ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE SERVICIOS DE RED CON HERRAMIENTAS DE DEVOPS

DESPLIEGUE DE PKI DE FORMA AUTOMÁTICA MEDIANTE ANSIBLE CON HERRAMIENTAS OPEN SOURCE

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

ARLEY IVÁN SALGADO MAÑAY

salgado_arley@hotmail.com

DIRECTOR: ING. FERNANDO VINICIO BECERRA CAMACHO

fernando.becerrac@epn.edu.ec

DMQ, AGOSTO 2023

CERTIFICACIONES

Yo, Arley Iván Salgado Mañay 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.

Arley Iván Salgado Mañay

arley.salgado@epn.edu.ec

salgado_arley@hotmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Arley Iván Salgado Mañay, 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.

Arley Iván Salgado Mañay

CI: 1724254386

DEDICATORIA

Dedico este trabajo principalmente a mis padres Iván y Noemi, por ser mi principal apoyo, por ayudarme a levantarme en cada una de mis caídas y por ser incondicionales en toda mi vida.

De igual manera dedico este esfuerzo a mis hermanos Joan y Yanina, quienes me son mi motor para seguir adelante y me han dado la motivación suficiente para poder continuar en mi vida personal.

A mis amigos y compañeros de carrera, de quienes he aprendido mucho a lo largo de este camino y quienes me han ayudado a poder cumplir esta meta.

Arley Salgado

AGRADECIMIENTO

Agradezco a Dios por todas las bendiciones que me da en mi diario vivir, agradezco la familia que me ha regalado, los amigos que he conocido, las experiencias vividas y las enseñanzas que he adquirido.

A mis padres Iván y Noemi por nunca haber soltado mi mano y ser mis eternos acompañantes, agradezco cada uno de sus consejos, cada una de sus enseñanzas y principalmente agradezco la educación que me han dado.

A mis hermanos Joan y Yanina por ser mi mayor motivación para no decaer a lo largo de este camino y por estar conmigo en todo momento.

A mi familia que ha estado al pendiente de mi constante crecimiento personal y profesional.

A mi tutor de tesis Fernando Becerra por la paciencia que me ha tenido a lo largo del desarrollo del presente trabajo, por su ayuda en las materias impartidas y por ser el principal soporte para poder desarrollar este trabajo.

A mis amigos del colegio que siempre nos hemos ayudado mutuamente y siempre hemos estados al pendiente de todos.

Finalmente, a mi grupo de amigos de la ESFOT que han sido parte fundamental para poder llegar a este punto, agradezco su compañía, su paciencia, sus consejos y por su amistad incondicional.

Arley Salgado

ÍNDICE DE CONTENIDOS

RESUMEN

El presente trabajo de integración contiene el proceso de como desplegar un PKI automáticamente por medio de la herramienta de automatización *Ansible*.

La primera sección contiene la explicación del trabajo elaborado, los objetivos a cumplir, el alcance del trabajo de titulación y los principales conceptos utilizados.

La segunda sección contiene la metodología utilizada para llevar a cabo cada objetivo planteado en el trabajo de titulación y el proceso que se siguió para el despliegue del PKI.

La tercera sección contiene los resultados obtenidos, inicialmente se presenta información de las herramientas utilizadas para el desarrollo del presente trabajo de titulación. Se presenta la instalación de las herramientas para lograr el despliegue de un PKI, la elaboración de la aplicación que contiene las tareas a ejecutarse. Por último, se tienen las pruebas del código creado.

La cuarta y quinta sección contiene las conclusiones y recomendaciones relacionadas al tema, con el fin de crear una relación directa entre el aprendizaje adquirido conjunto a los resultados obtenidos. La sexta sección contiene las referencias de donde se pudo extraer información para llevar a cabo el presente trabajo.

Por último, la séptima sección contiene los anexos contiene el código creado, el certificado de originalidad, el enlace y código QR del video que muestra el funcionamiento del código.

PALABRAS CLAVE: PKI, *Ansible*, DevOps, *playbook*, *OpenSSL*.

ABSTRACT

This integration work contains the process of how to deploy a PKI automatically by means of the Ansible automation tool.

The first section contains the explanation of the work done, the objectives to be achieved, the scope of the degree work and the main concepts used.

The second section contains the methodology used to carry out each objective and the process followed for the deployment of the PKI.

The third section contains the results obtained. Initially, information about the tools used for the development of this degree work is presented. The installation of the tools to achieve the deployment of a PKI is presented, as well as the elaboration of the file containing the tasks to be executed. Finally, the tests of the created code are presented.

The fourth and fifth sections contain the conclusions and recommendations related to the subject, to create a direct relationship between the learning acquired and the results obtained. The sixth section contains the references from which information could be extracted to carry out the present work.

Finally, the seventh section contains the annexes containing the code created, the certificate of originality, the link and the QR code of the video showing how the code works.

KEYWORDS: *PKI, Ansible,* DevOps*, playbook, OpenSSL.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Este proyecto de titulación tiene como finalidad utilizar herramientas DevOps para automatizar una infraestructura de llave pública (PKI). Para esto se lleva a cabo una investigación previa sobre las herramientas de automatización de DevOps. Por medio de esta investigación, se presenta las ventajas y desventajas de cada una de las posibles herramientas de automatización a utilizar. De esta forma se podrá seleccionar la opción viable para este proyecto. Una vez que se tenga claro la herramienta adecuada para el desarrollo de este trabajo de titulación se procede a la instalación y desarrollo sobre esta herramienta.

La herramienta de automatización a utilizarse en el presente proyecto de titulación es *Ansible*. Dentro de esta herramienta se va a desarrollar un libro de jugadas (*playbook*) que funciona por medio de tareas. Estas tareas son esenciales para automatizar el despliegue del PKI dentro de uno o más nodos. La razón por la que esta herramienta fue seleccionada para desarrollar este trabajo de titulación, es que el único requerimiento para los clientes es el manejo del protocolo SSH.

Para despliegue del PKI se necesita de la herramienta *OpenSSL*. Por medio de esta herramienta, *Ansible* va a automatizar la implementación de esta infraestructura. Esta herramienta permite crear las siguientes instancias: autoridad certificadora (CA), solicitud de firma de certificado (CSR), llave privada y el certificado firmado. Además, openssl revisar la CSR, firmar el CSR y finalmente generar el certificado para el usuario.

Una vez que se realicen las pruebas respectivas al *playbook* creado, se va a realizar el despliegue del PKI. Este PKI se va a encontrar en una carpeta en el escritorio del cliente. Este elemento va a contener diferentes carpetas que cumplen una tarea en específico, el certificado de la CA, el CSR, un archivo serial que lleva el conteo de certificados emitidos y finalmente el certificado que pertenece al usuario.

1.1 Objetivo general

Desarrollar servicios de *networking* mediante herramientas de DevOps.

1.2 Objetivos específicos

- Analizar la herramienta de DevOps que soluciona cada componente del proyecto de titulación.
- Diseñar la solución para cada servicio de *networking* mediante herramientas de DevOps.

- Implementar las soluciones mediante herramientas de DevOps para el despliegue de los servicios de *networking*.
- Verificar el funcionamiento de cada servicio de *networking* implementado mediante DevOps.

1.3 Alcance

El presente proyecto pretende que los estudiantes utilicen DevOps y que desplieguen las soluciones en *host* o en equipos de *networking*. Para realizar este propósito se necesita de herramientas de DevOps capaces de automatizar el despliegue de los servicios propuestos y además se puedan realizar pruebas de las soluciones. En el presente proyecto se pretende implementar el despliegue de PKI de forma automática mediante *Ansible* con herramientas *Open Source.*

1.4 Marco Teórico

Open Source

Open Source es un código que ha sido diseñado para ser accesible hacia cualquier usuario. Esta característica permite que exista un perfeccionamiento del código que se desarrolla, al permitir un trabajo colaborativo. Estos códigos abiertos pueden ser modificados o distribuidos dependiendo del tipo de licencia que tengan. Gracias a las ventajas que posee O*pen Source* ha sido tomado en cuenta para la producción de *software* [1].

El uso de *Open Source* presenta varias ventajas tales como:

- **Acceso:** Cualquier persona que lo desee puede acceder al código. El que varias personas accedan a él garantiza rapidez y efectividad [2].
- **Revisión entre compañeros:** El código puede ser verificado entre programadores, lo cual provoca que un *Open Source* mejore según se depure este código [1].
- **Transparencia:** Los programadores poseen una vista del 100% del código base. Por medio del código abierto, quienes se dedican al desarrollo de aplicaciones no presentan riesgos de bloqueos [2].
- **Flexibilidad:** Las empresas o usuarios que usen *Open Source* mantienen control total sobre las modificaciones del código [3].
- **Costo:** Al ser un código abierto, no presenta gastos de soporte, antivirus, actualización, entre otras [4].

Certificado x509

Los certificados x.509 son un documento digital, el cual se encarga de representar un usuario. Estos certificados son emitidos por una unidad certificadora (CA) y son un tipo de estándar de certificados de clave pública (PKI). Las aplicaciones principales del certificado x.509 son [5] [6]:

- Navegación web por medio de SSL y HTTPS, mientras la web se encuentre encriptada.
- Cifrar y firmar de forma electrónicamente correos electrónicos por medio del protocolo S/MIME.
- Firmar códigos o documentos electrónicos.
- Verifica la autentificación de cada uno de los clientes.
- Los gobiernos pueden emitir identificaciones, PKI.

2 METODOLOGÍA

Inicialmente, se realizó una investigación exhaustiva sobre el manejo de las herramientas DevOps y en especial de *Ansible*, así como el funcionamiento de un PKI. Dentro de esta investigación se tomó en cuenta otras herramientas de automatización aparte de *Ansible*, permitiendo tener un panorama amplio de las características de cada herramienta. De esta forma poder tener la conclusión de la herramienta viable para el desarrollo del presente trabajo.

Después de tener claro el funcionamiento de las herramientas de DevOps, se realizaron pruebas del despliegue del PKI directamente en la máquina principal. Para esta tarea se utiliza la herramienta OpenSSL la cual generan los diferentes tipos certificados, además se puede generar códigos hash o textos cifrados.

En base a las pruebas realizadas, se creó un *script* que realice el despliegue del PKI de forma local. Este *script* tuvo como objetivo realizar el despliegue de un PKI automáticamente sin que el usuario ingrese información manualmente.

Para proceder con el desarrollo del *playbook*, se debe instalar *Ansible* únicamente en la máquina principal y el protocolo SSH en ambas máquinas. Una vez que se tenga instalado lo antes mencionado, se guarda la dirección IP de la máquina secundaría y la clave pública para posteriormente realizar una prueba de conexión.

Se procede a desarrollar el *playbook*, el cual va a constar por dos *playbooks* en su interior. Ambos *plabooks* tienen variables de entrada, para que el usuario ingrese la información de la CA y del certificado para el usuario. Las tareas que va a cumplir este libro de jugadas son: Actualizar los repositorios, instalar *OpenSSL*, crear directorios, crear archivo serial, crear CA,

modificar archivo raíz *OpenSSL*, guardar variables, crear CSR, revisar CSR y firmar el certificado.

Finalmente se va a ejecutar el *playbook* desarrollado, el cual va a crear una carpeta en el escritorio del cliente que va a pertenecer a la CA. Esta carpeta contiene todo lo relacionado al PKI, en ella se va a encontrar el certificado que pertenece a la autoridad certificadora.

3 RESULTADOS

Dentro de este apartado se encuentra el desarrollo de los objetivos planteados, los cuales permitieron desplegar un PKI. Inicialmente se tiene información relevante sobre DevOps, *Ansible* y PKI, posteriormente la instalación de las herramientas necesarias, el desarrollo del *playbook* y la verificación del funcionamiento del despliegue del PKI.

3.1 Objetivo específico uno

Analizar la herramienta de DevOps que soluciona cada componente del proyecto de titulación. En primera instancia se realizará una investigación sobre el manejo de DevOps, *Ansible* y PKI, ya que con estas herramientas se procederá a desarrollar el trabajo de titulación.

DevOps

DevOps busca adoptar la automatización y el diseño de diferentes plataformas, que presten una metodología ágil y de alta calidad. Para lograr esto el equipo de desarrollo se ha unido con el de operaciones, permitiendo tener procesos de producción con mayor rapidez y eficiencia. Gracias a esto DevOps ha permitido dejar de lado la gestión manual para obtener una infraestructura dinámica [7].

La automatización de DevOps reduce el tiempo de desarrollo de aplicaciones o servicios, permitiendo obtener una escalabilidad mayor a la que se tendría con una aplicación convencional. La autogestión de DevOps convierte una red normal en una red flexible, al realizar cambios de configuración de servicios ante cualquier eventualidad [8].

DevOps posee múltiples ventajas, entre las que se destaca:

- **Rapidez:** Se tienen entregas de software son realizadas con mayor efectividad, dando resultados con una mejor calidad y estabilidad [9].
- **Rentabilidad:** Por medio de la optimización de trabajo se consigue una mejor productividad, gracias a la reducción de tiempos permitiendo nuevos proyectos [10].

- **Elimina cuellos de botella:** Gracias a que son procesos automatizados se puede ahorrar tiempo y costos, ya que se facilita el desarrollo de *software* [11].
- **Seguridad:** Se realizan auditorías y pruebas de seguridad dentro de los flujos de trabajos realizados por DevOps [9].

Herramientas de automatización

Salt Stack

SaltStack, se diferencia a otras herramientas de automatización por su velocidad y su diseño de procesamiento, permitiendo que se ejecuten varias tareas de forma simultánea. Esta herramienta de DevOps ejecuta tareas con mayor velocidad y se reduce los tiempos de implementación. Pese a que *SaltStack* posee características para brindar un buen servicio, presenta una configuración adicional dentro del cliente. Esta configuración consiste en la instalación del agente, lo que provoca que *SaltStack* no sea una herramienta de automatización completamente amigable con el usuario a diferencia de otras herramientas que simplifican el trabajo [12].

Jenkins

Jenkis es una herramienta de fácil instalación y configuración, mantiene integración continua y presenta automatización de pruebas con el objetivo de reducir errores. Posee aplicaciones versátiles gracias al manejo de *plugins* y su implementación dentro de plataformas basadas en la nube. Sin embargo, aunque contiene características resaltables, presenta limitaciones en su uso para los programadores, siendo la mayor limitación que los códigos solamente funcionaban si ha finalizado el proyecto, provocando retrasos en su entrega. Por este motivo dentro del presente trabajo de integración no se hace uso de esta herramienta [13] [14] [15].

Ansible

Ansible es una herramienta de automatización que se encarga de gestión de la configuración, preparación de infraestructura e implementación de aplicaciones. Para poder realizar las tareas anteriormente mencionadas, se desarrolla programas llamados *playbooks* que contiene pequeñas tareas a los que se les denominadas *tasks*. El uso de *playbooks* es la base de la automatización de esta herramienta, siendo el encargado de la implementación, orquestación y configuración en cada estación de trabajo [16] [17].

Ansible posee una ventaja sobre otras herramientas, la cual es que no necesita ningún agente o *software* en los equipos finales para su funcionamiento. En su lugar, se hace uso del protocolo SSH, de esta manera el equipo principal se puede conectar hacia otros terminales y ejecutar los *tasks* que se encuentran dentro del *playbook*. Por lo tanto, el uso de *Ansible* no requiere ningún tipo de instalación adicional en los equipos finales [17] [18].

PKI

La *Public Key Infraestructure (PKI)* es una infraestructura que se encarga de generar certificados digitales confiables, que verifican la identidad de un sitio remoto. Una autoridad certificadora (CA) es la encargada de validar la información del sitio, para proceder a firmar la solicitud. Después de que se firme, la información validada se convierte en un certificado digital, los cuales pueden ser validados por una clave pública. Los certificados generados protegen los datos de los usuarios y brindan un entorno seguro al momento de realizar una transacción [19].

Una PKI protege la información de los usuarios por medio de dos tipos de claves, una clave privada y una clave pública, siendo la primera la que se mantiene en secreto y la segunda la que se puede ser compartida. Dentro de una PKI, la clave pública es la que consta en la solicitud que se emite a la CA y en el certificado del cliente. Mientras que la clave privada únicamente se mantiene dentro del dispositivo. Una clave pública únicamente se usa para enviar información, mientras que la clave privada es la que permite leer la información en texto plano. La información se mantiene segura al no poder deducirse la una de la otra, pese a que mantengan una relación matemática [20] [21].

Infraestructura de una clave pública:

- **Certificado digital:** Archivo electrónico que tiene como función la verificación del titular, este certificado es autentificado a través de una autoridad certificadora [19].
- **Autoridad certificadora (CA):** Entidades que emiten los certificados o a su vez validan la información que se encuentra dentro de este [22].
- **Claves privadas y públicas:** Son utilizadas para poder cifrar o descifrar información, por lo que por medio de una clave privada se puede descifrar la información cifrada por la clave pública [23].
- **Solicitud de certificado:** Un usuario realiza una solicitud hacia una autoridad certificadora para poder validar su información y pueda generar un certificado electrónico.

3.2 Objetivo específico dos

Diseñar la solución para cada servicio de *networking* mediante herramientas de DevOps. Ya que se tiene claro el panorama de que se va a implementar se procede a diseñar el *playbook* en particular el cual será la encargada de implementar PKI.

Instalar *OpenSSL*

En el presente trabajo de titulación se busca generar certificados digitales, por lo que se hace uso de la herramienta *OpenSSL*. Esta herramienta es de código abierto, además posee la infraestructura necesaria para la generación de los certificados que se requieren para la PKI. *OpenSSLl* será instalada en un Sistema Operativo Linux con una distribución Ubuntu 22.04 LTS. Para realizar la instalación de esta herramienta se utiliza el comando ***sudo apt install openssl -y***, tal como se muestra en la Figura 3.1.

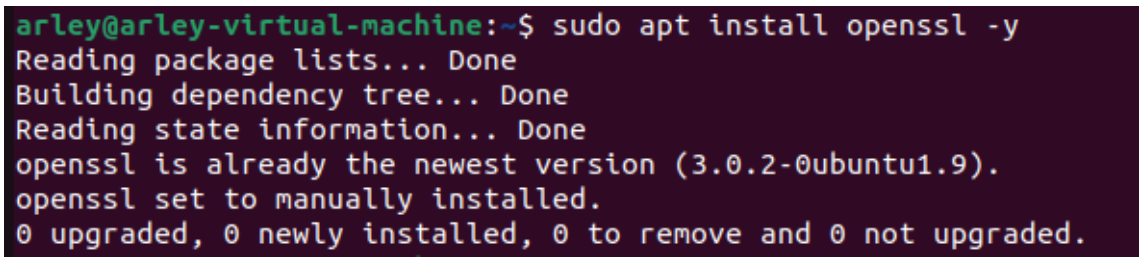

Figura 3.1 Instalación de *OpenSSL*

Instalar *Ansible*

Tal como se mencionó en el en el numeral 3.1, se utiliza la herramienta de automatización *Ansible*, la cual tiene como único requisito el protocolo SSH, el cual se consigue por medio del comando ***sudo apt install openssh-server***. Una vez instalado el servidor SSH, se realiza una prueba de conexión desde el servidor principal hacia el clente.

Para hacer uso de la herramienta de automatización *Ansible*, requiere instalar los repositorios oficiales de la herramienta para proceder a la descarga de la última versión. Para la instalación de *Ansible* se ejecuta el comando ***sudo apt install ansible -y***, tal como se muestra en la Figura 3.2. Se corrobora la instalación de la herramienta ejecutando el comando ***ansible --version*** y de esta manera poder obtener la información que se tiene dentro de la Figura 3.3.

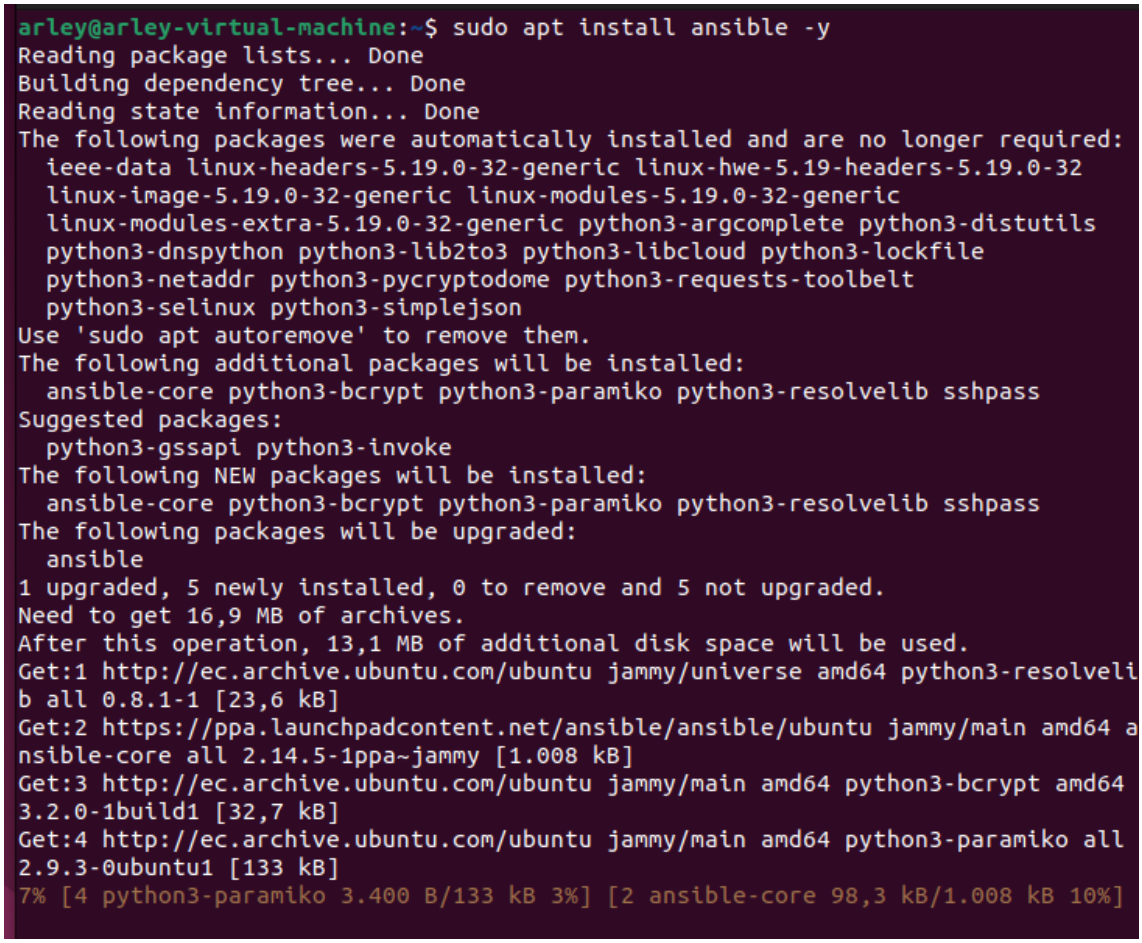

Figura 3.2 Instalación de *Ansible*

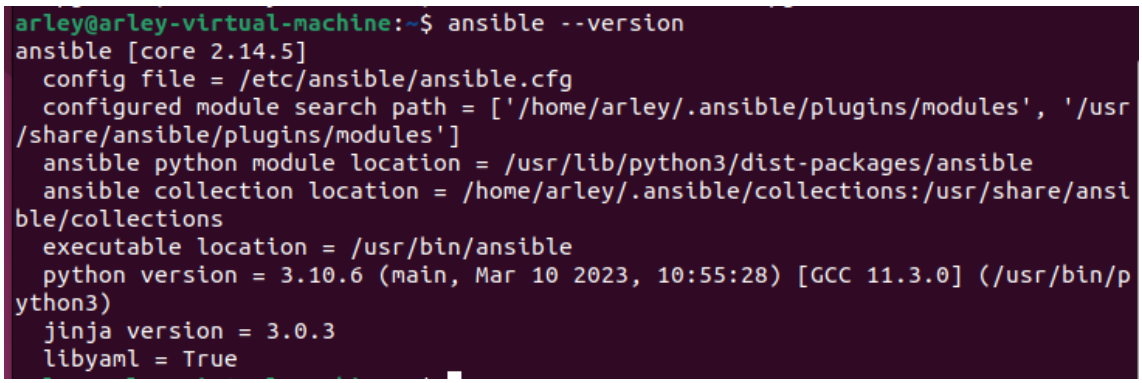

Figura 3.3 Revisión de la versión de *Ansible*

Ansible se puede conectar a varios *hosts* simultáneamente. Para realizar esta tarea se crea un archivo que funcione como inventario en donde se incluyen las direcciones IP de los servidores. Por medio del editor nano se crea este inventario con el comando ***sudo nano /etc/ansible/hosts***, para este trabajo de titulación se ingresa la información que se muestra en la Figura 3.4. En la sección de *[servers]* se introducen los servidores a los que se desea conectar y dentro de la sección *[phyton]* se utiliza *Phytonh3* como interprete.

```
[servers]
server1 ansible_host=192.168.226.129

[all:vars]
ansible_python_interpreter=/usr/bin/python3
```

Figura 3.4 Inventario de servidores a los que se va a conectar

Ansible utiliza una conexión segura mediante SSH y este protocolo requiere autentificación de usuario y contraseña. Por lo que se requiere una configuración entre el servidor y los clientes indicados dentro del inventario. Inicialmente se configura una clave para *Ansible* con ***ssh-keygen -t rsa -b 4096 -C "Ansible key"***, y por último se guardará la contraseña de cada servidor ingresado al inventario por medio del comando ***ssh-copy-id*** tal como se muestra en Figura 3.5.

```
arley@arley-virtual-machine:~$ ssh-copy-id arley@192.168.226.129
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed

/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist
on the remote system.
                (if you think this is a mistake, you may want to use -f option)
```

Figura 3.5 Guardar las contraseñas de los servidores

Para verificar que *Ansible* funciona correctamente con el host ingresado en el inventario y que la configuración de claves SSH fue exitosa, se ejecuta el comando ***ansible all -m ping -u arley***. Una vez que no existan errores en la configuración el comando arrojara la información que se muestra en Figura 3.6.

```
arley@arley-virtual-machine:~$ sudo usermod -L ansible
arley@arley-virtual-machine:~$ ansible all -m ping -u arley
server1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Figura 3.6 Prueba de conexión entre servidor y cliente en *Ansible*

Una vez que se tenga las herramientas necesarias para interconectar el servidor principal con el resto de los clientes, se crea un *playbook* por medio del comando ***gedit nombre_playbook.yml***. Dentro de este archivo se debe crear las tareas requeridas para crear un PKI. Para mantener un orden dentro de la carpeta en donde se va desarrolla todas las tareas ejecutadas por *Ansible*, se crean diferentes carpetas.

Las carpetas que van a crearse son las que se mencionan a continuación: *private*, *newcerts*, *crl* y *certs*. Cada una de las carpetas creadas, tiene una función en específico para el desarrollo de este trabajo. A continuación, se menciona la función que cada carpeta va a cumplir:

- La carpeta *private* almacenará las claves privadas que corresponden a cada certificado emitido.
- La carpeta *crl* tendrá un listado de los certificados que hayan sido revocados, es decir los que ya no son válidos.
- La carpeta *newcerts* contendrá los certificados nuevos que sean emitidos por la CA y que no ha sido confirmada por la CRL.
- La carpeta *certs* tendrá los certificados emitidos por la CA y que ya han sido confirmados por la CRL.

Para que una CA pueda funcionar, se requiere que exista un contador de los certificados que han sido emitidos. Para esto se crea un archivo serial, dentro de este archivo se va a llevar la cuenta de los certificados que hayan sido firmados.

3.3 Objetivo específico tres

Implementar las soluciones mediante herramientas de DevOps para el despliegue de los servicios de *networking*. Se implementará la aplicación para generación de PKI en un ambiente virtualizado.

A continuación, se muestra el contenido del *playbook* creado por medio de *Ansible*:

Al inicio del *playbook* se coloca el nombre de este, quienes los servidores que pueden acceder y si necesita ejecutar las distintas tareas con permisos de *root*. Seguido se incluirán las variables que van a ser ingresadas por el usuario, tal como se puede visualizar en la Figura 3.7 y en la Figura 3.8.

```yaml
#Servidor PKI
---
- name: Configuración CA de PKI
  hosts: all
  become: yes

  vars_prompt:
    #Dirección donde se crea el PKI
    - name: direccion
      prompt: "Ingrese la ruta del directorio para almacenar el PKI"
      private: false

    #Nombre de carpeta del PKI
    - name: carpeta
      prompt: "Indique el nombre de su carpeta"
      private: false

    #Nombre de certificado CA
    - name: CertificadoCA
      prompt: "Ingrese el nombre para su certificado CA"
      private: false

    #Nombre de llave privada CA
    - name: LlaveCA
      prompt: "Ingrese el nombre para su llave privada CA"
      private: false

    #Clave de certificado CA
    - name: ClaveCA
      prompt: "Ingresa la frase de contraseña PEM"
      encrypt: sha512_crypt
      private: yes

    #País de CA
    - name: pais
      prompt: "Ingrese su país (EU)"
      private: no

    #Provincia CA
    - name: provincia
      prompt: "Ingrese su Provincia"
      private: no
```

Figura 3.7 Variables de entrada CA (Parte 1)

```
#Ciudad CA
- name: ciudad
  prompt: "Ingrese su ciudad"
  private: no

#Empresa CA
- name: empresa
  prompt: "Ingrese el nombre de su empresa"
  private: no

#Seccion CA
- name: seccion
  prompt: "Ingrese su departamento/dirección"
  private: no

#Dominio CA
- name: dominio
  prompt: "Ingrese su dominio (example.com)"
  private: no
```

Figura 3.8 Variables de entrada CA (Parte 2)

Al usar Ubuntu se necesita actualizar los repositorios. Dentro del *playbook* creado se realiza esta acción tal como se muestra en la Figura 3.9.

```
tasks:
  #Actualizar repositorios
  - name: Actualizar repositorios
    apt:
      update_cache: yes
      upgrade: yes
      cache_valid_time: 3600
```

Figura 3.9 Actualizar repositorios en el *playbook*

Este trabajo de titulación tal como se mencionó en el numeral 3.2, utilizará la herramienta *OpenSSL* para la implementación de un PKI. Por lo que se requiere que el cliente tenga esta herramienta en el equipo, la cual va a ser instalado por la tarea que se evidencia dentro de la Figura 3.10.

```
#Instalar OpenSSL
- name: Instalar OpenSSL
  apt:
    name: openssl
```

Figura 3.10 Instalación *OpenSSL* en *Ansible*

Para corroborar la instalación de la herramienta *OpenSSL*, se verifica la versión de esta. En la Figura 3.11 la primera tarea verifica la versión, para guardarla en una variable y la segunda muestra al usuario la información que está dentro de la variable.

```
#Verificar version de OpenSSL
- name: Verificar version de OpenSSL
  command: openssl version
  register: version_openssl

#Mostrar versión de OpenSSL
- name: Mostrar versión de OpenSSL
  debug:
    msg: "{{ version_openssl.stdout }}"
```

Figura 3.11 Verificación de la versión del *OpenSSL* en el *playbook*

En la Figura 3.12 se crean las carpetas mencionadas en el numeral 3.2 mediante el *playbook* en *Ansible*. Estas carpetas requieren tener permisos de edición por lo que dentro de la tarea se indica esto en el parámetro ***mode: '0777'***.

```
#Crear carpetas de certificados
- name: Crear carpetas
  ansible.builtin.file:
    path: "{{ item.path }}"
    state: directory
    mode: '0777'
  loop:
    - { path: "{{ direccion}}/{{ carpeta }}/certs" }
    - { path: "{{ direccion}}/{{ carpeta }}/crl" }
    - { path: "{{ direccion}}/{{ carpeta }}/newcerts" }
    - { path: "{{ direccion}}/{{ carpeta }}/private" }
```

Figura 3.12 Carpetas creadas mediante *Ansible*

En el numeral 3.2 se mencionó que la CA necesita un archivo que lleve la contabilidad de los certificados generados. La Figura 3.13 contiene las tareas que se necesita para crear el archivo serial.

```
#Crear bases de datos de certificados
- name: Crear y establecer permisos de index.txt
  ansible.builtin.file:
   path: "{{ carpeta }}/index.txt"
   state: touch
   mode: '0777'

- name: Crear y establecer permisos del archivo serial
  ansible.builtin.file:
   path: "{{ carpeta }}/serial"
   state: touch
   mode: '0777'

- name: Agregar contenido al archivo
  ansible.builtin.copy:
    dest:  "{{ carpeta }}/serial"
    content: |
     C001
```

Figura 3.13 Creación del archivo serial en el *playbook*

La Figura 3.14 presenta la tarea para la creación del certificado de la autoridad certificadoras. El certificado a crear requiere el estándar x.509, mencionado en el marco teórico, dentro de esta tarea se va a incluir las variables ingresadas por el usuario al inicio del *playbook*.

```
#Crear certificado CA
- name: Crear certificado CA
  command: openssl req -new
       -x509
       -subj "/C={{ pais }}/ST={{ provincia }}/L={{ ciudad }}/
O={{ empresa }}/OU={{ seccion }}/CN={{ dominio }}"
       -keyout {{ direccion}}/{{ carpeta }}/private/{{ LlaveCA }}.pem
       -out {{ direccion}}/{{ carpeta }}/{{ CertificadoCA }}.pem \
       -passout pass:{{ ClaveCA }}
```

Figura 3.14 Creación del certificado de la CA en playbook

Para que la autoridad certificadora pueda firmar las solicitudes CSR realizadas, se requiere que se modifique el archivo raíz de *OpenSSL*. En el archivo raíz se va a realizar modificaciones en los siguientes parámetros:

- *dir:* Dirección en donde se encuentra la carpeta que guarda el PKI,
- *certificate:* Nombre del certificado de la entidad certificadora y
- *private_key:* Nombre de la llave de la entidad certificadora

Estas modificaciones mencionadas se las realiza por la tarea de la Figura 3.16. Dentro del presente trabajo se crea un respaldo del archivo raíz de la herramienta *OpenSSL* antes de que sea modificado, tal como se muestra en la Figura 3.15.

```
#Crear una copia del archivo raíz
- name: Crear copia de openssl.conf
  command: sudo cp openssl.cnf openssl.cnf.bk2
  args:
    chdir: /usr/lib/ssl/
```

Figura 3.15 Copia de seguridad de archivo raíz en el *playbook*

```
#Modificar archivo raíz
- name: Modificar archivo .cnf
  ansible.builtin.replace:
    path: /usr/lib/ssl/openssl.cnf
    regexp: './demoCA'
    replace: '{{ direccion }}{{ carpeta }}'

- name: Modificar archivo .cnf
  ansible.builtin.replace:
    path: /usr/lib/ssl/openssl.cnf
    regexp: '/cacert.pem'
    replace: '/{{ CertificadoCA }}.pem'

- name: Modificar archivo .cnf
  ansible.builtin.replace:
    path: /usr/lib/ssl/openssl.cnf
    regexp: '/private/cakey.pem'
    replace: '/private/{{ LlaveCA }}.pem'
```

Figura 3.16 Modificar archivo raíz en *Ansible*

El *playbook* desarrollado va a funcionar con dos *playbooks* diferentes en su interior. El primero se encarga de crear la autoridad certificadora y realizar las modificaciones en el archivo raíz. El segundo *playbook* realiza la solicitud CSR y la firma de esta. Para que las variables ingresadas inicialmente funcionen dentro del segundo *playbook*, estas deben ser almacenadas tal como se muestra en la Figura 3.17.

```
#Variables
- name: Almacenar variables
  set_fact:
    direccion: "{{ direccion }}"
    carpeta: "{{ carpeta }}"
    pais: "{{ pais }}"
    provincia: "{{ provincia }}"
    ciudad: "{{ ciudad }}"
    empresa: "{{ empresa }}"
    ClaveCA: "{{ ClaveCA }}"
```

Figura 3.17 Guardar variables del primer *playbook*

Al igual que en el comienzo del archivo, el segundo *playbook* va a contar con un nombre propio, los servidores a los que puede conectarse y si requiere permisos de *root*. Igualmente va a

requerir variables que también van a ser ingresadas por el usuario. Esto mencionado se puede evidenciar en la Figura 3.18.

```
#Cliente PKI

- name: Configuración cliente de PKI
  hosts: all
  become: yes

  vars_prompt:

    #Nombre de certificado Usuario
    - name: CertificadoUsuario
      prompt: "Indique el nombre del certificado"
      private: false

    #Nombre de llave Usuario
    - name: LlaveUsuario
      prompt: "Indique el nombre de la llave"
      private: false

    #Clave para certificado Usuario
    - name: clave1
      prompt: "Ingresa la frase de contraseña PEM"
      encrypt: sha512_crypt
      private: yes

    #Seccion Usuario
    - name: seccion2
      prompt: "Ingrese su departamento"
      private: no

    #Nombre Usuario
    - name: nombre
      prompt: "Ingrese su nombre (Nombre y Apellido)"
      private: no
```

Figura 3.18 Variables de entrada del usuario en el *playbook*

En la Figura 3.19 se tiene la tarea con la que se va a enviar la solicitud hacia la entidad certificadora. Esta tarea tiene la clave privada del usuario y el certificado de este.

```
  tasks:
    #Crear un certificado para usuario
    #Crear solicitud de certificado (CSR)
    - name: Crear solicitud certificado
      command:
        cmd: openssl req
            -new
            -keyout {{ LlaveUsuario }}.pem
            -subj "/C={{ pais }}/ST={{ provincia }}/L={{ ciudad }}/
O={{ empresa }}/OU={{ seccion2 }}/CN={{ nombre }}"
            -out {{ CertificadoUsuario }}-req.pem
            -passout pass:{{ clave1 }}
        chdir: "{{ direccion }}{{ carpeta }}/"
```

Figura 3.19 Realizar la solicitud de certificado CSR en el *playbook*

Antes de que los certificados puedan ser firmados, es necesario que la autoridad certificadora valide la información. La Figura 3.20 posee 2 tareas en la que la primera guarda en una variable la información de la solicitud y la segunda muestra la información dentro de la variable.

```
#Revisar solicitud (CSR)
- name: Revisar solicitud (CSR)
  ansible.builtin.command:
    cmd: openssl req
       -text
       -in "{{ direccion }}{{ carpeta }}/{{ CertificadoUsuario }}-req.pem"
    chdir: "{{ direccion }}{{ carpeta }}/"
  register: solicitud

- name: Imprimir resultado de certificado
  ansible.builtin.debug:
    msg: "{{ solicitud.stdout }}"
```

Figura 3.20 Revisación de la solicitud del certificado en el *playbook*

Finalmente, en la Figura 3.21 se observa la tarea encargada de firmar la solicitud realizada por el usuario hacia la autoridad certificadora.

```
    #Firmar el certificado
    - name: Firmar certificado
      command:
        cmd: openssl ca
           -in {{ CertificadoUsuario }}-req.pem
           -out {{ direccion}}/{{ carpeta }}/certs/{{ CertificadoUsuario }}.pem
           -passin pass:{{ ClaveCA }}
           -subj "/C={{ pais }}/ST={{ provincia }}/L={{ ciudad }}/
O={{ empresa }}/OU={{ seccion2 }}/CN={{ nombre }}"
           -batch
        chdir: "{{ direccion }}{{ carpeta }}/"
```

Figura 3.21 Firmar el certificado en el *playbook*

3.4 Objetivo específico cuatro

Verificar el funcionamiento de cada servicio de *networking* implementado mediante DevOps. Finalmente se realizará pruebas del algoritmo y se verificará su buen funcionamiento. Verificar el funcionamiento de cada servicio de *networking* implementado mediante DevOps.

Una vez que el *playboook* se encuentre completamente terminado, se realizan diferentes pruebas para verificar su funcionamiento.

Ejecución del playbook

A continuación, se ejecuta el *playbook* por medio del comando ***ansible-playbook resultado.yml -Kk*** tal como muestra la Figura 3.22. En el terminale observa como el usuario ingresa la información solicitada por las variables dentro del *playbook*. Dentro de esta misma figura, se observa que no existe ningún inconveniente para leer las tareas creadas.

```
arley@arley-virtual-machine:~/Desktop$ ansible-playbook resultado.yml -Kk
SSH password:
BECOME password[defaults to SSH password]:
Ingrese la ruta del directorio para almacenar el PKI: /home/arley/Desktop/
Indique el nombre de su carpeta: ESFOTCA
Ingrese el nombre para su certificado CA: ESFOT_CERT
Ingrese el nombre para su llave privada CA: ESFOT_KEY
Ingresa la frase de contraseña PEM:
[DEPRECATION WARNING]: Encryption using the Python crypt module is deprecated.
The Python crypt module is deprecated and will be removed from Python 3.13.
Install the passlib library for continued encryption functionality. This
feature will be removed in version 2.17. Deprecation warnings can be disabled
by setting deprecation_warnings=False in ansible.cfg.
Ingrese su país (EU): EC
Ingrese su Provincia: PICHINCHA
Ingrese su ciudad: QUITO
Ingrese el nombre de su empresa: EPN
Ingrese su departamento/dirección: TI
Ingrese su dominio (example.com): ESFOT.COM

PLAY [Configuración CA de PKI] *************************************************

TASK [Gathering Facts] *********************************************************
ok: [server1]
```

Figura 3.22 Ingreso de la información CA en el *playbook*

En la Figura 3.23 se evidencia que el cliente no tuvo ningún inconveniente para realizar la actualización de los repositorios.

```
TASK [Actualizar repositorios] *************************************************
ok: [server1]
```

Figura 3.23 Actualización del repositorios en el cliente

Instalación de *OpenSSL*

Para poder desplegar un PKI, se requiere que la máquina cliente tenga instaladas *OpenSSL*. En la Figura 3.24 se evidencia que esta herramienta ha sido instalada y se puede observar la versión que tiene.

```
TASK [Instalar OpenSSL] ********************************************************
ok: [server1]

TASK [Verificar version de OpenSSL] ********************************************
changed: [server1]

TASK [Mostrar versión de OpenSSL] **********************************************
ok: [server1] => {
    "msg": "OpenSSL 1.1.1f  31 Mar 2020"
}
```

Figura 3.24 Instalación *OpenSSL* en el cliente

Creación de la Unidad Certificadora (CA)

Tal como se mencionó anteriormente, se requiere crear carpetas para mantener una correcta organización dentro del cliente. Las carpetas han sido creadas dentro de la dirección indicada y con el nombre ingresado, la creación de estos directorios se aprecia en la Figura 3.25.

```
TASK [Crear carpetas] **********************************************************
changed: [server1] => (item={'path': '/home/arley/Desktop//ESFOTCA/certs'})
changed: [server1] => (item={'path': '/home/arley/Desktop//ESFOTCA/crt'})
changed: [server1] => (item={'path': '/home/arley/Desktop//ESFOTCA/newcerts'})
changed: [server1] => (item={'path': '/home/arley/Desktop//ESFOTCA/private'})
```

Figura 3.25 Creación de los directorios en el cliente

Se corrobora la creación de los archivos seriales que contabilizan los certificados que la unidad certificadora emita dentro de la Figura 3.26.

```
TASK [Crear y establecer permisos de index.txt] ********************************
changed: [server1]

TASK [Crear y establecer permisos del archivo serial] **************************
changed: [server1]

TASK [Agregar contenido al archivo] ********************************************
changed: [server1]
```

Figura 3.26 Creación de archivos seriales en el cliente

La Figura 3.27 indica que la creación del certificado de la Unidad Certificadora no ha tenido ningún tipo de problema.

Figura 3.27 Creación de la Unidad Certificadora en cliente

Modificar archivo raíz *OpenSSL*

El archivo raíz de *OpenSSL* es modificado para que la unidad certificadora pueda firmar las solicitudes recibidas. Esta modificación se la evidencia en la Figura 3.28.

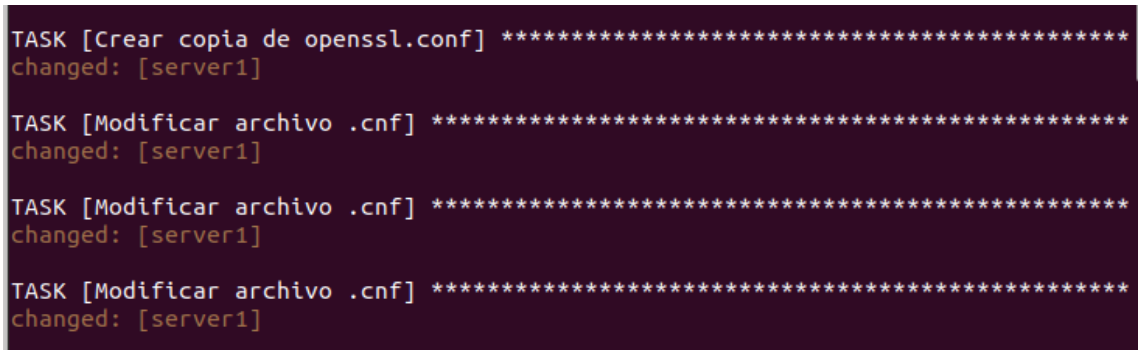

Figura 3.28 Modificación del archivo raíz de *OpenSSL*

Almacenamiento de variables

En la Figura 3.29 se muestra como las variables del primer *playbook*, han sido guardadas para poderlas usar en el segundo.

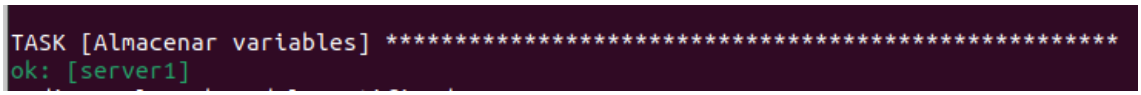

Figura 3.29 Almacenar variables en el *playbook*

Ingresar información del certificado del usuario

La Figura 3.30 muestra la información que va a tener el certificado del usuario, el cual va a ser emitido por la unidad certificadora.

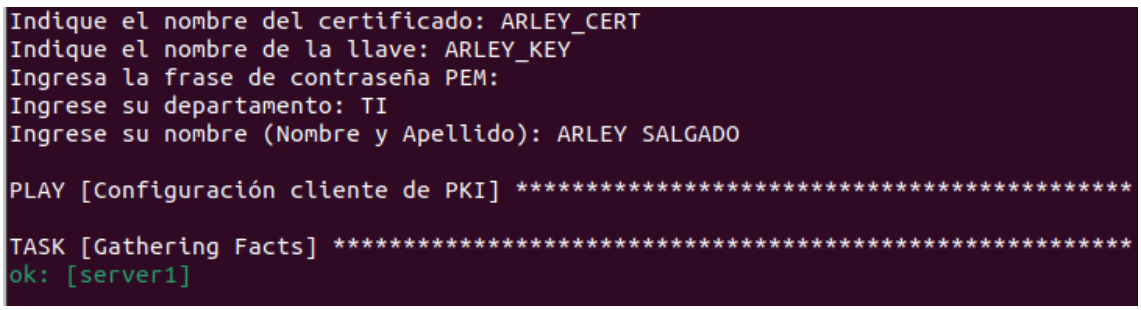

Figura 3.30 Información del certificado del usuario

Crear una solicitud (CSR)

La solicitud creada es enviada hacia la unidad certificadora, tal como muestra la Figura 3.31.

```
TASK [Crear solicitud certificado] ********************************************
changed: [server1]
```

Figura 3.31 Creación de CSR en el cliente

Revisión del CSR

Tal como ya se había mencionado anteriormente, la unidad certificadora necesita verificar el contenido de la solicitud para que esta pueda ser firmada. En la Figura 3.32 se puede verificar la información que va a tener el certificado que será emitido hacía el usuario.


```
TASK [Revisar solicitud (CSR)] ************************************************
changed: [server1]

TASK [Imprimir resultado de certificado] **************************************
ok: [server1] => {
    "msg": "Certificate Request:\n    Data:\n        Version: 1 (0x0)\n        Subject: C
= EC, ST = PICHINCHA, L = QUITO, O = EPN, OU = TI, CN = ARLEY SALGADO\n        Subject Pub
lic Key Info:\n            Public Key Algorithm: rsaEncryption\n                RSA Public
-Key: (2048 bit)\n                Modulus:\n                    00:bf:1c:fc:03:05:e5:09:d1
:23:9e:be:1a:a6:77:\n                    8a:c0:58:e8:a5:52:d3:5e:0a:c0:3c:1f:24:df:2d:\n
                    0e:0b:2b:8d:82:73:01:76:6f:21:75:fb:34:ac:fc:\n                    f9:f1
:35:26:23:71:70:bc:b6:db:bf:56:62:f6:e1:\n                    1c:f9:e1:0d:e8:00:46:2c:c8:4
0:f7:64:ed:00:9a:\n                    b2:a9:6e:cb:10:91:55:7e:77:d5:38:08:03:76:31:\n
                1a:a2:45:4a:82:9e:54:f0:43:c3:c7:3e:b2:b5:b0:\n                    49:f7:3
a:1e:fa:7e:b0:cb:82:82:a3:a2:35:2b:9c:\n                    ca:0d:ba:5e:e3:31:10:4f:af:2e:
f1:eb:a1:60:15:\n                    1f:7b:e0:23:8c:36:ad:08:25:b7:c2:f6:44:dd:3b:\n
            82:a1:25:41:55:87:b8:08:54:4e:c5:0b:59:46:d2:\n                    95:8b:d3:
56:a3:e1:d3:69:33:6b:7e:eb:8e:c3:5a:\n                    4b:2b:5b:41:1c:6a:1e:43:35:9c:a8
:04:18:a8:6a:\n                    2a:3d:bc:bd:a2:d9:16:4d:43:bf:52:90:df:ee:2c:\n
            a8:79:41:08:10:24:09:ce:08:0d:8b:bd:30:20:8c:\n                    e8:90:aa:5a
:e1:7b:0e:2e:f1:dd:d8:c1:90:fc:b1:\n                    dc:27:59:47:52:07:53:72:c4:5a:b0:7
f:05:d6:b2:\n                    d0:d9\n                Exponent: 65537 (0x10001)\n
 Attributes:\n            a0:00\n    Signature Algorithm: sha256WithRSAEncryption\n
  73:7f:ce:06:c7:b5:f9:70:17:d1:4b:f6:1d:17:9c:7b:01:f4:\n         5f:b9:06:eb:59:1f:5a:c9
:b5:a3:14:db:81:36:9f:aa:f6:af:\n         07:c2:b9:86:41:c2:4f:0a:bd:ae:3c:61:a8:8e:70:67:
02:0e:\n         52:61:9d:72:16:59:6d:2f:c4:fc:65:9d:b8:28:69:27:02:52:\n         b0:af:0a
:63:2b:fd:8a:ea:9d:e6:0e:97:a2:08:64:a3:d2:65:\n         1c:94:03:20:06:96:d4:3f:b4:27:d4:
5b:87:17:9b:1a:f3:86:\n         c8:89:7e:82:09:03:1f:98:ee:9a:3a:b7:40:4b:a5:e3:07:14:\n
       e5:14:61:b3:b0:0b:e6:8a:f9:8e:79:a5:95:28:4d:5a:be:60:\n         38:9e:1f:b8:59:51:
c7:76:3f:ac:e7:2f:5b:6f:59:30:d4:67:\n         76:9a:88:33:b0:4e:a6:f5:81:62:0f:e5:90:bf:5
5:05:2e:d6:\n         a3:78:8b:44:66:b6:fa:1d:db:95:21:80:77:e2:6c:9c:e9:90:\n         68:
09:3a:1a:9d:81:84:26:bc:05:8b:85:2d:75:20:c6:4b:25:\n         c3:6f:12:0b:a6:bf:02:0e:fd:2
c:7b:4c:07:21:0c:8f:93:c7:\n         76:5f:c2:1c:f5:fa:a7:f9:92:c8:f7:b2:1a:d6:21:e2:bc:89
:\n         e2:f5:6d:a4\n-----BEGIN CERTIFICATE REQUEST-----\nMIICqTCCAZECAQAwZDELMAkGA1UE
BhMCRUMxEjAQBgNVBAgMCVBJQ0hJTkNIQTEO\nMAwGA1UEBwwFUVVJVE8xDDAKBgNVBAoMA0VQTjELMAkGA1UECwwC
VEkxFjAUBgNV\nBAMMDUFSTEVZIFNBTEdBRE8wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB\nAQC/HPwDBe
UJ0SOevhqmd4rAWOilUtNeCsA8HyTfLQ4LK42CcwF2byF1+zSs/Pnx\nNSYjcXC8ttu/VmL24Rz54Q3oAEYsyED3ZO
0AmrKpbssQkVV+d9U4CAN2MRqiRUqC\nnlTwQ8PHPrK1sEn3Oh76frDLgoKjojUrnMoNul7jMRBPry7x66FgFR974C
OMNq0I\nJbfC9kTdO4KhJUFVh7gIVE7FC1lG0pWL01aj4dNpM2t+647DWksrW0Ecah5DNZyo\nBBioaio9vL2i2RZN
Q79SkN/uLKh5QQgQJAnOCA2LvTAgjOiQqlrhew4u8d3YwZD8\nsdwnWUdSB1NyxFqwfwXWstDZAgMBAAGgADANBgkq
hkiG9w0BAQsFAAOCAQEAc3/O\nBse1+XAX0Uv2HRecewH0X7kG61kfWsm1oxTbgTafqvavB8K5hkHCTwq9rjxhqI5w
\nZwIOUmGdchZZbS/E/GWduChpJwJSsK8KYyv9iuqd5g6Xoghko9JlHJQDIAaW1D+0\nJ9RbhxebGvOGyIl+ggkDH5
jumjq3QEul4wcU5RRhs7AL5or5jnmllShNWr5gOJ4f\nuFlRx3Y/rOcvW29ZMNRndpqIM7BOpvWBYg/lkL9VBS7Wo3
iLRGa2+h3blSGAd+Js\nnOmQaAk6Gp2BhCa8BYuFLXUgxkslw28SC6a/Ag79LHtMByEMj5PHdl/CHPX6p/mS\nyPey
GtYh4ryJ4vVtpA==\n-----END CERTIFICATE REQUEST-----"
```


Figura 3.32 Revisión del CSR en el cliente

Firmar certificado

La Figura 3.33 muestra que el *playbook* creado funciona correctamente al haber finalizado el mismo, con la firma de la solicitud.

```
TASK [Firmar certificado] ******************************************************
changed: [server1]

PLAY RECAP *********************************************************************
server1                    : ok=22   changed=14   unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

Figura 3.33 Firmar certificado en el cliente

Verificación del despliegue del PKI en el cliente

La Figura 3.34 muestra que la carpeta creada en la dirección ingresada al inicio del *playbook*. Así como también, se puede evidenciar que tiene el nombre que el usuario deseo darle a esta carpeta.

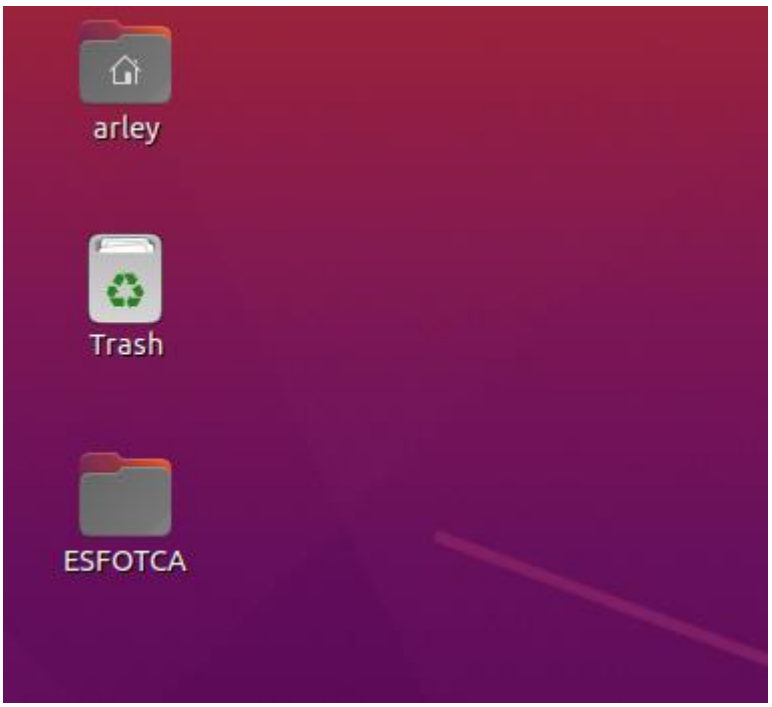

Figura 3.34 Carpeta PKI (Parte1)

En la Figura 3.35 muestra el interior de la carpeta que pertenece al despliegue del PKI. En su interior encontramos lo siguiente:

- Los directorios creados.
- El certificado de la unidad certificadora.
- La solicitud emitida hacia la CA.
- Los archivos que llevan el conteo de certificados emitidos.

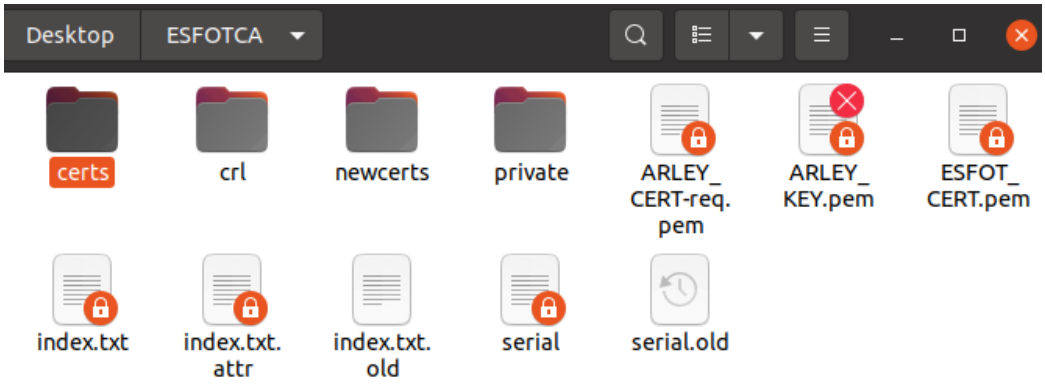

Figura 3.35 Carpeta PKI (Parte2)

Al interior de la carpeta *certs* se tiene el certificado que fue emitido por la CA hacia el usuario, tal como se muestra en la Figura 3.36

Figura 3.36 Carpeta PKI (Parte3)

Por último, se evidencia el contenido de los certificados creados tanto de la CA como la emitida al usuario. La Figura 3.37 contiene el certificado que pertenece a la unidad certificadora y la Figura 3.38 el certificado del usuario.

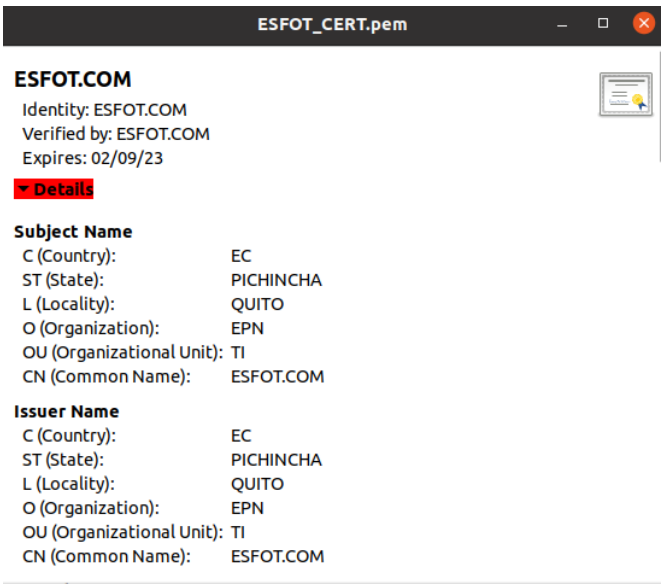

Figura 3.37 Certificado CA

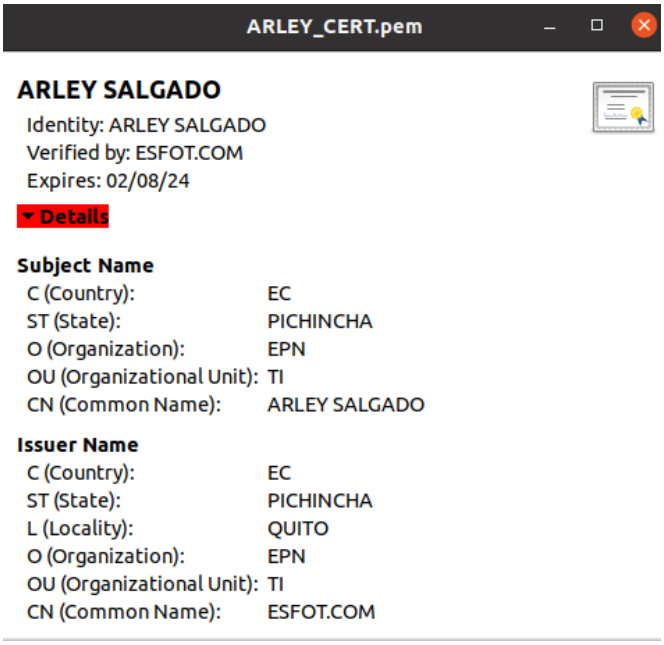

Figura 3.38 Certificado usuario

4 CONCLUSIONES

- El levantamiento de información sobre el funcionamiento de las herramientas DevOps, *Ansible* y sobre la estructura de un PKI, fue fundamental para despliegue de PKI.
- Un PKI es importante dentro de la seguridad en redes, ya que presenta una disminución en la probabilidad de ataques por parte de personas intermediarias y garantiza el acceso autorizado, no permitiendo que cualquier usuario pueda acceder a diferentes sistemas o datos.
- La herramienta de automatización *Ansible* es amigable para el usuario, ya que solo requiere hacer uso del protocolo SSH para lograr conectarse desde el nodo principal hacia los nodos secundarios y por medio de esto implementar el *playbook* creado dentro de ellos, permitiendo de esta forma automatizar procesos de implementación.
- La comunicación que se tiene entre el nodo principal y el nodo secundario es seguro gracias al protocolo SSH, ya que este mantiene la información cifrada y permite que solo los usuarios que fueron autentificados sean quienes adquieran la implementación que está dentro del *playbook.*
- Un archivo *playbook* consta de tareas que se ejecutan una por una dentro del cliente, siendo estas legibles para cualquier usuario, es decir que son fáciles de entender y fáciles de diseñar. Por esta razón es que este tipo de archivo es ideal para poder automatizar configuraciones, implementaciones o despliegues sin contar con un nivel alto de programación.

- *Ansible* es independiente de la distribución de *Linux* que se utilice, por lo que se garantiza su funcionamiento y compatibilidad, es decir si el nodo principal cuenta con *Ubuntu*, puede ejecutar el *playbook* dentro de un nodo secundario que funcione con *CentOS*, *Debian*, etc. Siempre y cuando todos los nodos cuenten con el protocolo SSH.
- *Ansible* permite al usuario del nodo principal, tener conocimiento de como avanza la implementación creada dentro del *playbook*, por medio de esto el usuario tiene seguridad de que el despliegue realizado ha sido exitoso o por lo contrario podrá evidenciar cual fue el error que existió durante el despliegue y que tarea tuvo ese error.
- La herramienta *OpenSSL* es esencial para el despliegue de un PKI, ya cuenta con protocolos que ayudan a la generación y gestión de certificados digitales, también brinda seguridad al usuario al proporcionar alta confidencialidad de datos e integridad de la comunicación existente.
- El despliegue de un PKI dentro de este trabajo de titulación se realiza por medio de la herramienta de *OpenSSL*, siendo esta la que permite crear y generar los certificados digitales, automatizando este despliegue mediante tareas creadas al interior de un *playbook*, el cual es ejecutado a través de la herramienta *Ansible*, demostrando como las herramientas DevOps facilitan los procesos de implementación dentro del área de TI.

5 RECOMENDACIONES

- Pasar los archivos *playbook* por un verificador de códigos YML, de esta manera se puede reducir problemas al correr el código, ya que este verificador se encarga de dar a conocer al usuario si el espaciado del del archivo es correcto.
- Realizar un *script* con todos los comandos relacionados al despliegue, de esta manera es más fácil entender cómo se puede automatizar procesos dentro de la línea de comandos.
- Es mejor crear un solo *playbook* en el que conste la parte del nodo principal y la parte del nodo secundario, en especial cuando se tienen variables de entrada, de esta manera se facilita el uso de las variables ingresadas.
- Realizar una investigación sobre los múltiples usos de *OpenSSL*, ya que este es ampliamente utilizado en la gestión de certificados digitales. Esta herramienta requiere que el usuario tenga conocimientos sobre seguridades y que este consciente de posibles vulnerabilidades.
- Tener en cuenta que *OpenSSL* necesita que su archivo raíz sea modificado con las direcciones en donde se crea la CA, de lo contrario no se podrá firmar la solicitud que se emite hacia la entidad certificadora al no saber en qué dirección está el certificado que le pertenece.

- Antes de realizar cualquier tipo de configuración en alguna distribución de *Linux*, lo primero que se debe hacer es actualizar los repositorios por medio de los comandos ***sudo apt upgrade*** y ***sudo apt update.***

6 REFERENCIAS BIBLIOGRÁFICAS

[1] «¿Qué es el open source o código abierto?» https://www.redhat.com/es/topics/open-source/what-is-open-source (accedido 19 de junio de 2023).

[2] «Open source: sus 3 ventajas y 3 desventajas», *Zendesk MX*. https://www.zendesk.com.mx/blog/que-es-open-source/ (accedido 19 de junio de 2023).

[3] «Cuatro ventajas del desarrollo Open Source para las empresas - MuyComputerPRO». https://www.muycomputerpro.com/2023/03/30/cuatro-ventajas-del-desarrollo-open-source-para-las-empresas (accedido 19 de junio de 2023).

[4] «10 razones por las que elegir una solución Open Source». https://www.chakray.com/es/razones-ventajas-open-source/ (accedido 19 de junio de 2023).

[5] «¿Qué es un certificado X.509? - SSL.com». https://www.ssl.com/es/preguntas-frecuentes/%C2%BFQu%C3%A9-es-un-certificado-x-509%3F/ (accedido 12 de agosto de 2023).

[6] «Certificado X.509. ¿Qué es y para qué se usa? | Grupo Atico34». https://protecciondatos-lopd.com/empresas/certificado-x509/ (accedido 12 de agosto de 2023).

[7] «¿Qué es la automatización de DevOps?» https://www.redhat.com/es/topics/automation/what-is-DevOps-automation (accedido 12 de agosto de 2023).

[8] T. Das, «¿Qué es la automatización de DevOps? Una inmersión profunda en sus tipos, casos de uso y las 5 mejores herramientas», *Geekflare*, 26 de diciembre de 2022. https://geekflare.com/es/DevOps-automation/ (accedido 12 de agosto de 2023).

[9] «¿Qué es DevOps? | Atlassian». https://www.atlassian.com/es/DevOps (accedido 19 de junio de 2023).

[10] «10 ventajas de implementar DevOps». https://www.ambit-bst.com/blog/10-ventajas-de-implementar-DevOps (accedido 19 de junio de 2023).

[11] «5 Ventajas de DevOps para tus proyectos». https://keepcoding.io/blog/ventajas-de-DevOps/ (accedido 12 de agosto de 2023).

[12] «What is SaltStack? | Definition from TechTarget», *IT Operations*. https://www.techtarget.com/searchitoperations/definition/SaltStack (accedido 12 de agosto de 2023).

[13] «Jenkins», *Jenkins*. https://www.jenkins.io/ (accedido 12 de agosto de 2023).

[14] Sentrio, «Introducción a Jenkins: ¿qué es, para qué sirve y cómo funciona?», *Sentrio*, 16 de septiembre de 2021. https://sentrio.io/blog/que-es-jenkins/ (accedido 12 de agosto de 2023).

[15] P. Á. Corredera, «¿Qué es Jenkins?, Herramienta de Integración Continua», *CIBERNINJAS*, 22 de mayo de 2020. https://ciberninjas.com/jenkins/ (accedido 12 de agosto de 2023).

[16] R. KeepCoding, «¿Qué es Ansible? Cómo iniciarte en esta herramienta DevOps», 31 de mayo de 2021. https://keepcoding.io/blog/que-es-ansible/ (accedido 12 de agosto de 2023).

[17] A. Hat Red, «Ansible is Simple IT Automation». https://www.ansible.com (accedido 12 de agosto de 2023).

[18] «Ansible: conceptos básicos de la automatización con Ansible». https://www.redhat.com/es/topics/automation/learning-ansible-tutorial (accedido 12 de agosto de 2023).

[19] «Infraestructura de clave pública (PKI) | Juniper Networks». https://www.juniper.net/documentation/mx/es/software/junos/vpn-ipsec/topics/concept/pki-security-overview.html (accedido 19 de junio de 2023).

[20] «¿Qué es una PKI? Infórmese sobre esta tecnología aquí». https://www.entrust.com/es/resources/certificate-solutions/learn/what-is-pki (accedido 19 de junio de 2023).

[21] «¿Qué es una PKI ? | Beneficios para las empresas | Uanataca». https://web.uanataca.com/ec/blog/transformacion-digital/pki (accedido 19 de junio de 2023).

[22] «¿Qué es la infraestructura de clave pública o PKI cuál es su relación con la firma electrónica?», 30 de octubre de 2020. https://www.docusign.mx/blog/pki (accedido 19 de junio de 2023).

[23] «Qué es la PKI | Infraestructura de clave pública | DigiCert». https://www.digicert.com/es/what-is-pki (accedido 19 de junio de 2023).

7 ANEXOS

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces del video demostrativo del funcionamiento del *playbook*

ANEXO III. Código de *Ansible*

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 23 de agosto de 2023

De mi consideración:

Yo, FERNANDO VINICIO BECERRA CAMACHO, en calidad de Director del Trabajo de Integración Curricular titulado DESPLIEGUE DE PKI DE FORMA AUTOMÁTICA MEDIANTE ANSIBLE CON HERRAMIENTAS OPEN SOURCE elaborado por el estudiante ARLEY IVAN SALGADO MAÑAY de la carrera en TECNÓLOGÍA 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 12%.

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.

LINK

https://epnecuador-my.sharepoint.com/:b:/g/personal/fernando_becerrac_epn_edu_ec/EZzlmjPDBVpElAKo_PiSS_wBm6Xw1TpzfAi_DmmlKsQ3Fw

Atentamente,

Fernando Vinicio Becerra Camacho

Docente

Escuela de Formación de Tecnólogos

ANEXO II: Enlaces del video demostrativo del funcionamiento del *playbook*

El video que muestra el funcionamiento del despliegue de un PKI de automáticamente por medio de *Ansible* se encuentra en el enlace https://youtu.be/RlqetRsiAM4

Anexo II.I Código QR de la implementación y pruebas de funcionamiento

ANEXO III: Código de *Ansible*

```
#Servidor PKI

---

- name: Configuración CA de PKI

  hosts: all

  become: yes


  vars_prompt:

    #Dirección donde se crea el PKI

    - name: direccion

      prompt: "Ingrese la ruta del directorio para almacenar el PKI"

      private: false


    #Nombre de carpeta del PKI

    - name: carpeta

      prompt: "Indique el nombre de su carpeta"

      private: false


    #Nombre de certificado CA

    - name: CertificadoCA

      prompt: "Ingrese el nombre para su certificado CA"

      private: false


    #Nombre de llave privada CA

    - name: LlaveCA
```

```
  prompt: "Ingrese el nombre para su llave privada CA"
  private: false

#Clave de certificado CA
- name: ClaveCA
  prompt: "Ingresa la frase de contraseña PEM"
  encrypt: sha512_crypt
  private: yes

#País de CA
- name: pais
  prompt: "Ingrese su país (EU)"
  private: no

#Provincia CA
- name: provincia
  prompt: "Ingrese su Provincia"
  private: no

#Ciudad CA
- name: ciudad
  prompt: "Ingrese su ciudad"
  private: no

#Empresa CA
- name: empresa
  prompt: "Ingrese el nombre de su empresa"
```

```
    private: no

  #Seccion CA
  - name: seccion
    prompt: "Ingrese su departamento/dirección"
    private: no

  #Dominio CA
  - name: dominio
    prompt: "Ingrese su dominio (example.com)"
    private: no

tasks:
  #Actualizar repositorios
  - name: Actualizar repositorios
    apt:
      update_cache: yes
      upgrade: yes
      cache_valid_time: 3600

  #Instalar OpenSSL
  - name: Instalar OpenSSL
    apt:
      name: openssl

  #Verificar version de OpenSSL
  - name: Verificar version de OpenSSL
```

```
  command: openssl version
  register: version_openssl

#Mostrar versión de OpenSSL
- name: Mostrar versión de OpenSSL
  debug:
    msg: "{{ version_openssl.stdout }}"

#Crear carpetas de certificados
- name: Crear carpetas
  ansible.builtin.file:
    path: "{{ item.path }}"
    state: directory
    mode: '0777'
  loop:
    - { path: "{{ direccion}}/{{ carpeta }}/certs" }
    - { path: "{{ direccion}}/{{ carpeta }}/crl" }
    - { path: "{{ direccion}}/{{ carpeta }}/newcerts" }
    - { path: "{{ direccion}}/{{ carpeta }}/private" }

#Crear bases de datos de certificados
- name: Crear y establecer permisos de index.txt
  ansible.builtin.file:
    path: "{{ direccion}}/{{ carpeta }}/index.txt"
    state: touch
    mode: '0777'
```

```
  - name: Crear y establecer permisos del archivo serial
    ansible.builtin.file:
     path: "{{ direccion}}/{{ carpeta }}/serial"
     state: touch
     mode: '0777'

  - name: Agregar contenido al archivo
    ansible.builtin.copy:
      dest:  "{{ direccion}}/{{ carpeta }}/serial"
      content: |
       C001

  #Crear certificado CA
  - name: Crear certificado CA
    command: openssl req -new
        -x509
        -subj "/C={{ pais }}/ST={{ provincia }}/L={{ ciudad }}/O={{ empresa }}/OU={{ seccion }}/CN={{ dominio }}"
        -keyout {{ direccion}}/{{ carpeta }}/private/{{ LlaveCA }}.pem
        -out {{ direccion}}/{{ carpeta }}/{{ CertificadoCA }}.pem \
        -passout pass:{{ ClaveCA }}

  #Crear una copia del archivo raíz
  - name: Crear copia de openssl.conf
    command: sudo cp openssl.cnf openssl.cnf.bk2
    args:
      chdir: /usr/lib/ssl/
```

```
#Modificar archivo raíz
- name: Modificar archivo .cnf
  ansible.builtin.replace:
    path: /usr/lib/ssl/openssl.cnf
    regexp: './demoCA'
    replace: '{{ direccion }}{{ carpeta }}'

- name: Modificar archivo .cnf
  ansible.builtin.replace:
    path: /usr/lib/ssl/openssl.cnf
    regexp: '/cacert.pem'
    replace: '/{{ CertificadoCA }}.pem'

- name: Modificar archivo .cnf
  ansible.builtin.replace:
    path: /usr/lib/ssl/openssl.cnf
    regexp: '/private/cakey.pem'
    replace: '/private/{{ LlaveCA }}.pem'

#Variables
- name: Almacenar variables
  set_fact:
    direccion: "{{ direccion }}"
    carpeta: "{{ carpeta }}"
    pais: "{{ pais }}"
    provincia: "{{ provincia }}"
```

```
      ciudad: "{{ ciudad }}"
      empresa: "{{ empresa }}"
      ClaveCA: "{{ ClaveCA }}"


#Cliente PKI

- name: Configuración cliente de PKI
  hosts: all
  become: yes

  vars_prompt:

    #Nombre de certificado Usuario
    - name: CertificadoUsuario
      prompt: "Indique el nombre del certificado"
      private: false

    #Nombre de llave Usuario
    - name: LlaveUsuario
      prompt: "Indique el nombre de la llave"
      private: false

    #Clave para certificado Usuario
    - name: clave1
      prompt: "Ingresa la frase de contraseña PEM"
      encrypt: sha512_crypt
```

```
    private: yes


  #Seccion Usuario
  - name: seccion2
    prompt: "Ingrese su departamento"
    private: no


  #Nombre Usuario
  - name: nombre
    prompt: "Ingrese su nombre (Nombre y Apellido)"
    private: no



tasks:
  #Crear un certificado para usuario
  #Crear solicitud de certificado (CSR)
  - name: Crear solicitud certificado
    command:
      cmd: openssl req
        -new
        -keyout {{ LlaveUsuario }}.pem
        -subj "/C={{ pais }}/ST={{ provincia }}/L={{ ciudad }}/O={{ empresa }}/OU={{
seccion2 }}/CN={{ nombre }}"
        -out {{ CertificadoUsuario }}-req.pem
        -passout pass:{{ clave1 }}
      chdir: "{{ direccion }}{{ carpeta }}/"
```

```
  #Revisar solicitud (CSR)
  - name: Revisar solicitud (CSR)
    ansible.builtin.command:
      cmd: openssl req
        -text
        -in "{{ direccion }}{{ carpeta }}/{{ CertificadoUsuario }}-req.pem"
      chdir: "{{ direccion }}{{ carpeta }}/"
    register: solicitud


  - name: Imprimir resultado de certificado
    ansible.builtin.debug:
      msg: "{{ solicitud.stdout }}"


  #Firmar el certificado
  - name: Firmar certificado
    command:
      cmd: openssl ca
        -in {{ CertificadoUsuario }}-req.pem
        -out {{ direccion }}/{{ carpeta }}/certs/{{ CertificadoUsuario }}.pem
        -passin pass:{{ ClaveCA }}
        -subj "/C={{ pais }}/ST={{ provincia }}/L={{ ciudad }}/O={{ empresa }}/OU={{
seccion2 }}/CN={{ nombre }}"
        -batch
      chdir: "{{ direccion }}{{ carpeta }}/"
```