

# **ESCUELA POLITÉCNICA NACIONAL**

**ESCUELA DE FORMACIÓN DE TECNÓLOGOS**

**IMPLEMENTACIÓN DE SERVICIOS DE RED CON  
HERRAMIENTAS DE DEVOPS**

**CREACIÓN DE SERVIDOR HTTP Y DNS MEDIANTE LA  
HERRAMIENTA DEVOPS SALTSTACK**

**TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO  
REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR  
EN REDES Y TELECOMUNICACIONES**

**SANTIAGO ANDRÉS SIERRA SIERRA**

**DIRECTOR: ING. FERNANDO VINICIO BECERRA CAMACHO, MSC.**

**DMQ, marzo 2023**

## CERTIFICACIONES

Yo, SANTIAGO SIERRA declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



---

**Santiago Andrés Sierra Sierra**

**santiago.sierra@epn.edu.ec**

**santiago.sierra2500@gmail.com**

Certifico que el presente trabajo de integración curricular fue desarrollado por SANTIAGO SIERRA, bajo mi supervisión.



---

**Fernando Vinicio Becerra Camacho**

**DIRECTOR**

**fernando.becerrac@epn.edu.ec**

## **DECLARACIÓN DE AUTORÍA**

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.



---

**Santiago Andrés Sierra Sierra**

**santiago.sierra@epn.edu.ec**

**santiago.sierra2500@gmail.com**

## **DEDICATORIA**

A mis padres Nelson y Mariana, quienes han sido un pilar fundamental en mi formación académica. A mi hermana Mayra, por brindarme su apoyo incondicional durante todo el proceso de estudios, compartiéndome sus conocimientos día a día. A mis primos Cristofer e Israel con quienes he compartido los mejores momentos, risas, tristezas.

## **AGRADECIMIENTO**

Un especial agradecimiento a la Escuela Politécnica Nacional y a la Escuela de Formación de Tecnólogos por el conocimiento de alta calidad impartido durante todo el proceso académico. Al Ing. Fernando Becerra por su paciencia y buena actitud en este proyecto de titulación. Al Ing. Elvis Espinosa por compartir sus conocimientos en este periodo académico.

# ÍNDICE DE CONTENIDOS

CERTIFICACIONES .....	I
DECLARACIÓN DE AUTORÍA .....	II
DEDICATORIA .....	III
AGRADECIMIENTO .....	IV
RESUMEN.....	VII
ABSTRACT.....	VIII
1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO.....	1
1.1 Objetivo general.....	1
1.2 Objetivos específicos .....	1
1.3 Alcance.....	2
1.4 Marco Teórico.....	2
Arquitectura de Saltstack .....	2
Gestión de la configuración.....	3
DevOps .....	4
Métodos para implementar Saltstack. ....	5
Servidor HTTP .....	5
Servidor DNS.....	5
2 METODOLOGÍA.....	6
3 RESULTADOS .....	7
3.1 Análisis de la herramienta DevOps y sus características. ....	8
3.2 Diseño de una aplicación con la herramienta Saltstack para la implementación de un servidor HTTP y DNS .....	9
3.3 Implementar mediante la herramienta de DevOps Saltstack un servidor HTTP y DNS.....	24
3.4 Verificación del algoritmo implementado con Saltstack.....	29
4 CONCLUSIONES .....	30
5 RECOMENDACIONES.....	32

6	REFERENCIAS BIBLIOGRÁFICAS.....	33
7	ANEXOS.....	34
	ANEXO I: Certificado de Originalidad.....	i
	ANEXO II: Enlaces.....	ii
	ANEXO III: Códigos Fuente .....	iii

# RESUMEN

El presente proyecto de titulación detalla el proceso de implementación de un servidor HTTP y DNS mediante la herramienta de gestión de la configuración Saltstack. El desarrollo del este proyecto fue llevado a cabo a través de la filosofía de DevOps. La implementación se realizó mediante una investigación de los sistemas de gestión de la configuración, ejecutados con métodos para agilizar los procesos de entrega de un proyecto dentro de una organización. Así como, la infraestructura misma.

En todas las secciones del presente documento, se especifica el proceso para la correcta instalación e implementación del sistema. En la primera parte se explica las ventajas de implementar métodos de automatización en las aplicaciones y redes de los servidores encontrados dentro de una infraestructura de comunicaciones. De igual manera, se describe el alcance del proyecto, el objetivo general, los objetivos específicos y el marco teórico.

La segunda parte comprende la metodología utilizada para la implementación y correcto funcionamiento del proyecto. Esta se dividió en cuatro fases, las cuales especifican los procedimientos empleados para alcanzar los objetivos planteados por el sistema.

En la parte de resultados se definen las herramientas y plataformas de *software* utilizadas, tales como: AWS, Apache, BIND. Además, se describe detalladamente el procedimiento empleado para la instalación de Saltstack y de los servidores. Así como, la integración entre los mismos.

En la cuarta y quinta sección se exponen las conclusiones y recomendaciones del proyecto, presentando ideas a tomar en cuenta en un futuro para mejorar próximas implementaciones.

En la sexta sección se cita las referencias bibliográficas. En la séptima parte se presentan los anexos y un video explicativo de la instalación de Saltstack, del servidor HTTP y DNS, pruebas de conexión y funcionamiento del sistema.

**PALABRAS CLAVE:** HTTP, DNS, DevOps, Saltstack, AWS, Apache, BIND.



## ABSTRACT

*This degree project details the implementation process of an HTTP and DNS server using the Saltstack configuration management tool. The development of this project was carried out through the DevOps philosophy. The implementation was done through an investigation of configuration management systems executed with methods to streamline the delivery processes of a project within an organization. As well as, the infrastructure itself.*

*In all sections of this document, the process for the correct installation and implementation of the system is specified. The first part explains the advantages of implementing automation methods in the applications and networks of the servers found within a communications infrastructure. It also describes the scope of the project, the general objective, the specific objectives and the theoretical framework.*

*The second part comprises the methodology used for the implementation and correct operation of the project. This was divided into four phases, which specify the procedures used to achieve the objectives set by the system.*

*In the results part, the tools and software platforms used are defined, such as: AWS, Apache, and BIND. In addition, the procedure used for the installation of Saltstack and the servers is described in detail. As well as, the integration between them.*

*In the fourth and fifth sections, the conclusions and recommendations of the project are presented, presenting ideas to be considered in the future to improve future implementations.*

*In the sixth section the bibliographical references are cited. The seventh section presents the annexes and a video explaining the installation of Saltstack, the HTTP and DNS server, connection tests and system operation.*

**KEYWORDS:** HTTP, DNS, DevOps, Saltstack, AWS, Apache, BIND.

# **1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO**

En el proyecto desarrollado, se realizó la implementación y configuración de servidores HTTP y DNS. El mismo que se llevó a cabo mediante la utilización de la herramienta de gestión de la configuración Saltstack, correspondiente a una aplicación de la ideología DevOps. Este sistema tiene como principal ventaja unificar los procesos de desarrollo y operaciones dentro de las etapas de análisis y ejecución de un proyecto. Teniendo como resultado una mejor organización del personal dentro del departamento de TI. Los sistemas de automatización son implementados con la finalidad de tener un monitoreo constante de redes de servidores y aplicaciones pertenecientes a una red.

Se realizó la instalación de la herramienta con todos sus componentes necesarios para su correcto funcionamiento (nodos maestros y secundarios). Posteriormente se procedió a realizar las pruebas de conexión necesarias entre los nodos instalados. Se ejecutó la instalación de los paquetes del servidor Apache correspondiente a la implementación de un servidor HTTP, y la descarga de repositorios necesarios para habilitar el servidor DNS dentro de la herramienta Saltstack.

Para verificar su funcionamiento se realizó los siguientes procedimientos: en el caso del servidor HTTP se ingresó la dirección IP (Protocolo de Internet) del nodo subalterno dentro del navegador. En el caso del servidor DNS se realizó una prueba de conectividad mediante el comando PING.

## **1.1 Objetivo general**

Implementar un servidor HTTP y DNS a través de la herramienta DevOps Saltstack.

## **1.2 Objetivos específicos**

- Analizar la herramienta de DevOps Saltstack y sus características.
- Diseñar una aplicación mediante la herramienta de DevOps Saltstack para un servidor HTTP y DNS.
- Implementar mediante la herramienta de DevOps Saltstack un servidor HTTP y DNS.
- Verificar el funcionamiento del algoritmo implementado con Saltstack.

### 1.3 Alcance

Realizar un análisis mediante la investigación de las herramientas precisas de *software*, para cumplir con las necesidades de funcionalidad del proyecto, incluyendo sistemas operativos utilizados, comandos empleados en el desarrollo e implementación tanto del servidor HTTP como DNS, y correcta selección de la plataforma a ser utilizada para implementar la herramienta dentro de los sistemas operativos seleccionados.

### 1.4 Marco Teórico

El enfoque de las empresas en la actualidad es migrar todos sus servicios a la nube, teniendo una administración más eficaz de sus productos, práctica denominada automatización o gestión de la configuración. La automatización está tomando fuerza en el área de TI para la administración de los servicios y plataformas en una organización.

Saltstack es una herramienta de gestión de la configuración confiable y de fácil comprensión, la cual brinda las siguientes características: administrar, configurar e instalar *software* a partir de un nodo central, configurar servicios a través de comandos y optimizar tiempos de implementación de un sistema informático.

#### Arquitectura de Saltstack

Saltstack está compuesto de una estructura primario-secundario denominado *Salt Máster* y *SaltMinion*.

**Salt-Master** -> Es el servidor de Saltstack, que fue desarrollado en el lenguaje de programación Python, y que tiene el control de todos los sistemas integrados, además de ser el proveedor de los comandos y archivos para los nodos secundarios remotos que se encuentran vinculados al mismo [1].

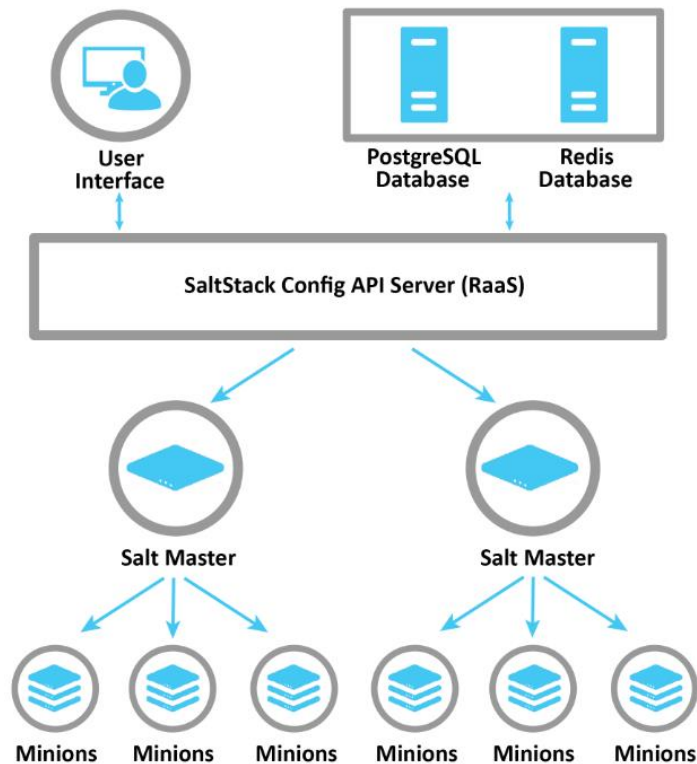
**Salt-Minion** -> son los clientes que se encargan de dar información sobre todos los sucesos importantes y reciben los comandos por parte del maestro para ser ejecutados en los mismos [1].

**Salt-syndic**-> es una máquina virtual configurada en modo maestro que actúa como intermedia entre los dos nodos [1].

**Salt-SSH**-> es una herramienta de gran utilidad cuando en la máquina del nodo secundario no se pueden ejecutar los comandos enviados [1].

**Salt proxy Minion** -> permite inspeccionar dispositivos de red; tales como: enrutadores y conmutadores [1].

Además, Saltstack ofrece un acceso a bases de datos desarrolladas en el lenguaje de programación Python, tales como: PostgreSQL la cual permite almacenar información de los *Minions* y *Redis*, que recopila datos de almacenamiento temporales, como es el caso de la memoria caché [2], como se ilustra en la Figura 1.1.



**Figura 1.1** Arquitectura de Saltstack [2].

También posee un *RaaS* (Regresador como servicio), siendo este el sistema central de la arquitectura de Saltstack, la cual envía comandos a través de puertos de comunicación 4505 y 4506.

### **Gestión de la configuración.**

Se denomina gestión de configuración al proceso de control y mantenimiento tanto de *software* como de *hardware*. Asegurando el correcto funcionamiento de estos para que trabajen en un prolongado periodo de tiempo. Identificando si necesitan un parche [3].

El procedimiento tiene una fase inicial en donde se realiza una recolección de datos pertenecientes a las aplicaciones o infraestructura de comunicaciones, en dicho proceso se verifica contraseñas de manera segura [3].

La automatización de los sistemas es una práctica que se implementa para mejorar la calidad de los mismos, teniendo como principal ventaja la facilidad de reconocer y administrar de manera segura aquellos elementos que requieren atención de manera inmediata y otros que no.

## **DevOps**

El término hace referencia a la unión de *development* (desarrollo) y *operations* (operaciones). Con dicho método se unifican: tecnología, personas y procesos. Logrando dar un mejor servicio a los clientes de una organización de manera persistente. Donde todos los departamentos pertenecientes a una organización trabajan juntos para alcanzar un objetivo [4]. Siendo así una técnica efectiva para solventar y brindar un mejor servicio a los clientes de una empresa.

Desarrollando esta técnica se brinda una mejor administración de los equipos y sistemas implementados. Alcanzando objetivos de una manera efectiva en un menor tiempo [4]. Las ventajas de este sistema de trabajo son:

- Adaptación al mercado y competencia.
- Mejora del tiempo y medio de recuperación.
- Reducción del tiempo de comercialización [4].

Esta técnica influencia en las fases de entrega de una aplicación, como lo son:

1. **Planificación** -> se define las características de los sistemas próximos a implementar, haciendo un seguimiento de errores, uso de paneles, creación de registros [4].
2. **Desarrollo** -> en esta fase se enfoca netamente todos los detalles de programación, pudiéndolos implementar en diferentes entornos; se busca la innovación con rapidez sin sacrificar calidad, haciendo uso de herramientas de automatización [4].
3. **Entrega**-> en esta etapa se incluye la configuración e implementación de la estructura del entorno, en donde se define procesos de administración de versiones, realizando entregas con facilidad, tranquilidad y confianza [4].
4. **Uso**-> permite al administrador del sistema monitorear los diferentes entornos de producción, teniendo equipos trabajando de una manera confiable. Garantizando un tiempo de inactividad casi nulo. Con la capacidad de identificar y corregir errores rápidamente, a través de un sistema de alertas, implementados con herramientas de automatización [4].

## **Métodos para implementar Saltstack.**

La implementación Saltstack facilita a los miembros del departamento de TI la administración y automatización del sistema, en especial cuando existe una cantidad numerosa de equipos de red.

Saltstack necesita un controlador configurado en un sistema operativo Linux, y los *Minions* pueden ser configurados en Linux, Windows o macOS [5].

Para la configuración del nodo maestro se requiere una distribución Linux compatible con Python [5]. Se debe ejecutar un Bootstrap, que se entiende como un *script* el cual contiene todos los repositorios necesarios para la instalación de paquetes en el nodo maestro, conjunto con las banderas -M -P. En el nodo secundario se realiza la instalación del Bootstrap, pero sin las banderas, porque se instalan diferentes opciones de configuración. Posterior a la instalación, se debe realizar cambios en el archivo principal, tanto en el nodo maestro como en el nodo secundario, con la dirección IP del maestro para efectuar la conexión entre estos.

### **Servidor HTTP**

El servidor HTTP utiliza el puerto 80 para la conexión, recibiendo peticiones de direcciones URL y enviando una respuesta del contenido web alojado en sus servicios. Se pueden implementar el servidor HTTP mediante NGINX o Apache. Siendo este último el utilizado en este proyecto de titulación. Apache es un proyecto de código abierto que implementa el servidor HTTP para sistemas multiplataforma, [6]. Lanzado en el año 1995 con el objetivo de brindar un servicio eficiente aplicando los estándares HTTP [6].

Apache establece conexión entre el navegador y el servidor. Tiene una estructura cliente-servidor, lo que permite la transmisión de información entre los mismos [7].

### **Servidor DNS**

El servidor DNS utiliza el puerto 53 para comunicarse, resuelve direcciones IP en nombres de dominio. En este proyecto de titulación se hizo uso del servicio BIND (*Berkeley Internet Name Domain*). BIND es un servidor que resuelve nombres de dominio. Fue diseñado en el año 1980 por la universidad de Berkeley, California. Actualmente se encuentra en su versión número nueve [8].

## 2 METODOLOGÍA

El proyecto se desarrolló en fases, descritas en la Figura 2.1; esto permite una mejor organización y fácil implementación del sistema. Cumpliendo los requisitos para que el mismo funcione, brindando eficiencia y facilidad de uso.



**Figura 2.1** Metodología seleccionada.

Para analizar la herramienta de DevOps Saltstack y sus características se realizó un estudio de las plataformas con las que se pueden implementar la herramienta de automatización. Detallando que el objetivo principal del proyecto es poder administrar diferentes tipos de servicios, tales como: HTTP y DNS por medio de Saltstack.

El diseño e implementación del sistema consta de la instalación de la herramienta Saltstack en una distribución de Linux, dentro de la plataforma *AWS (Amazon Web Services)*.

Posteriormente se instaló el Bootstrap de Saltstack con las banderas *-M* y *-P* dentro del sistema operativo maestro; se configuró el archivo *Minion* en modo local para obtener una respuesta de conexión interna. Obteniendo así, una configuración maestro-secundario dentro de la misma máquina virtual. En otra máquina virtual se instaló el Bootstrap de Saltstack, pero sin el parámetro de las banderas, para habilitar herramientas pertenecientes únicamente al nodo secundario.

Una vez instalados los paquetes se configuró el archivo *Minion*, con la dirección IP pública del nodo maestro, para que exista conexión entre ambos. Esta máquina virtual actúa como nodo secundario y se conecta con el maestro para recibir notificaciones sobre actualizaciones y comandos a implementarse en su sistema.

Para la implementación de los servidores se realizó la configuración de archivo de estados de Saltstack, para que se instalen los repositorios pertenecientes al servidor Apache; correspondiente a la implementación de un servidor HTTP. Al igual que la configuración de estados de Saltstack, para habilitar el servicio *BIND*, pertenecientes al servidor DNS.

En la fase de verificación del funcionamiento se iniciaron los estados configurados en la máquina virtual que actúa como maestro, tanto de Apache como de BIND. Se corroboró el funcionamiento del servicio HTTP ingresando la dirección IP del nodo secundario dentro de la barra de búsqueda del navegador, obteniéndose una página web funcional. En el caso del servicio DNS se ejecutó el comando *PING*, pudiéndose evidenciar un estado activo de los paquetes y conexión al dominio configurado (proyecto titulacion.com).

### **3 RESULTADOS**

El sistema de implementación de un servidor HTTP y DNS mediante la herramienta de gestión de configuración Saltstack permite al usuario final tener la capacidad de monitorear los sistemas pertenecientes en su red de manera constante y en tiempo real. En caso de existir un evento inusual, los nodos secundarios alertarán al nodo maestro. Para dicho monitoreo se debe establecer una conexión entre los mismos, a través de las llaves de identificación de Saltstack. Cada nodo tiene configurado una llave de identificación por defecto habilitada cuando se descargan los repositorios de la



herramienta Saltstack. Siendo el nodo maestro el encargado de administrar las mismas. Para que esto ocurra se debe efectuar el comando `salt-key -a` conjunto con el nombre del nodo secundario al que se requiere anexar la conexión.

Finalmente, teniendo ya anexados los nodos maestro-secundario se ejecutan configuraciones de archivos de estado para la instalación de paquetes de los servicios de Apache y BIND. Una vez instalados y configurados dentro del nodo maestro, se ejecutan los siguientes comandos:

1. `sudo salt 'nombre del/los nodos en donde se desea aplicar el servicio*' state.sls apache.`
2. `sudo salt 'nombre del/los nodos en donde se desea aplicar el servicio*' state.sls dns.`

Con estos comandos se instala el servicio en todos los nodos secundarios anexados al nodo maestro. Obteniendo así, una administración eficiente y optimización de tiempo de despliegue de los sistemas.

### **3.1 Análisis de la herramienta DevOps y sus características.**

Se entiende a DevOps como un proceso de unificación entre las actividades relacionadas con el desarrollo y operaciones. DevOps posee varias ventajas para el desarrollo de un proyecto, entre las cuales tenemos:

- Asegurar la calidad, desarrollo, despliegue e integración de los sistemas informáticos.
- Construye una colaboración entre equipos aislados.
- Une fases.
- Optimiza tiempos.
- Automatiza las tareas [9].

La automatización de tareas es una ventaja para una organización, minimizando errores y brindando una mejor experiencia para el usuario final. DevOps trata de implementar un cambio de la cultura en los procesos de desarrollo de proyecto y evitar el aislamiento de los equipos, implementando herramientas para garantizar la estabilidad y calidad de las aplicaciones (identificar causas de errores de la aplicación) [10].

DevOps representa una filosofía de trabajo en conjunto que consta de diferentes fases.

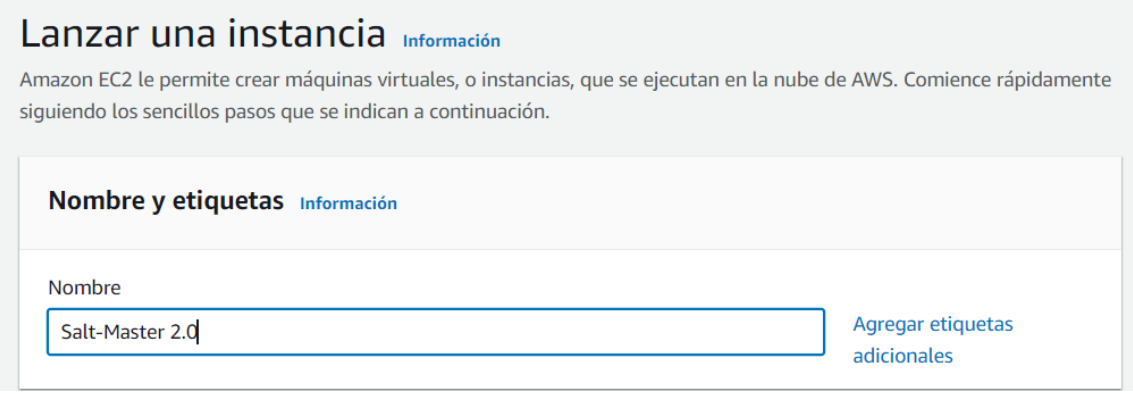
- 1) Planeación-> análisis de requerimientos de un sistema en específico.

- 2) Desarrollo-> proceso por el cual se escribe el código de la implementación.
- 3) Entrega-> etapa en donde el código se une.
- 4) Uso-> pruebas de funcionamiento [10].

Como etapa final se tiene la fase de monitoreo, para identificar momentos en donde el sistema falla.

### 3.2 Diseño de una aplicación con la herramienta Saltstack para la implementación de un servidor HTTP y DNS

La plataforma seleccionada para realizar el despliegue del sistema de gestión de la configuración fue AWS. Debido a la facilidad de uso y variedad de sistemas operativos a disposición para los usuarios, facilitando así la implementación de diferentes proyectos. En la plataforma de AWS se eligió la opción de configurar una máquina virtual en AWS de modo que en la plataforma se lo denomina instancia EC2. Al mismo que se le asignó el nombre de Salt-Master 2.0. Como se puede evidenciar en la Figura 3.1



**Lanzar una instancia** [Información](#)

Amazon EC2 le permite crear máquinas virtuales, o instancias, que se ejecutan en la nube de AWS. Comience rápidamente siguiendo los sencillos pasos que se indican a continuación.

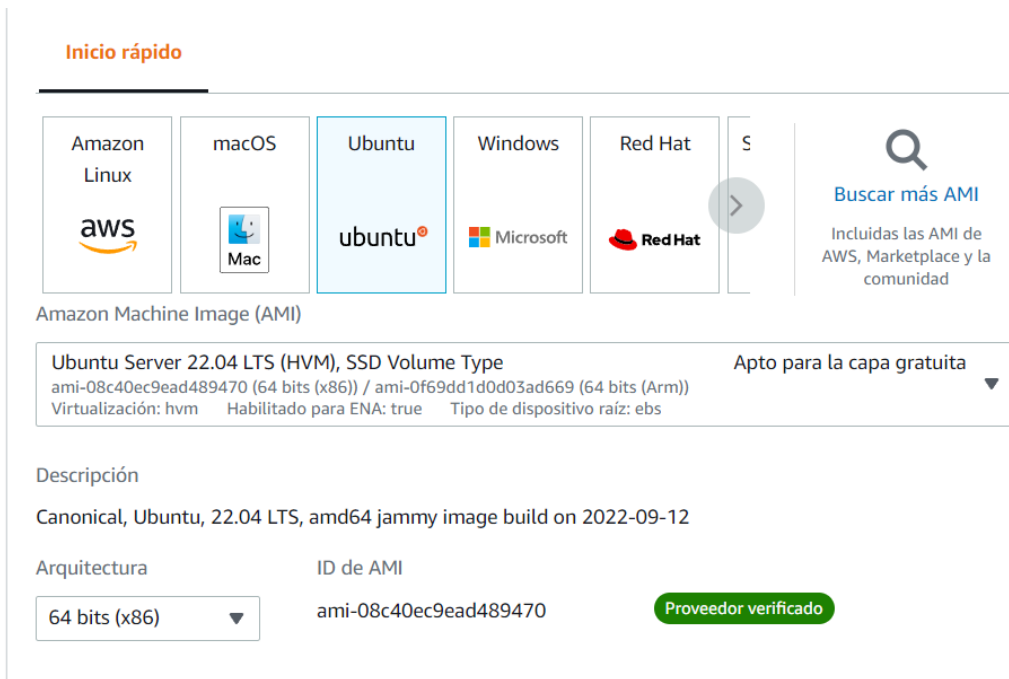
**Nombre y etiquetas** [Información](#)

Nombre

[Agregar etiquetas adicionales](#)

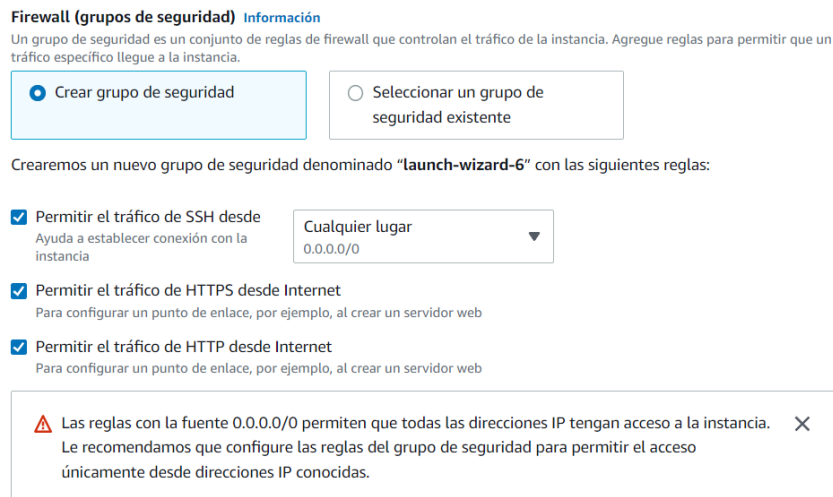
**Figura 3.1** Nombre de la máquina virtual.

Dentro de la misma ventana de configuración se escogió el sistema operativo Ubuntu Server 22.04 como se ilustra en la Figura 3.2. Ubuntu brinda mayor facilidad de configuración, debido a que el sistema operativo Unix es el más conocido en la actualidad.



**Figura 3.2** Sistema operativo de la máquina virtual.

Al momento de crearse una máquina virtual se configuró varias opciones; tales como: edición de un grupo de seguridad, habilitar el puerto 22 de SSH, permitir el tráfico HTTPS y HTTP, correspondientes al puerto 80 y 443. Esta acción facilitó la descarga de los repositorios de la herramienta en la máquina virtual como se ilustran en la Figura 3.3 y Figura 3.4.



**Figura 3.3** Habilitación de reglas de seguridad.

Reglas de entrada (3)							
Name	ID de la regla del g...	Versión de IP	Tipo	Protocolo	Intervalo de puertos		
-	sgr-02648bb67866fb1...	IPv4	SSH	TCP	22		
-	sgr-0b2a427783a3fcf4d	IPv4	HTTP	TCP	80		
-	sgr-0288205a844b03...	IPv4	HTTPS	TCP	443		

**Figura 3.4** Puertos habilitados por defecto.

Dentro de la pestaña de edición de las reglas de seguridad, se añadió un campo denominado TCP personalizado, habilitando una ruta por defecto. Se habilitó los puertos 4505-4506, correspondientes al funcionamiento de Saltstack como se muestra en la Figura 3.5.

ID de la regla del grupo de seguridad	Tipo <small>Información</small>	Protocolo <small>Información</small>	Intervalo de puertos <small>Información</small>	Origen <small>Información</small>
sgr-02648bb67866fb121	SSH	TCP	22	Person... <input type="text" value="0.0.0.0"/>
sgr-0b2a427783a3fcf4d	HTTP	TCP	80	Person... <input type="text" value="0.0.0.0"/>
sgr-0288205a844b033f6	HTTPS	TCP	443	Person... <input type="text" value="0.0.0.0"/>
-	TCP personalizado	TCP	4505-4506	Person... <input type="text" value="0.0.0.0"/>

**Figura 3.5** Habilitación de puertos 4505-4506.

Una vez configurados y habilitados todos los parámetros necesarios como tipo de sistema operativo, puertos HTTP, HTTPS, SSH, 4505-4506 en las reglas de seguridad de la máquina virtual. Se procedió a conectar a la máquina Ubuntu Server 22.04. En la Figura 3.6 se muestran los detalles de red, almacenamiento, procesos y sistema operativo de la máquina virtual. Estos detalles son de gran ayuda en instancias posteriores para poder descargar los repositorios de la herramienta Saltstack, configurar su infraestructura de comunicación (nodos maestros y secundarios) y anexar las máquinas entre sí; logrando una conexión entre las mismas, ya sea de manera local o existiendo un sistema alojado dentro de un segmento de red diferente.

```
System information as of Sun Nov 20 02:17:09 UTC 2022

System load: 0.01025390625    Processes:           103
Usage of /: 19.6% of 7.57GB   Users logged in:    0
Memory usage: 21%             IPv4 address for eth0: 172.31.27.155
Swap usage: 0%

0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

**Figura 3.6** Interfaz inicial de la máquina virtual.

La máquina contó con los recursos estipulados en los parámetros de configuración, como una dirección IP (172.31.27.155) y una memoria 7.57 GB. Este espacio de almacenamiento es alojado en la nube de Amazon.

Posteriormente, se ejecutó el comando `apt update -y`, para actualizar todos los repositorios existentes en el sistema operativo como muestra en la Figura 3.7. El comando utiliza la información caché de los paquetes para saber su versión y si deben o no ser actualizados.

```
ubuntu@ip-172-31-27-155:~$ sudo -s
root@ip-172-31-27-155:/home/ubuntu# apt update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [489 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [106 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [404 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [62.0 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [611 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [78.5 kB]
Get:13 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [2408 B]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [4192 B]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [900 B]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [228 B]
```

**Figura 3.7** Actualización de repositorios.

Para la instalación de Saltstack se descargaron los paquetes de un *script* denominado *Bootstrap*. Descargando los repositorios de Saltstack, tanto del nodo maestro y secundario de la herramienta en la máquina virtual como se muestra en la Figura 3.8.

```

root@ip-172-31-27-155:/home/ubuntu# curl -L https://bootstrap.saltstack.com -o install_salt.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total  Spent    Left  Speed
  0     0     0     0     0     0      0     0  --:--:--  --:--:--  --:--:--    0
100 320k 100 320k   0     0 1466k     0  --:--:--  --:--:--  --:--:-- 1466k

```

**Figura 3.8** Descarga de repositorios.

Para descargar el nodo principal o nodo maestro se ejecutó el comando *sudo sh install\_salt.sh* con las banderas *-M* y *-P*, las mismas que permiten que los paquetes se instalen con las funciones de tipo nodo central y del tipo nodo secundario como se muestra en la Figura 3.9.

```

root@ip-172-31-27-155:/home/ubuntu# curl -L https://bootstrap.saltstack.com -o install_salt.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total  Spent    Left  Speed
  0     0     0     0     0     0      0     0  --:--:--  --:--:--  --:--:--    0
100 320k 100 320k   0     0 1466k     0  --:--:--  --:--:--  --:--:-- 1466k
root@ip-172-31-27-155:/home/ubuntu# sudo sh install_salt.sh -M -P
sh: 0: cannot open install: No such file
root@ip-172-31-27-155:/home/ubuntu# sudo sh install_salt.sh -M -P
* INFO: Running version: 2022.10.04
* INFO: Executed by: sh
* INFO: Command line: 'install_salt.sh -M -P'

* INFO: System Information:
* INFO: CPU:           GenuineIntel
* INFO: CPU Arch:      x86_64
* INFO: OS Name:       Linux
* INFO: OS Version:   5.15.0-1019-aws
* INFO: Distribution: Ubuntu 22.04

* INFO: Installing minion
* INFO: Installing master
* INFO: Found function install_ubuntu_stable_deps
* INFO: Found function config_salt
* INFO: Found function preseed_master
* INFO: Found function install_ubuntu_stable
* INFO: Found function install_ubuntu_stable_post

```

**Figura 3.9** Instalación de Salt master.

Para verificar la correcta instalación y evidenciar qué versión de la herramienta se obtuvo, se ejecutó el comando *sudo salt --versions-report*. Como se muestra en la Figura 3.10 es la versión 3005.1. Saber las versiones del *software* permite determinar qué tipo de plataformas son compatibles con el mismo, si existe una brecha de seguridad y si es posible realizar actualizaciones.

```
root@ip-172-31-27-155:/home/ubuntu# sudo salt --versions-report
Salt Version:
  Salt: 3005.1

Dependency Versions:
  cffi: Not Installed
  cherrypy: Not Installed
  dateutil: 2.8.1
  docker-py: Not Installed
  gitdb: 4.0.9
  gitpython: 3.1.24
  Jinja2: 3.0.3
  libgit2: Not Installed
  M2Crypto: Not Installed
  Mako: Not Installed
  msgpack: 1.0.3
  msgpack-pure: Not Installed
  mysql-python: Not Installed
  pycparser: Not Installed
  pycrypto: Not Installed
```

**Figura 3.10** Versión instalada de Saltstack.

Una vez evidenciada la versión de la herramienta instalada, se procedió a verificar el funcionamiento de los puertos de Saltstack, para esta acción se utilizó el comando `netstat -aon | grep 450` como se muestra en la Figura 3.11.

```
root@ip-172-31-27-155:/home/ubuntu# netstat -aon | grep 450
tcp        0      0 0.0.0.0:4506          0.0.0.0:*            LISTEN    off (0.00/0/0)
tcp        0      0 0.0.0.0:4505          0.0.0.0:*            LISTEN    off (0.00/0/0)
unix  3      [ ]          DGRAM        CONNECTED    20450
```

**Figura 3.11** Visualización puertos en los que trabaja la herramienta.

Posteriormente, se verificó la ubicación de Saltstack con el comando `whereis salt` como se muestra en la Figura 3.12. Con esta información, se pueden instalar archivos de estado y la ruta base de los repositorios y servicios que se requieran implementar.

```
root@ip-172-31-27-155:/home/ubuntu# cd
root@ip-172-31-27-155:~# whereis salt
salt: /usr/bin/salt /etc/salt /usr/share/man/man1/salt.1.gz /usr/share/man/man7/salt.7.gz
root@ip-172-31-27-155:~#
```

**Figura 3.12** Ubicación de Saltstack.

Con el comando `ls` se puede visualizar los archivos que tiene Saltstack como se muestra en la Figura 3.13, en la misma podemos observar los archivos de configuración de Saltstack (*Minion* y *master.d*), los mismos que se van a editar para realizar las conexiones entre los nodos maestro y secundario.

```
root@ip-172-31-27-155:/etc/salt# ls
master master.d minion minion.d minion_id pki proxy proxy.d
root@ip-172-31-27-155:/etc/salt#
```

Figura 3.13 Archivos de la herramienta.

Después se procedió a configurar un nodo maestro y un nodo secundario de manera local, editando el archivo *Minion*, con la finalidad de obtener una comunicación entre ellos.

En el apartado de *master* se configuró el nombre de *localhost* como se lo muestra en la Figura 3.14.

Se ubicó en la configuración de *id* y se asignó el nombre de *myminion* como se puede evidenciar en la Figura 3.15. A este archivo se lo editó con el comando *vi* conjunto con el comando *sudo* que asigna privilegios de administrador.

Para que los cambios sean eficientes se reinició los servicios de Salt-Minion y Master como se muestra en la Figura 3.16. Una vez reiniciados los servicios se podrá tener conexión entre el nodo maestro y el secundario.

```
# This configuration file is used to manage the behavior of the Salt Minion.
# With the exception of the location of the Salt Master Server, values that are
# commented out but have an empty line after the comment are defaults that need
# not be set in the config. If there is no blank line after the comment, the
# value is presented as an example and is not the default.

# Per default the minion will automatically include all config files
# from minion.d/*.conf (minion.d is a directory in the same directory
# as the main minion config file).
#default_include: minion.d/*.conf

# Set the location of the salt master server. If the master server cannot be
# resolved, then the minion will fail to start.
master: localhost
```

Figura 3.14 Configuración de mater como localhost.

```
# Explicitly declare the id for this minion to use, if left commented the id
# will be the hostname as returned by the python call: socket.getfqdn()
# Since salt uses detached ids it is possible to run multiple minions on the
# same machine but with different ids, this can be useful for salt compute
# clusters.
id: myminion
```

Figura 3.15 Configuración de id.



```
root@ip-172-31-27-155:/etc/salt# sudo service salt-minion restart
root@ip-172-31-27-155:/etc/salt# sudo service salt-master restart
root@ip-172-31-27-155:/etc/salt#
```

**Figura 3.16** Reinicio de servicios Salt minion y Salt master.

Con el comando `sudo salt-key` se verificó que las llaves no están aceptadas aún, como se muestra en la Figura 3.17. Las llaves son el identificador con el cual se comunican los nodos maestro y secundario, el nodo maestro debe tener las llaves del nodo secundario aceptadas para realizar una conexión eficiente entre ellos.

```
root@ip-172-31-27-155:/etc/salt# sudo salt-key
Accepted Keys:
Denied Keys:
Unaccepted Keys:
myminion
Rejected Keys:
root@ip-172-31-27-155:/etc/salt#
```

**Figura 3.17** Verificación de llaves no aceptadas.

Para generar la aceptación de las llaves se ejecutó el comando `sudo salt-key -a myminion`. Como se lo muestra en la Figura 3.18, la llave del nodo secundario pasa de estado no aceptado a estado aceptado.

```
root@ip-172-31-27-155:/etc/salt# sudo salt-key -a myminion
The following keys are going to be accepted:
Unaccepted Keys:
myminion
Proceed? [n/Y] y
Key for minion myminion accepted.
root@ip-172-31-27-155:/etc/salt#
```

**Figura 3.18** Aceptación de llaves.

Para verificar que las llaves de nodo secundario estén aceptadas y comunicándose con el nodo maestro se ejecutó el comando `sudo salt-key` como se muestra en la Figura 3.19. Al comprobar que el estado de las llaves se encuentre en aceptado se tiene una evidencia clara de que existe comunicación entre el nodo maestro con el nodo secundario, es decir que el nodo maestro posee entre sus registros la identificación del nodo secundario *myminion*. En instancias posteriores el maestro será capaz de enviar comandos e instrucciones al nodo secundario, el mismo que informará al nodo maestro de los eventos suscitados en su sistema.

```
root@ip-172-31-27-155:/etc/salt# sudo salt-key
Accepted Keys:
myminion
Denied Keys:
Unaccepted Keys:
Rejected Keys:
```

**Figura 3.19** Verificación de las llaves.

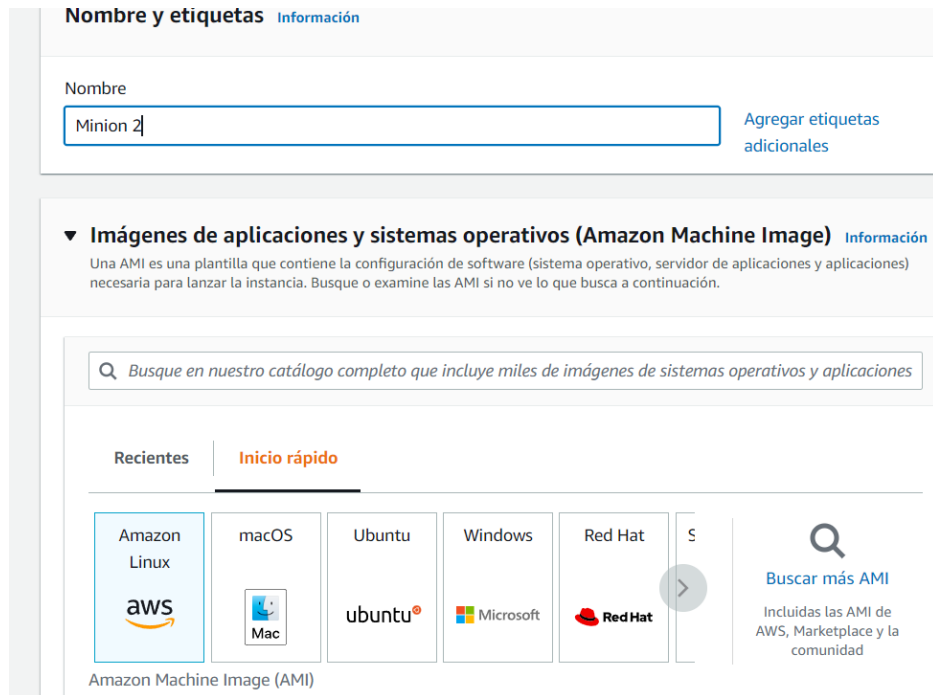
Como procedimiento final de conexión local se ejecutó el comando `Salt '*' test..ping` como se lo muestra en la Figura 3.20. El estado verdadero del comando evidencia que la conexión entre ambos nodos se está efectuando de manera correcta. Si el resultado de la orden se muestra en estado de Falso, quiere decir que la comunicación entre el nodo maestro y el secundario falló.

```
root@ip-172-31-27-155:/etc/salt# salt '*' test.ping
myminion:
True
```

**Figura 3.20** Prueba de conexión local.

Para crear una conexión no local se configuró otra máquina virtual con el sistema operativo Linux implementado por AWS denominada "Minion 2", la cual actuó como nodo subalterno como se lo muestra en la Figura 3.21. Dentro de esta máquina virtual se debe editar parámetros necesarios para instalar los repositorios de Saltstack tales como:

- Habilitación de puertos HTTP y HTTPS.
- Edición de reglas de seguridad
- Habilitación de la regla de tipo TCP personalizado
- Habilitación de puertos de Saltstack (4505-4506).
- Puerto de conexión SSH (22).



**Figura 3.21** Máquina Minion 2.

Se configuró los grupos de seguridad asignando los puertos de funcionamiento de Saltstack 4505-4506 para que el servicio y los repositorios de Saltstack se instalen correctamente como se muestra en la Figura 3.22. Además de tener habilitados varios puertos por defecto en la máquina virtual, como lo son: puerto 22 que corresponde a la conexión remota, los puertos 443 y 80 para el acceso a Internet. También se puede evidenciar que los paquetes se comunican a través de la ruta por defecto 0.0.0.0/0, esto para facilitar el enrutamiento de estos y no tener latencias o pérdidas de comunicación entre las máquinas virtuales.

sgr-084e052ab3b7017cc	TCP personalizado	TCP	4505 - 4506	Person...	Q	0.0.0.0/0 X	Eliminar
sgr-09b4c37e7ae8db2ea	SSH	TCP	22	Person...	Q	0.0.0.0/0 X	Eliminar
-	HTTP	TCP	80	Anywh...	Q	0.0.0.0/0 X	Eliminar
-	HTTPS	TCP	443	Anywh...	Q	0.0.0.0/0 X	Eliminar

**Figura 3.22** Configuración de reglas de grupos de seguridad.

Posteriormente se procedió a descargar el Bootstrap de Saltstack para el *minion*, como se muestra en la Figura 3.23. Este *script* permite ejecutar una lista de instrucciones para conocer el tipo de sistema operativo y versiones de Saltstack necesarios para instalar los repositorios del nodo secundario.

```
[ec2-user@ip-172-31-88-192 ~]$ curl -L https://bootstrap.saltstack.com -o install_salt.sh
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload  Total      Spent    Left     Speed
  0     0     0     0     0     0     0     0     0  0curl: (6)
```

Figura 3.23 Instalación de Bootstrap en el nodo secundario no local.

Para la instalación del cliente de Saltstack no se pasaron los parámetros de las banderas -M y -P, ya que estas corresponden a la instalación del nodo maestro, solamente se ejecutó el comando `sudo sh install_salt.sh` para la descarga de repositorios de Saltstack en el nodo secundario como se lo muestra en la Figura 3.24.

```
[ec2-user@ip-172-31-27-22 ~]$ sudo sh install_salt.sh
* INFO: Running version: 2022.10.04
* INFO: Executed by: sh
* INFO: Command line: 'install_salt.sh '

* INFO: System Information:
* INFO: CPU: GenuineIntel
* INFO: CPU Arch: x86_64
* INFO: OS Name: Linux
* INFO: OS Version: 5.10.162-141.675.amzn2.x86_64
* INFO: Distribution: Amazon Linux AMI 2

* INFO: Installing minion
* INFO: Found function install_amazon_linux_ami_2_deps
* INFO: Found function config_salt
* INFO: Found function preseed_master
* INFO: Found function install_amazon_linux_ami_2_stable
* INFO: Found function install_amazon_linux_ami_2_stable_post
* INFO: Found function install_amazon_linux_ami_2_restart_daemons
* INFO: Found function daemons_running
* INFO: Found function install_amazon_linux_ami_2_check_services
* INFO: Running install amazon linux ami 2 deps()
```

Figura 3.24 Instalación del nodo secundario de Saltstack.

Se configuró el archivo *Minion* para crear una conexión con el nodo maestro. A diferencia de lo realizado en la conexión local, en el apartado de *master* se escribió la dirección IP pública (3.87.78.216) correspondiente al nodo maestro como se muestra en la Figura 3.25. La correcta estructura del archivo de tipo *minion* es una de las partes más esenciales de la herramienta Saltstack, si existe una incorrecta estructura de este no existirá conexión entre el nodo maestro y los nodos subalternos.

```

##### Primary configuration settings #####
#####
# This configuration file is used to manage the behavior of the Salt Minion.
# With the exception of the location of the Salt Master Server, values that are
# commented out but have an empty line after the comment are defaults that need
# not be set in the config. If there is no blank line after the comment, the
# value is presented as an example and is not the default.

# Per default the minion will automatically include all config files
# from minion.d/*.conf (minion.d is a directory in the same directory
# as the main minion config file).
#default_include: minion.d/*.conf

# Set the location of the salt master server. If the master server cannot be
# resolved, then the minion will fail to start.
master: 3.87.78.216

```

**Figura 3.25** Configuración de la dirección IP del nodo maestro.

En orden de reconocer los nodos secundarios Saltstack maneja identificadores, los mismos que pueden ser configurados para una correcta administración. Se cambió el nombre en el campo de identificación del nodo secundario, dentro del archivo de edición. Se le dio el nombre de *minion1* como se muestra en la Figura 3.26.

```

# Specify the location of the daemon process ID file.
#pidfile: /var/run/salt-minion.pid

# The root directory prepended to these options: pki_dir, cachedir, log_file,
# sock_dir, pidfile.
#root_dir: /

# The path to the minion's configuration file.
#conf_file: /etc/salt/minion

# The directory to store the pki information in
#pki_dir: /etc/salt/pki/minion

# Explicitly declare the id for this minion to use, if left commented the id
# will be the hostname as returned by the python call: socket.getfqdn()
# Since salt uses detached ids it is possible to run multiple minions on the
# same machine but with different ids, this can be useful for salt compute
# clusters.
id: minion1

# Cache the minion id to a file when the minion's id is not statically defined
# in the minion config. Defaults to "True". This setting prevents potential
# problems when automatic minion id resolution changes, which can cause the
# minion to lose connection with the master. To turn off minion id caching,

```

**Figura 3.26** Configuración del campo id.

Para que se efectúen los cambios configurados en el archivo *minion*, se reinició el servicio de *salt-minion* como se puede evidenciar en la Figura 3.27. En este caso particular solo se reinició el servicio de *salt-minion* ya que en la máquina virtual que actúa como nodo secundario, se instalaron los paquetes de la herramienta en modo *minion*. Reiniciar el servicio es importante debido a que los cambios ejecutados en el archivo se guardan, pero no se ejecutan, reiniciando el mismo se logra que estos actúen.

```
[ec2-user@ip-172-31-88-192 salt]$ sudo service salt-minion restart
Redirecting to /bin/systemctl restart salt-minion.service
[ec2-user@ip-172-31-88-192 salt]$
```

**Figura 3.27** Reinicio de servicio salt-minion.

En el nodo maestro y en orden de obtener la conexión con el nodo subalterno configurado de manera no local se reinició el servicio tanto del nodo maestro como del nodo secundario como se muestra en la Figura 3.28. Este paso se lo realizó para que se actualicen y ejecuten los cambios en el archivo del nodo maestro.

```
ubuntu@ip-172-31-27-155:~$ sudo service salt-master restart
ubuntu@ip-172-31-27-155:~$ sudo service salt-minion restart
```

**Figura 3.28** Reinicio de servicios en el nodo maestro.

Se realizó la verificación de las llaves con el comando `sudo salt-key` para evidenciar si se encuentran en un estado de aceptado o no aceptado. Como se muestra en la Figura 3.29, las llaves del nodo subalterno configurado en manera no local se encuentran en un estado de no aceptado.

```
ubuntu@ip-172-31-27-155:~$ sudo service salt-master restart
ubuntu@ip-172-31-27-155:~$ sudo service salt-minion restart
ubuntu@ip-172-31-27-155:~$ sudo salt-key
Accepted Keys:
myminion
Denied Keys:
Unaccepted Keys:
minion1
Rejected Keys:
ubuntu@ip-172-31-27-155:~$
```

**Figura 3.29** Verificación de las llaves aceptadas y denegadas.

Para que la llave cambie de estado al de aceptado se ejecutó el comando `sudo salt-key -a minion1` como se muestra en la Figura 3.30, pasando los mismos parámetros necesarios para realizar la aceptación del nodo secundario configurado de manera local. Efectuando esta orden se habilitan opciones para que el nodo maestro guarde el identificador del nodo `minion1` en sus registros internos, logrando una comunicación eficiente entre los mismos. Las llaves contienen información del nodo secundario, tales como: dirección IP, dirección física, memoria interna, datos de memoria caché. Por tal razón los nodos secundarios informan constantemente al nodo maestro sobre los sucesos ocurrientes en la red.

```
ubuntu@ip-172-31-27-155:~$ sudo salt-key -a minion1
The following keys are going to be accepted:
Unaccepted Keys:
minion1
Proceed? [n/Y] y
Key for minion minion1 accepted.
```

Figura 3.30 Aceptación de las llaves.

Para verificar que las llaves del nodo secundario *minion1* se encuentren en un estado de aceptado se ejecutó el comando `sudo salt-key` como se muestra en la Figura 3.31. Obteniendo como resultado la correcta conexión de los nodos secundarios con el nodo maestro. Una de las ventajas de tener una comunicación entre el nodo maestro y el secundario es tener un completo control sobre los procesos, comandos y servicios que se ejecuten en el sistema operativo del nodo secundario.

```
ubuntu@ip-172-31-27-155:~$ sudo salt-key
Accepted Keys:
minion1
myminion
Denied Keys:
Unaccepted Keys:
Rejected Keys:
ubuntu@ip-172-31-27-155:~$
```

Figura 3.31 Verificación de las llaves.

Una vez configurados correctamente los nodos: maestro y secundarios se realizó las pruebas de funcionamiento y comprobación de conexión con el comando `sudo salt '*' test.ping` como se lo muestra en la Figura 3.32. Evidenciando que ambos dan como resultado un valor verdadero, conectándose ambos nodos secundarios al nodo maestro.

```
ubuntu@ip-172-31-27-155:~$ sudo salt '*' test.ping
myminion:
  True
minion1:
  True
```

Figura 3.32 Prueba de respuesta de los nodos subalternos.

En Saltstack los archivos base son los que se configuran para ubicar los paquetes de servidores y servicios que se requieren instalar, dentro de los mismos se configura una

ruta raíz de ubicación de directorios y paquetes. Para tener la configuración de los archivos base se editó el archivo *master.d* dentro de la carpeta de *salt* como se muestra en la Figura 3.33.

```
ubuntu@ip-172-31-27-155:~$ cd /etc/salt
ubuntu@ip-172-31-27-155:/etc/salt$ ls
master  master.d  minion  minion.d  minion_id  pki  proxy  proxy.d
ubuntu@ip-172-31-27-155:/etc/salt$ cd master.d
ubuntu@ip-172-31-27-155:/etc/salt/master.d$
```

**Figura 3.33** Creación del archivo raíz dentro de la herramienta.

Se configuró el archivo raíz con el nombre *roots.conf*, dentro del cual todas las configuraciones de los servicios instalados se guardan. Como se muestra en la Figura 3.34, esto corresponde a la ruta de servicios, dentro de la misma se ubica Saltstack y dentro de ella se encuentran los archivos base.

```
file_roots:
  base:
    - /srv/salt/base
```

**Figura 3.34** Archivo base en donde se guardan las configuraciones.

Se creó el directorio en donde se alojan los archivos y configuraciones principales como se muestra en la Figura 3.35, con el comando de Linux *sudo mkdir -p /srv/salt/base*. Dentro del directorio creado se deben instalar, modificar e iniciar todos los paquetes de aplicaciones que se requieran instalar en orden de obtener un correcto funcionamiento de la herramienta Saltstack.

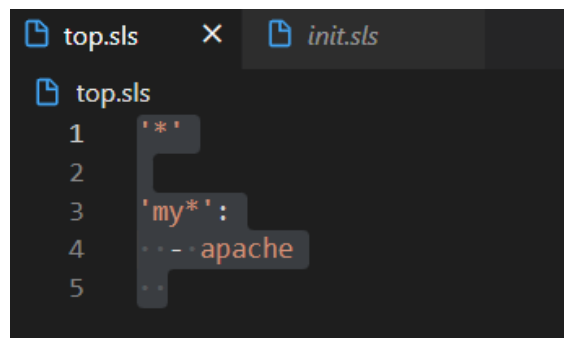
```
ubuntu@ip-172-31-27-155:/etc/salt/master.d$ sudo vi roots.conf
ubuntu@ip-172-31-27-155:/etc/salt/master.d$ vi roots.conf
ubuntu@ip-172-31-27-155:/etc/salt/master.d$ sudo vi roots.conf
ubuntu@ip-172-31-27-155:/etc/salt/master.d$ sudo mkdir -p /srv/salt/base
ubuntu@ip-172-31-27-155:/etc/salt/master.d$ cd /srv/salt/base
ubuntu@ip-172-31-27-155:/srv/salt/base$
```

**Figura 3.35** Configuración de los archivos principales.



### 3.3 Implementar mediante la herramienta de DevOps Saltstack un servidor HTTP y DNS.

Como se describió en secciones anteriores, los archivos de estado de Saltstack son esenciales cuando se configura un determinado servicio. En primera instancia y antes de iniciar el servicio se debe tener un archivo denominado *top.sls*, el cual describe todas las configuraciones que se desea ejecutar en los nodos secundarios. Como se ilustra en la Figura 3.36 se configura un asterisco que se va a crear para todo el entorno de configuración. Después se apunta a que identificación de nodo subalterno se desea instalar algún paquete. En este caso se requiere instalar apache.

The image shows a code editor window with two tabs: 'top.sls' and 'init.sls'. The 'top.sls' tab is active and displays the following code:

```
1  '*'
2
3  'my*':
4  - apache
5
```

**Figura 3.36** Instalación de paquetes de apache.

Para iniciar el servidor HTTP se debe configurar todos los paquetes del programa Apache. Como se muestra en la Figura 3.37 primero se configuró el servicio httpd, que es el servidor de Apache.

En el archivo *index\_html* se editó el contenido que tiene el servidor. Posteriormente, en el fichero *file\_managed*, es el que tiene la ubicación de configuración principal del servidor. Este archivo de estado se lo edita en el nodo maestro y se lo ejecuta en los nodos secundarios, implementando varios parámetros. El parámetro de *name* se refiere al nombre con el que se creó el archivo, en el parámetro de usuario y grupo se ubica el paquete instalado, en este caso es Apache. En el parámetro de modo se implementaron los indicadores seis cientos cuarenta y cuatro, esto hace referencia a que el usuario tiene permisos de escritura y lectura. Como último criterio de esta sección se configuró la ruta de ubicación de donde debe crear el archivo de estado.

En los parámetros *apache\_service* se indica la instrucción para que se inicie el servicio, con el nombre del servidor httpd, y habilitando el servicio con el estado habilitado en modo verdadero.

```
init.sls
1  install_apache: #Servidor Apache
2  pkg.installed:
3    - pkgs:
4      - httpd
5
6  index_html: # archivo html
7  file.managed:
8    - name: /var/www/html/index.html
9    - user: apache
10   - group: apache
11   - mode: 644 #Permisos de lectura para grupos, usuario permisos de lectura y escritura
12   - source: salt://apache/templates/index.html
13
14  apache_service:
15  service.running:
16    - name: httpd
17    - enable: True
```

**Figura 3.37** Instalación de paquetes de apache.

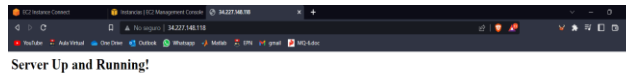
Para que el archivo se ejecute correctamente en el servicio de Saltstack, las configuraciones se ubican en la carpeta /srv/salt/base del servidor. Se corre el comando `sudo salt 'mi*' state.sls apache`, para que el servicio se ejecute en el sistema que actúa como el nodo secundario *minion1*, que es en donde se desea que se habilite el servicio, como se muestra en la Figura 3.38.

```
ubuntu@ip-172-31-27-155:~$ sudo salt 'mi*' state.sls apache
minion1:
-----
      ID: install_apache
  Function: pkg.installed
     Result: True
  Comment: All specified packages are already installed
  Started: 00:08:09.236210
  Duration: 464.969 ms
  Changes:
-----
      ID: index_html
  Function: file.managed
     Name: /var/www/html/index.html
     Result: True
  Comment: File /var/www/html/index.html is in the correct state
  Started: 00:08:09.704665
  Duration: 36.452 ms
  Changes:
-----
      ID: apache_service
  Function: service.running
     Name: httpd
     Result: True
  Comment: The service httpd is already running
  Started: 00:08:09.742360
```

**Figura 3.38** Servicio apache corriendo correctamente.

Para la verificación del sistema instalado se insertó la URL <http://34.227.148.118> en el navegador como se muestra en la Figura 3.39. El funcionamiento del servidor HTTP es

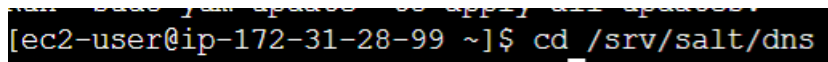
una página web, en este caso no se han configurado marcadores de visualización dentro de la misma.



**Figura 3.39** Servidor activo y corriendo de manera funcional.

En el caso del servidor DNS se realizaron configuraciones similares. Utilizando archivos de estados apuntando a un nodo subalterno específico.

Se creó el directorio `/srv/salt/dns` como se muestra en la Figura 3.40, para ubicar el servicio dentro del mismo, teniendo una organización eficaz y de fácil administración de los sistemas implementados.



**Figura 3.40** Creación de directorio.

Como se lo puede evidenciar en la Figura 3.41, se configuró un archivo de estado denominado `init.sls`. Dentro del mismo, se agregaron líneas donde se instaló los paquetes del servidor BIND. Configurando el servicio `service_running`, donde se pasa los parámetros de `name` conjunto con el grupo `named`, se habilita el servicio y se lo recarga para que el mismo se inicie correctamente.

```

bind:
  pkg.installed:
    - pkgs: ['bind']
  service.running:
    - name: named
    - enable: True
    - reload: True
bind_key_directory:
  file.directory:
    - name: /etc/named.keys
    - require:
      - pkg: bind
named:
  pkg.installed:
    - name: bind
  service.running:
    - enable: True
    - require:
      - pkg: named
    - watch:
      - file: /etc/named.keys

```

**Figura 3.41** Configuración *init.sls* parte 1.

El archivo *named* es el que se estructura para que exista la correspondencia entre las direcciones IP y los nombres de dominio e iniciar el servicio. Como se lo puede evidenciar en la Figura 3.42, se editó la ruta de origen para Saltstack con el motor de configuración jinja, el cual es comúnmente utilizado para mostrar contenido de páginas web mediante variables. Los usuarios con acceso al mismo son los de tipo administrador. Se inician con el grupo *named*, en modo de lectura y escritura, conjunto con los paquetes del servicio *named*.

```

/etc/named.conf:
  file.managed:
    - source: salt://dns/named.conf.jinja
    - template: jinja
    - user: root
    - group: named
    - mode: 640
    - require:
      - pkg: named

```

**Figura 3.42** Configuración *init.sls* parte 2.

En la Figura 3.43, se visualiza la configuración del motor jinja, conjunto con la inicialización de los paquetes del servicio *named*. Se pasaron los parámetros para lectura, escritura y ejecución de los paquetes. Dentro del archivo base, se apuntó hacia

el archivo con el nombre de dominio proyecto titulacion.com. Asimismo, se indicó que el usuario y grupo es el servicio de *named*.

```
/var/named/named.proyectotitulacion.com:
file.managed:
  - source: salt://dns/named.proyectotitulacion.com.jinja
  - template: jinja
  - user: named
  - group: named
  - mode: 644
  - require:
    - pkg: named
```

**Figura 3.43** Configuración *init.sls* parte 3.

Como se evidencia en la Figura 3.44 se muestran las configuraciones del archivo *named*, para apuntar la dirección IP del servidor HTTP instalado con anterioridad para el nombre de dominio seleccionado. Para realizar esta correspondencia se debe editar los parámetros de tipo maestro; el archivo debe tener el nombre de su dominio y en la regla que permite el paso de información se debe anexar la dirección IP del servidor Apache.

```
logging {
  channel default_debug {
    file "data/named.run";
    severity dynamic;
  };
};

zone "." IN {
  type hint;
  file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";

zone "proyectotitulacion.com." IN {
  type master;
  file "named.proyectotitulacion.com";
  allow-update { 34.227.148.118; };
};
```

**Figura 3.44** Configuración *named.conf.jinja*.

Como se evidencia en la Figura 3.45, se configuró varios parámetros del servicio *named* para el dominio seleccionado. En primera instancia se editó la dirección IP del servidor HTTP. Como segundo requerimiento se implementó el tiempo de vida de los paquetes, tales como: tiempo de actualización, tiempo de reinicio, tiempo de expiración y tiempo

mínimo de ejecución. Posteriormente se configuraron los registros de tipo A que corresponden a una dirección IP, en este caso el del servidor HTTP. El registro NS correspondiente al nombre de dominio seleccionado.

```
$ORIGIN .
$TTL 86400 ; 1 day
proyectotitulacion.com      IN SOA    proyectotitulacion.com. salt.proyectotitulacion.com. (
    2016042001 ; serial (YYYYMMDDHH)
    86400      ; refresh (1 day)
    3600       ; retry (1 hour)
    604800    ; expire (1 week)
    300       ; minimum (5mins)
    )
    NS        proyectotitulacion.com.
    A        34.227.148.118
    ;AAAA    ::1
$ORIGIN proyectotitulacion.com.

test                          A        34.227.148.118
proyectotitulacion.com       A        34.227.148.118
```

Figura 3.45 Configuración *named.proyectotitulacion.com.jinja*.

### 3.4 Verificación del algoritmo implementado con Saltstack

Para verificar el funcionamiento de la implementación de un servidor HTTP se ingresó la dirección IP 34.227.148.118. Correspondiente al servidor Apache en el navegador como se muestra en la Figura 3.46.

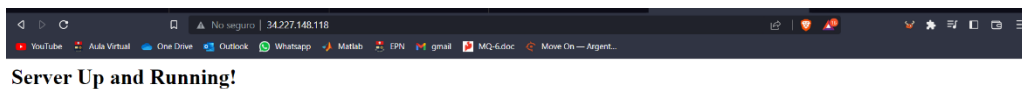


Figura 3.46 Funcionamiento del servidor HTTP.

Como última instancia, para la verificación de funcionamiento del servidor DNS se realizó pruebas con el comando PING como se muestra en la Figura 3.47. Al obtener una respuesta inmediata del servidor implementado conjunto con los servidores de

Amazon.

```
[ec2-user@ip-172-31-7-5 dns]$ ping proyectotitulacion.com
PING proyectotitulacion.com (104.21.14.87) 56(84) bytes of data.
64 bytes from 104.21.14.87 (104.21.14.87): icmp_seq=1 ttl=44 time=7.99 ms
64 bytes from 104.21.14.87 (104.21.14.87): icmp_seq=2 ttl=44 time=7.92 ms
64 bytes from 104.21.14.87 (104.21.14.87): icmp_seq=3 ttl=44 time=7.87 ms
64 bytes from 104.21.14.87 (104.21.14.87): icmp_seq=4 ttl=44 time=7.87 ms
64 bytes from 104.21.14.87 (104.21.14.87): icmp_seq=5 ttl=44 time=7.92 ms
64 bytes from 104.21.14.87 (104.21.14.87): icmp_seq=6 ttl=44 time=8.08 ms
64 bytes from 104.21.14.87 (104.21.14.87): icmp_seq=7 ttl=44 time=7.88 ms
^C
```

Figura 3.47 Funcionamiento del servidor DNS con Saltstack.

## 4 CONCLUSIONES

- A través de este proyecto de titulación se implementó un servidor HTTP y DNS mediante la herramienta de gestión de la configuración Saltstack; teniendo en cuenta que este es un método práctico de aplicación de la ideología DevOps. La infraestructura de Saltstack se compone de nodos maestros y secundarios, los mismos que se comunican entre sí. Estos transmiten información de configuraciones a realizarse en los sistemas de los nodos secundarios; los mismos que informan a los maestros sobre eventos suscitados en la red
- DevOps es una ideología de gran utilidad, con la misma se unifican las tareas de desarrollo y operaciones. Esta filosofía permite optimizar tiempos, automatizar tareas, ahorrar recursos y realizar un monitoreo eficiente de la red. Este sistema de trabajo permite un mantenimiento y control eficiente en los sistemas integrados.
- Las diversas herramientas de gestión de la configuración de DevOps ayudan al fácil monitoreo e identificación rápida de sucesos inusuales entre los sistemas implementados dentro de la infraestructura de comunicaciones. Saltstack se destaca por su capacidad para monitorear múltiples nodos secundarios anexados a un nodo maestro, a través de los archivos de estado configurados dentro de la herramienta, la misma que en el nodo maestro se implementa servicios, comandos, acciones a ser ejecutadas por los nodos secundarios.
- La herramienta Saltstack permite configurar varios nodos secundarios, ejecutando diferentes servicios en cada uno de ellos. Esta es una ventaja para

los administradores de red con la tarea de evaluar, mantener y corregir sucesos e incidentes dentro de una infraestructura de comunicaciones.

- Los archivos de estado Saltstack son herramientas de gran ayuda para aquel que realiza la configuración de la misma. Dichos archivos ayudan a optimizar el tiempo, ya que, no se tiene que manipular de manera manual la serie de instrucciones a llevar a cabo. Al editar este tipo de archivos, ya sea cambiarlos o escribirlos se los ejecuta de manera automática para todos los nodos conectados a dicho nodo maestro.
- Al efectuar una búsqueda sobre los diferentes métodos de implementación de Saltstack, se concluyó que al usar la plataforma AWS para el desarrollo del proyecto es la mejor opción, debido a que posee una diversidad de sistemas operativos disponibles, entre ellos Ubuntu 22.04. Las máquinas virtuales utilizadas cumplen con los requisitos de almacenamiento, procesamiento y memoria requeridos para el diseño de los servidores configurados con paquetes de Apache y BIND con la herramienta Saltstack.
- Para la verificación de funcionalidad del sistema implementado, se ingresó la dirección IP del servidor Apache en el navegador, evidenciando su correcta operatividad. En el caso del servidor DNS se verificó su desempeño a través del comando PING.



## 5 RECOMENDACIONES

- Cada nodo secundario debe tener conexión hacia el nodo maestro mediante el archivo de configuración de este, denominado *minion*. Para que exista comunicación entre ambos, debe estar configurada la dirección IP del nodo maestro en el archivo principal del nodo secundario.
- Mediante comandos propios de la herramienta de Saltstack se puede verificar la conexión. Si la comunicación no está establecida entre el nodo maestro y el/los nodos secundarios, no se puede realizar la configuración de archivos de estado necesarios para habilitar servicios requeridos.
- Saltstack trabaja con los puertos de comunicación 4505-4506. Los mismos que deben estar en estado de escucha y habilitados para que se establezca la conexión entre los nodos maestro y los secundarios.
- Cuando se realiza la configuración de archivos de estados para iniciar un servicio o instalar paquetes usando Saltstack no es necesario especificar la ruta de los paquetes. Esto se debe a que se crea en el directorio por defecto, se indica el servicio configurado para que se ejecute y el archivo en donde se tiene el contenido específico, como en el caso de Apache.

## 6 REFERENCIAS BIBLIOGRÁFICAS

- [1] «¿En qué consiste SaltStack?», *IONOS Digital Guide*. <https://www.ionos.es/digitalguide/servidores/configuracion/que-es-saltstack/> (accedido 16 de enero de 2023).
- [2] alyssar, «Arquitectura del sistema de SaltStack Config». <https://docs.vmware.com/es/VMware-vRealize-Automation-SaltStack-Config/8.6/use-manage-saltstack-config/GUID-FFD5DAE2-5FCD-46B2-8334-21799FB8FECE.html> (accedido 18 de enero de 2023).
- [3] «What is Configuration Management?», *VMware*. <https://www.vmware.com/topics/glossary/content/configuration-management.html> (accedido 18 de enero de 2023).
- [4] «¿Qué es DevOps? Explicación de DevOps | Microsoft Azure». <https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops> (accedido 22 de enero de 2023).
- [5] «Cómo implementar un SaltStack de código abierto para la configuración y administración automatizadas del servidor», *Compubyte.es*, 11 de mayo de 2021. <https://compubyte.es/blog/como-implementar-un-saltstack-de-codigo-abierto-para-la-configuracion-y-administracion-automatizadas-del-servidor/> (accedido 24 de enero de 2023).
- [6] «Welcome! - The Apache HTTP Server Project». <https://httpd.apache.org/> (accedido 25 de enero de 2023).
- [7] G. B, «¿Qué es Apache? Descripción completa», *Tutoriales Hostinger*, 31 de agosto de 2018. <https://www.hostinger.es/tutoriales/que-es-apache/> (accedido 25 de enero de 2023).
- [8] N.-I. D. and T. Management, «BIND DNS: Pros, Cons and Alternatives», *NS1*. <https://ns1.com/resources/bind-dns-pros-cons-and-alternatives> (accedido 25 de enero de 2023).
- [9] *Introducción a DevOps con la nube de AWS y la herramienta SaltStack*, (4 de diciembre de 2021). Accedido: 12 de febrero de 2023. [En línea Video]. Disponible en: <https://www.youtube.com/watch?v=orXTWmFP1WU>
- [10] *¿Qué es y qué no es DevOps?*, (19 de agosto de 2022). Accedido: 12 de febrero de 2023. [En línea Video]. Disponible en: <https://www.youtube.com/watch?v=MtDFK-evWw4>

## **7 ANEXOS**

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO III. Conjunto de datos extensos

## **ANEXO I: Certificado de Originalidad**

### **CERTIFICADO DE ORIGINALIDAD**

Quito, D.M. 10 de abril de 2022

De mi consideración:

Yo, FERNANDO BECERRA, en calidad de director del Trabajo de Integración Curricular titulado CREACIÓN DE UN SERVIDOR HTTP Y DNS MEDIANTE LA HERRAMIENTA SALTSTACK elaborado por el estudiante SANTIAGO SIERRA de la carrera en TECNOLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 17%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

[https://epnecuador-my.sharepoint.com/:b:/g/personal/fernando\\_becerrac\\_epn\\_edu\\_ec/EQKKdBgs4ntHpXDN1-9zDoMBKqFfU6z4yblzLpJ4GHZEng?e=XQMMC7](https://epnecuador-my.sharepoint.com/:b:/g/personal/fernando_becerrac_epn_edu_ec/EQKKdBgs4ntHpXDN1-9zDoMBKqFfU6z4yblzLpJ4GHZEng?e=XQMMC7)

Atentamente.



FERNANDO VINICIO BECERRA CAMACHO

Docente

Escuela de Formación de Tecnólogos

## ANEXO II: Enlaces



*Anexo II.1 Código QR de la implementación y pruebas de funcionamiento*

## **ANEXO III: Códigos Fuente**

### **Servidor HTTP:**

#### **Archivo Top.sls**

'\*':

'my\*':

- apache

Archivo init.sls

install\_apache:

pkg.installed:

- pkgs:

- httpd

index\_html:

file.managed:

- name: /var/www/html/index.html

- user: apache

- group: apache

- mode : 644

- source: salt://apache/templates/index.html

apache\_service:

service.running:

- name: httpd

- enable: True

### **Servidor DNS**

## **Archivo init.sls:**

bind:

pkg.installed:

- pkgs: ['bind']

service.running:

- name: named
- enable: True
- reload: True

bind\_key\_directory:

file.directory:

- name: /etc/named.keys
- require:
- pkg: bind

named:

pkg.installed:

- name: bind

service.running:

- enable: True
- require:
- pkg: named
- watch:
- file: /etc/named.keys

/etc/named.conf:

file.managed:

- source: salt://dns/named.conf.jinja

- template: jinja
- user: root
- group: named
- mode: 640
- require:
- pkg: named

/var/named/named.proyectotitulacion.com:

file.managed:

- source: salt://dns/named.proyectotitulacion.com.jinja
- template: jinja
- user: named
- group: named
- mode: 644
- require:
- pkg: named

### **Archivo named.conf.jinja**

```
options {  
    listen-on port 53 { any; };  
    directory    "/var/named";  
    dump-file    "/var/named/data/cache_dump.db";  
    statistics-file "/var/named/data/named_stats.txt";  
    memstatistics-file "/var/named/data/named_mem_stats.txt";  
    allow-query   { any; };  
    recursion yes;  
};
```



```

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";

zone "proyectotitulacion.com." IN {
    type master;
    file "named.proyectotitulacion.com";
    allow-update { 34.227.148.118; };
};

Archivo named.proyectotitulacion.com.jinja:

$ORIGIN .

$TTL 86400 ; 1 day

```

Proyectotitulacion.com IN SOA proyectotitulacion.com. salt.proyectotitulacion.com.

(

2016042001 ; serial (YYYYMMDDHH)

86400 ; refresh (1 day)

3600 ; retry (1 hour)

604800 ; expire (1 week)

300 ; minimum (5mins)

)

NS proyectotitulacion.com.

A 34.227.148.118

;AAAA ::1

\$ORIGIN proyectotitulacion.com.

test A 34.227.148.118