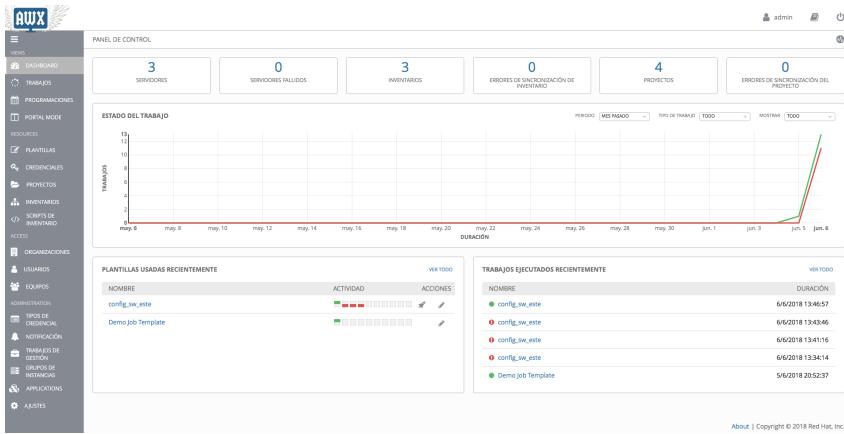


Figura 12: Panel de control AWX

6.1.4. GNS3

GNS3 permite personalizar los espacios de trabajo, crear topologías según las necesidades y hasta llegar al límite del rendimiento de la máquina. Para concretar, GNS3 es una plataforma para experimentos, pruebas, demostraciones y tecnologías de red. Es un software multiplataforma y tiene soporte para Windows, Linux y MacOS. Dispone de una interfaz gráfica muy sencilla de utilizar como se puede apreciar en la Figura 13 e incorpora Dynamips. Dynamips es un emulador de software para routers CISCO, que son los que se van a utilizar en este TFG. [24]

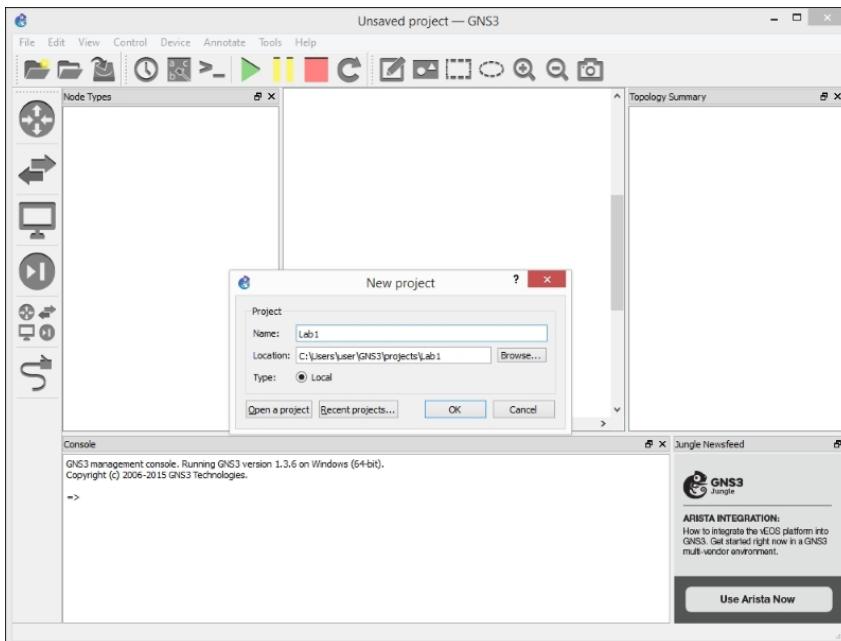


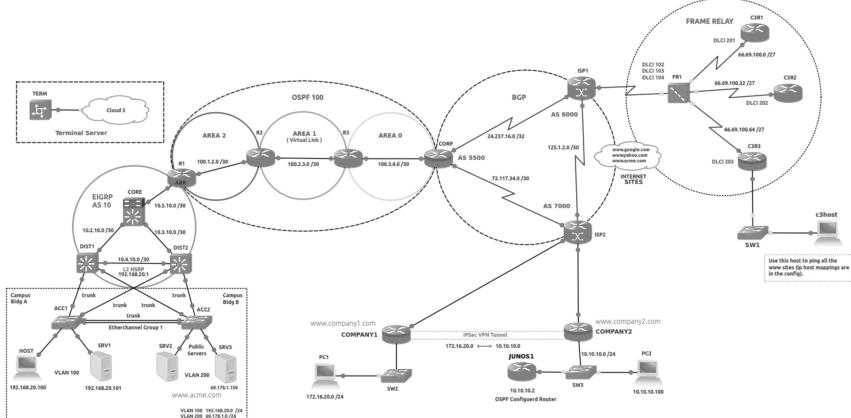
Figura 13: Interfaz gráfica GNS3

Por lo tanto, mediante Dynampis nuestro software es capaz de realizar las siguientes funciones:

- *Emular Hardware:* La interfaz gráfica proporciona un escenario donde se pueden crear laboratorios con redes virtuales juntando switches, routers y hosts. GNS3 no limita el uso de estos dispositivos, lo cual es una gran ventaja y debido a que solo emula el hardware, se puede ejecutar un archivo de un sistema operativo de IOS real, pudiendo así utilizar todas las configuraciones, protocolos y salidas que proporciona un dispositivo real de CISCO. Esta funcionalidad es la que distingue GNS3 de otros simuladores.
- *Emular sistemas operativos:* Además de emular el hardware, GNS3 integra sistemas operativos simuladores que pueden conectarse entre ellos. Puede también integrar máquinas virtuales mediante VirtualBox que ejecutan sistemas operativos como por ejemplo Linux o Windows y realizar las pruebas necesarias sobre esos sistemas, pero siempre permaneciendo en el entorno de GNS3.
- *Escalabilidad con el servidor GNS3:* En la instalación de GNS3, el instalador crea un servidor especial que solo GNS3 puede utilizar permitiéndole iniciar, detener y controlar los dispositivos. Esta funcionalidad tiene una gran ventaja si se trabaja con topologías grandes, ya que permite ejecutar el servidor GNS3 en un host con más memoria y potencia de procesamiento y en un host diferente la interfaz gráfica.

Otro emulador soportado por GNS3 es Qemu que proporciona emulación de dispositivos Cisco ASA, routers Juniper y Vyatta.

En la Figura 14 se puede observar la complejidad de una red a la que se puede llegar mediante este software.



- Dynamips es incapaz de emular el hardware de los circuitos integrados específicos de los switches avanzados de Catalyst de Cisco (ASIC).
- Debido a que Dynamips es un emulador que no proporciona ninguna aceleración de hardware, el rendimiento de los equipos está limitado entre 1.5Mb/s hasta 800Mb/s. Esta limitación se creó pensando en que la gente podría utilizar este software para virtualizar los dispositivos de CISCO y añadirlos en entornos de producción.

6.1.4.1 Instalación GNS3

En GNS3 se pueden crear topologías de red bastante complejas como se puede ver en la Figura 14 y para ello se necesita una máquina bastante potente, pero para hacer pruebas sencillas y practicar es suficiente con los siguientes requisitos de máquina:

- Procesador de 1,5 GHz
- 4GB RAM
- 1GB de espacio en el disco duro

Una vez identificados los requisitos mínimos del sistema para que pueda funcionar GNS3 se va a detallar los pasos que hay que seguir para instalar la herramienta en Windows, MacOS y Ubuntu.

Windows

Para instalar GNS3 en Windows [5] se deben seguir los siguientes pasos:

- Descargar desde la página oficial (<http://www.gns3.com>) el instalar y ejecutarlo para iniciar la instalación.
- Una vez abierto el instalador es muy sencillo ya que solo hay que seguir las instrucciones que se muestran en la pantalla, pudiendo elegir distintos componentes adicionales y en qué directorio instalarlo como se puede ver en la Figura 15 16.

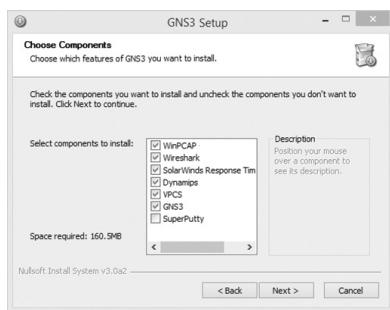


Figura 15: Componentes adicionales

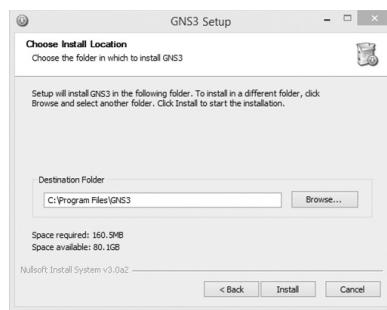


Figura 16: Directorio de instalacion.

- Tras realizar los pasos anteriores ya debería de estar instalado GNS3 y se puede empezar a utilizar.

MacOS

La instalación de GNS3 en MacOS es bastante sencilla. Lo único que se debe hacer es descargar como en el caso de Windows, el instalador para MacOS desde la página oficial y copiar el archivo que hay dentro de ese instalador en la carpeta de aplicaciones. En la Figura 17 se puede ver un ejemplo:

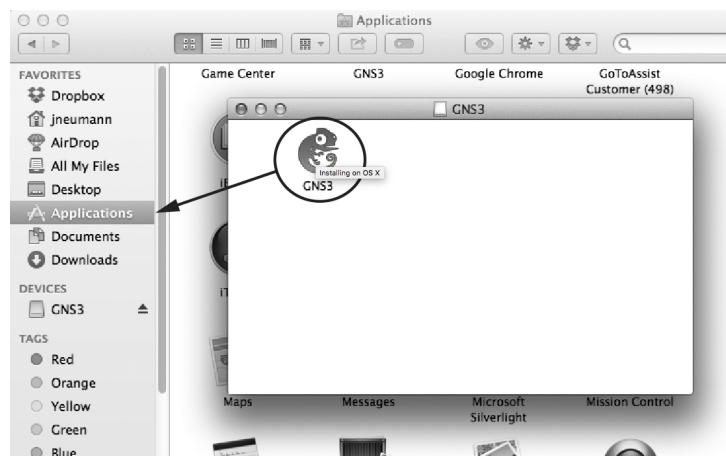


Figura 17: Instalacion GNS3 en MacOS

Ubuntu

GNS3 funciona en muchas distribuciones de Linux, pero en esta sección se explicará para Ubuntu ya que es la que más personas utiliza. Existen dos formas para instalar GNS3 en Ubuntu. La primera es mediante un paquete a través del gestor de paquetes y la segunda es mediante código fuente. Puesto que la instalación mediante código fuente es más compleja y más personalizable en este apartado se explicará solamente la instalación mediante paquete, que además es muy sencilla. Solamente se debe ejecutar el siguiente comando:

```
sudo apt-get install gns3
```

Listado 21: Ejemplo instalación GNS3 en Ubuntu

Para más información acerca de cómo instalar GNS3 en distintas distribuciones de Linux o mediante código fuente se puede consultar en el siguiente enlace:

```
https://docs.gns3.com/
```

Listado 22: Página web oficial GNS3

6.1.5. VMWare Workstation

VMware Workstation es un sistema de virtualización por software. Concretamente, es un programa que simula un sistema físico (un computador) con unas características de hardware determinadas. Una vez ejecutado el programa, proporciona las mismas características en todos los aspectos a un computador físico (excepto en el puro acceso físico al hardware simulado), con CPU (puede ser más de una), BIOS, tarjeta gráfica, memoria RAM, tarjeta de red, sistema de sonido, conexión USB, disco duro (pueden ser más de uno).

Un virtualizador por software permite ejecutar (simular) varios computadores (sistemas operativos) dentro de un mismo hardware de manera simultánea, permitiendo así el mayor aprovechamiento de recursos, pero el rendimiento del sistema virtual estará limitado por las características del sistema físico en el que se ejecuta VMware Workstation.

Este software tiene soporte para máquinas con Windows, Linux y también para MacOS, pero en este último se llama VMware Fusion y solamente funciona en MacOS basado en procesadores INTEL.

Se ha elegido este software para realizar el TFG, principalmente porque es compatible con las demás herramientas y también porque es uno de los software más avanzados en el campo de la virtualización. [10]

6.1.6. Docker

Docker es un proyecto de la comunidad de OpenSource que utiliza una tecnología de creación de contenedores permitiendo la creación y el uso de contenedores de Linux.

La comunidad OpenSource de Docker trabaja constantemente para mejorar estas tecnologías de forma gratuita para beneficiar a todos los usuarios.

También existe la empresa Docker Inc., donde se desarrolla todo el trabajo hecho por la comunidad de Docker con el fin de hacerlo más seguro y poder así compartirlo a un mayor número de usuarios.

Con Docker, se consigue que las máquinas virtuales sean más ligeras y modulares, además de obtener una flexibilidad gracias a los contenedores, ya que se pueden crear, implementar, copiar y mover de un entorno a otro.

La tecnología Docker usa el Kernel de Linux y funciones de este para separar procesos y que puedan ejecutarse de manera independiente. El objetivo de los contenedores es conseguir esta independencia, permitiendo ejecutar varios procesos separados mejorando así el uso de la infraestructura y la seguridad de la misma. [25]

Las ventajas de utilizar Docker son:

- Modularidad
- Capas y control de versión de imagen
- Restauración
- Implementación rápida

El presente TFG cumple con las fases mencionadas al comienzo de esta sección y a continuación se van a detallar.

6.2. Fase de análisis

Para que el proyecto final tenga una calidad óptima se deben cumplir los requisitos y objetivos propuestos en los capítulos anteriores. Para ello es importante esta fase de análisis, que ayuda a tener una buena estructura de lo que se desea desarrollar ya que si no podría afectar negativamente en el desarrollo.

Para mostrar los requisitos definidos previamente, se realizará mediante la ayuda del lenguaje unificado de modelado (UML) [20] Con la ayuda de este lenguaje se podrá definir el funcionamiento del proyecto, representando su funcionalidad mediante diagramas para facilitar su comprensión y así también se asegurará que cumple con los requisitos.

6.2.1. Casos de Uso

Con los casos de uso se obtiene una descripción de las actividades que deberán realizar los usuarios del sistema. En la Figura 18 se pueden ver los dos actores que hay con sus correspondientes casos de uso. Este proyecto está formado por dos actores:

- **Administrador:** El actor, administrador, tiene los máximos privilegios ya que es el encargado de controlar toda la red. Tiene la opción más importante que es la de crear los Playbooks y detener el funcionamiento de la red. Como el administrador es un actor que hereda de técnico, tiene permisos para poder realizar todas las actividades del técnico como son las de lanzar los Playbooks con Ansible desde CLI o desde AWX e iniciar el funcionamiento de la red.
- **Técnico:** El otro actor del sistema es el técnico que no tiene los mismos permisos que el administrador. Solamente podrá ejecutar los Playbooks desde CLI o desde AWX y el otro caso de uso que tiene asignado es el de iniciar el funcionamiento de la red.

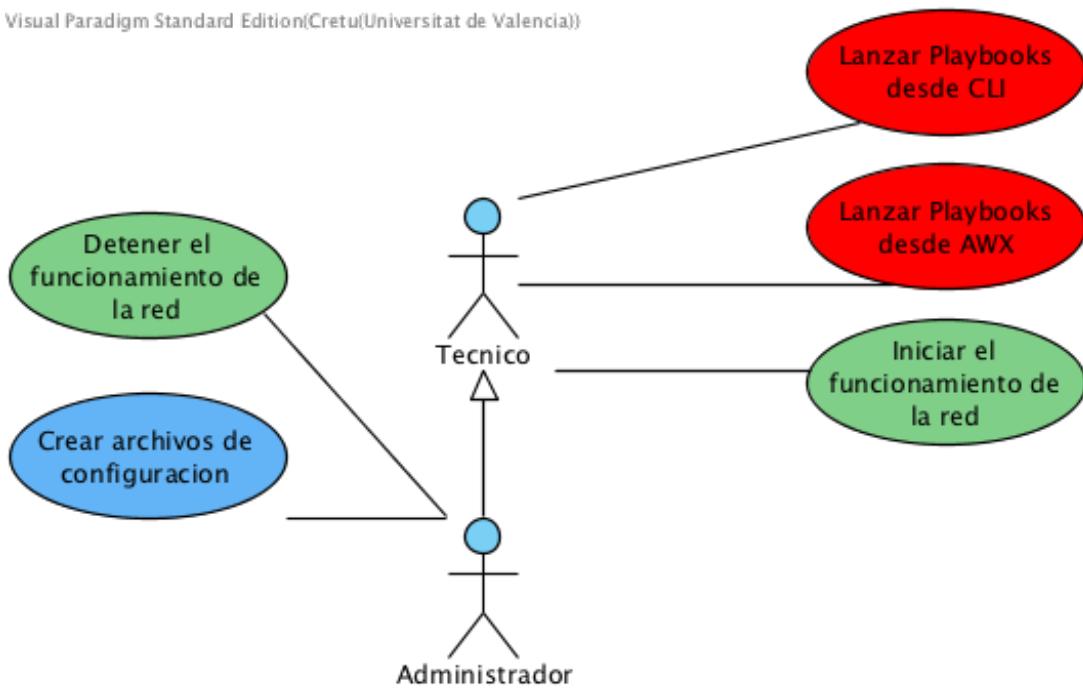


Figura 18: Casos de uso

Para poder distinguir la prioridad de las actividades se han coloreado con un patrón que se va a explicar a continuación:

- *Rojo*: El color rojo indica una prioridad alta. Las actividades que tienen este color son las que más se van a ejecutar.
- *Azul*: Este color indica una prioridad media.
- *Verde*: Este color indica una prioridad baja. El inicio y la detención del funcionamiento de la red no suele ser tan frecuente.

Después de definir los casos de uso de cada usuario se procederá a explicar de manera precisa cual es la intencionalidad y funcionamiento de cada caso de uso en concreto.

6.2.2. Formatos expandidos

En las Figuras 19, 20, 21, 22, 23 se muestran las especificaciones de cada caso de uso detalladamente.

El caso de uso *Lanzar Playbook desde CLI* comenzaría cuando el usuario técnico actualizaría o introduciría una nueva configuración en los dispositivos de la red.

| Use Case ID | 1 | | | | | | | | | | | | |
|---------------------------|---|--|--------------------|------------------------|---|--|--|---|--|----------------------------------|---|--|--|
| Super Use Case | | | | | | | | | | | | | |
| Primary Actor | Tecnico | | | | | | | | | | | | |
| Secondary Actor(s) | | | | | | | | | | | | | |
| Brief Description | El tecnico comienza la actualizacion o introduce una nueva configuracion en algun dispositivo de la red | | | | | | | | | | | | |
| Preconditions | Tienen que estar los Playbooks creados | | | | | | | | | | | | |
| Flow of Events | <table border="1"> <thead> <tr> <th></th> <th>Actor Input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>El flujo comienza cuando el tecnico quiere configurar un dispositivo</td> <td></td> </tr> <tr> <td>2</td> <td>El tecnico decide que Playbook lanzar</td> <td></td> </tr> <tr> <td>3</td> <td></td> <td>El dispositivo devuelve un mensaje de confirmacion</td> </tr> </tbody> </table> | | Actor Input | System Response | 1 | El flujo comienza cuando el tecnico quiere configurar un dispositivo | | 2 | El tecnico decide que Playbook lanzar | | 3 | | El dispositivo devuelve un mensaje de confirmacion |
| | Actor Input | System Response | | | | | | | | | | | |
| 1 | El flujo comienza cuando el tecnico quiere configurar un dispositivo | | | | | | | | | | | | |
| 2 | El tecnico decide que Playbook lanzar | | | | | | | | | | | | |
| 3 | | El dispositivo devuelve un mensaje de confirmacion | | | | | | | | | | | |
| Post-conditions | El dispositivo se ha configurado correctamente | | | | | | | | | | | | |
| Playbook erroneo | <table border="1"> <thead> <tr> <th></th> <th>Actor Input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>El flujo comienza en el punto 3</td> <td></td> </tr> <tr> <td>2</td> <td></td> <td>El dispositivo devuelve un error</td> </tr> </tbody> </table> | | Actor Input | System Response | 1 | El flujo comienza en el punto 3 | | 2 | | El dispositivo devuelve un error | | | |
| | Actor Input | System Response | | | | | | | | | | | |
| 1 | El flujo comienza en el punto 3 | | | | | | | | | | | | |
| 2 | | El dispositivo devuelve un error | | | | | | | | | | | |
| Conexion erronea | <table border="1"> <thead> <tr> <th></th> <th>Actor Input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>El flujo comienza en el punto 2</td> <td></td> </tr> <tr> <td>2</td> <td>El tecnico no puede lanzar el Playbook</td> <td></td> </tr> </tbody> </table> | | Actor Input | System Response | 1 | El flujo comienza en el punto 2 | | 2 | El tecnico no puede lanzar el Playbook | | | | |
| | Actor Input | System Response | | | | | | | | | | | |
| 1 | El flujo comienza en el punto 2 | | | | | | | | | | | | |
| 2 | El tecnico no puede lanzar el Playbook | | | | | | | | | | | | |
| Author | Cretu | | | | | | | | | | | | |
| Post-conditions | Los dispositivos se han configurado. | | | | | | | | | | | | |

Figura 19: Casos de uso - Lanzar Playbook desde CLI

El caso de uso *Lanzar Playbook desde AWX* comenzaría cuando el usuario técnico actualizaría o introduciría una nueva configuración en los dispositivos de la red mediante AWX.

| Use Case ID | 2 | | | | | | | | | | | | |
|---------------------------|--|---|--------------------|------------------------|---|--|--|---|---|---------------------|---|--|---|
| Super Use Case | | | | | | | | | | | | | |
| Primary Actor | Tecnico | | | | | | | | | | | | |
| Secondary Actor(s) | | | | | | | | | | | | | |
| Brief Description | El tecnico comienza la actualizacion o introduce una nueva configuracion en algun dispositivo de la red mediante AWX. | | | | | | | | | | | | |
| Preconditions | Tienen que estar los Playbooks creados. | | | | | | | | | | | | |
| Flow of Events | <table border="1"> <thead> <tr> <th></th> <th>Actor Input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>El flujo comienza cuando el tecnico quiere configurar un dispositivo a traves de AWX</td> <td></td> </tr> <tr> <td>2</td> <td>El tecnico debe ejecutar AWX y elegir los Playbooks</td> <td></td> </tr> <tr> <td>3</td> <td></td> <td>AWX devuelve la respuesta de los dispositivos</td> </tr> </tbody> </table> | | Actor Input | System Response | 1 | El flujo comienza cuando el tecnico quiere configurar un dispositivo a traves de AWX | | 2 | El tecnico debe ejecutar AWX y elegir los Playbooks | | 3 | | AWX devuelve la respuesta de los dispositivos |
| | Actor Input | System Response | | | | | | | | | | | |
| 1 | El flujo comienza cuando el tecnico quiere configurar un dispositivo a traves de AWX | | | | | | | | | | | | |
| 2 | El tecnico debe ejecutar AWX y elegir los Playbooks | | | | | | | | | | | | |
| 3 | | AWX devuelve la respuesta de los dispositivos | | | | | | | | | | | |
| Docker no funciona | <table border="1"> <thead> <tr> <th></th> <th>Actor Input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>El flujo comienza en el punto 2</td> <td></td> </tr> <tr> <td>2</td> <td></td> <td>AWX no responde</td> </tr> </tbody> </table> | | Actor Input | System Response | 1 | El flujo comienza en el punto 2 | | 2 | | AWX no responde | | | |
| | Actor Input | System Response | | | | | | | | | | | |
| 1 | El flujo comienza en el punto 2 | | | | | | | | | | | | |
| 2 | | AWX no responde | | | | | | | | | | | |
| Conexion erronea | <table border="1"> <thead> <tr> <th></th> <th>Actor Input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>El flujo comienza en el punto 2</td> <td></td> </tr> <tr> <td>2</td> <td>El tecnico no puede lanzar los Playbooks</td> <td></td> </tr> </tbody> </table> | | Actor Input | System Response | 1 | El flujo comienza en el punto 2 | | 2 | El tecnico no puede lanzar los Playbooks | | | | |
| | Actor Input | System Response | | | | | | | | | | | |
| 1 | El flujo comienza en el punto 2 | | | | | | | | | | | | |
| 2 | El tecnico no puede lanzar los Playbooks | | | | | | | | | | | | |
| Playbook erroneo | <table border="1"> <thead> <tr> <th></th> <th>Actor Input</th> <th>System Response</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>El flujo comienza en el punto 3</td> <td></td> </tr> <tr> <td>2</td> <td></td> <td>AWX devuelve error.</td> </tr> </tbody> </table> | | Actor Input | System Response | 1 | El flujo comienza en el punto 3 | | 2 | | AWX devuelve error. | | | |
| | Actor Input | System Response | | | | | | | | | | | |
| 1 | El flujo comienza en el punto 3 | | | | | | | | | | | | |
| 2 | | AWX devuelve error. | | | | | | | | | | | |
| Author | Cretu | | | | | | | | | | | | |
| Post-conditions | Los dispositivos se han configurado. | | | | | | | | | | | | |

Figura 20: Casos de uso - Lanzar Playbook desde AWX

El caso de uso *Crear archivos de configuracion* comenzaría cuando el usuario administrador empezase a crear los archivos de configuración (Playbooks) para ser ejecutados cuando sea necesario.

| | | | |
|---------------------------|--|--|-----------------------------|
| Use Case ID | 3 | | |
| Super Use Case | | | |
| Primary Actor | Administrador | | |
| Secondary Actor(s) | | | |
| Brief Description | El administrador crea los archivos de configuracion para ser ejecutados cuando sea necesario | | |
| Preconditions | | | |
| Flow of Events | | Actor Input | System Response |
| | 1 | El flujo comienza cuando el administrador crea los archivos de configuracion | |
| Post-conditions | Los archivos se han creado correctamente | | |
| | | Actor Input | System Response |
| Error en alguna linea | 1 | El flujo comienza en el punto 2 | |
| | 2 | | El sistema alerta del error |
| Author | Cretu | | |

Figura 21: Casos de uso - Crear archivos de configuracion

El caso de uso *Iniciar funcionamiento de la red* comenzaría cuando el usuario técnico inicia el sistema para que los dispositivos se configuren o para que haya conectividad entre ellos.

| | | | |
|---------------------------|--|---|------------------------|
| Use Case ID | 4 | | |
| Super Use Case | | | |
| Primary Actor | Tecnico | | |
| Secondary Actor(s) | | | |
| Brief Description | El tecnico inicia el sistema para que haya conectividad entre los dispositivos | | |
| Preconditions | | | |
| Flow of Events | | Actor Input | System Response |
| | 1 | El flujo comienza cuando el tecnico quiere iniciar el sistema | |
| Post-conditions | El sistema ya esta en funcionamiento | | |
| | Los dispositivos de la red no estan conectados a la corriente. | | |
| Author | Cretu | | |

Figura 22: Casos de uso - Iniciar funcionamiento de la red

El caso de uso *Detener funcionamiento de la red* comenzaría cuando el usuario administrador decide que se debe apagar todo el sistema.

| | | |
|---|--|------------------------|
| Use Case ID | 5 | |
| Super Use Case | | |
| Primary Actor | Administrador | |
| Secondary Actor(s) | | |
| Brief Description | El administrador tiene que de detener el funcionamiento del sistema. | |
| Preconditions | El sistema tiene que estar iniciado y revisado por el administrador. | |
| Flow of Events | Actor Input | System Response |
| | 1 El flujo comienza cuando el administrador tiene que detener el sistema | |
| | 2 | El sistema se detiene |
| Post-conditions | El sistema se ha detenido | |
| Alternative flows and exceptions | Los dispositivos están realizando tareas de conectividad o traspaso de información | |
| Author | Cretu | |

Figura 23: Casos de uso - Detener funcionamiento de la red

6.3. Fase de Diseño

Tras concluir la fase de análisis, donde se han establecido los casos de uso para cada actor del sistema, comienza la fase de diseño. En esta fase se definirá el sistema de forma detallada para poder implementarlo en la siguiente fase.

Para el desarrollo de esta fase, se explicará la forma en la que se ha creado el sistema, incluyendo la automatización mediante CLI y AWX. También se detallará la topología elegida y la estructura de Ansible.

Si la automatización se realiza por CLI o si se realiza mediante AWX, la topología siempre será la misma para ambos casos.

6.3.1. Diseño de la topología

Para la realización de este TFG se ha escogido una topología extraída de un laboratorio de CISCO. Concretamente se trata de laboratorio de troubleshooting, que pertenece al CCNA: Routing and Switching Essentials.

La red está implementada en GNS3 y está formada por tres Routers, dos Switches, dos ordenadores simples, un ordenador desde donde se ejecutará Ansible y una interfaz en forma de nube (NAT) que establece la conexión con la tarjeta de red física del portátil MacBook, para así después poder ejecutar Ansible mediante AWX.

En la Figura 24 se puede ver el diseño de la red hecho en GNS3 para realizar la automatización con Ansible por CLI desde el dispositivo *Admin*, que en la fase de implementación se verá como crearla.

Pero para realizar la automatización de la red con Ansible mediante AWX, el diseño de la red tiene un ligero cambio ya que AWX es ejecutado desde el navegador del portátil físico, Macbook y se conecta directamente a los dispositivos de red creados en GNS3, mediante la interfaz en forma de nube (NAT), mencionada anteriormente. En la siguiente Figura 25 se puede observar que simplemente desaparece el dispositivo *Admin* para así conectar directamente AWX con los dispositivos de red.

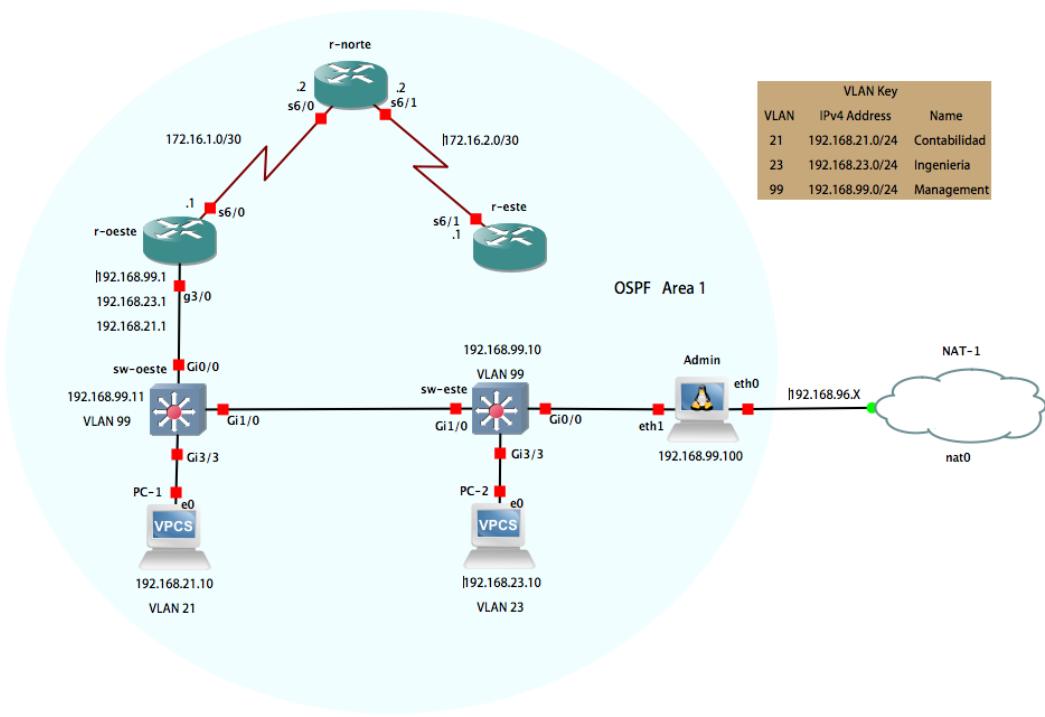


Figura 24: Diseño de la topología en GNS3

En la Figura 24 y 25 se puede apreciar claramente los tres Routers en la parte superior llamados *r-oeste*, *r-norte* y *r-este*, en la parte del medio están los dos Switches, *sw-oeste* y *sw-este*, en la parte inferior están los dos ordenadores simples, *PC-1* y *PC-2* y por último, en la Figura 24 está el dispositivo *Admin* desde el que se ejecuta Ansible por CLI.

También se puede observar que hay una asignación de IP's para cada dispositivo y un recuadro con las IP's pertenecientes a cada VLAN, pero en la parte de implementación se detallará mejor cada asignación de IP hecha.

Para cumplir con los requisitos del proyecto y poder realizar la automatización de la configuración de los dispositivos de red se han propuesto las siguientes configuraciones:

1º ***Iniciar el funcionamiento de la red***: Para poder realizar la automatización es muy importante asegurar que la red está iniciada y si no lo estuviese proceder a iniciar todos los dispositivos de la red.

2º ***Configuración básica de los dispositivos***: Como se verá en la fase de implementación, todos los dispositivos de red, cuando se inician por primera vez necesitan una serie de configuraciones básicas para poder después conectarse a ellos, en este caso mediante SSH.

3º ***Configurar seguridad en el Switch, VLANs y encaminamiento Inter-VLAN***: El objetivo de este laboratorio es tener conexión entre los distintos dispositivos creando departamentos mediante VLAN. Toda esta configuración se realizará de manera automática mediante Ansible. [26]

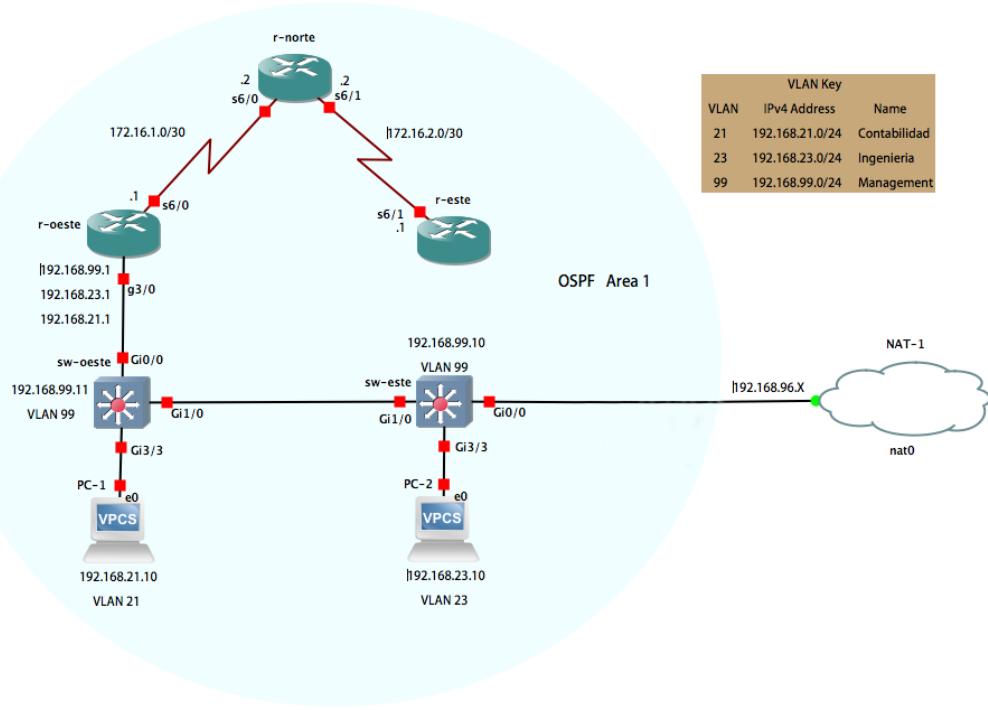


Figura 25: Diseño de la topología en GNS3

- 4º **Configurar Open Shortest Path First (OSPF)**: También es necesario habilitar el protocolo de red OSPF que ayuda a los dispositivos de encaminamiento (Routers) a saber que redes hay conectadas y disponibles en toda la topología para así poder crear un camino más corto hacia el destino. [27]
- 5º **Configurar Dynamic Host Configuration Protocol (DHCP) en r-oeste**: Para añadir un poco más de complejidad a la automatización, también se implementará DHCP en el *r-oeste* para poder asignar IP's automáticamente a los dispositivos de las VLAN. [28]

6.3.2. Estructura del sistema

En este apartado se describirá la manera de interactuar de todo el sistema para realizar la automatización.

6.3.2.1 Automatización mediante Ansible por CLI

Como se ha detallado en la parte del Estado del Arte, Ansible utilizar SSH para poder comunicarse con los distintos dispositivos. Por lo tanto, otra ventaja de utilizar esta herramienta de automatización con dispositivos de red es que como los dispositivos CISCO tiene la posibilidad de habilitar este protocolo, para que puedan conectarse remotamente, facilita el uso de Ansible para automatizar las redes de cualquier empresa y además se crea una comunicación segura de datos.

En la Figura 26 se puede observar cómo está estructurado el proyecto para automatizar la red a través de Ansible mediante CLI. ??

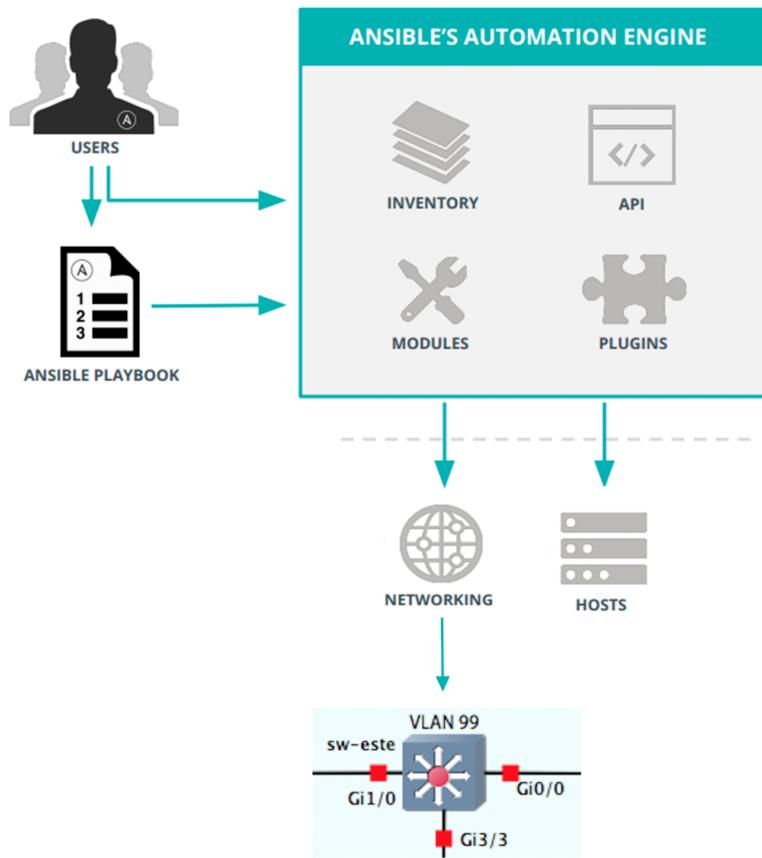


Figura 26: Estructura del proyecto desde CLI

Para esta primera estructura, se ubican, dentro de la máquina de *Admin*, todos los ficheros que aparecen en la Figura anterior. El administrador del proyecto procedería a crear los Playbooks y los demás ficheros necesarios para el funcionamiento del Ansible como por ejemplo el *Inventory*. Una vez creados todos los archivos, el administrador podría lanzar los Playbooks a los distintos dispositivos definidos en *Inventory* o simplemente avisar al técnico para que los lanzara él. Todas estas configuraciones y ejecuciones de Playbooks se harían desde CLI.

En la siguiente estructura, que se puede ver en la Figura 27, la ejecución completa de Ansible se haría a través de AWX. El procedimiento sería el mismo que en el caso anterior, el administrador debe crear los Playbooks, pero ahora las ejecuciones de dichos Playbooks se pueden realizar desde la interfaz gráfica. Permitiendo así poder crear programaciones de lanzamientos, workflows y también tener más seguridad ya que permite controlar quién, cuándo y dónde han ejecutado las tareas.

La siguiente fase es la fase de implementación. Este apartado tiene una gran importancia ya que es donde se procederá a crear paso por paso toda la estructura para poder realizar la automatización de una red, en este caso un laboratorio del CCNA: Switching

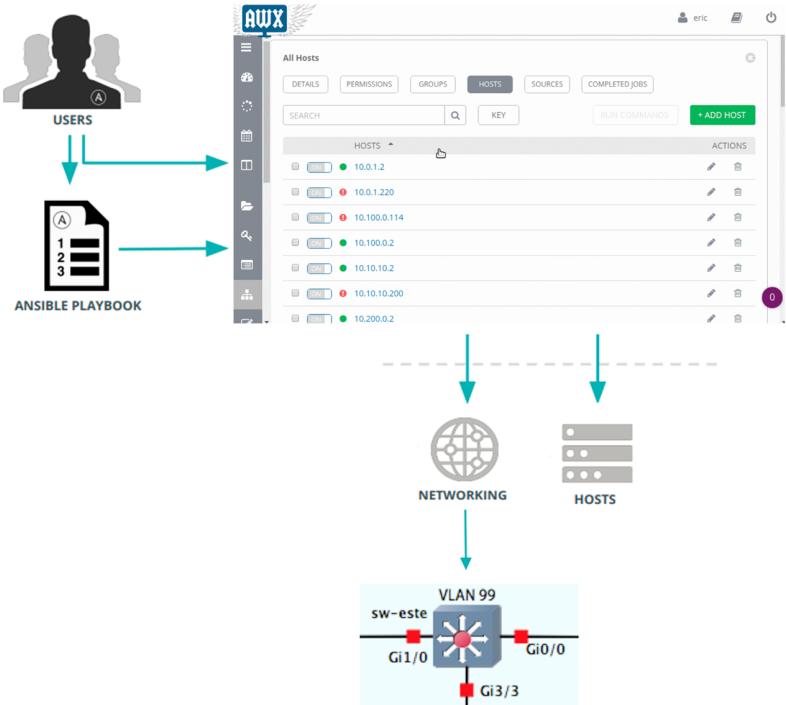


Figura 27: Estructura del proyecto con AWX

and Routing Essential. Para que cualquier persona o incluso técnicos de empresas, puedan poner en práctica todo lo aprendido hasta ahora e intentar aplicar esta modalidad de trabajo que es bastante productiva.

6.4. Fase de implementación

En el presente apartado se procederá a crear toda la estructura en forma de manual, para poder realizar una automatización de dispositivos de red, CISCO.

Los siguientes pasos son necesarios tanto si se utiliza Ansible desde CLI como si se utiliza desde AWX.

Antes de empezar con las configuraciones y las ejecuciones hay que cerciorarse que todo está correctamente instalado y funcionando.

Para ello el primer paso es comprobar que se ha descargado la máquina virtual *GNS3 VM* y el programa *GNS3* desde la página oficial www.gns3.com.

Para comprobar que todo está correctamente instalado y funcionando se debe ejecutar el programa e iniciar la máquina virtual, de modo que el resultado final sería el que se puede ver en la Figura 28 y 29:

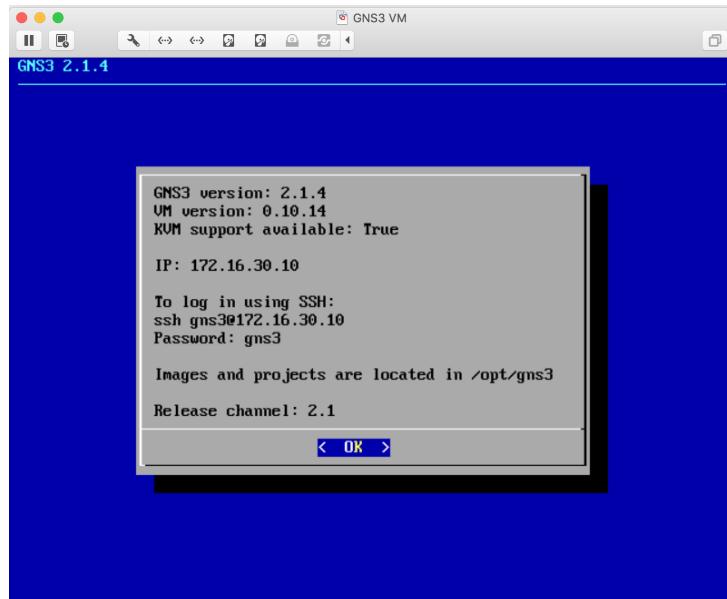


Figura 28: Máquina virtual GNS3 VM

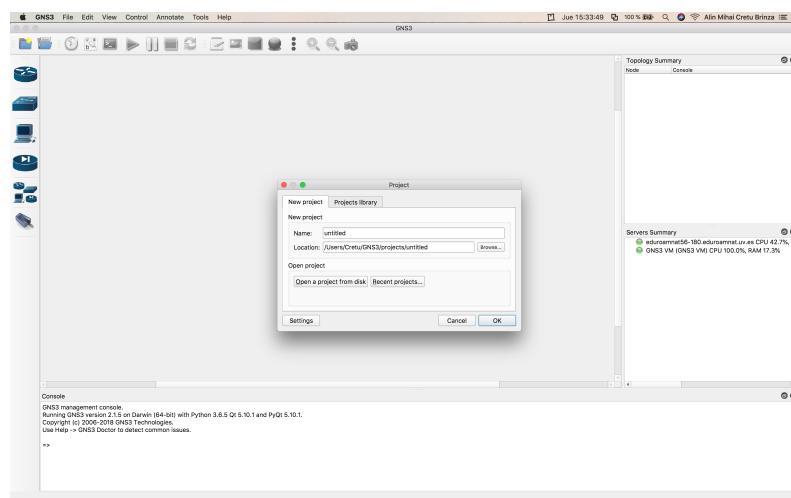


Figura 29: Software GNS3

Esta máquina virtual viene ya configurada, pero si no se diese el caso, habría que configurar dos interfaces de red en el VMWare. Uno sería para darle IP a la máquina virtual para acceder al servidor y la otra interfaz sería para que las máquinas creadas dentro de GNS3 pudiesen tener una conexión a internet que se realizaría mediante la interfaz en forma de nube mencionada en el apartado anterior.

En la Figura 30 y 31 se puede apreciar las dos interfaces que debe tener la máquina:



Figura 30: Interfaz interna para el servidor GNS3



Figura 31: Interfaz para tener acceso a internet

A continuación, se procedería a crear la topología dentro de GNS3. Para ello con el programa abierto y la máquina virtual iniciada, solamente se tendría que arrastrar los dispositivos necesarios al escenario. Un pequeño ejemplo de cómo realizarlo se puede ver en la Figura 32:

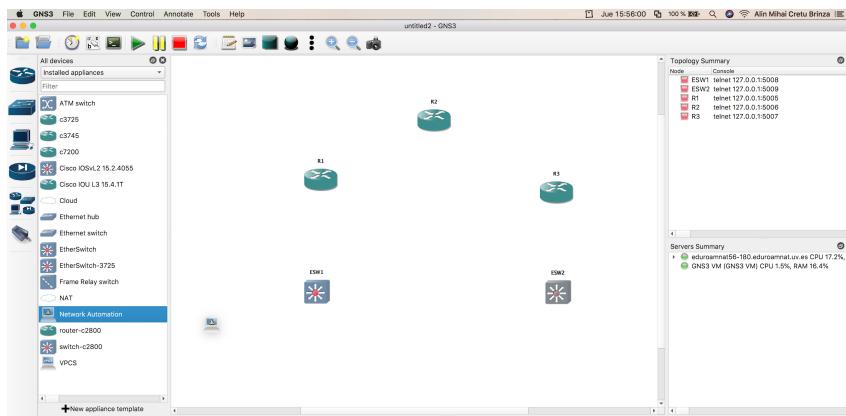


Figura 32: Escenario de GNS3 con la topología

Cuando se instala GNS3, trae solamente siete dispositivos instalados, pero muy básicos, que no servirían para nada en este proyecto. Como se explicó en la sección de GNS3, el usuario es el que tiene que proporcionar las imágenes de los sistemas operativos de CISCO y también se debe instalar un *appliance* del *Marketplace* de GNS3 llamado *NetworkAutomation*.

6.4.1. Importación del proyecto de Ansible

Hoy en día tener una copia de seguridad, un control de versiones y una organización de los proyectos realizados es fundamental para un correcto funcionamiento y para asegurar el contenido. Para la realización de este proyecto se han creado varios repositorios en GitHub y en GitLab para tener un control de las versiones, asegurar el contenido, la posibilidad de una rápida creación del proyecto en cualquier máquina clonando el repositorio y una buena organización. Además de compartir el proyecto con toda la comunidad para que puedan utilizar esta herramienta.

Para utilizar dichos repositorios primero se debe crear el proyecto en el ordenador y seguidamente mediante unos sencillos comandos se puede subir el proyecto al repositorio.

Los pasos a seguir para guardar el proyecto en GitLab y en GitHub son los siguientes:

```

1- Crear el proyecto en el ordenador
2- Acceder a dicha carpeta - cd /TFG
# Ejecutar los siguientes comandos :
3- git config --global user.name "El nombre de la persona"
4- git config --global user.email "Email de la persona"
5- git init
6- git remote add origin https://gitlab.com/USUARIO/PROYECTO.git
o para github
6- git remote add origin https://github.com/USUARIO/PROYECTO.git

```

```

7- git add .
8- git commit -m "Initial commit"
9- git push -u origin master

```

Listado 23: Subir un proyecto a un repositorio.

Para poder utilizar el mismo proyecto y realizar prácticas de automatización de dispositivos virtuales o físicos se puede acceder a este proyecto clonandolo. Hay que destacar que para dispositivos físicos algunas configuraciones cambian por las interfaces disponibles de cada dispositivo.

Los pasos a seguir para clonar el proyecto son los siguientes:

```

1- Acceder al directorio donde se desea clonar
2- git clone https://gitlab.com/amicre/Training.git
o desde Github
2- git clone https://github.com/utercnila/TFG.git

```

Listado 24: Clonar el proyecto.

En las Figuras 34 y 33 se puede observar los dos repositorios con el proyecto creado:

| Name | Last commit | Last update |
|---------------|---------------------------------|-------------|
| Images | Add Topology image | 1 month ago |
| Initial_Setup | Initial_Setup | 1 month ago |
| backups | Last commit | 1 month ago |
| group_vars | Last modification for roles | 1 month ago |
| roles | h | 1 week ago |
| README.md | 12- DHCP | 1 month ago |
| ansible.cfg | roles | 1 week ago |
| hosts | All config without organization | 1 month ago |

Figura 33: GitLab

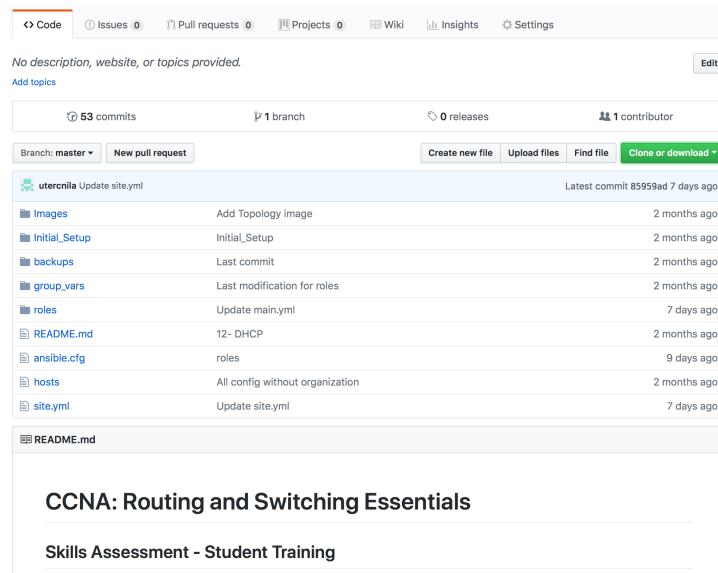


Figura 34: GitHub

A continuación, se implementará Ansible desde CLI y desde AWX, creando todos los ficheros necesarios para su funcionamiento.

6.4.2. Implementación para utilizar Ansible desde CLI

En este proyecto se utiliza el *appliance* oficial de GNS3, *NetworkAutomation*. Es un dispositivo más, con un sistema operativo Ubuntu pero facilita la configuración ya que tiene por defecto instalado Ansible y Python. A lo largo de este proyecto, este dispositivo se llamará *Admin*.

Como se ha definido en los casos de uso, debe haber dos usuarios, por lo tanto, lo primero es crear los dos usuarios con los comandos que se ven a continuación:

```
root@Admin:~# adduser tecnico
root@Admin:~# adduser administrador
```

Listado 25: Crear usuarios en Admin

Para comprobar el funcionamiento de Ansible y para asegurar que el usuario administrador se ha creado correctamente, se puede utilizar un comando Ad-Hoc explicado en la sección de Arquitectura de Ansible. Para realizar la comprobación, dentro de la máquina *Admin*, se debe ejecutar los siguientes comandos:

```
root@Admin:~# login administrador
##introducir password
administrador@Admin:~$ ansible localhost -m setup
## El resultado se puede ver en el Anexo %Resultado ansible
localhost -m setup
```

Listado 26: Comprobación

Tras realizar la comprobación anterior, el siguiente paso es empezar la configuración y adaptación de Ansible a este proyecto.

Los archivos que se van a mostrar a continuación deben ser implementados en la máquina *Admin*. Son los archivos, mencionados en el apartado Estado del Arte, necesarios para que Ansible pueda realizar la automatización de los dispositivos de red remotamente.

6.4.2.1 Host Inventory

Este es un fichero de especial relevancia para que Ansible pueda ejecutarse de manera correcta. Este archivo tiene extensión *.yml* y dentro se han de definir los dispositivos existentes en la red que se quiera automatizar. Para tener una buena organización este archivo debería de estar ubicado en la carpeta del proyecto, más adelante se verá la estructura final de Ansible para este TFG. Para este proyecto dicho archivo tiene el siguiente contenido:

```
[switch]
sw-este
sw-oeste

[router]
r-oeste
r-norte
r-este
```

Listado 27: Host Inventory

Es recomendable no definir variables en este archivo, así que cuanto más sencillo mejor. Como se puede observar en el Listado 25, se han definido únicamente los dispositivos de red existentes en la topología de este TFG.

Para que este archivo pueda reconocer los nombres de los dispositivos se debe modificar el archivo */etc/hosts* ubicado en la máquina *Admin* añadiendo las siguientes líneas:

```
192.168.99.10 sw-este
192.168.99.11 sw-oeste
192.168.99.1 r-oeste
172.16.1.2 r-norte
172.16.2.1 r-este
```

Listado 28: /etc/hosts

6.4.2.2 ansible.cfg

El siguiente archivo a configurar es el *ansible.cfg*. Dicho fichero debe nombrarse como *ansible.cfg* ya que si es nombrado de forma diferente no funcionará de forma correcta. Cuando se instala Ansible se crea este archivo, pero tiene muchas líneas de código con ejemplos y configuraciones que para este caso en concreto no son útiles, por este motivo es por lo que se crea de nuevo. Cuando se ejecuta un Playbook, Ansible, internamente

busca este fichero en la carpeta desde la cual se está ejecutando dicho Playbook y si no estuviese creado buscaría en la carpeta original `/etc/ansible/`.

Este fichero se ha creado con las siguientes líneas:

```
[defaults]
inventory = hosts
become_method = enable
host_key_checking = false
timeout = 100
remote_user = ansible
```

Listado 29: ansible.cfg

Se pueden definir configuraciones en este fichero para `[inventory]`, `[privilege_escalation]`, `[ssh_connection]` pero para este proyecto se han definido solamente configuraciones básicas por defecto.

- **`[defaults]`**: Indicar configuraciones básicas por defecto.
- **`inventory`**: Se le indica el fichero Host Inventory que va a utilizar Ansible, en este caso se llama `hosts`.
- **`become_method`**: Es un parámetro útil para configurar y automatizar dispositivos de red. Le permite a la herramienta obtener permisos para realizar ciertas configuraciones.
- **`host_key_checking`**: Se le indica con la opción “`false`” que no realice comprobaciones de clave SSH.
- **`timeout`**: Se establece un tiempo de conexión.
- **`remote_user`**: Se define el usuario creado en el dispositivo remoto.

Tras acabar la configuración de Ansible solamente quedarían tres simples configuraciones en la máquina *Admin* y hay que destacar lo sencillo y rápido que es configurar Ansible.

Finalmente, para acabar con las configuraciones de la máquina administradora y pasar a implementar los Playbooks, se procederá a configurar la interfaz para dar IP, dar todos los permisos al usuario administrador, dar permisos de solo ejecución al usuario técnico y definir unas rutas por defecto para que la máquina *Admin* sepa por donde ir a Internet y por donde a la red creada.

Para que la máquina orquestadora pueda acceder a Internet y a la red local, se deben agregar las siguientes líneas al fichero `/etc/network/interfaces`:

```
auto eth0
iface eth0 inet dhcp
auto eth1
iface eth1 inet static
address 192.168.99.100
netmask 255.255.255.0
gateway 192.168.99.1
```

Listado 30: /etc/network/interfaces

La interfaz “eth0” tiene habilitado DHCP, que será la interfaz para acceder a Internet y la interfaz “eth1” tiene asignada una dirección IP estática para conectarse a red local y realizar toda la automatización.

Para cumplir con los requisitos y respetar los casos de uso hay que dar los permisos adecuados para cada usuario.

Al usuario administrador se le deben conceder todos los permisos, así que es conveniente añadirlo a la lista de *sudoers* y para ello se debe modificar el fichero */etc/sudoers* con la siguiente línea:

```
administrador ALL=(ALL:ALL) ALL
```

Listado 31: /etc/sudoers

Antes de seguir con los permisos para cada usuario es conveniente crear el proyecto. Para este TFG se ha creado un proyecto en GitLab y en GitHub como se ha explicado en 6.4.1. Se debe clonar o crear en la máquina *Admin* con el usuario administrador. Para clonar el proyecto se utilizan los siguientes comandos:

```
administrador@Admin:~$ login administrador
administrador@Admin:~$ git clone https://gitlab.com/amicre/
    Training.git
administrador@Admin:~$ git clone https://github.com/utercnila/TFG
    .git
```

Listado 32: Clonar proyecto

Como se puede observar en la Figura 35, el usuario que ha creado el proyecto es *administrador* y también se pueden apreciar los permisos para cada grupo, usuario y resto.

```
drwxrwxr-x 8 administrador administrador 4096 Jun 18 07:23 Training
```

Figura 35: Proyecto clonado.

Según los casos de uso, los permisos no están asignados correctamente ya que los usuarios sin privilegios pueden modificar archivos del proyecto y el objetivo de estos usuarios, en este caso el técnico, es solamente ejecutar los Playbooks y poder leerlos. Para modificar estos permisos se debe ejecutar la siguiente línea:

```
administrador@Admin:~$ chmod 575
```

Listado 33: Permisos para el usuario técnico.

Ahora el técnico no podrá modificar los ficheros del proyecto, pero sí que los podrá leer y ejecutar.

Por último y para acabar con las configuraciones de la máquina *Admin* queda establecer las rutas por defecto para que se pueda acceder a Internet e ir hacia la red local. Esto se consigue con los siguientes comandos:

```
dr-xrwxr-x 8 administrador administrador 4096 Jun 18 07:23 Training
```

Figura 36: Permisos modificados.

```
route add -net 172.16.0.0/16 dev eth1
route del -net 0.0.0.0/0 gw 192.168.99.1
```

Listado 34: Rutas por defecto.

Una vez finalizada la configuración básica para la máquina *Admin* se debe aplicar la configuración básica para los dispositivos de red, como se especificó en el apartado de diseño. Esto es necesario ya que si un dispositivo de red no tiene activado el SSH y tampoco tiene una IP es imposible acceder a dicho dispositivo y no se podría realizar la automatización.

Dado que los sistemas operativos de los dispositivos, para este proyecto, no se pueden modificar para habilitar el DHCP o para habilitar el SSH por defecto, se debe realizar de modo manual. Los archivos de configuración para cada dispositivo están disponibles en el Anexo. Los pasos a seguir para aplicar la configuración serían los siguientes:

- 1º Primero de todo se debe encender el dispositivo.
- 2º Una vez arrancado, el técnico o el administrador de la red debe entrar en modo privilegiado con el comando *enable* para poder aplicar la configuración.
- 3º Después se copian todas las líneas del archivo correspondiente a cada dispositivo y se pegan en la Interfaz de Línea de Comandos (CLI) o también se pueden escribir una por una de forma manual.
- 4º Por último hay que repetir lo mismo para todos los dispositivos de la red.

Realizando estas configuraciones básicas será suficiente para que Ansible pueda automatizar las demás configuraciones y conseguir que todos los equipos de la red se puedan comunicar unos con otros, ya que de momento eso es imposible y se puede observar en la Figura 37:

```
[sw-este#ping 192.168.99.100
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.99.100, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Figura 37: Error en la conexión.

Siguiendo con la estructura del proyecto y lo que se definió en la parte de diseño, ahora se crearán los Playbooks para cumplir con los objetivos del diseño. Primero de todo se crearán los roles para cada dispositivo ejecutando los siguientes comandos dentro de la carpeta del proyecto:

```

administrador@Admin:~$ cd Training/
administrador@Admin:~$ mkdir roles/
administrador@Admin:~/Training/roles$ ansible-galaxy init Backup
administrador@Admin:~/Training/roles$ ansible-galaxy init
    config_sw_este
administrador@Admin:~/Training/roles$ ansible-galaxy init
    config_sw_oeste
administrador@Admin:~/Training/roles$ ansible-galaxy init
    config_r_oeste
administrador@Admin:~/Training/roles$ ansible-galaxy init
    config_r_norte
administrador@Admin:~/Training/roles$ ansible-galaxy init
    config_r_este

```

Listado 35: Rutas por defecto.

Una buena organización del proyecto facilita la interpretación del mismo y la comprensión para que los demás usuarios entiendan su funcionamiento, por lo tanto, es muy recomendable crear roles. En la Figura 38 se puede ver el resultado final de esta carpeta.

```

administrador@Admin:~/Training/roles$ ls -l
total 24
dr-xrwxr-x 8 administrador administrador 4096 Jun 18 07:22 Backup
dr-xrwxr-x 8 administrador administrador 4096 Jun 18 07:22 config_r_este
dr-xrwxr-x 8 administrador administrador 4096 Jun 18 07:22 config_r_norte
dr-xrwxr-x 8 administrador administrador 4096 Jun 18 07:22 config_r_oeste
dr-xrwxr-x 8 administrador administrador 4096 Jun 18 07:22 config_sw_este
dr-xrwxr-x 8 administrador administrador 4096 Jun 18 07:22 config_sw_oeste

```

Figura 38: Estructura de la carpeta roles.

Otro consejo es realizar copias de seguridad de la configuración inicial de cada dispositivo o realizarlas siempre y cuando se apliquen cambios en la configuración por que suelen haber errores aplicando configuraciones y puede que dejen de funcionar los equipos.

6.4.2.3 Roles

Backup

El primer rol a desarrollar es el de Backup. A continuación 39 se puede observar la estructura del rol de Backup:

```
[administrador@Admin:~/Training/roles/Backup$ tree
.
|-- README.md
|-- defaults
|   '-- main.yml
|-- handlers
|   '-- main.yml
|-- meta
|   '-- main.yml
|-- tasks
|   '-- show_run.yml
|-- tests
|   |-- inventory
|   '-- test.yml
`-- vars
    '-- main.yml

6 directories, 8 files
```

Figura 39: Estructura del rol Backup.

Dado que este rol es muy sencillo y solamente realiza copias de seguridad de todos los dispositivos, se ha creado únicamente un Playbook para realizar dicha tarea:

```
---
- name: Show devices' startup-config
  ios_command:
    authorize: yes
    auth_pass: ansible
    commands:
      - show start

  register: print
  - debug: var=print.stdout_lines

- name: Save configuration to a file
  copy: content="{{ print.stdout[0] }}" dest="/administrador/
    Training/backups/{{ inventory_hostname }}.txt"
```

Listado 36: sh_run.yml

config_sw_este

Este rol está dedicado al primer Switch que conecta con la máquina *Admin*, por lo tanto, su estructura es la siguiente 40:

```
[administrador@Admin:~/Training/roles/config_sw_este$ tree
.
|-- README.md
|-- defaults
|   '-- main.yml
|-- handlers
|   '-- main.yml
|-- meta
|   '-- main.yml
|-- tasks
|   |-- config_vlan.yml
|   '-- create_vlan.yml
|-- tests
|   '-- test.yml
`-- vars
    '-- main.yml

6 directories, 8 files
```

Figura 40: Estructura del rol config_sw_este.

Para este rol se han creado dos Playbooks:

```
---
- name: create VLAN
  ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
      - name: {{ item.name }}
    parents:
      - vlan "{{ item.id }}"

  with_items:
  - { id: "{{ id_vlan21 }}", name: "{{ name_vlan21 }}" }
  - { id: "{{ id_vlan23 }}", name: "{{ name_vlan23 }}" }

- name: show vlan
  ios_command:
    authorize: yes
    commands:
      - show vlan brief
  register: sh_vlan

- debug: var=sh_vlan.stdout_lines
```

Listado 37: create_vlan.yml.

```

### CONFIG VLANS & INTERFACES ###

---
- name: Register Facts
  ios_facts:

#Force trunking on Interface G1/0#

- name: Enable and describe G1/0
  ios_interface:
    authorize: yes
    auth_pass: ansible
    name: G1/0
    description: trunk port

- name: Enable trunk mode on G1/0
  ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
      - switchport mode trunk
      - switchport trunk encapsulation dot1Q
        - switchport trunk native vlan 1
      parents:
        - interface GigabitEthernet 1/0

#####Assign G3/3 to VLAN 23#####
- name: Enable, describe G3/3
  ios_interface:
    authorize: yes
    auth_pass: ansible
    name: G3/3
    description: Access to Vlan 23

- name: Enable mode access on int G3/3
  ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
      - switchport mode access
      - switchport access vlan 23
    parents:
      - interface GigabitEthernet 3/3

###Assign all other ports as access ports and Shutdown###
- name: Modo access para las demas interfaces & Shutdown

```

```

interfaces
ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
        - switchport mode access
    parents:
        - interface range {{ item.name }}
    after:
        - shutdown

with_items:
    - { name: GigabitEthernet 0/1-3 }
    - { name: GigabitEthernet 1/1-3 }
    - { name: GigabitEthernet 2/0-3 }
    - { name: GigabitEthernet 3/0-2 }

- name: Show Running Config
  ios_command:
    authorize: yes
    commands:
        - show run | begin interface

register: sh_run_interface

- debug: var=sh_run_interface.stdout_lines

- name: save running to startup when modified
  ios_config:
    authorize: yes
    auth_pass: ansible
    save_when: modified

```

Listado 38: config_vlan.yml.

config_sw_oeste

El segundo Switch de la topología tiene la siguiente estructura 41:

Este rol también tiene dos Playbooks pero dado que create_vlan.yml es idéntico al del rol *config_sw_este* no se añadirá a la memoria para no repetir cosas innecesarias.

Identico al Playbook del rol de config_sw_este

Listado 39: create_vlan.yml.

En el Playbook de *config_vlan.yml* también hay similitudes con el del rol anterior, pero cambian variables por lo tanto no se puede reutilizar.

```
[administrador@Admin:~/Training/roles/config_sw_oeste$ tree
.
|-- README.md
|-- defaults
|   '-- main.yml
|-- handlers
|   '-- main.yml
|-- meta
|   '-- main.yml
|-- tasks
|   |-- config_vlan.yml
|   '-- create_vlan.yml
|-- tests
|   '-- test.yml
`-- vars
    '-- main.yml

6 directories, 8 files
```

Figura 41: Estructura del rol config_sw_oeste.

```
### CONFIG VLANS & INTERFACES ####
---
- name: Register Facts
  ios_facts:

#Force trunking on Interface G0/0 with r-oeste#
- name: Enable and describe G0/0
  ios_interface:
    authorize: yes
    auth_pass: ansible
    name: G0/0
    description: trunk port with r-oeste

- name: Enable trunk mode on G0/0
  ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
      - switchport mode trunk
- switchport trunk encapsulation dot1Q
  - switchport trunk native vlan 1
parents:
  - interface GigabitEthernet 0/0

#####Assign G3/3 to VLAN 21#####

```

```

- name: Enable , describe G3/3
  ios_interface:
    authorize: yes
    auth_pass: ansible
    name: G3/3
    description: Access to Vlan 21

- name: Enable mode access on int G3/3
  ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
      - switchport mode access
      - switchport access vlan 21
    parents:
      - interface GigabitEthernet 3/3

#####Assign all other ports as access ports and Shutdown#####

- name: Modo access para las demas interfaces & Shutdown
  interfaces
  ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
      - switchport mode access
    parents:
      - interface range {{ item.name }}
    after:
      - shutdown

    with_items:
      - { name: GigabitEthernet 0/1-3 }
      - { name: GigabitEthernet 1/1-3 }
      - { name: GigabitEthernet 2/0-3 }
      - { name: GigabitEthernet 3/0-2 }

- name: Show Running Config
  ios_command:
    authorize: yes
    commands:
      - show run | begin interface

  register: sh_run_interface

  - debug: var=sh_run_interface.stdout_lines

```

```

- name: save running to startup when modified
  ios_config:
    authorize: yes
    auth_pass: ansible
    save_when: modified

```

Listado 40: config_vlan.yml.

Dado que los Playbooks de los Switches utilizar las mismas variables se han creado todas ellas en la carpeta *group_vars/switch/main.yml*, ubicada en la raíz del proyecto, de este modo se utiliza las distintas formas de crear variables explicadas en el Estado del Arte y para no tener variables repetidas. Dicho fichero contiene lo siguiente:

```

---
# Id for VLANs #
id_vlan21: 21
id_vlan23: 23
# Name for VLANs #
name_vlan21: Contabilidad
name_vlan23: Ingenieria

```

Listado 41: group_vars/switch/main.yml.

Una vez finalizado la creación de los Playbooks para los roles de los Switches es el turno de los Routers.

config_r_oeste

Este Router será el encargado de crear las subinterfaces para que las VLAN se puedan ver entre si y además será el encargado de repartir IP's a través de DHCP a las distintas VLAN's. La estructura de este rol es la siguiente 42:

```
[administrador@Admin:~/Training/roles/config_r_oeste$ tree
.
|-- README.md
|-- defaults
|   '-- main.yml
|-- handlers
|   '-- main.yml
|-- meta
|   '-- main.yml
|-- tasks
|   |-- config_dhcp.yml
|   |-- config_interfaces.yml
|   |-- config_ospf.yml
|   '-- config_subinterfaces.yml
|-- tests
|   '-- test.yml
`-- vars
    '-- main.yml

6 directories, 10 files
```

Figura 42: Estructura del rol config_r_oeste.

Los cuatro roles que hay disponibles para este rol son los siguientes:

```
---

- name: Reserve the first 20 ip addresses in VLAN 21 & 23 for
  static config
  ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
      - ip dhcp excluded-address {{ item.ip_inicio }} {{ item.
          ip_fin }}
  with_items:
    - { ip_inicio: 192.168.21.1, ip_fin: 192.168.21.20 }
    - { ip_inicio: 192.168.23.1, ip_fin: 192.168.23.20 }

- name: Create the DHCP pools for VLAN 21 & 23
  ios_config:
    authorize: yes
    auth_pass: ansible
  lines:
    - network {{ item.ip_vlan }} 255.255.255.0
    - domain-name ccna-sa.com
    - dns-server {{ item.ip_dns }}
    - default-router {{ item.ip_default }}
  parents:
```

```

    - ip dhcp pool "{{ item.name }}"
with_items:
- { name: Contabilidad, ip_vlan: 192.168.21.0, ip_dns:
  10.10.10.10, ip_default: 192.168.21.1 }
- { name: Ingenieria, ip_vlan: 192.168.23.0, ip_dns:
  10.10.10.10, ip_default: 192.168.23.1 }

- name: Show run | section DHCP
  ios_command:
    authorize: yes
    commands:
      - show run | section dhcp
  register: sh_dhcp
- debug: var=sh_dhcp.stdout_lines

- name: Save configuration
  ios_command:
    authorize: yes
    auth_pass: ansible
    commands:
      - write memory

```

Listado 42: config_dhcp.yml.

```

## Configure the router's interfaces ##
---
- include_vars: ../vars/main.yml
- name: Enable serial 6/0
  ios_interface:
    authorize: yes
    auth_pass: ansible
    name: Serial 6/0
    description: to r-norte

- name: Configure serial 6/0
  ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
      - ip address {{ ips60 }} {{ mask_r_oeste_s6_0 }}
      - clock rate 128000
    parents:
      - interface {{ item.id }}
    after: no shutdown
  with_items:
    - { id: Serial 6/0 }

- name: Save configuration

```

```

ios_command:
  authorize: yes
  auth_pass: ansible
  commands:
    - write memory

```

Listado 43: config_interfaces.yml.

```

I---
- include_vars: ../vars/main.yml
- name: Enable OSPF on R-oeste
  ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
      - network {{ ip_lan23 }} {{ wildmask_lan23 }} area {{ area }}
      - network {{ ip_lan21 }} {{ wildmask_lan21 }} area {{ area }}
      - network {{ ip_lan99 }} {{ wildmask_lan99 }} area {{ area }}
      - network {{ ip_s6_0 }} {{ wildmask_s6_0 }} area {{ area }}
    parents:
      - router ospf 1

- name: Save configuration
  ios_command:
    authorize: yes
    auth_pass: ansible
    commands:
      - write memory

```

Listado 44: config_ospf.yml.

```

## Configure the subinterfaces for the VLAN 21 & 23
---
- include_vars: ../vars/main.yml
- name: Configure subinterfaces for VLAN 21 & 23
  ios_config:
    authorize: yes
    auth_pass: ansible
    parents:
      - interface g3/0.{{ item.id }}
    lines:
      - description Contabilidad LAN
      - encapsulation dot1q {{ item.id }}
      - ip address {{ item.ip }} 255.255.255.0

```

```

with_items:
- { id: "{{ id_vlan21 }}", ip: "{{ ip_vlan21 }}" }
- { id: "{{ id_vlan23 }}", ip: "{{ ip_vlan23 }}" }

- name: Save configuration
  ios_command:
    authorize: yes
    auth_pass: ansible
    commands:
      - write memory

```

Listado 45: config_subinterfaces.yml.

Como se puede observar en los Playbooks anteriores, se ha usado el parámetro *include_vars* por lo tanto debe haber un fichero con variables en este rol. El fichero de variables que está ubicado en la carpeta *vars/main.yml* tiene el siguiente contenido:

```

---
# vars file for config_r_oeste
parent: router ospf 1
area: 1
## Configuration for OSPF (networks) ##
ip_lan99: 192.168.99.0
wildmask_lan99: 0.0.0.255
ip_lan21: 192.168.21.0
wildmask_lan21: 0.0.0.255
ip_lan23: 192.168.23.0
wildmask_lan23: 0.0.0.255
ip_s6_0: 172.16.1.0
wildmask_s6_0: 0.0.0.3
## Subinterface for Vlan21 ##
id_vlan21: 21
ip_vlan21: 192.168.21.1
## Subinterface for Vlan23 ##
id_vlan23: 23
ip_vlan23: 192.168.23.1
## Ip for serial6/0 ##
ips60: 172.16.1.1
mask_r_oeste_s6_0: 255.255.255.252

```

Listado 46: vars/main.yml.

config_r_norte

Los siguientes dos roles tienen una estructura muy similar, de modo que la del Router Norte quedaría de la siguiente manera 43:

```
[administrador@Admin:~/Training/roles/config_r_norte$ tree
.
|-- README.md
|-- config_interfaces.yml
|-- defaults
|   '-- main.yml
|-- handlers
|   '-- main.yml
|-- meta
|   '-- main.yml
|-- tasks
|   |-- config_interfaces.yml
|   '-- config_ospf.yml
|-- tests
|   '-- test.yml
`-- vars
    '-- main.yml

6 directories, 9 files
```

Figura 43: Estructura del rol config_r_norte.

Los dos Playbooks creados para este rol se pueden observar a continuación:

```
## Configure the north router interfaces ##

---
- include_vars: ../vars/main.yml
- name: Enable serial 6/0 & 6/1
  ios_interface:
    authorize: yes
    auth_pass: ansible
    name: "{{ item.id }}"
    description: "{{ item.desc }}"

  with_items:
  - { id: "{{ id_s6_0 }}", desc: "{{ desc_s6_0 }}" }
  - { id: "{{ id_s6_1 }}", desc: "{{ desc_s6_1 }}" }

- name: Configure serial 6/0 & 6/1
  ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
```

```

- ip address {{ item.ip }} {{ item.mask }}
  parents:
  - interface {{ item.id }}
  after: no shutdown
with_items:
- { id: "{{ id_s6_1 }}", ip: "{{ ip_r_norte_s6_1 }}", mask: "{{ mask_r_norte_s6_1 }}" }

- name: Enable Clock Rate Serial 6/1
  ios_config:
    authorize: yes
    auth_pass: ansible
    lines:
    - clock rate 128000
  parents:
  - interface Serial 6/1

- name: Save configuration
  ios_command:
    authorize: yes
    auth_pass: ansible
    commands:
    - write memory

```

Listado 47: config_interfaces.yml.

```

---
- name: Enable Ospf on r-norte
  - include_vars: ./vars/main.yml
  - name: Enable OSPF on R-norte
    ios_config:
      authorize: yes
      auth_pass: ansible
      lines:
      - network {{ ip_lanS6_1 }} {{ wildmask_lanS6_1 }} area {{ area }}
    parents:
    - router ospf 1

```

Listado 48: config_ospf.yml.

Para este rol también existe un fichero con variables definidas para poder modificarlas cuando se desee sin necesidad de modificar el Playbook y dicho fichero contiene las siguientes líneas:

```

---
# vars file for config_r_norte

## Ip for Serial 6/0 ##

```

```

desc_s6_0: Connection to r-oeste
id_s6_0: Serial 6/0
ip_r_norte_s6_0: 172.16.1.2
mask_r_norte_s6_0: 255.255.255.252

## Ip for Serial 6/1 ##
desc_s6_1: Connection to r-este
id_s6_1: Serial 6/1
ip_r_norte_s6_1: 172.16.2.2
mask_r_norte_s6_1: 255.255.255.252
ip_lanS6_1: 172.16.2.0
wildmask_lanS6_1: 0.0.0.3

```

Listado 49: vars/main.yml.

config_r-este

Este Router está creado pensando en ampliar la red y para crear una conexión directa con Internet a través de él. Dado que no tiene subredes definidas no tiene ningún Playbook creado. Para el presente proyecto tiene la siguiente estructura:

```

[administrador@Admin:~/Training/roles/config_r-este$ tree
.
|-- README.md
|-- defaults
|   '-- main.yml
|-- handlers
|   '-- main.yml
|-- meta
|   '-- main.yml
|-- tasks
|   '-- main.yml
|-- tests
|   '-- test.yml
`-- vars
    '-- main.yml

6 directories, 7 files

```

Figura 44: Estructura del rol config_r-este.

Para acabar con la Implementación para utilizar Ansible desde CLI queda crear el archivo *site.yml*, que tienen todos los proyectos de Ansible en la carpeta raíz, para ejecutar todos los Playbooks de manera automática ya que es el objetivo de este TFG. Si se desea ejecutar una cierta configuración entonces se puede lanzar un Playbook específico. El archivo *site.yml* contiene las siguientes líneas:

```

---
- name: Configurar Switch Este
  hosts: sw-este
  gather_facts: false
  connection: local

  roles:
    - config_sw_este

- name: Configurar Switch Oeste
  hosts: sw-oeste
  gather_facts: false
  connection: local

  roles:
    - config_sw_oeste

- name: Configurar Router Norte
  hosts: r-oeste
  gather_facts: false
  connection: local

  roles:
    - config_r_oeste

- name: Configurar Router Norte
  hosts: r-norte
  gather_facts: false
  connection: local

  roles:
    - config_r_norte

- name: Realizar copia de seguridad
  hosts: all
  gather_facts: false
  connection: local

  roles:
    - Backup

```

Listado 50: site.yml.

Para este TFG, este sería el único archivo que habría que ejecutar para realizar toda la configuración y que los equipos se comuniquen entre ellos.

6.4.3. Implementación para utilizar Ansible desde AWX

Esta manera de ejecutar Ansible es mucho más gráfica y muy útil para llevar un buen control de las tareas que se llevan a cabo.

Dado que ya se ha creado un proyecto en el apartado anterior desde CLI ahora todo es mucho más fácil. Para crear el proyecto en AWX se deben copiar los enlaces del repositorio en una casilla que se verá a continuación y para ello se recuerdan las url's disponibles:

```
https://gitlab.com/amicre/Training.git  
https://github.com/utercnila/TFG.git
```

Listado 51: URL's de los repositorios.

Para esta implementación se necesita Docker, VMWare, GNS3 y AWX.

Primero de todo se debe descargar e instalar Docker y AWX. Como se comentó anteriormente, Docker es una herramienta gratuita y desde la página web oficial <https://www.docker.com> se puede descargar. AWX se tiene que instalar desde CLI clonando un repositorio y posteriormente ejecutando un Playbook para instalarlo.

El repositorio para clonar AWX es <https://github.com/ansible/awx.git>, en esa misma página hay instrucciones para utilizar AWX e instalarlo en distintos dispositivos.

Para este proyecto el proceso de instalación es muy sencillo, solamente hay que ejecutar los siguientes comandos:

```
1 - git clone https://github.com/ansible/awx.git  
2 - ansible-playbook -i inventory awx/installer/install.yml  
3 - docker logs -f awx_task
```

Listado 52: Instalacion de AWX.

Una vez Instalado Docker y AWX, se debe ejecutar Docker y cuando inicie hay que acceder a la página de AWX *localhost* en el navegador.

Por defecto, AWX solamente tiene el usuario *admin / password* pero desde el dashboard se le pueden añadir más, así que es lo primero que se debe hacer para seguir con los mismos casos de uso.

Por lo tanto, los pasos a seguir para configurar AWX son los siguientes:

- 1º *Crear el usuario adicional.* Esto es necesario para cumplir con los casos de uso. En la Figura 45 se puede observar la pantalla donde se realizaría este cambio.

Figura 45: Crear usuario en AWX.

2º *Crear las credenciales de cada máquina.* Como Ansible se conecta por SSH a los dispositivos remotos, se debe crear unas credenciales para tener acceso a dicho dispositivos. Esto se puede hacer copiando la clave ssh o directamente colocando el usuario y la contraseña como se puede ver a continuación 46:

Figura 46: Crear credenciales para los dispositivos.

Como se puede observar en la imagen hay posibilidad de copiar la clave o directamente con el usuario. Este paso se debe realizar para todos los dispositivos de la red.

Dado que a la hora de importar el proyecto en AWX se realizará desde GitHub se debe crear una credencial para ello, para que pueda acceder AWX a GitHub e importar el proyecto.

3º *Crear el archivo de inventario.* Esta opción es fundamental, cuando se configura Ansible desde CLI es importante crear este fichero y en AWX también es importante crearlo ya que si no AWX no sabría hacia dónde dirigirse. Para ello en la parte de inventario se crea un inventario y después se añaden las máquinas como se puede observar en 47 48:

Figura 47: Crear inventario.

Figura 48: Crear máquinas en inventario.

4º *Crear o importar el proyecto.* Éste sería el último paso de la configuración de AWX, aunque es uno de los más importantes porque sin Playbooks Ansible no sabría qué hacer. Para crear el proyecto o importarlo desde GitHub, como se ha hecho en este proyecto, hay que realizarlo desde la parte de “Proyectos” y el resultado sería parecido a 49:

The screenshot shows the AWX web interface. On the left is a sidebar with navigation links: NEWS, DASHBOARD, TRABAJOS, PROGRAMACIONES, PORTAL MODE, RESOURCES, PLANTILLAS, CREDENCIALES, PROYECTOS, INVENTARIOS, SCRIPTS DE INVENTARIO, ACCESS, ORGANIZACIONES, USUARIOS, EQUIPOS, ADMINISTRATION, TIPOS DE CREDENCIAL, NOTIFICACIÓN, TRABAJOS DE GESTIÓN, GRUPOS DE DISPOSITIVOS, and APPLICATIONS. The main area is titled 'PROYECTOS / CREAR PROYECTO'. It has tabs for 'DETALLES', 'PERMISOS', 'NOTIFICACIÓN', 'PLANTILLAS DE TRABAJO', and 'PROGRAMACIONES'. Under 'DETALLES', there are fields for 'NOMBRE' (TFG), 'DESCRIPCIÓN', 'TIPO SCM' (GIT), and 'ORGANIZACIÓN' (Default). Below this is a section for 'DETALLES DE LA FUENTE' with 'SCM URL' (https://github.com/uternaria/TFG.git), 'CONSIGNA/ETIQUETA/RAMA SCM' (empty), and 'SCM CREDENTIAL' (Github). There are also options for 'OPCIONES DE ACTUALIZACIÓN SCM' (Clean, Delete, or Update Revision on Launch) and a 'TIEMPO DE ESPERA PARA LA EXPIRACIÓN DE LA CÁCHÉ (SEGUNDOS)' (0). At the bottom of this section are 'CANCELAR' and 'GUARDAR' buttons. Below this is a table titled 'PROYECTOS' showing one entry: 'Demo Project' (TIPO: GIT). The table includes columns for 'NOMBRE', 'TIPO', 'REVISIÓN', 'ÚLTIMA ACTUALIZACIÓN', and 'ACCIONES' (with icons for edit, delete, etc.). A message at the bottom right says 'ELEMENTOS 1 - 1'.

Figura 49: Importar proyecto de GitHub.

Para este TFG se decidió guardar el proyecto en GitLab desde el inicio, pero a la hora de utilizar AWX surgieron problemas. Cuando se realizaba la importación del proyecto desde GitLab no se importaba correctamente. Tras varias pruebas se decidió probar con GitHub y funcionó correctamente. Como consejo, es preferible utilizar GitHub para importar un proyecto en AWX.

Para acabar con la configuración de AWX y la fase de implantación se debe comprobar, como se hizo en la *Implementación para utilizar Ansible desde CLI*, que hay conexión con los dispositivos.

Para realizar esta comprobación se ejecutará desde AWX un Playbook sencillo para ver la respuesta de la máquina. A continuación, se puede comprobar que el resultado ha sido el deseado 50:

```

aa@12:44:35
PLAY [Configurar Switch Este] *****
TASK [show vlan] *****
ok: [sw-este]
TASK [debug] *****
ok: [sw-este] => {
    "ah_vlan.stdout_lines": [
        {
            "VLAN Name": "default",
            "Status": "active",
            "Ports": "G10/1, G10/2, G10/3, G11/0, G11/1, G11/2, G11/3, G12/0, G12/1, G12/2, G12/3, G13/0, G13/1, G13/2",
            "Priority": "0"
        },
        {
            "VLAN Name": "23 Interneia",
            "Status": "active",
            "Ports": "G13/3",
            "Priority": "23"
        },
        {
            "VLAN Name": "99 Management",
            "Status": "active",
            "Ports": "G10/0",
            "Priority": "99"
        }
    ]
}
PLAY RECAP *****
: ok=2    changed=0   unreachable=0   failed=0

```

Figura 50: Comprobación de la conexión con Sw-este.

Además de aplicar AWX sobre la topología creada en GNS3 se realizará una prueba de la automatización sobre dispositivos físicos. Se utilizará la misma configuración que la descrita anteriormente. Desde el ordenador Macbook se lanzará AWX y se conectará a los distintos dispositivos disponibles. A continuación, se puede ver cómo serían todas las conexiones y lo que se ha utilizado para esta prueba 51:

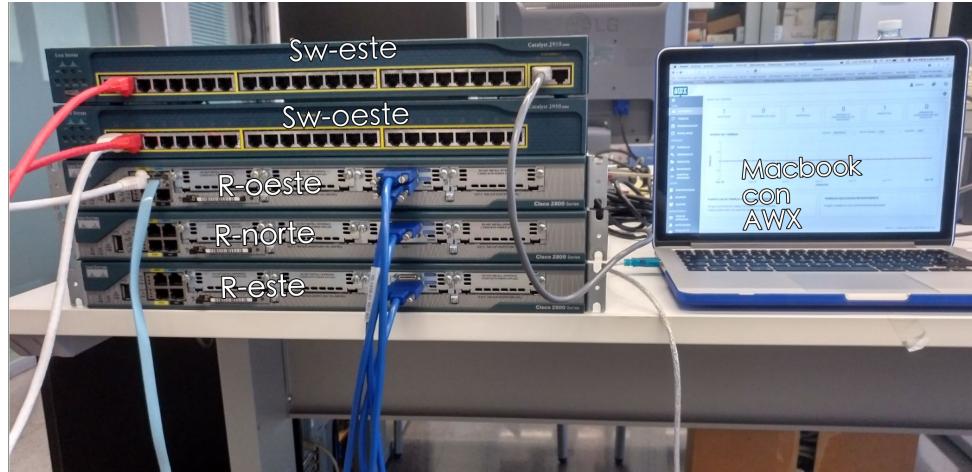


Figura 51: Dispositivos físicos.

7. Pruebas y resultados

Es necesaria la realización de un conjunto de pruebas para asegurar la funcionalidad de la red una vez realizada la automatización mediante Ansible y verificar así que se cumplen todos los objetivos anteriormente especificados.

Las pruebas se van a realizar de manera individual, es decir comprobar todos los Playbooks de manera individual y también de forma conjunta.

En este proyecto se han realizado las siguientes pruebas:

- *Pruebas unitarias*: Este tipo de pruebas consiste en lanzar los Playbooks a los dispositivos de la red de manera individual y comprobar su funcionamiento. Esto se realizará desde la máquina *Admin* y a través de CLI. Si apareciesen errores en los Playbooks se volvería a editar y corregir dicho Playbook.
- *Pruebas de integración*: Una vez finalizadas las pruebas unitarias, mediante AWX se ejecutaría el Playbook maestro *site.yml* integrando todos los roles y ejecutándolos todos a la vez.
- *Pruebas del sistema*: Por último, aunque no menos importante son las pruebas finales del sistema. Se comprobaría que los dispositivos se han configurado correctamente de manera automática y existe conexión entre ellos.

7.1. Pruebas unitarias

Estas pruebas son fundamentales para comprobar que se han creado correctamente los Playbooks. Se ejecutan los Playbooks desde la máquina *Admin* para cada rol.

De este modo se consigue averiguar si hay algún error sintáctico o de algún módulo interno y se puede solucionar. Estas pruebas se ejecutan analizando de manera aislada e individual cada uno de los Playbooks que se han creado durante el proyecto.

La realización de estas pruebas se ejecuta desde el mismo instante que se crea el primer Playbook.

Ansible dispone de una opción muy útil llamada *Check Mode ("Dry Run")*. Si se ejecuta un playbook con la opción *-check* después de todos los parámetros, Ansible ejecuta el Playbook pero no realiza ningún cambio en la máquina remota.

Para realizar la primera prueba se va a comprobar el si el primer Playbook se ha creado correctamente. Esto se realizaría con el comando:

```
ansible-playbook roles/Backup/tasks/show_run.yml --check
```

Listado 53: Comprobar Playbook.

Una vez ejecutado el comando aparece un mensaje como el siguiente 52:

```

root@Admin:~/TFG# ansible-playbook roles/Backup/tasks/show_run.yml --check
ERROR! Syntax Error while loading YAML.

The error appears to have been in '/root/TFG/roles/Backup/tasks/show_run.yml': line 10, column 3, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

register: print
- debug: var=print.stdout_lines
^ here

exception type: <class 'yaml.parser.ParserError'>
exception: while parsing a block mapping
  in "<unicode string>", line 2, column 3
did not find expected key
  in "<unicode string>", line 10, column 3

```

Figura 52: Error.

Esto quiere decir que hay un problema de sintaxis en el Playbook. Como se explicó en el Estado del Arte, los Playbooks de Ansible están escritos en Python por lo tanto los espacios son muy importantes cuando se escribe código y esto es lo que ha fallado en este caso.

Una vez corregido y ejecutado con el mismo comando da como resultado la siguiente Figura 53:

```

SSH password:
PLAY [Backup] *****
TASK [Gathering Facts] *****
ok: [sw-este]

TASK [Show devices' startup-config] *****
  [WARNING]: argument auth_pass has been deprecated and will be removed in a
future version

ok: [sw-este]

TASK [debug] *****
ok: [sw-este] => {
    "print.stdout_lines": [
        [
            ""
        ]
    ]
}

TASK [Save configuration to a file] *****
changed: [sw-este]

PLAY RECAP *****
sw-este : ok=4    changed=1    unreachable=0    failed=0

root@Admin:~/TFG# 

```

Figura 53: Correcto.

Se puede comprobar que todo ha tenido éxito y que ahora si se quitase el parámetro `-check` funcionaría perfectamente.

Para comprobar todos los Playbooks de todos los roles existentes en este proyecto habría que ejecutar los siguientes comandos:

```

# Comprobaciones para el Switch Este
ansible-playbook -i hosts roles/config_sw_este/tasks/create_vlan.
    yml --check
ansible-playbook -i hosts roles/config_sw_este/tasks/config_vlan.
    yml --check
# Comprobaciones para el Switch Oeste
ansible-playbook -i hosts roles/config_sw_oeste/tasks/create_vlan
    .yml --check
ansible-playbook -i hosts roles/config_sw_oeste/tasks/config_vlan
    .yml --check
# Comprobaciones para el Router Oeste
ansible-playbook -i hosts roles/config_r_oeste/tasks/
    config_interfaces.yml --check
ansible-playbook -i hosts roles/config_r_oeste/tasks/
    config_subinterfaces.yml --check
ansible-playbook -i hosts roles/config_r_oeste/tasks/config_ospf.
    yml --check
ansible-playbook -i hosts roles/config_r_oeste/tasks/config_dhcp .
    yml --check
# Comprobaciones para el Router Norte
ansible-playbook -i hosts roles/config_r_norte/tasks/
    config_interfaces.yml --check
ansible-playbook -i hosts roles/config_r_norte/tasks/config_ospf .
    yml --check

```

Listado 54: Comprobar Playbooks.

Además de realizar las pruebas para cada Playbook creado en el proyecto, se deben realizar pruebas en cada dispositivo de la red para comprobar si la configuración básica que se ha aplicado manualmente funciona correctamente.

Una de las pruebas que se podría realizar sería un ping a su propia ip para asegurarse que la tiene bien asignada. *ping 192.168.99.11* esto serviría para el Switch Oeste.

En la Figura 54 se observar un ejemplo de que la configuración básica se ha aplicado a uno de los dispositivos:

```

[sw-oeste>ping 192.168.99.11
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.99.11, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/6 ms

```

Figura 54: Comprobación de la configuración.

7.2. Pruebas de integración

Una vez realizadas las pruebas unitarias pasan a realizarse las pruebas de integración. Estas pruebas de integración sirven para aplicar la configuración y utilizar Ansible para

realizar la automatización tanto en los dispositivos virtuales creados en GNS3 como en los dispositivos físicos.

7.2.1. Pruebas desde la máquina Admin

Las pruebas de integración se realizarán primero desde la CLI de la máquina *Admin*.

Como ya se ha comprobado en la parte de pruebas unitarias que la configuración básica se ha aplicado correctamente a los dispositivos de la red ahora se puede realizar la automatización del resto de configuraciones mediante Ansible. Antes de todo, se ha guardado una copia de la configuración inicial de cada dispositivo para así después poder hacer la comparativa.

Desde la máquina *Admin* se debe ejecutar el archivo *site.yml* para aplicar todos los roles. Para ejecutar dicho Playbook maestro se debe utilizar el siguiente comando:

```
ansible-playbook -i hosts site.yml -k -u ansible
```

Listado 55: Comprobar Playbook.

Al ejecutar el comando anterior Ansible comienza la comunicación con los dispositivos, pero de manera secuencial. Por lo tanto, el primer rol a ejecutar es el de *config_sw_este*, seguido de *config_sw_oeste*, *config_r_oeste*, *config_r_norte* y finalmente el de *Backup*.

En las Figuras 55, 56, 57, 58, 59 se puede observar la ejecución de esta herramienta de automatización aplicando los roles creados y en la Figura 60 se puede comprobar que todo se ha ejecutado correctamente.

En la siguiente Figura 55 se puede observar cómo se han creado VLAN's en el SW-ESTE y se muestra la información de las VLAN's

```
PLAY [Configurar Switch Este] ****
TASK [config_sw_este : create VLAN] ****
changed: [sw-este] => (item={'id': 21, 'name': 'Contabilidad'})
changed: [sw-este] => (item={'id': 23, 'name': 'Ingenieria'})
[WARNING]: argument auth_pass has been deprecated and will be removed in a
future version

TASK [config_sw_este : show vlan] ****
ok: [sw-este]

TASK [config_sw_este : debug] ****
ok: [sw-este] => {
    "sh_vlan.stdout_lines": [
        [
            "VLAN Name                  Status Ports",
            "----",
            "1  default                 active  Gi0/1, Gi0/2, Gi0/3,
Gi2/0, Gi2/1, Gi2/2, Gi2/3, Gi3/0, Gi3/1, Gi3/2",
            "21 Contabilidad           active  ",
            "23 Ingenieria             active  Gi3/3",
        ]
    ]
}
```

Figura 55: Sw-Este.

Esta Figura 56 muestra cómo se han creado las VLAN's y su información, pero para el SW-OESTE.

```
PLAY [Configurar Switch Oeste] ****
TASK [config_sw_oeste : create VLAN] ****
changed: [sw-oeste] => (item={'id': 21, 'name': 'Contabilidad'})
changed: [sw-oeste] => (item={'id': 23, 'name': 'Ingenieria'})

TASK [config_sw_oeste : show vlan] ****
ok: [sw-oeste]

TASK [config_sw_oeste : debug] ****
ok: [sw-oeste] => {
    "sh_vlan.stdout_lines": [
        [
            "VLAN Name                               Status Ports",
            "---- -----",
            "1  default                             active Gi0/1, Gi0/2, Gi0/3,
Gi2/0, Gi2/1, Gi2/2, Gi2/3, Gi3/0, Gi3/1, Gi3/2",
            "21 Contabilidad                        active Gi3/3",
            "23 Ingenieria                          active ",
            "99 Management                          active ",
            "1002 fddi-default                      act/unsup ",
            "1003 token-ring-default                act/unsup "
        ]
    ]
}
```

Figura 56: Sw-Oeste.

En la Figura 57 se muestra como se está configurando las subinterfaces del R-OESTE.

```
PLAY [Configurar Router Norte] ****
TASK [config_r_oeste : include_vars] ****
ok: [r-oeste]

TASK [config_r_oeste : Configure subinterfaces for VLAN 21 & 23] ****
changed: [r-oeste] => (item={'ip': '192.168.21.1', 'id': 21})
changed: [r-oeste] => (item={'ip': '192.168.23.1', 'id': 23})

TASK [config_r_oeste : Save configuration] ****
ok: [r-oeste]

TASK [config_r_oeste : include_vars] ****
ok: [r-oeste]

TASK [config_r_oeste : Enable serial 6/0] ****
changed: [r-oeste]
```

Figura 57: R-Oeste.

En la Figura 58 se puede observar que están cambiando las configuraciones de las interfaces serial del R-NORTE

```
PLAY [Configurar Router Norte] ****
TASK [config_r_norte : include_vars] ****
ok: [r-norte]

TASK [config_r_norte : Enable serial 6/0 & 6/1] ****
changed: [r-norte] => (item={'id': u'Serial 6/0', 'desc': u'Connection to r-oeste'})
changed: [r-norte] => (item={'id': u'Serial 6/1', 'desc': u'Connection to r-este'})

TASK [config_r_norte : Configure serial 6/0 & 6/1] ****
changed: [r-norte] => (item={'ip': u'172.16.2.2', 'mask': u'255.255.255.252', 'id': 'Serial 6/0'})

TASK [config_r_norte : Enable Clock Rate Serial 6/1] ****
changed: [r-norte]

TASK [config_r_norte : Save configuration] ****
ok: [r-norte]

TASK [config_r_norte : include_vars] ****
ok: [r-norte]
```

Figura 58: R-Norte.

En la Figura 59 se puede apreciar que se han guardado las configuraciones de tres de los cuatro dispositivos disponibles ya que de sw-este ya estaba guardada.

```
TASK [Backup : Save configuration to a file] ****
changed: [r-oeste]
changed: [r-norte]
changed: [sw-oeste]
ok: [sw-este]
```

Figura 59: Backup.

En esta última Figura 60 se puede comprobar que la ejecución de Ansible ha finalizado correctamente y sin ningún fallo.

```
PLAY RECAP ****
r-norte          : ok=10  changed=5    unreachable=0    failed=0
r-oeste          : ok=18  changed=7    unreachable=0    failed=0
sw-este          : ok=15  changed=6    unreachable=0    failed=0
sw-oeste         : ok=15  changed=7    unreachable=0    failed=0
```

Figura 60: Resultado de la ejecución.

7.2.2. Pruebas desde AWX

Para finalizar con este apartado de pruebas de integración se va a realizar la ejecución de Ansible y la automatización de la red desde AWX. Esta opción permite a los usuarios ejecutar los Playbooks de manera más guiada y de fácil manejo.

Para realizar estas pruebas se debe crear una plantilla de trabajo ya que y el proyecto todas las configuraciones se han realizado en la parte de implementación. Además, en nuestra software de virtualización, GNS3 se va a proceder a colocar de nuevo los dispositivos para que solamente tengan la configuración básica y que no hayan problemas a la hora de ejecutar nuevamente Ansible.

Antes de crear la plantilla de trabajo se deben definir todos los dispositivos de la red en el inventario de AWX y el resultado final sería 61:

| SERVIDORES | GRUPOS RELACIONADOS | ACTIONES |
|-------------------------------------|---------------------|----------|
| <input type="checkbox"/> | n-este | |
| <input type="checkbox"/> | r-norte | |
| <input type="checkbox"/> | n-oeste | |
| <input checked="" type="checkbox"/> | sw-este | |
| <input type="checkbox"/> | sw-este | |

Figura 61: inventario completo.

Para crear una plantilla de trabajo es muy sencillo y en la Figura 62 se puede ver como llenar los campos:

AWX

PLANTILLAS / CREAR PLANTILLA DE TRABAJO

NEVA PLANTILLA DE TRABAJO

DETALLES **VERMOSOS** **NOTIFICACIÓN** **TAREAS COMPLETADAS** **PROGRAMACIONES** **AÚN ENCUENTA**

| | | |
|--|---|--|
| NOMBRE TGF | DESCRIPCIÓN | TIPO DE TRABAJO <input checked="" type="checkbox"/> Ejecutar <input type="checkbox"/> PREGUNTAR AL EJECUTAR |
| INVENTARIO <input checked="" type="checkbox"/> TGF <input type="checkbox"/> PREGUNTAR AL EJECUTAR | PROYECTO <input checked="" type="checkbox"/> TGF <input type="checkbox"/> PLAYBOOK | LIMITES <input type="checkbox"/> PREGUNTAR AL EJECUTAR |
| CREDENCIAL <input checked="" type="checkbox"/> TGF <input type="checkbox"/> PREGUNTAR AL EJECUTAR | FORMS <input checked="" type="checkbox"/> DEFAULT <input type="checkbox"/> PREGUNTAR AL EJECUTAR | OMITIR ETIQUETAS <input type="checkbox"/> PREGUNTAR AL EJECUTAR |
| NIVEL DE DETALLE <input checked="" type="checkbox"/> (Normal) <input type="checkbox"/> PREGUNTAR AL EJECUTAR | ETIQUETAS DE TRABAJO <input type="checkbox"/> PREGUNTAR AL EJECUTAR | MOSTRAR CAMBIOS <input checked="" type="checkbox"/> DIF <input type="checkbox"/> PREGUNTAR AL EJECUTAR |
| ETIQUETAS | GRUPOS DE INSTANCIAS <input type="checkbox"/> | |
| Opciones | | |
| <input checked="" type="checkbox"/> Activar la elevación de privilegios <input type="checkbox"/> <input type="checkbox"/> Permitir la ejecución utilizando Callbacks <input type="checkbox"/> <input type="checkbox"/> Activar los trabajos concurrentes <input type="checkbox"/> <input type="checkbox"/> Usar Cache de eventos <input type="checkbox"/> | | |
| VARIABLES ADICIONALES <input type="checkbox"/> NAME <input type="checkbox"/> JSON | | |
| <pre>1</pre> <input type="button" value="CANCELAR"/> <input type="button" value="GUARDAR"/> | | |

Figura 62: Creación de la plantilla de trabajo.

Hay que destacar que en el campo de credenciales se ha usado *sw-este* ya que como es un entorno de laboratorio, todos los dispositivos tienen el mismo usuario y contraseña.

Desde la misma pestaña de *Plantilla* se puede lanzar la plantilla de trabajo creada anteriormente. Se puede observar en la Figura 63 el botón en forma de cohete desde el que se puede lanzar:

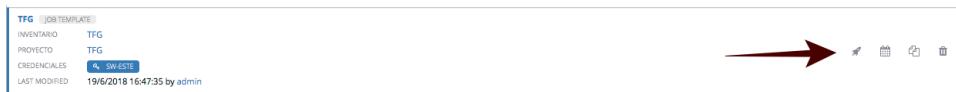


Figura 63: Ejecución de la plantilla de trabajo.

AWX empieza la ejecución de la plantilla y se puede seguir la ejecución desde la pestaña de *Trabajos*

Figura 64: Plantilla ejecutandose.

Cuando empieza a ejecutar los roles se puede apreciar que la salida es igual que cuando se ejecuta desde CLI.

```

1 SSH password:
2
3 PLAY [Configurar Switch Este] ****
4
5 TASK [config_sw_este : create VLAN] ****
6 changed: [sw-este] => (item={u'id': 21, u'name': u'Contabilidad'})
7 changed: [sw-este] => (item={u'id': 23, u'name': u'Ingenieria'})
8
9 TASK [config_sw_este : show vlan] ****
10 ok: [sw-este]

```

Figura 65: Ejecución del rol config_sw_este.

```
220 PLAY [Configurar Switch Oeste] ****
221
222 TASK [config_sw_oeste : create VLAN] ****
223 changed: [sw-oeste] => (item={u'id': 21, u'name': u'Contabilidad'})
224 changed: [sw-oeste] => (item={u'id': 23, u'name': u'Ingenieria'})
225
226 TASK [config_sw_oeste : show vlan] ****
227 ok: [sw-oeste]
228
```

Figura 66: Ejecución del rol config_sw_oeste.

Una vez realizada la prueba sobre los dispositivos virtuales de GNS3 se realizará la misma prueba desde AWX sobre los dispositivos reales. Para realizar la prueba se debe conectar físicamente el ordenador orquestador, el Macbook, al primer switch, en este caso al Switch Este como se puede ver en la Figura 51. Una vez realizada la conexión y haber ejecutado las configuraciones iniciales básicas en los cinco dispositivos de red físicos, se debe ejecutar de nuevo la plantilla creada.

Para aplicar las configuraciones a los routers y a los switches se debe conectar el Macbook con un cable consola a cada dispositivo y mediante un programa que simula la consola de los dispositivos, se puede configurar.

Una vez ejecutada la plantilla, el proceso es el mismo que cuando se configuraban los dispositivos virtuales en GNS3. En la Figura 67, 68, 69, 70, 71se puede apreciar la ejecución de la plantilla y resultado final indicando el correcto funcionamiento.

En la siguiente Figura 67 se puede observar cómo se está ejecutando el rol para SW-ESTE y mientras cambian configuraciones en AWX también cambian en la pantalla azul que simula la consola del dispositivo físico.

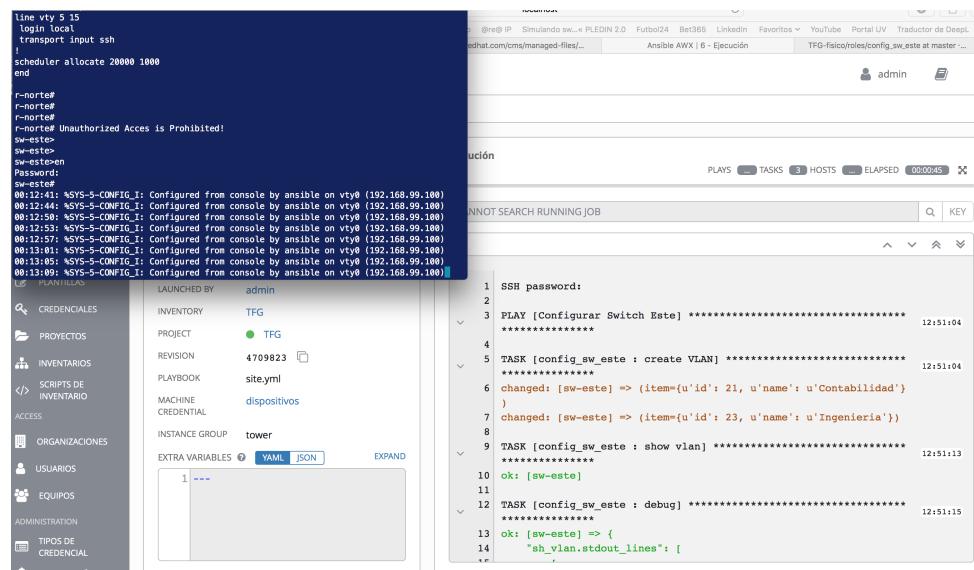


Figura 67: Ejecución de la plantilla para configurar dispositivos físicos.

En la siguiente Figura 68 se puede observar cómo se está ejecutando el rol para SW-OESTE y mientras cambian configuraciones de las VLAN's en AWX también cambian en la pantalla azul que simula la consola del dispositivo físico.

```

Serial File Edit View Terminal Window Help
USB-Serial Controller — 8Bx24 — 9600.8.N1
Password:
sw-este>
00:12:41: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:12:44: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:12:50: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:12:54: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:12:57: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:01: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:05: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:09: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:13: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100) Unauthorized Access is Prohibited!
sw-este>
sw-este<
00:13:32: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:35: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:38: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:43: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
  
```

LAUNCHED BY admin
INVENTORY TFG
PROJECT ● TFG
REVISION 4709823
PLAYBOOK site.yml
MACHINE dispositivos
CREDENTIAL
INSTANCE GROUP tower
EXTRA VARIABLES YAML JSON EXPAND

localhost Lun 12:52:03 85% Q Alin Mihai Cretu Brinza >

@req IP Simulado sw-este PLEDIN 2.0 Futbol24 Bet365 LinkedIn Favoritos YouTube Portal UV Traductor de Deep... Ansible AWX | 6 - Ejecución TFG-fisico/roles/config_sw_este at master... +

admin

PLAYS TASKS HOSTS ELAPSED 00:01:08

NOT FOUND SEARCH RUNNING JOB Q KEY

85 |—
...
198
199 TASK [config_sw_este : save running to startup when modified] *** 12:51:41

200 ok: [sw-este]
201
202 PLAY [Configurar Switch Oeste] ***** 12:51:45

203
204 TASK [config_sw_este : create VLAN] ***** 12:51:45

205 changed: [sw-este] => (item={'id': 21, 'name': 'Contabilidad'})
206 changed: [sw-este] => (item={'id': 23, 'name': 'Ingenieria'})
207
208 TASK [config_sw_este : show vlan] ***** 12:51:53

209 ok: [sw-este]
210
211 TASK [config_sw_este : debug] ***** 12:51:53

About | Copyright © 2018 Red Hat, Inc.

Figura 68: Ejecución de la plantilla para configurar dispositivos físicos.

En la siguiente Figura 69 se puede observar cómo se está ejecutando el rol para R-OESTE modificando las configuraciones del Serial 0/1/0 y a su vez se modifican en el dispositivo físico.

```

Serial File Edit View Terminal Window Help
USB-Serial Controller — 8Bx24 — 9600.8.N1
Password:
sw-este>
00:13:32: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:35: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:40: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:43: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
r-este>
r-este<
00:13:32: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:35: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:40: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
00:13:43: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
r-este>
r-este<
*Jan 1 00:07:04.455: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
*Jan 1 00:07:07.383: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
*Jan 1 00:07:15.575: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
*Jan 1 00:07:18.675: %SYS-5-CONFIG_I: Configured from console by ansible on vty0 (192.168.99.100)
*Jan 1 00:07:20.675: %LINK-3-UPDOWN: Interface Serial0/1/0, changed state to up
*Jan 1 00:07:21.787: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1/1, changed state to up
  
```

LAUNCHED BY admin
INVENTORY TFG
PROJECT ● TFG
REVISION 4709823
PLAYBOOK site.yml
MACHINE dispositivos
CREDENTIAL
INSTANCE GROUP tower
EXTRA VARIABLES YAML JSON EXPAND

localhost Lun 12:53:46 84% Q Alin Mihai Cretu Brinza >

@req IP Simulado sw-este PLEDIN 2.0 Futbol24 Bet365 LinkedIn Favoritos YouTube Portal UV Traductor de Deep... Ansible AWX | 6 - Ejecución TFG-fisico/roles/config_sw_este at master... +

admin

PLAYS TASKS HOSTS ELAPSED 00:02:28

SEARCH Q KEY

405 TASK [config_r_este : Enable OSPF on R-este] ***** 12:52:43

406 changed: [r-este]
407
408 TASK [config_r_este : Save configuration] ***** 12:52:45

409 ok: [r-este]
410
411 TASK [config_r_este : Reserve the first 20 ip addresses in VLAN 21 & 23 for static config] *** 12:52:51
412 changed: [r-este] => (item={'ip_inicio': '192.168.21.1', 'ip_fin': '192.168.21.20'})
413 changed: [r-este] => (item={'ip_inicio': '192.168.23.1', 'ip_fin': '192.168.23.20'})
414
415 TASK [config_r_este : Create the DHCP pools for VLAN 21 & 23] ** 12:52:55

416 changed: [r-este] => (item={'ip_dns': '10.10.10.10', 'ip_vlan': '192.168.21.0', 'name': 'Contabilidad', 'ip_default': '192.168.21.1'})
417 changed: [r-este] => (item={'ip_dns': '10.10.10.10', 'ip_vlan': BACK TO TOP}

About | Copyright © 2018 Red Hat, Inc.

Figura 69: Ejecución de la plantilla para configurar dispositivos físicos.

En la siguiente Figura 70 se puede observar cómo se está ejecutando el rol para R-NORTE y mientras cambian configuraciones de la interfaz Serial, en AWX, también cambian en la pantalla azul que simula la consola del dispositivo físico.

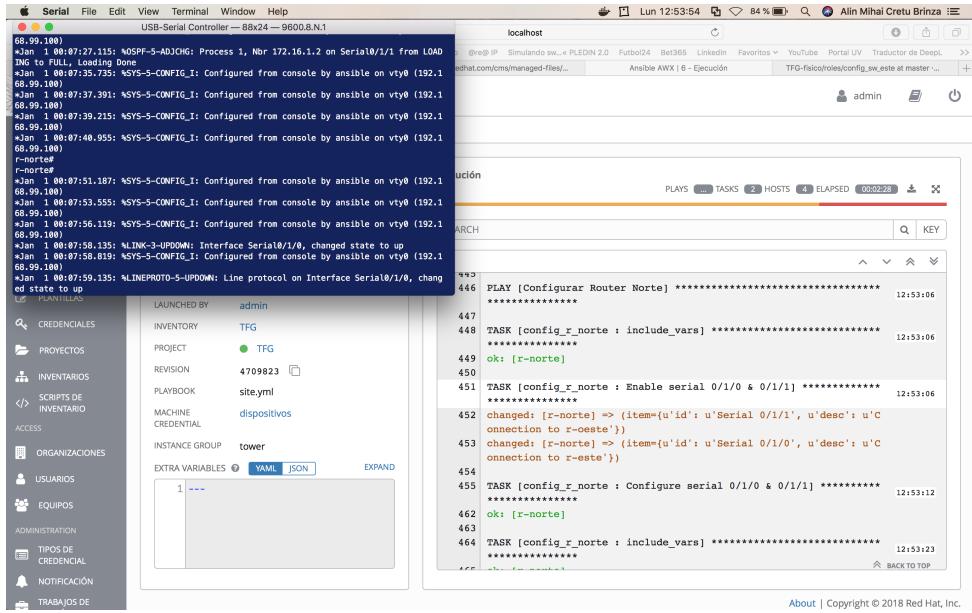


Figura 70: Ejecución de la plantilla para configurar dispositivos físicos.

Por último, en la siguiente Figura 71 se pueden observar los resultados de haber ejecutado Ansible desde AWX sobre los dispositivos físicos, siendo el resultado un éxito.

```
PLAY RECAP ****
*****
r-este : ok=2    changed=0    unreachable=0
failed=0
r-norte : ok=9    changed=3    unreachable=0
failed=0
r-oeste : ok=17   changed=4    unreachable=0
failed=0
sw-este : ok=13   changed=6    unreachable=0
failed=0
sw-oeste : ok=13   changed=6    unreachable=0
failed=0
```

Figura 71: Ejecución de la plantilla para configurar dispositivos físicos.

7.3. Pruebas del sistema

Por último, se realizan pruebas para comprobar que toda la red tiene conexión y que todos los dispositivos pueden conectarse entre sí.

Para la realización de estas pruebas se han realizado peticiones ping desde la máquina *Admin* hacia todos los dispositivos disponibles en la red y los que han sido configurados mediante Ansible.

Los resultados de las pruebas se pueden observar a continuación 72, 73, 74, 75 y 76:

```
PING sw-este (192.168.99.10) 56(84) bytes of data.  
64 bytes from sw-este (192.168.99.10): icmp_seq=1 ttl=255 time=8.44 ms  
^C  
--- sw-este ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 8.442/8.442/8.442/0.000 ms
```

Figura 72: Prueba de conexión.

```
PING sw-oeste (192.168.99.11) 56(84) bytes of data.  
64 bytes from sw-oeste (192.168.99.11): icmp_seq=1 ttl=255 time=9.53 ms  
^C  
--- sw-oeste ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 9.539/9.539/9.539/0.000 ms
```

Figura 73: Prueba de conexión.

```
PING r-oeste (192.168.99.1) 56(84) bytes of data.  
64 bytes from r-oeste (192.168.99.1): icmp_seq=1 ttl=255 time=18.4 ms  
^C  
--- r-oeste ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 18.447/18.447/18.447/0.000 ms
```

Figura 74: Prueba de conexión.

```
PING r-norte (172.16.1.2) 56(84) bytes of data.  
64 bytes from r-norte (172.16.1.2): icmp_seq=1 ttl=254 time=31.3 ms  
^C  
--- r-norte ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 31.352/31.352/31.352/0.000 ms
```

Figura 75: Prueba de conexión.

```
PING r-este (172.16.2.1) 56(84) bytes of data.  
64 bytes from r-este (172.16.2.1): icmp_seq=3 ttl=253 time=53.4 ms  
^C  
--- r-este ping statistics ---  
3 packets transmitted, 1 received, 66% packet loss, time 1999ms  
rtt min/avg/max/mdev = 53.464/53.464/53.464/0.000 ms
```

Figura 76: Prueba de conexión.

Con estas pruebas se asegura que se han cumplido los casos de uso y los requisitos especificados. Se ha podido realizar la automatización de la configuración de los dispositivos de red y poder así establecer conexión entre los dispositivos.

8. Conclusiones y Trabajo Futuro

8.1. Conclusiones

Una vez finalizado el proyecto, se puede decir que se han conseguido cumplir satisfactoriamente los objetivos planteados al inicio.

Con la realización de este TFG se ha conseguido automatizar las configuraciones de dispositivos CISCO ubicados en una red virtual, creada con GNS3, mediante la herramienta Ansible y poder así tener conexión entre los equipos de la red.

Además, se ha podido realizar la automatización de dispositivos físicos, demostrando la facilidad con la que se ha conseguido y lo útil que es incluir esta nueva modalidad de trabajo que en un futuro va a ser muy relevante ya que las redes aumentan de tamaño día tras día y configurar equipos individualmente supondrá una pérdida de tiempo y de dinero para la empresa.

Cabe destacar la facilidad con la que se ha llevado a cabo la automatización, una vez se tiene todo configurado. Dado que es un entorno de laboratorio han surgido problemas que mediante pruebas y búsquedas de información en foros, documentación de la herramienta, cursos y expertos en el tema se han conseguido superar sin problema.

Ansible es una herramienta OpenSource lo cual ayuda mucho a su desarrollo e implementación. Hoy en día hay una gran comunidad dispuesta a ayudar a cualquier persona y que trabaja para mejorar la herramienta.

Dado que Ansible es una herramienta bastante novedosa y que está en fase continua de mejora y de desarrollo, se pensó que su explotación podía ser bastante útil para el estudiante ya que la automatización de las redes está en auge.

Para la realización de este TFG se ha recibido el apoyo de una empresa de renombre, Red Hat, que ha estado siempre en contacto con el estudiante, ayudado en todo momento y proporcionado soluciones cuando se han necesitado.

Respecto al desarrollo del proyecto, se puede decir que la parte más complicada no ha sido la ejecución de Ansible sino la fase previa de estudio sobre la herramienta a utilizar, el diseño de la red y como se ha de implementar para cumplir con los requisitos especificados.

Se ha escogido la automatización, mediante Ansible, de dispositivos de red porque es un tema muy poco desarrollado en la nube. Aunque con esta herramienta se están automatizando máquinas con sistemas operativos Linux, que hacen el trabajo de servidores, para los dispositivos de red, como podrían ser Switches o Routers, a día de hoy se está utilizando cada vez más y también se está intentando implantar este tipo de tareas en todos los departamentos TIC.

A pesar de la falta de conocimiento por parte del alumno de la tecnología elegida para desarrollar el proyecto, durante la realización del proyecto se ha tenido la posibilidad de aprender de personas con experiencia en Ansible, por lo tanto, esto ha ayudado a agilizar el aprendizaje y su uso. Además de los conocimientos obtenidos sobre automatización y el uso de Ansible, se han obteniendo conocimientos de programación en Python, virtualización de redes y sistemas operativos y se han mejorado los conocimientos sobre configuración de dispositivos CISCO.

La parte más espesa durante la realización de este TFG ha sido la búsqueda de información y la realización de la memoria, que como se puede observar en la planificación

temporal ha tenido una duración de casi dos semanas, aunque se han redactado apartados durante el desarrollo del proyecto para facilitar el trabajo final.

En el aspecto temporal, la planificación inicial ha sido fundamental para la ejecución del proyecto. Dado que se ha trabajado en el TFG a jornada completa, 8 horas al día, se ha conseguido adelantar un día.

8.2. Trabajo Futuro

A pesar de que se han cumplido los objetivos, durante la implementación de este proyecto, han surgido algunas ideas para posibles mejoras y se han detectado aspectos a perfeccionar en cuanto a la creación de los Playbooks ya que se podrían optimizar un poco más. Por lo tanto, se considera que existe un trabajo futuro.

Como trabajos futuros se podrían incluir:

- *Optimización de los Playbooks:* Dado que Ansible ofrece la posibilidad de incluir cualquier archivo dentro de los Playbooks, esto permite reutilizar algunos Playbooks agrupándolos por tipo de dispositivo si comparten configuraciones. Para conseguir este resultado, Ansible tiene un módulo para recoger datos de los dispositivos y con esos datos, mediante módulos condicionales, aplicar configuraciones a un dispositivo u otro.
- *Crear proyectos más grandes:* Puesto que en este TFG se ha realizado toda la automatización en un entorno de laboratorio, se pretende aplicar esta herramienta a un entorno profesional.
- *Mejorar el uso de AWX:* Gran parte del trabajo se ha realizado desde CLI y como añadido extra se ha utilizado AWX. El próximo paso sería mejorar el uso y los conocimientos sobre AWX para aprovechar todas las mejoras que ofrece esta interfaz gráfica, como por ejemplo realizar Workflows.
- *Aplicación de Ansible Tower:* Red Hat, en su página oficial, ofrece la posibilidad a los usuarios de poder utilizar la herramienta de pago, Ansible Tower, durante un periodo de tiempo. Es un entorno más profesional por lo tanto es más completa que AWX.
- *Automatización de máquinas Windows:* Durante la búsqueda de información sobre Ansible, se han realizado ejercicios de práctica con esta herramienta sobre máquinas Linux para aprender su funcionamiento y después poder aplicarlo a dispositivos de red. Automatizar máquinas con sistema operativo Windows es un nuevo reto, aunque es muy similar a emplear Ansible para Linux.
- *Automatizar aplicaciones con Ansible:* OpenShift es una plataforma de aplicaciones de contenedores de código abierto que le pertenece a Red Hat, útil para el desarrollo, implementación y gestión de aplicaciones. Se pretende realizar automatizaciones de las aplicaciones desplegadas en dicha plataforma ya que Ansible tiene módulos específicos para ello.

Glosario

Ansible

Es un proyecto de comunidad de código abierto apoyada por Red Hat, convirtiéndose en la herramienta más fácil de utilizar para automatizar.

CISCO

Cisco es una empresa global, principalmente dedicada a la fabricación, venta, mantenimiento y consultoría de equipos de telecomunicaciones.

GitHub

Es una plataforma para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de computadora.

GitLab

Es una plataforma para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de computadora.

GNS3

Es un simulador gráfico de red que permite diseñar topologías de red complejas y poner en marcha simulaciones sobre ellos.

JSON

Acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos.

OpenFlow

Es el primer protocolo abierto de comunicaciones SDN.

OpenSource

Es un modelo de desarrollo de software basado en la colaboración abierta.

Python

Es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible

Red Hat

Es una compañía, líder mundialmente, de soluciones de TI de Open Source para empresas.

Ruby

Es un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad.

Troubleshooting

Es una forma de resolución de problemas, aplicada a reparar productos o procesos fallidos en una máquina o sistema.

YAML

Es un formato de serialización de datos legible por humanos inspirado en lenguajes como XML, C, Python, Perl, así como el formato para correos electrónicos especificado en RFC 2822.

Bibliografía

- [1] Michael T. Hannan and Melvin Kranzberg. History of the organization of work. *Encyclopædia Britannica, inc.*, (Junio), 2017.
- [2] Información sobre openflow - <https://es.wikipedia.org/wiki/openflow>.
- [3] Información sobre sdn - <https://www.opennetworking.org/sdn-definition/>.
- [4] Inc. Red Hat. Ansible in depth. *Ansible resources*, 2017.
- [5] Jason C. Neumann. *The Book of GNS3*. No Starch Press, 2015.
- [6] Scott S. Lowe, Matt Oswalt, and Jason Edelman. *Network Programmability and Automation*. O'Reilly Media, Inc., 2018.
- [7] Cisco and its affiliates. Network programmability and automation with cisco nexus 9000 series switches. *Cisco White Paper*, (Agosto), 2016.
- [8] Página oficial cisco - <https://www.cisco.com>.
- [9] ¿qué es la infraestructura convergente? - <https://www.hpe.com/es/es/what-is/converged-infrastructure.html>.
- [10] Documentation about vmware - <https://docs.vmware.com/es/>.
- [11] Página oficial oracle - <https://www.oracle.com/es/index.html>.
- [12] Página oficial red hat - <https://www.redhat.com>.
- [13] About the open source initiative - <https://opensource.org/about>.
- [14] Jennifer Riggins. Netdevops: The next frontier in agile enterprise automation? *The New Stack*, (Junio), 2018.
- [15] Información sobre osd - <https://opensource.org/osd>.
- [16] Información sobre openflow - <https://www.wireshark.org>.
- [17] Página oficial para descarga de ios cisco - <https://software.cisco.com/download/home/>.
- [18] Asignatura gestión de proyectos.
- [19] Federación de Servicios de CCOO. Ministerio de empleo y seguridad social. *Boletín oficial del estado*, (Enero), 2017.
- [20] Asignatura ingeniería del software.
- [21] Robin Gilijamse. Insights from the netdevops fall 2016 survey. *Interesting Traffic*, (Marzo), 2017.
- [22] Madhurranjan Mohaan and Ramesh Raithatha. *Learning Ansible*. Packt Publishing, 2014.

- [23] Documentation about ansible - <https://docs.ansible.com>.
- [24] Documentation about gns3 - <https://docs.gns3.com>.
- [25] Documentation about docker - <https://docs.docker.com>.
- [26] Cisco. Creating ethernet vlans on catalyst switches. *Cisco Documentation*, (Mayo), 2014.
- [27] Cisco. Ospf design guide. *Cisco Documentation*, (Agosto), 2015.
- [28] Cisco. Dynamically configuring dhcp server options. *Cisco Documentation*, (Octubre), 2005.

Anexo

Resultado “ansible localhost -m setup”

```
localhost | SUCCESS => {
"ansible_facts": {
"ansible_apparmor": {
"status": "disabled"
},
"ansible_architecture": "x86_64",
"ansible_bios_date": "07/02/2015",
"ansible_bios_version": "6.00",
"ansible_cmdline": {
"BOOT_IMAGE": "/boot/vmlinuz-4.4.0-31-generic",
"root": "UUID=9e2ceb37-ebf5-4282-84e5-0d6b66e41a64",
"splash": true,
"vt.handoff": "7"
},
"ansible_date_time": {
"date": "2018-06-14",
"time": "14:32:41",
"tz": "UTC",
"tz_offset": "+0000",
"weekday": "Thursday",
"weekday_number": "4",
"weeknumber": "24",
"year": "2018"
},
"ansible_device_links": {
"ids": {},
"model": "VMware Virtual S",
"partitions": {
"sdb1": {
},
"sectors": "204797952",
"sectorsize": 512,
"size": "97.66 GB",
"start": "2048",
"uuid": null
}
},
},
"ansible_distribution": "Ubuntu",
"ansible_dns": {},
"ansible_domain": "",
"ansible_effective_group_id": 1000,
"ansible_effective_user_id": 1000,
"ansible_env": {
```

```

"HOME": "/home/administrador",
"HUSHLOGIN": "FALSE",
"LOGNAME": "administrador",r",
"_": "/usr/bin/ansible"
},
"ansible_fips": false,
"ansible_form_factor": "Other",
"ansible_fqdn": "Admin",
"ansible_hostname": "Admin",
"description": "Ubuntu 16.04.4 LTS",
"major_release": "16",
"release": "16.04"
},
"ansible_machine": "x86_64",
"ansible_memfree_mb": 75,
"ansible_memtotal_mb": 2000,
"Intel(R) Core(TM) i5-5257U CPU @ 2.70GHz"
],
"ansible_processor_cores": 1,
"ansible_processor_vcpus": 1,
"ansible_product_name": "VMware Virtual Platform",
"ansible_product_serial": "NA",
"ansible_product_uuid": "NA",
"ansible_python_version": "2.7.12",
"ansible_real_group_id": 1000,
},
"ansible_swapfree_mb": 2045,
"ansible_system": "Linux",
"ansible_system_capabilities": [
"cap_chown",
"37+i"
],
"module_setup": true
},
"changed": false
}

```

Listado 56: ansible localhost -m setup

Configuración básica para sw-este

```

conf t
hostname sw-este
enable secret ansible
service password-encryption
no ip domain-lookup
banner motd @ Unauthorized Access is Prohibited! @

```

```

ip ssh version 2
ip domain-name cisco.com
username ansible secret ansible
crypto key generate rsa
1024
ip ssh version 2
line vty 0 15
login local
transport input ssh
vlan 99
name Management
interface vlan 99
ip address 192.168.99.10 255.255.255.0
no sh
exit
ip default-gateway 192.168.99.1
interface G0/0
switchport mode access
switchport access vlan 99
end

```

Listado 57: Configuración básica para sw-este.

Configuración básica para sw-oeste

```

conf t
hostname sw-oeste
enable secret ansible
service password-encryption
no ip domain-lookup
banner motd @ Unauthorized Access is Prohibited! @
ip ssh version 2
ip domain-name cisco.com
username ansible secret ansible
crypto key generate rsa
1024
ip ssh version 2
line vty 0 15
login local
transport input ssh
vlan 99
name Management
interface vlan 99
ip address 192.168.99.11 255.255.255.0
no sh
exit
ip default-gateway 192.168.99.1

```

```
interface g1/0
switchport trunk encapsulation dot1q
switchport mode trunk
switchport trunk native vlan 1
end
```

Listado 58: Rutas por defecto.

Configuración básica para r-oeste

```
conf t
hostname r-oeste
enable secret ansible
service password-encryption
no ip domain-lookup
banner motd @ Unauthorized Access is Prohibited! @
ip ssh version 2
ip domain-name cisco.com
username ansible secret ansible
crypto key generate rsa
1024
ip ssh version 2
line vty 0 15
login local
transport input ssh
interface g3/0.99
description Management LAN
encapsulation dot1q 99
ip address 192.168.99.1 255.255.255.0
interface g3/0
no sh
end
```

Listado 59: Rutas por defecto.

Configuración básica para r-norte

```
conf t
hostname r-norte
enable secret ansible
service password-encryption
no ip domain-lookup
banner motd @ Unauthorized Access is Prohibited! @
ip ssh version 2
ip domain-name cisco.com
username ansible secret ansible
```

```
crypto key generate rsa
1024
ip ssh version 2
line vty 0 15
login local
transport input ssh
interface serial 6/0
ip address 172.16.1.2 255.255.255.252
no sh
router ospf 1
network 172.16.1.0 0.0.0.3 area 1
end
```

Listado 60: Rutas por defecto.

Configuración básica para r-este

```
conf t
hostname r-este
enable secret ansible
service password-encryption
no ip domain-lookup
banner motd @ Unauthorized Access is Prohibited! @
ip ssh version 2
ip domain-name cisco.com
username ansible secret ansible
crypto key generate rsa
1024
ip ssh version 2
line vty 0 15
login local
transport input ssh
interface serial 6/1
ip address 172.16.2.1 255.255.255.252
no sh
router ospf 1
network 172.16.2.0 0.0.0.3 area 1
end
```

Listado 61: Rutas por defecto.