OneCloud: Monitorización, Automatización y Seguridad

Fuente: https://whitestack.com/es/blog/que-es-una-nube-privada

FICHA DEL TRABAJO FINAL

Título del Trabajo:	OneCloud: Monitorización, Automatización y Seguridad
Nombre del autor:	Alexandru Ionut Stoian
Nombre del consultor/a:	Jordi Guijarro Olivares
Nombre del PRA:	Josep Jorba Esteve
Fecha de entrega:	06/2025
Titulación o programa:	Ingeniería Informática
Área de trabajo Final:	Arquitectura de computadores y sistemas operativos
Idioma del trabajo:	Castellano
Palabras clave:	Cloud, hibridación, seguridad, monitorización

Resumen del Trabajo

El proyecto **OneCloud**, trata de abordar la implementación de una infraestructura segura y automatizada utilizando **Infraestructura como Código (IaC)** como **Terraform**, la plataforma de gestión de máquinas virtuales y contenedores denominada **OpenNebula**, y tecnologías avanzadas de seguridad como **Confidential Computing**. Con ellos se busca mejorar la seguridad, automatización y monitorización de entornos virtualizados, garantizando la protección de datos sensibles incluso durante su procesamiento. A través de la configuración automatizada mediante plantillas utilizando **Ansible** y la monitorización en tiempo real mediante **Prometheus y Grafana**, se establece un entorno robusto y fácil de gestionar, diseñado para adaptarse a escenarios híbridos o de nube pública. La metodología empleada incluye el diseño detallado de la arquitectura lógica y física, configuración del entorno de laboratorio doméstico, implementación práctica de módulos Terraform y OpenNebula, y pruebas exhaustivas para evaluar la seguridad y rendimiento. Los resultados demuestran que la infraestructura desplegada cumple con los requisitos funcionales y no funcionales establecidos, ofreciendo una solución fiable. En conclusión, este trabajo presenta una infraestructura innovadora que combina automatización avanzada, monitorización efectiva y seguridad reforzada mediante Confidential Computing. Este proyecto nos sirve como base para futuras implementaciones empresariales o institucionales que demanden altos estándares de seguridad y gestión eficiente de recursos.

Abstract

The OneCloud project aims to implement a secure and automated infrastructure using Infrastructure as Code (IaC) such as Terraform, the virtual machine and container management platform called OpenNebula, and advanced security technologies such as Confidential Computing. The aim is to improve the security, automation and monitoring of virtualised environments, ensuring the protection of sensitive data even during processing. Through automated configuration by means of templates using Ansible and real-time monitoring using Prometheus and Grafana, a robust and easy-to-manage environment is established, designed to adapt to hybrid or public cloud scenarios. The methodology employed includes detailed design of the logical and physical architecture, configuration of the home lab environment, hands-on implementation of Terraform and OpenNebula modules, and extensive testing to assess security and performance. The results demonstrate that the deployed infrastructure meets the established functional and non-functional requirements, providing a reliable solution. In conclusion, this work presents an innovative infrastructure that combines advanced automation, effective monitoring and enhanced security through Confidential Computing. This project provides a basis for future corporate or institutional implementations that demand high standards of security and efficient resource management.

índice

1. Introducción:

1.1. Contexto y justificación

Hoy en día donde la seguridad, la automatización y la gestión eficiente de la infraestructura en la nube son clave para el desarrollo de sistemas robustos y confiables, surge la necesidad de buscar nuevas soluciones tecnológicas. Mi trabajo de Fin de Grado (TFG) se centra en el despliegue de una infraestructura segura y automatizada utilizando infraestructura como código (IaC) con Terraform, OpenNebula y Confidential Computing.

Este proyecto nace de mi interés por como la computación en la nube y la seguridad llegan a integrarse para ofrecer entornos mas confiables y resilientes. El aumento de la digitalización ha llevado a que las organizaciones necesiten gestionar entornos mas complejos, donde la privacidad y la integridad de los datos son claves en estos entornos cloud. No obstante, la ciberseguridad , la automatización y la privatización plantean desafíos importantes, es por ello haciendo uso de Confidential Computing y OpenNebula marcan la diferencia en estos aspectos.

Vamos a ver que conocer y a describir brevemente estas dos nuevas tecnologías importantes. Por una parte, tenemos Confidential Computing la cual es una tecnología emergente que permite ejecutar cargas de trabajo en entornos protegidos , asegurando que los datos usados no sean accesibles ni por el proveedor cloud. Esta cualidad ofrece un nivel de seguridad sin precedentes y más en sectores donde la privacidad es un pilar fundamental, como puede ser la banca, el sector salud y la administración pública.
Por otra parte tenemos OpenNebula siendo una plataforma cien por cien de código abierto u *open source* la cual cumple los estándares de protección y tratamiento de datos de la unión europea además de ser un proyecto con origen en Europa. Esta plataforma de código abierto nos permite gestionar maquinas virtuales y contenedores con un control toral sobre la infraestructura, ya sea en entornos on-premise o en la nube pública.

Mi TFG propone una solución en la que mediante Terraform como herramienta de infraestructura como código (IaC) se realizara el despliegue de una infraestructura en un entorno totalmente seguro, automatizado y monitorizado con OpenNebula y Confidential Computing.

Este conjunto de infraestructura nos va a permitir gestionar maquinas virtuales de seguras , monitorizar entornos con Prometheus y Grafana , además de una automatización definida con Ansible.

El proyecto no solo representa un desafío técnico sino también una oportunidad para demostrar como la seguridad y la automatización pueden convivir en las infraestructuras modernas. El uso de tecnologías como Intel Software Guard Extensions (Intel SGX), AMD Secure Encrypted Virtualization (SEV) o AWS Nitro Enclaves garantizaran que los datos se mantengan totalmente protegidos en todo momento mientras que la orquestación con OpenNebula y Terraform simplifica y facilita la escalabilidad y mantenimiento de la infraestructura.

Para finalizar, este trabajo busca proporciona una solución practica para una gestión segura y automatizada de la infraestructura en la nube con un enfoque innovador basando en Confidential Computing. Mas allá de su aplicación académica, espero y deseo que este proyecto sirva como un modelo para entornos empresariales donde la seguridad y la eficiencia en la gestión de los recursos en la nube sea una prioridad.

1.2. Objetivo

- Objetivo Principal
  El principal objetivo es desarrollar un prototipo funcional en el cual creare una infraestructura segura y automatizada que utilice Terraform, OpenNebula y Confidential Computing de forma que permita gestionar maquinas virtuales y contenedores con alto nivel de protección de datos y privatización y transparencia en el despliegue.

- Objetivos Secundarios
  - Investigación de los desafíos actuales
    Se analizará los posibles problemas de seguridad y gestión de entornos en la nube para comprender como afecta a la integridad de la infraestructura y en la protección de la información.

  - Explorar las tecnologías clave
    Analizar y estudiar tecnologías como Confidential Computing, Terraform y OpenNebula viendo sus fortalezas y limitación para aplicarlas en la creación de un entorno seguro.

  - Diseño de una arquitectura de sistemas
    Definir como OpenNebula y Terraform se integran para permitir un despliegue automatizado y estables asegurando la protección de los datos en todo momento y la privatización de estos.

- Integración monitorización y automatización
  Implementar soluciones de monitorización apoyándonos en plataformas como Prometheus y Grafana y la automatización realizada con Ansible para garantizar un control del rendimiento y la seguridad de la infraestructura.

- Testeo de prototipo
  Se lleva a cabo una batería de prueba y simulaciones para evaluar la solidez, resiliencia y eficiencia del entorno desplegado, asegurando de esta forma que el entorno cumple con los requisitos de seguridad.

- Evaluación del impacto potencial
  Analizar como esta infraestructura puede aplicarse en entornos empresariales ,impulsando la confianza en soluciones de la nube y la optimización en la gestión de recursos.

- Diseño de recomendaciones y mejoras futuras
  Elaborar propuesta mejoras para perfeccionar la solución, explotando nuevas herramientas y tecnologías que apoye la seguridad y escalabilidad del proyecto.

1.3. Enfoque y método

Para el desarrollo de este proyecto comenzare por investigar a fondo las tecnologías clave para su desarrollo siendo estas OpenNebula, Terraform y Confidential Computing. Dicha investigación nos permitirá establecer la base y entender como puede cada herramienta contribuir tanto a la protección de datos como la automatización del despliegue. Me voy a apoyar en documentación oficial para tener una perspectiva solida sobre su aplicación.

Vamos a segmentar el trabajo en varios módulos para una gestión mas fácil. Comenzamos por la definición de la infraestructura como código (IaC) con Terraform y la integración de este con OpenNebula. Continuare enfocándome en la parte de seguridad incorporando tecnologías de Confidential Computing que permitan proteger los datos inclusive en el momento de su ejecución y uso. Voy a trabajar de forma conjunta también en la monitorización y automatización de servidores integrando las plataformas mencionadas como Prometheus, Grafana y Ansible.

Durante todo el proceso de integración e implementación voy a documentar cada decisión del diseño y cada obstáculo que se me presente será documentado junto a su solución. Este registro no solo será fundamental para la memoria del proyecto, sino que también me facilitará la trazabilidad de los problemas y la justificación de las soluciones adoptadas.

Por último, para finalizar realizare las pruebas de validación que incluirán la evaluación de resiliencia , la escalabilidad y la seguridad de la infraestructura. Esta batería de pruebas abarcara este el correcto despliegue de las VM hasta la validación de la protección de datos en uso y en base a los resultados ajustare los detalles antes de presentar el sistema garantizando que sea seguro, automático y fácil de gestionar en un entorno real.

1.4. Requerimientos

- Despliegue mediante infraestructura como Código (IaC)
  El sistema debe poder permitir el despliegue automático de todos los recursos necesario ya sea maquinas virtuales, redes, contenedores, etc.. utilizando Terraform como IaC.

- Integración OpenNebula
  Esta solución debe gestionar y orquestrar máquinas virtuales y contenedores mediante OpenNebula garantizando la escalabilidad y el control total de los recursos en entornos locales, híbridos o multicloud.

- Codifidenctial VMs con Confidential Computing
  Es fundamental el uso de soluciones Confidential Computing para proteger los datos en todo momento inclusive de los proveedores externos como es el cloud.

- Seguridad y aislamiento de Red
  Se debe establecer redes privadas con mecanismos de aislamiento para asegurar que las máquinas virtuales y los contenedores desplegados en estos entornos operan de forma protegida contra las amenazas externas.

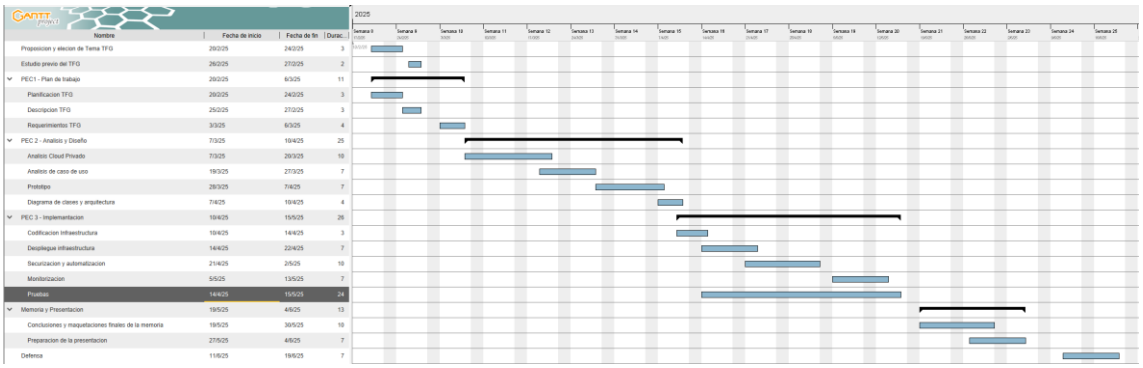
- Monitorización
  El sistema ha de incluir herramientas de monitorización en tiempo real como Prometheus y Grafana para recopilar métricas de visualización sobre el rendimiento y estado de la infraestructura.

- Automatización de la configuración
  Es importante integración con Ansible para instalar software o gestionar actualizaciones de seguridad de forma automatizada en las VM.

- Registro y auditoria
  Todo cambio en la infraestructura de la nube se realizará mediante Terraform dejando constancia y registro permitiendo una trazabilidad de la infraestructura.

Estos son los requerimientos del proyecto que garantiza un entorno seguro transparente y automatizado abarcando aspectos importantes como es el tratamiento y protección de los datos en todo momento.

1.5. Planificación



Fuente: Elaboración propia

Mi TFG ha arrancado con la selección de mi tema y una posición inicial a principios de marzo 2025. Seguiré con una etapa de planificación y la determinación de meta y requisitos hasta la segunda semana de marzo.

Después profundizare en el análisis de cloud privado y herramientas adicionales y su uso hasta inicios de abril luego avanzare hacia la creación de un prototipo y el diseño de la estructura técnica del sistema por mediados de abril.

La fase de construcción y despliegue que comprende la codificación y desarrollo de las soluciones con sus respectivas integraciones, así como las pruebas y ajustes del sistema.

Concluiré mi proyecto redundando los documentos finales y preparando mi exposición durante junio, para presentar y defender mi trabajo antes el tribunal a partir del 20 de junio. Este enfoque me brinda un camino claro para desarrollar y demostrar que mi infraestructura es totalmente operativa y funcional.

1.6. Tecnologías

Para mi proyecto de final de grado en el que estoy diseñando una infraestructura totalmente segura y privada, he investigado inicialmente una serie de tecnologías que considero idóneas para su desarrollo.

- **Terraform**
  Se trata de una herramienta de infraestructura como código utilizado para definir y desplegar de forma automatizada los recursos necesarios ya sea en la nube u on-premise.

- **OpenNebula**
  Se trata de una plataforma de código abierto de orquestación de maquinas virtuales y contenedores que facilita la gestión y escalabilidad de la infraestructura.

- **Confidential Computing**
  Conjunto de tecnologías que permite proteger los datos en proceso o uso que vamos a tratar mediante entornos de ejecución seguros.

- **Ansible**
  Se trata de una solución de automatización y configuración de servidores que agiliza la instalación de software, parches de seguridad, actualizaciones y tareas repetitivas.

- **Prometheus**
  Es un sistema de monitorización y alerta que recopila métricas en tiempo real sobre el rendimiento y estado de una infraestructura.

- **Grafana**
  Se trata de una plataforma de visualización que integrada con Prometheus nos ofrece paneles de interactivos y de alerta sobre los indicadores clave del sistema.

- **Sistema operativo Base**
  Distribuciones Linux donde se ejecutarán los componentes de nuestra infraestructura ya sea Ubuntu, Debian o CentOS, proporcionando un entorno adecuado para la virtualización de las máquinas virtuales.

Estas tecnologías representan mi punto de partida basado en una investigación inicial y están sujetas a cambios medida que avance en el proyecto y vaya ajustando las necesidades del sistema. Voy a mantener una actitud flexible y estaré abierto a insertar cambios y modificaciones

2. Fundamentos y principios subyacentes

2.1. Infraestructura como Código (IaC)

La infraestructura como código (IaC) se ha convertido en un punto fundamental para la gestión y aprovisionamiento de recursos de IT los entornos basados en la nube o cloud en la actualidad. La infraestructura como código consiste en describir y configurar infraestructura de forma declarativa mediante archivos de definición, generalmente son en formato texto, en lugar de realizar acción de forma manual o mediante herramientas graficas. Este enfoque automatizado y repetible contribuye significativamente a reducir errores de configuración, además de simplificar la colaboración entre equipo y agilizar el despliegue de entornos de desarrollo , integración o producción.

La base de la infraestructura como código radica en el concepto de tratar la infraestructura como la misma severidad que el software propio. Las definiciones de infraestructura se versionan en repositorios, se llegan a probar además de tener la capacidad de revertir a un estado anterior si es necesario. De esta forma se realiza un seguimiento de los cambios parecidos a los que se utiliza en el desarrollo de software tradicionales. De esta manera se establece así un nuevo estándar de consistencia y fiabilidad en la administración de sistemas.

Por otra parte, otro punto clave de IaC es la automatización de procesos. En lugar de depender de las tareas manuales que pueden variar de un técnico a otros, la infraestructura como código permite orquestar la creación, configuración y despliegue de ya sea servidores, redes, bases de datos o aplicaciones con alto nivel de precisión y rapidez. Con IaC se evita gran parte de la complejidad asociada a la gestión de entornos heterogéneos, permitiendo que las organizaciones escalen y se adapten al ritmo que exige la era de la transformación digital que se esta viviendo hoy en día.

La transparencia y consistencia son uno de los beneficios notables de IaC. Al utilizar código legible y comprensible para todos los técnicos involucrados en el ciclo de vida de IT, se facilita la colaboración entre equipos de desarrollo, operaciones y negocio. Cualquier cambio en los archivos de configuración puede ser rastreado y auditado de forma clara, además el proceso de revisión por pares garantiza que la adopción de nuevas configuraciones siga los nuevos estándares y practicas definidas por las organizaciones.

Por último, es importante destacar la seguridad que se ve reforzada gracias a IaC puesto que los controles y políticas pueden definirse, aplicarse y supervisarse de manera coherente a lo largo de todo el proceso de

aprovisionamiento. Esto reduce la posibilidad de errores de configuración manual que puede tener un técnico y se alinea con los principios de DevSecOps que buscan integrar la seguridad en cada fase de desarrollo y operación.

En resumen, la infraestructura como código está redefiniendo la forma de diseñar y operar sistemas informáticos, ofreciendo una alternativa ágil, segura y transparente al modelo tradicional de la gestión de infraestructuras. Teniendo en cuenta su potencial para elevar la confiabilidad y la velocidad de los despliegues en organizaciones de todos los tamaños no deja de aumentar su uso haciendo un tránsito hacia una cultura de automatización y colaboración continua en el ámbito tecnológico.

2.1.1. Concepto y ventajas de IaC

Infrastructure as Code (**IaC**) hace referencia a la práctica de gestionar y aprovisionar la infraestructura IT ya sea servidores, redes, etc... mediante archivos de configuración o scripts en lugar de hacerlo de forma manual o mediante herramientas de administración graficas. Esto implica que toda la infraestructura se describe en líneas de código que pueden ser versionadas, revisadas y mantenidas de forma similar a la de desarrollo de software convencional.

A continuación, se describen algunos de los conceptos clave de esta.

- Automatización: La infraestructura se despliega y configura de forma desatendida sin necesidad de acciones manuales.
- Control de versiones: los archivos de configuración se gestionan en sistemas de versionado como puede ser Git, permitiendo una trazabilidad de cambios.
- Repetición: Siendo que la infraestructura está reflejada como código se puede reproducir exactamente el mismo código una y otra vez.
- Escalabilidad, añadiendo nuevos parámetros al código se puede escalar la infraestructura de forma ágil.
- Coherencia y confiabilidad: Se minimizan los errores humanos siendo que el despliegue se realiza de forma automatizada y estandarizada.
- Agilidad: IaC permite acelera el tiempo de implementación.

Para finalizar este apartado vamos a comentar algunas ventajas de IaC.

La mayoría de ellos los hemos comentado previamente sin embargo aquí se realizará un recordatorio.

- Reducción de errores humanos. Al escribir la infraestructura en script o plantillas se evitan errores de configuración manual.
- Mayor consistencia y calidad: Al describir la infraestructura de forma declarativa se asegura una homogeneidad y estabilidad entre entornos.
- Escalabilidad más rápida y sencilla. Podemos duplicar entornos o aumentar recursos de forma simple tan solo modificando líneas de código.
- Trazabilidad y *rollback* de cambio: al utilizar un control de versiones es fácil auditar estados y si es necesario es posible hacer reversión de cambios.
- Estandarización: laC promueve el uso de modelos y plantillas estandarizadas para la infraestructura.
- Colaboración entre equipos: al trabajar con archivos de configuración versionados el equipo de desarrollo y el de operaciones pueden trabajar en conjunto alineando el provisionamiento de infraestructura.

2.1.2. Herramientas principales en el mercado

Podemos encontrar varias herramientas en el mercado que permiten la imple implementación y configuración mediante IaC, estas son algunas de las más conocidas.

- Terraform: de la mano de HasiCorp es un lenguaje declarativo y multiplataforma. Soporta numeroos proveedores cloud y proveedores locales permitiendo una adopción entre entornos híbridos o multicloud. Su mayor ventaja es la de que mantiene el estado de la infraestructura permitiendo detectar y aplicar los cambios necesarios.

- Azure Resource Manager (ARM) : tiene un enfoque declarativo. ARM Templates utilizan JSON para describir y aprovisionar recursos del cloud de Microsoft.

- Bicep : al igual que ARM tiene un enfoque declarativo, sin embargo, este simplifica la sintaxis de las plantillas ARM.

- Chef: Tiene un enfoque basado en libro de cocina o *cookbooks* , es un lenguaje escrito en ruby. Automatiza la configuración y el mantenimiento de servidores. Utiliza un modelo cliente-servidor o puede emplearse en modo local. Una de sus cualidades e que permite definir de forma granular la configuración de cada sistema y aplicar cambios de manera continua.

- Puppet: es un lenguaje declarativo, permite describir el estado deseado de los sistemas mediante “manifests” en su propio lenguaje Puppet. Necesita de un servidor maestro con los agentes previamente instalados en cada host, pero también puede funcionar modo serverless.

- Saltstack: es una herramienta parecida a puppet debido a que funciona en modo servidor- agente, pero al igual que puppet puede funcionar también modo *serverless*. Esta diseñado para orquestrar y administrar sistemas a gran escala. Su mayor ventaja es su arquitectura distribuida y capacidad para ejecutar tareas de manera muy rápida varios nodos.

- AWS CloudFormation: se trata de una lenguaje declarativo y exclusivo de AWS. Permite describir y aprovisionar casi todos los recursos de AWS a través de plantillas en formato JSON o YAML.

- Ansible (Red Hat): es un lenguaje basado en *playbooks* en formato YAML, utilizado para describir configuraciones y el despliegue de infraestructura. No necesita de agentes en los servidores de despliegue , utiliza SSH para ejecutar tareas de configuración y orquestación. Destaca por su sencillez y un amplio catalogo de módulos.

2.1.3. Terraform: características y funcionamiento

Terraform es una herramienta de IaC, una de las mas conocidas y utilizadas en la actualidad. Fue desarrollada por la empresa HashiCorp en 2014. Es un lenguaje declarativo multicloud además destaca por su versatilidad y gran comunidad.

Estas son las principales características de Terraform:

- Modelo declarativo: El usuario describe el estado deseado de la infraestructura en archivos de configuración y este se encarga de orquestrar los pasos para alcanzar dicho estado.

- Multiplataforma: Soporta diversos proveedores cloud además de integrar otras soluciones de infraestructura y servicios de terceros.

- Gestión de estado: Terraform crea y almacena un archivo del estado actual de la infraestructura. Este archivo se compara con la configuración definida para determinar las diferencias y aplicar los cambios necesarios.

- Modularidad: permite crear módulos reutilizables para componentes comunes.

- Dispone de diversos bloques: En terraform se puede disponer de bloques de recursos y bloques de dependencias, se pueden definir recursos y data sources. Terraform decide en que orden se crean y destruyen los recursos basándose en sus dependencias internas.

- Comprobación y validación: parte de la ejecución de terraform implica exponer una vista previa sobre los cambios que se realizan, esta información la podemos ver en el *stage* de ***plan***. Por otra parte, tenemos el *stage* ***apply*** que implica ejecución y configuración de esta infraestructura. Evitamos sorpresas a la hora de generar o crear infraestructura.

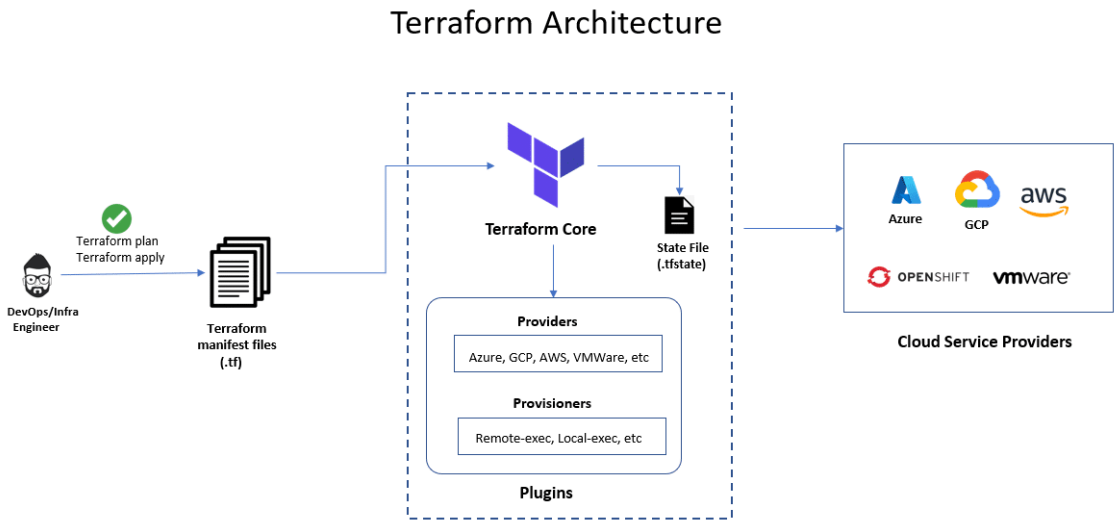
- Creación de un entorno de testing: Podemos crear una batería de pruebas a aplicar sobres nuestra infraestructura ya sea en el stage de plan o apply, inclusive una vez creada podemos hacer test sobre nuestra infraestructura.
- Amplia integración: Terraform se integra fácilmente con herramientas de integración y entrega continua, permitiendo una ágil automatización del ciclo de vida de la infraestructura.
- Gran comunidad y ecosistema: Se dispone de un extenso registro oficial de proveedores y módulos compartidos por la comunidad.

Funcionamiento de terraform:

El flujo de funcionamiento te terraform es el siguiente:

1. Se realiza una escritura de archivos de configuración. El usuario describe la infraestructura que desea implementar en los archivos .tf
2. Inicialización del proyecto. Con el comando ***terraform init*** se prepara el directorio de trabajo o *workspace*, descargando proveedores necesarios y configuración del entorno local.
3. Planificación de cambios: Con el comando ***terraform plan***, se compara la configuración descrita en los archivos con el estado actual de a infraestructura. En este estado se expone todos los cambios que se van a realizar.
4. Aplicación de cambios: con el comando ***terraform apply*** se comienza a ejecutar el plan aprobando la creación de recursos. En este estado se comienza a aprovisionar los nuevos recursos o se modifican los ya existentes o por lo contrario se eliminan los necesarios además de crear o actualizar el fichero de estado de la infraestructura ya sea nueva o existente.
5. Mantenimiento Continuo: Cuando se requiere un cambio o actualización de la infraestructura, tan solo con la modificación del código se ha de repetir el proceso de *init , plan y apply* para que se realicen los cambios necesarios y almacenar el estado de la infraestructura.

6. Destrucción de la infraestructura: Con el comando ***terraform destroy*** se puede eliminar toda la infraestructura que se ha definido en los archivos de configuración.



Fuente: https://medium.com/@impradeep.techie/terraform-architecture-overview-structure-and-workflow-fdc60697941a

2.2. OpenNebula

2.2.1. Introducción e Historia

Open Nebula es una solución de código abierto (open source) que permite implementar fe forma sencilla infraestructuras Cloud Computing privadas e hibridas. Dispone de varias herramientas para crear y administrar infraestructuras de virtualización ya sea en entornos cloud como en entornos híbridos. Ha ido evolucionando con el tiempo y se ha consolidado como una solución destacada en el ámbito de la orquestación de recursos y automatización en la nube.

A grandes rasgos OpenNebula se caracteriza por ser sencilla y fácil de implementar , facilitando la creación de operaciones en entornos virtualizados. Por otra parte, ofrece una gran flexibilidad ya que puede integrar con múltiples hipervisores como puede ser KVM o VMware y con diferentes servicios de almacenamiento y redes.

OpenNebula destaca por integrar herramientas para la asignación dinámica de recursos , opciones de monitorización de máquinas virtuales, creación de imágenes de contenedores y la gestión de usuarios y permisos.

Es importante destacar que podemos construir nuestra propia nube o cloud privado y es lo que vamos a realizar en este proyecto.

Brevemente vamos a comentar sus orígenes historia y evolución de este IaaS (Infraestructure As A Service).

El proyecto de OpenNebula surgió como un proyecto de investigación en la Universidad Complutense de Madrid entre el 2005 y 2007, estaba orientado a explorar la computación en la nube y la virtualización a gran escala dentro de proyectos europeos de investigación.

La primera versión publica fue lanzada en el año 2008, ganando visibilidad en ámbitos académico y empresarial como una alternativa de código abierto para la gestión de entornos de virtualización. Con el paso del tiempo y la adopción de esta por varios proyectos europeos la plataforma fue evolucionando hacia una escalabilidad y eficiencia en la asignación de recursos. A medida que la adopción de la computación en la nube creía, OpenNebula mejoro sus capacidades de gestión y automatización e inserto nuevas opciones de compatibilidad con tecnologías de virtualización , almacenamiento y redes. Con el transcurso de los años se ha formado una gran comunidad de desabolladores y usuarios que contribuye con nuevas funcionalidades parches y documentación sobre la plataforma.

Y es por ello con toda la evolución y cambio por los cuales ha pasado el proyecto de OpenNebula ha conseguido convertirse en una alternativa robusta y confiable para la construcción y operación de nubes privadas e hibridas, adaptándose a diferentes necesidades y escalas de despliegue.

2.2.2. Arquitectura y componentes de OpenNebula

OpenNebula se basa en una estructura de varios niveles y componentes que combinados permite la gestión centralizada de entornos virtualizados y la orquestracion de recursos en la nube. Vamos a comentar algunos de sus elementos principales.

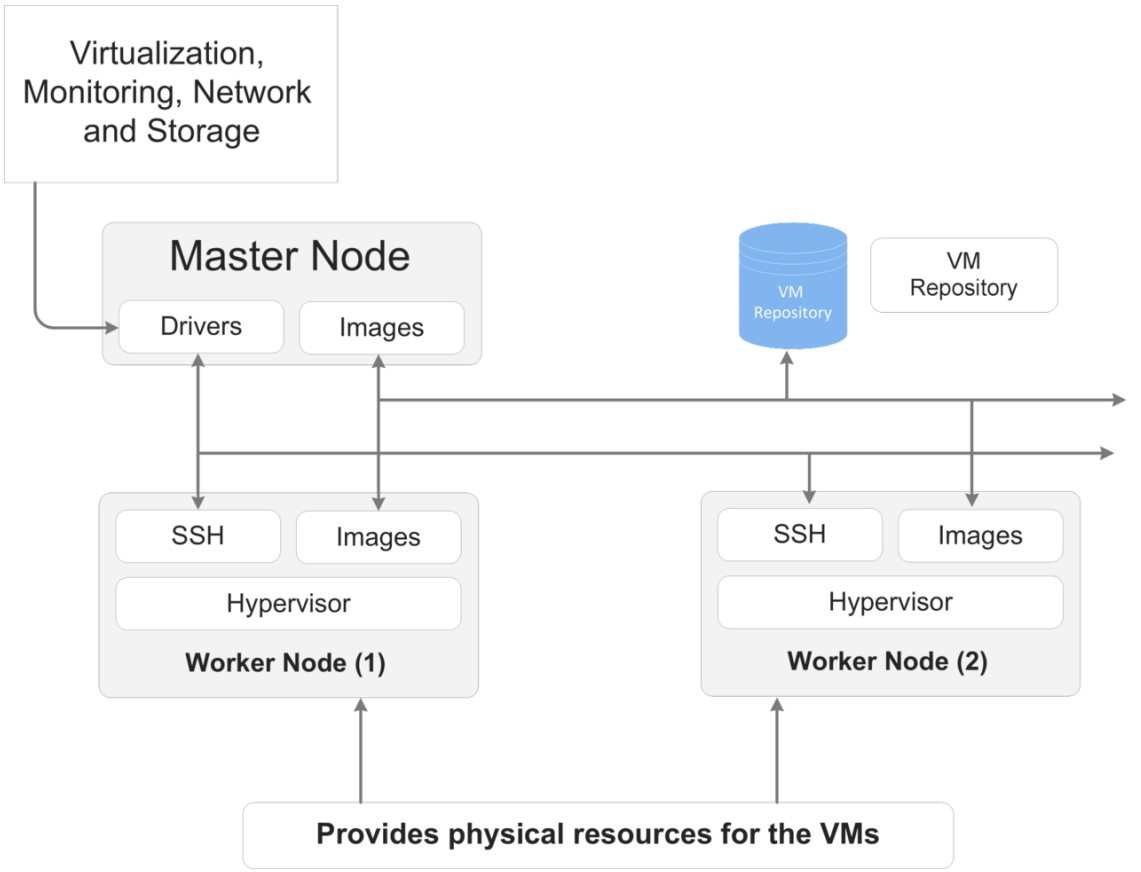
Nodo Front-End , podemos definir este nodo como un conjunto de servicios que corren en el actuando como el maestro de la infraestructura. En este nodo encontramos el proceso central que controla toda la lógica de OpenNebula, el scheduler responsable de asignar recursos a las máquinas virtuales y contenedores. Disponemos la interfaz de línea de comandos que permite interactuar y la interfaz web que nos brinda una consola grafica web de administración.

Por otra parte, tenemos el nodo de cómputo o host de virtualización, donde se gestionan y operan los distintos hipervisores. Es el responsable de proporcionar los recursos informáticos necesarios para procesar los trabajos enviados del nodo maestro.

Datastores o sistemas de almacenamiento, Open Nebula se organiza el almacenamiento en datastores (subsistema de almacenamiento). Un datastore almacena las imágenes base o plantillas de sistema operativo discos etc.... que luego se clona o se utiliza para aprovisionar máquinas virtuales o contenedores.

Redes virtuales, para la correcta interconectividad de componentes como cualquier otro cloud es necesario disponer de Virtual Networks , es por ello OpenNebula nos permite crear y administrar redes virtuales para las maquinas y contenedores que se requiera para nuestra solución. Dependiendo de la integración con los distintos modos de red que necesitemos disponemos de una gran gama de Network drivers además de varias opciones de seguridad para privatizar y securizar nuestra infraestructura.

Por último, comentar que disponemos de componentes adicionales y extensiones como Marketplace donde podemos obtener plantillas de máquinas virtuales y contenedores, módulos de auto escalado e integración con otros proveedores cloud, disponemos de conectores que permite desplegar máquinas virtuales en otros proveedore y estas ser gestionadas por OpenNebula.



Fuente: https://es.wikipedia.org/wiki/Opennebula#/media/Archivo:OpenNebula_Deployment_Model.png

2.2.3. Casos de uso (Cloud público, privado, híbrido)

A la hora de desplegar infraestructura con OpenNebula disponemos de diversos modelos de nube ya sea publica privada o una nube hibrida. Cada una de ellas cubre una necesidad y escenarios de uso distintos. Vamos a exponer de cada uno un caso de uso.

- Cloud Privado: la creación y despliegue de su propia infraestructura es necesaria para mantener segura y privada la información y recursos que estos utilicen. Es por ello tenemos las grandes organizaciones con un gran número de recursos informáticos en sus centros de datos propios y prefieren gestionar dicha infraestructura dentro de la organización. OpenNebula es la herramienta que mejor les ayudara a integrar siendo que es compatible con múltiples hipervisores y distintos sistemas de almacenamiento

- Cloud Publico: Las empresas con recursos limitados o Startups prefieren una solución que no les genere mucho gasto en infraestructura y buscan su aprovisionamiento reduciendo costes en infraestructura apelando a cloud públicos, en lugar de invertir recursos en sus propios centros de datos. Con OpenNebula puede controlar y gestionar la infraestructura como las maquinas virtuales desplegadas en dichos proveedores cloud.

- Cloud Hibrido: Si una organización requiere de una migración granular de su infraestructura a la nube publica , toma la decisión de mover el trabajo a nube publica, pero manteniendo parte de su infraestructura onpremise. Con esta estrategia se apoyan en la escalabilidad de la nube publica, pero por otra parte mantienen sistemas en una nube privada por motivos de seguridad o compliance. OpenNebula ayuda a gestionar la infra desplegada en proveedores cloud como la infraestructura que se dispone onpremise.

2.3. Confidential Computing

2.3.1. Definición y motivación

Podemos referirnos a Confidential Computing como un conjunto de tecnologías y practicas diseñadas para proteger datos y procesos mientras estos se ejecutan en un entorno de cómputo. Las medidas de seguridad tradicionales suelen enfocarse en proteger los datos en reposo o en tránsito, sin embargo, confidential computing se centra en estos datos en uso, esta es una de las grandes diferencias que podemos destacar de confidential computing. Para conseguir la protección de los datos en uso se hace uso de un proceso de aislamiento de carga de trabajo, es decir en entornos de ejecución confiables y seguros para que ni el sistema operativo subyacente, el hipervisor ni cualquier otra aplicación puedan acceder a la información que se está procesando/utilizando.

A continuación, vamos a citar algunas de las motivaciones de Confidential Computing.

- La protección de datos sensibles es una de las principales motivaciones de estas tecnologías. Con el tiempo las organizaciones tienen que gestionar cada vez más cantidad de información confidencial y con ello nace la necesidad de proteger y garantizar que esos datos no sean accesibles.

- El cumplimiento normativo y confianza es un pilar importante siendo que muchas organizaciones o sectores están sujetos a estrictas regulaciones de privacidad y seguridad. Confidential computing ayuda a cumplir estos requisitos garantizando ante auditores, clientes, socios que la información permanece protegida en todo momento.

- Proteger la información contra amenazas ya sea internas o externas es por otra parte otra de las motivaciones de estas tecnologías. Es necesario proteger la información no solo de amenazas o ataques externos sino también amenazas internas como una mala administración. Es necesario aislar las cargas de trabajo para prevenir estas situaciones minimizando así los riesgos en ambos casos.

- Colaboración privada y segura con diversas organizaciones en el proceso de análisis de datos conjuntos sin exponer los datos individuales a externos.

Confidencial computing ofrece una solución a la necesidad de proteger la información mientras se procesa, donde se busca ofrecer el mayor nivel de privacidad, seguridad y confianza ya sea en entornos locales o en entornos cloud donde las comunicaciones y tratamiento de datos puede verse comprometidos si no se implementan las correctas medidas de seguridad.

2.3.2. Tecnologías actuales (Intel SGX, AMD SEV, AWS Nitro Enclaves, etc.)

Como hemos comentado Confidential Computing es un enfoque que busca proteger los datos en uso mediante la ejecución de las cargas de trabajo dentro de entornos de ejecución confiables. Actualmente existen varias tecnologías propuestas por los fabricantes de hardware más destacados del sector IT como proveedores cloud que hacen lo posible por implementar entornos de cómputo confidenciales.

Vamos a listar algunas de las tecnologías propuestas por algunos de los fabricantes.

Intel SGX (Software Guard Extensions): Intel SGX introduce extensiones de CPU que permite crear entornos seguros en los cuales el código y los datos se encuentran aislados del sistema incluso de propio sistema operativo como del hipervisor. Para hacerlo se cifra la memoria asociada al entorno y se restringe el acceso a ella mediante mecanismo hardware.

AMD SEV (Secure Encrypted Virtualization) : AMD SEV ofrece la capacidad de cifrar las máquinas virtuales de forma que el hipervisor y otros componentes no puedan acceder a la información que se ejecuta dentro de estas. Para conseguirlo se utiliza un motor de cifrado en la CPU para proteger la memoria de la máquina virtual.

AWS Nitro Esclave: se trata de una funcionalidad de AWS Amazon Web Services que permite crear entornos aislados dentro de instancias EC2 para procesar datos sensibles sin exponerlos. Para ello hace uso de Nitro Hypervisor de AWS para asignar una CPU de memoria aislada y no persistente.

Microsoft Azure Confidential Computing: Azure confidential computing es un servicio que ofrece el cloud de Microsoft que permite el uso entornos basados en Intel SGX o AMD SEV. Para ofrecer este servicio se hace uso de contenedores o máquinas virtuales protegidos por hardware para el procesamiento de datos.

En conclusión, el ecosistema de confidencial computing abarca una gama de soluciones las cuales tienen todo un objetivo común y el de lograr proteger y asilar datos mientras se procesan, mitigando riesgos tanto de ataques internos como externos.

2.3.3. Ventajas y desafíos de la computación confidencial

Vamos a destacar algunas ventajas de la computación confidencial

La primera es la **protección de los datos sensibles** , esta es la ventaja principal, siendo la capacidad de proteger datos sensibles mientras están en uso reduciendo los riesgos de filtraciones y ataques.

La segunda ventaja que podemos encontrar es el **cumplimiento estándares y regulaciones**. Dada la importancia de la privacidad y las regulaciones como puede ser la *regulación general de protección de datos GDPR* en Europa , la computación confidencial ayuda el cumplimiento de estas.

Confianza en la computación en la nube. Es fundamental la computación confidencial para la confianza en las nubes públicas , siendo que se está cediendo datos a servidores de terceros. La computación confidencial garantiza que los datos estén protegidos aumentando la confianza en la nube.

Por otra parte, debemos tener en cuenta que la computación confidencial como cualquier cara de la moneda tiene su opuesto por lo tanto vamos a destacar algunas de los desafíos que implica.

Uno de los desafíos que implica la computación confidencial es la **complejidad y rendimiento** siendo que implementar las tecnologías necesarias es complejo y puede impactar el rendimiento debido al cifrado y descifrado continuo de los datos.

Otro de los desafíos es la **compatibilidad**. La integración de las tecnologías con sistemas existentes puede requerir de modificaciones significativa lo que puede ser un obstáculo para su adopción.

Por otra parte, tenemos los **costes**. Implementar estas soluciones y especialmente aquellas que utilizan hardware especifico puede impactar en costes adicionales y elevados.

3. Análisis del Estado del Arte en Infraestructuras Seguras y Automatizada

3.1. Infraestructura en la nube y seguridad de datos

La adopción de servicios en la nuve ha trasnformado de forma radical la forma en la que las organizaciones despliegan y gestionan sus recursos. No obstante esta evolución conlleva desafíos en la materia de la seguridad, más concretamente en la protección de datos, el cumplimiento normativo y la gestión de riesgos.

A continuación, voy a mencionar algunos aspectos importantes de la infraestructura en la nube y las consideraciones de seguridad de datos.

Los entornos cloud disponen de tres modelos de servicios actualmente: IaaS, Paas y SaaS. Voy a intentar describir breve y rápidamente. Comenzamos por IaaS que es infraestructura como servicio , la cual se proporciona recursos de procesamiento y almacenamiento y red. Es el proveedor responsable de la infraestructura física pero el cliente administra el sistema operativo o aplicaciones subyacentes. Por otra parte, tenemos PaaS que es plagaroma como servicio, donde se brinda un entorno de desarrollo y despliegue de aplicaciones gestionado por el proveedor. Por último, tenemos SaaS siendo software como servicio donde se permite acceder a aplicaciones gestionadas a través de internet.

Sabemos por otra parte que disponemos de tres arquitecturas en la nube ya mencionadas anteriormente, voy a mencionarlas, pero no vamos a entrar en detalle. Estas arquitecturas de la nube son Nube publica, privada e hibrida.

Por último, voy a destacar los desafíos de seguridad y mejores practicas que podemos aplicar la infraestructura cloud.
Uno de los principales desafíos es la protección de los datos en uso o reposo, siendo el cifrado una de las practicas más utilizadas para reducir los riesgos de robo o intercepción de la información. Por otra parte, la gestión de identidades y accesos es un punto clave e importante da la complejidad de los entornos cloud. El control de acceso basado en roles se ha de aplicar con una estrategia inicial para limitar privilegios y segmentar responsabilidades.

Continuamos y lo hacemos con el cumplimiento normativo otro de los desafíos de los entornos cloud. Dependiendo de la región en la que nos encontremos estaremos sujeto a una o otra regulación por lo tanto esto implica implementar políticas de retención , clasificación y eliminación de datos.
La monitorización de la infraestructura y la definición de planes de respuesta ante incidencias son cruciales y decisivos para los entornos cloud.
Por último pero no menos importante tenemos que destacar que el diseño de toda infraestructura en la nube a de basarse en el principio de ***Zero Trust*** , este enfoque parte de la premisa de que ninguna entidad ya sea externa o interna es de confianza, imponiendo controles de acceso para cada comunicación y recurso.

3.2. Automatización y monitorización: prácticas habituales

A continuación, voy a describir algunas practicas habituales en la automatización y monitorización dentro de entornos cloud.

Para definir la infraestructura se hoy en día se utiliza infraestructura como código, mediante la cual se realiza la creación y configuración de componentes mediante código en lugar de configuración manual. Una de las más utilizadas es Terraform, Ansible, etc....

Por otra parte, tenemos el Despliegue continuo (CD) y la integración continua (CI) la cual se integran los cambios de código necesarios junto a la ejecución de pruebas automáticas y el despliegue en entornos controlados. Algunas de las herramientas mas conocidas son GitHub, Azure DevOps, Jenkins....
Las operaciones en contenedores es otra de las practicas más habituales siendo que se permite automatizar la distribución y la gestión de contenedores en varios hosts. Se utilizan tecnologías como kubernetes que es una de las más conocidas.
La gestión de configuración centralizada es otra de las practicas más habituales permitiendo mantener la consistencia de ajustes de seguridad ya sea políticas y parámetros en múltiples servidores. Disponemos de plantillas de configuración, estrategias rollback etc....

La monitorización se centra por otra parte en definir KPIs o métricas clave para determinar anomalías críticas y anticiparse a problema. El registro centralizado de logs y su consolidación en herramientas lectura permite la correlación de evento y gestión de alertas proactivas. Para verificar el rendimiento de los servicios se hace uso de las sondas o ***probe***. Por otra parte, es importante disponer de sistemas de detección de intrusiones o IDS y sistemas de prevención de intrusiones IPS . El objetivo principal de estos sistemas es identificar patrones de tráfico que indiquen ataques. La optimización de los recursos es fundamental para identificar un crecimiento de la demanda o uso excesivo de los recursos.

Para concluir la automatización y la monitorización son fundamentales para lograr una infraestructura cloud confiable, segura y estable.

3.3. Soluciones existentes de Confidential Computing en la nube

Anteriormente he mencionado algunas soluciones ya implementadas en la nube de configendial computing ofrecidas por los principales proveedores cloud para proteger datos y aplicaciones en entornos asilados.

Voy a citar algunos de ellos con un caso de uso aplicado.

AWS Nitro Enclaves: esta opción ofrece entornos seguros dentro de Amazon EC2 para el procesamiento de datos sensibles de forma aislada. Un caso de uno seria la gestión de la información de datos bancarios, procesamiento de datos confidenciales, etc..

Microsoft Azure Confidential Computing, se basa en el uso de maquinas virtuales y contenedores que se ejecutan sobre hardware seguro y soportan diversas tecnologías como Intel SGX o AMD SEV. Un caso de uso aplicado seria el procesamiento de datos regulados como puede ser el sector de la salud.

Google Cloud Confidential Computing es una solución donde se incluye Google Confidential virtual machines y Configendial GKE (primer servicio de Kubernetes) nodes para cifrar memoria de las cargas de trabajo en tiempo de ejecución haciendo uso de tecnologías como AMD SEV para ello. Un caso de uso podría ser la protección de cargas de trabajo en las bases de datos.

IBM ofrece varias opciones que incluyen aislamiento basado en entornos aislados y contenedores seguros con cifrado de memoria y servicios de gestión de claves. Para ello hacen uso de tecnologías como Intel SGX y Linux ONE. Se puede utilizar en entornos mainframe híbridos como banca o sector de salud.

Cada proveedor cloud ofrece una solución de confidential computing basada en distintas tecnologías de hardware seguro. Sin embargo, todos tienen como objetivo común aislar y cifrar los datos en tiempo de ejecución.

4. Especificación de Requisitos y Casos de Uso

4.1. Análisis de requisitos funcionales y no funcionales

Se expondrán los principales requisitos funcionales y no funcionales de la infraestructura segura y automatizada que se quiere construir con tecnologías como Terraform OpenNebula y Confidential Computing.

Comenzamos con los requisitos funcionales.

- El despliegue automático donde el sistema debe permitir la creación, configuración y eliminación de recursos de forma automática mediante el uso de script de IaC con Terraform.
- La gestión centralizada de recursos, con la plataforma de OpenNebula debe centralizar la gestión de máquinas virtuales y contenedores ofreciendo una visibilidad y control sobre todos los recursos, con la posibilidad de escalar de forma vertical y horizontal según demanda.
- Confidential Computing, se deberá de contemplar la posibilidad de desplegar máquinas virtuales confidenciales que protegen los datos en uso, utilizando tecnologías como Intel SGX
- Automatización y configuración, mediante herramientas como Ansible se debe instalar paquetes , actualizaciones y configuraciones de forma desatendida.
- Monitorización y Alertas, el sistema debe incorporar Prometheus para la recolección de métricas y Grafana para su visualización.

Continuamos con los requisitos no funcionales.

- La seguridad y privacidad, el sistema ha de garantizar un alto nivel de protección de datos tanto en reposo como en uso.
- Escalabilidad, la arquitectura debe permitir escalar de forma dinámica según sea necesario sin comprometer el rendimiento ni la disponibilidad.
- Fiabilidad y Disponibilidad, el diseño debe contemplar mecanismos de tolerancia a fallos al igual que se debe aspirar a una alta disponibilidad minimizando los tiempos de inactividad.
- Rendimiento y procesos, los tiempos de de procesamiento , despliegue y actualización debe ser razonablemente corto para no afectar la productividad de los equipos.
- Compatibilidad y Estándares, siempre que se permita se debe priorizar el uso de herramientas y protocolos estándar como Terraform HCL, YAML en Ansible.

Estos son los requisitos funcionales y no funcionales que nos permiten establecer un marco de referencia para el desarrollo, despliegue y mantenimiento de la infraestructura.

4.2. Actores en el sistema (administradores, usuarios, DevOps)

A continuación, se describen los principales actores involucrados en el escenario de una infraestructura segura y automatizada de este proyecto.

- Administrador de la infraestructura (SysAdmin / DevOps)
  Su responsabilidad es la de diseñar, desplegar y mantener esta infraestructura. Ha de crear y configurar los archivos de terraform y Ansible. Este ha de velar por la seguridad y disponibilidad del sistema. Ha de tener conocimientos en redes, IaC con Terraform y seguridad.

- Administrador de seguridad (SecOps)
  Su responsabilidad será la de definir y supervisar las políticas de seguridad , realizar auditorías y gestionar los accesos garantizando que se cumplen todos los requisitos de seguridad y privacidad. Este ha de tener conocimientos y experiencia con entornos aislados y privados además de cifrado de datos.

- Desarrollador o Equipo de Desarrollo
  La responsabilidad de esta figura será la de desarrollar aplicaciones o servicios que se ejecuten sobre la infraestructura desplegada. Esto han de tener conocimientos en lenguajes de programación y buenas prácticas además de metodologías de CI/CD.

- Usuario Final
  El usuario final será el consumidor de la infraestructura o Aplicaciones que den servicio. La responsabilidad de este es utilizar los servicios o aplicaciones que se ejecuten en la infraestructura. Este ha de tener unos conocimientos base sobre la aplicación entregada para su uso.

- Operador de Monitorizacion (Observability / SRE)
  Su responsabilidad será la de configurar y mantener los sistemas de monitorizacion y la gestion de alertas totalmente operativas, reaccionando ante eventos críticos y colaborar con el administrador para solucionar problemas que tenga el sistema. Este es un perfil con conocimientos en herramientas de observabilidad y monitorización además de experiencia en diagnóstico de problemas de rendimiento y optimización de sistemas.

- Director de Producto (Product Owner)
  Es el responsable de definir los objetivos del proyecto, priorizar requisitos y coordinar el desarrollo de nuevas funcionalidades para el sistema y plataforma. Este es un perfil con conocimientos en administración de proyectos y gestión de clientes.

Esta es la organización o alcance del proyecto , puede ser que algunos roles se puedan combinar como el administrador de sistemas y el administrador de seguridad en un único equipo o que sea un único perfil. No obstante, estos actores son fundamentales para cubrir las diferentes facetas de la infraestructura, garantizando que se cumplan los objetivos de seguridad , escalabilidad y eficiencia.

4.3. Escenarios de despliegue y seguridad

Se presentan los distintos escenarios de despliegue y seguridad donde se ejemplifica como se puede implementar la infraestructura con Terraform OpenNebula y Confidential Computing.

El primer entorno donde se puede realizar este despliegue es en un entorno privado una nube privada(on-premise). Donde la organización cuenta con su propio centro de datos y desea mantener un control absoluto sobre su infraestructura. Se puede utilizar OpenNebula como plataforma de virtualización y administración junto a Terraform para definir y desplegar esta infraestructura. Respecto a aspectos de seguridad, se puede implementar reglas y políticas Firewall y segmentación de la red en VLAN que asegura un acceso restringido a los sistemas. Por otra parte se puede habilitar los en los host compatibles tecnologías de Confidential Computing además de incluir políticas de control de acceso como SSO o LDAP para esta infraestructura. Este escenario de despliegue puede darse en sectores como el financiero , el sector público como la sanidad o entidades gubernamentales.

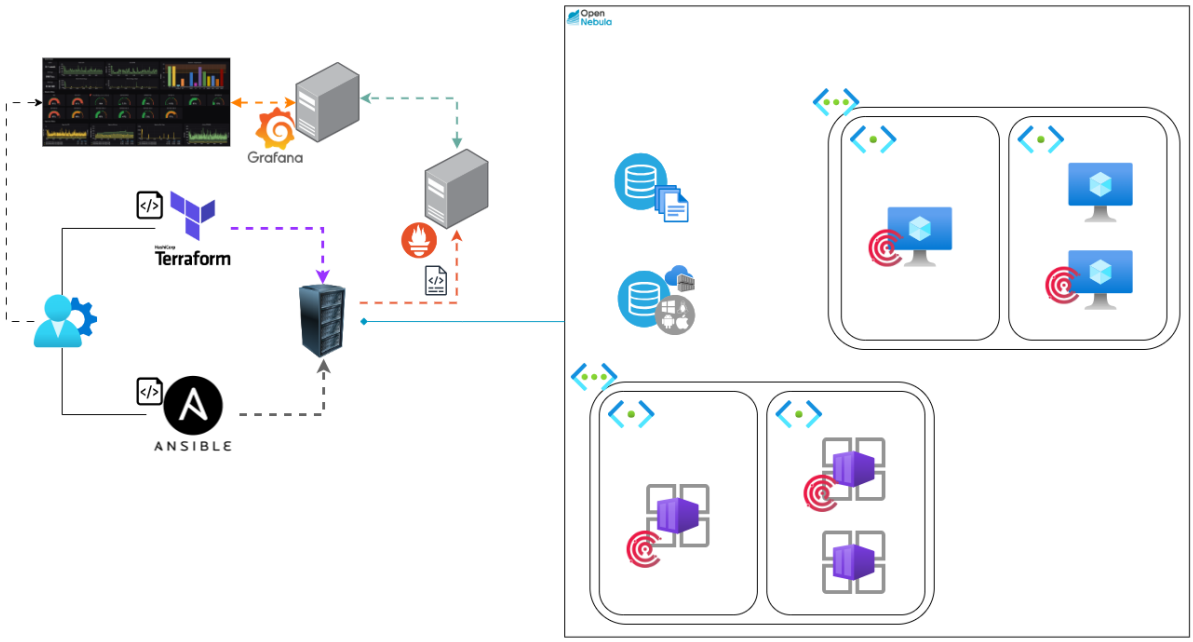
En segundo lugar, se puede realizar este despliegue en un entorno cloud público. La infraestructura se despliega en un proveedor cloud como puede ser AWS, Azure o GCP siendo que ofrecen la escalabilidad i disponibilidad global que estos ofrecen. Con OpenNebula podemos administrar o gestionar hibrida de la infraestructura cloud. Por otra parte, se pueden establecer máquinas virtuales confidenciales especificas a cada proveedor, siendo que incluyen servicios de confidential computing implementados. Respecto a la seguridad de estos entornos públicos podemos gestionar los datos almacenándolos en instancias cifradas o proteger el acceso a estos con reglas de firewall y limitaciones de accesos basado en roles. Este escenario está enfocado a organizaciones que no disponen de infraestructura propia y buscan una puesta en marcha rápida y ágil.

En tercer lugar, se puede realizar una combinación de las opciones anteriores, donde se combina la infraestructura onpremise con los recursos en a la nube publica para balancear la carga de trabajo. En este escenario OpenNebula será un orquestador multicloud, integrando de forma simultánea maquinas locales con las soluciones de proveedores externos. La seguridad en este escenario destaca por la interconectividad de redes con a la red onpremise y la red privada del entorno cloud publico ya se haciendo uso de VPN o MPLS. En este escenario seria apto para empresas que quieren mantener un control y gestión sobre datos sensibles en su centro de datos y a la vez aprovechar las cualidades de la nube pública.

En cualquier de estos escenarios se aplican los mismos principios de IaC, automatización y monitorización definidos. No obstante, la única diferencia es la ubicación de los recursos.

5. Diseño y Estructuración de la Arquitectura de la Infraestructura

5.1. Diagrama general de la solución (Terraform + OpenNebula + Confidential Computing)



Fuente: Elaboración propia

El administrador es quien define la infraestructura en los ficheros terraform y procede a su despliegue. OpenNebula gestiona las máquinas virtuales o infraestructura. Se procede a configurar el uso de máquinas virtuales confidenciales además de contenedores confidenciales.

Por otra parte, la monitorización del servidor se realizaría con Prometheus el cual obtendría las métricas del servidor y las envía a Grafana para su visualización.

Importante quiero destacar que para la gestión y automatización de la infraestructura se realiza fichero YAML el cual como se ha mencionado se gestionara la automatización de los servidores.

5.2. Definición de redes, seguridad y datos manejados

Definición de redes

Basándonos en el diagrama anterior sabemos que Terraform Y Ansible se sitúan en una zona de gestión localizada en el punto de control del administrador, la cual se conecta con el servidor de OpenNebula o su instancia dedicada al control.

Esta red esta protegida con un firewall que filtrara el tráfico entrante y saliente asegurando únicamente los puerto y protocolos necesarios en este caso sería SSH, HTTP/S. Estos protocolos están habilitados para la comunicación con las herramientas de orquestación de la infraestructura. Por otra parte, tenemos la red de monitorización compuesta por Grafana y Prometheus, separados en un segmento de red con acceso restringido, de modo que solo la zona de gestión y los nodos de monitorización podrán acceder a esta infraestructura. Este diseño evita que terceros no autorizados puedan acceder y visualizar métricas sensibles de la infraestructura. Por último, disponemos de la capa de virtualización o red de producción, esta infraestructura virtualizada se encuentra bajo el control de OpenNebula donde se pueden crear subredes lógicas y gestionarlas mediante virtual switch o bridges. Importante mencionar que las máquinas virtuales y los contenedores con Confidential Computing se sitúan en subredes aisladas que garantizan el cifrado de comunicaciones internas con un acceso mínimo entrante.

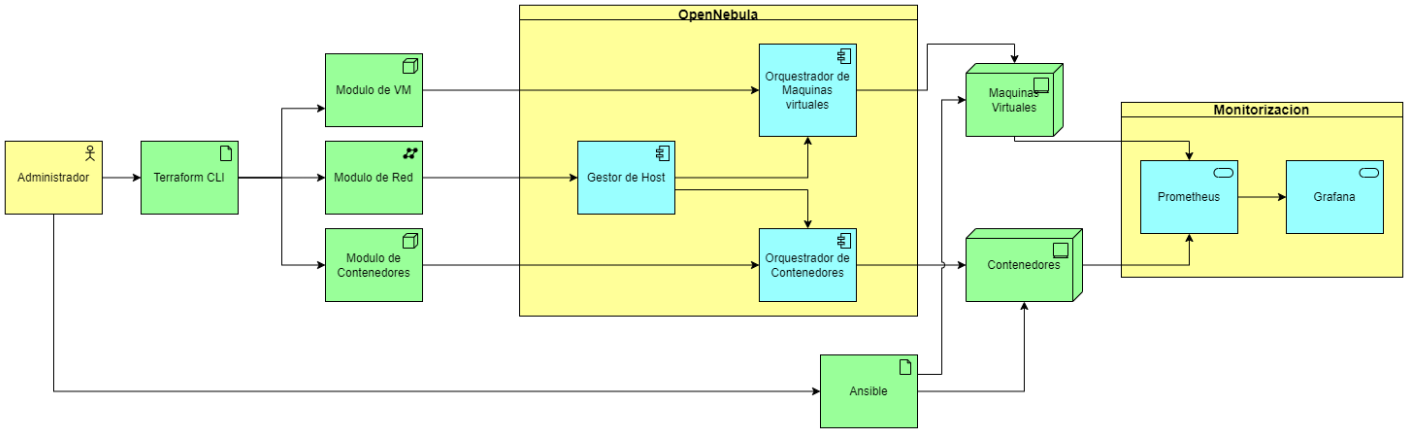
Definición de seguridad

Las decisiones tomadas para mejorar la seguridad y el control de accesos se pueden segmentar en varios puntos. Como se puede observar en el diagrama se muestra un servidor el cual tendrá un firewall implementado que hará de punto de control con la red exterior y la red interna. Las reglas siguen el principio de "*deny by default*" permitiendo únicamente el tráfico necesario para la administración y monitorización de esta. Por otra parte, tenemos en cuenta el aislamiento y confidencial computing que son aspectos clave , siendo las máquinas virtuales y contenedores seleccionados para la utilización de soluciones y procesos de aislamiento para tratar los datos. Por último, se definirán roles con accesos definidos para cada acceso y función que se vaya a desempeñar.

Datos manejados

En este escenario se va a gestionar tres tipos de datos, El primero serán los datos de configuración y despliegue que conforman los ficheros de Terraform y Ansible que describen como se han de crear y configurar la infraestructura. En segundo lugar, tenemos los datos operacionales, formado por las métricas de rendimiento, logs de eventos y registro de auditoritas. Por último, disponemos de los datos sensibles o confidenciales que se almacenaran en las máquinas virtuales y contenedores que procesaran estas con capas de confidential computing.

5.3. Esquema de componentes y relación (módulos Terraform, VMs, contenedores)



Fuente: Elaboración propia

Se realiza una breve descripción sobre cada componente de diagrama. En primera instancia tenemos a la figura del administrador quien escribe los ficheros de terraform y ansible que realiza la orquestación de la infraestructura. Los componentes terraform cli es una herramienta de línea de comando s que interpreta nuestro código terraform y lanza las llamadas a OpenNebula. Tenemos tres bloques de componentes configurado que se va a desplegar con los archivos terraform. El primer bloque o modulo es el módulo de red que crea y configura la red virtual y subredes de esta. Por otra parte, tenemos los bloques o módulos de máquinas virtuales que componen el tipo de VM y configuraciones aplicadas a esta en su despliegue. Por último, tenemos el módulo de contenedores el cual se encargará de gestionar los contenedores sobre la plataforma OpenNebula. Estos módulos se conectan con los orquestradores de OpenNebula que se responsabilizan de configurar y crear los recursos definidos para cada conjunto de componente declarado en los módulos de terraform.

A continuación, nos posicionamos en los recursos desplegados siendo estos los componentes como máquinas virtuales y contenedores que soportan confidential computing.

Por otra parte, tenemos los archivos de configuración de ansibe siendo estos scripts de automatización utilizados para realizar configuraciones desatendidas aplicadas tanto a las máquinas virtuales o contenedores desplegados.

Para finalizar tenemos los componentes de monitorización, por una parte, tenemos prometheus quien enviara las métricas a grafana la cual se utilizara para visualizar y presentar dichas métricas en paneles.

De esta forma con este diagrama se muestra la relación entre bloques de recursos que se crean , para comprender los componentes y evitar confusiones en el flujo del trabajo.

5.4. Arquitectura lógica y física del sistema

Se describirá la arquitectura lógica donde se estructuran los servicios y componentes del sistema a nivel conceptual. Donde destacamos las capas funcionales donde encontramos la codificación de la infraestructura mediante terraform, con la cual se describen y gestionan los recursos como máquinas virtuales, redes y volúmenes. Por otra parte, la orquestación se realiza bajo la gestión de OpenNebula que administrara el ciclo de vida de las VM, contenedore y resto de componentes, respecto a la automatización se realizara con Ansible aplicando configuraciones y la infraestructura y por último a la colección de métricas será gestionada por Prometheus junto a Grafana que nos presentara una capa de visualización de estas.

Por otra parte, vamos a describir la arquitectura física donde se muestra cómo se desplegarán los componentes en servidores máquinas virtuales o contenedores reales ya sea en entornos on premise o nube. La arquitectura física se enfoca en sistemas distribuidos de servidores en este escenario, donde encontramos un nodo de OpenNebula en el cual se insta sobre el servidor físico o instancia de la nube, otros nodos de monitorización siendo un nodo independiente del utilizado en OpenNebula evitando así una sobrecarga de operaciones y por último tenemos máquinas virtuales confidenciales que se desplegaran sobre el host de OpenNebula. Las redes se segmentan en dos, vamos a tener una red de gestión exclusiva para la administración con Terraform y Ansible además de los paneles de administración disponibles por la propia plataforma. Por otra existirá una red de producción la cual se desplegarán las instancias con acceso restringidos.

Respecto al almacenamiento se disponen de volúmenes cifrados para datos sensibles desde las que se van a alimentar las máquinas virtuales e instancias desplegadas.

Para concluir importante destacar que la arquitectura lógica coordina los procesos de despliegue y automatización mientras que la arquitectura física establece donde y como se ejecutan asegurando la seguridad y escalabilidad de los recursos.

6. Implementación y configuración

6.1. Configuración de entorno de desarrollo (laboratorio)

Para el desarrollo del proyecto se ha realizado la instalación de una distribución Linux en este caso , se ha utilizado Ubuntu. Dicha instalación se ha realizado en un servidor físico al cual disponemos acceso físico y remoto para realizar las gestiones correspondientes y configuraciones necesarias aplicado a nuestro caso práctico.

El laboratorio se ejecuta sobre un servidor local conectado de forma estable a la red local. Estas son las características del servidor en el cual se realizará el laboratorio.

- Procesador: Intel Core i5-7500 (7ª generación) con soporte de virtualización (VT-x y VT-d).
- Memoria RAM: 16 GB (DDR4 a 2400 MHz) suficientes para ejecutar varias máquinas virtuales de forma simultánea.
- Almacenamiento: Unidad SSD de 512 GB (Fanxiang S101). Esto proporciona espacio adecuado para el sistema anfitrión y el despliegue de VMs o contenedores.
- Sistema Operativo Base: distribución Linux (Ubuntu) configurada como hipervisor, que disponga de los paquetes esenciales para Terraform, OpenNebula y Ansible.
- Conexión de Red: Interfaz Ethernet Realtek Gigabit (1 Gbit/s), integrada en la placa, conectada al router doméstico con acceso a internet y a la LAN local.
- Virtualización: Compatible con KVM (soporta extensiones de virtualización Intel). También podría usarse VMware o VirtualBox, según la configuración deseada.
- Soporte de Confidential Computing : La CPU admite Intel SGX.

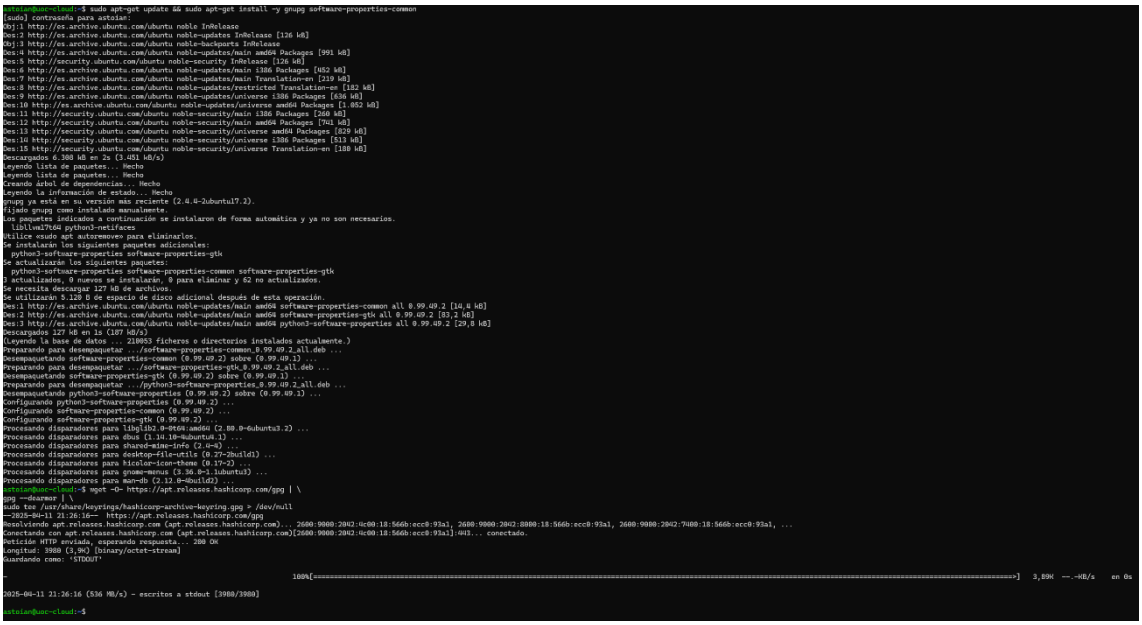
Respecto a la conectividad de este, el servidor se encuentra conectado a un router domestico el cual proporciona servicio de internet mediante protocolo NAT o resolución de dirección IP privada de manera interna. El equipo esta conectado de forma física al router mediante cable Ethernet cuya dirección ip privada se ha configurado de forma estática manteniendo la IP 192.168.1.177. Desde este servidor se gestiona toda la infraestructura de pruebas, incluyendo el acceso a internet para descargar paquetes y actualizaciones. Por otra parte, las máquinas virtuales y/o contenedores se alojan el mismo servidor heredando de esta forma la conectividad de la red local. No obstante, según la

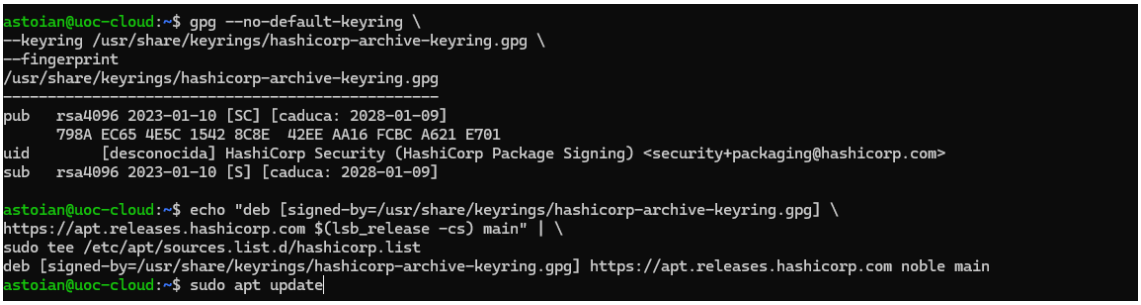
configuración de virtualización y OpenNebula se configurará un NAT interno donde las VM y contenedores accederán a la red local mediante la IP del servidor donde se alojan.

Se trata de un entorno simple donde el router hace de punto central de conectividad y el servidor asume todos los roles de laboratorio mientras que las maquinas virtuales y contenedores forman la capa de pruebas.

Se expondrá a continuación las herramientas a instalar y características ha habilitar en el servidor.

- Terraform: Necesitamos disponer de terraform para la implementación de la infraestructura de este por lo tanto vamos a realizar la instalación de este software. Para ello vamos a hacer uso del manual de instalación del cual disponemos en la página oficial de Terraform HashiCorp.





```
astoian@uoc-cloud:~$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \
https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/hashicorp.list
deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com noble main
astoian@uoc-cloud:~$ sudo apt update
Obj:1 http://es.archive.ubuntu.com/ubuntu noble InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu noble-updates InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu noble-backports InRelease
Des:4 https://apt.releases.hashicorp.com noble InRelease [12,9 kB]
Obj:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Des:6 https://apt.releases.hashicorp.com noble/main amd64 Packages [176 kB]
Des:7 https://apt.releases.hashicorp.com noble/main i386 Packages [74,2 kB]
Descargados 263 kB en 0s (570 kB/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 62 paquetes. Ejecute «apt list --upgradable» para verlos.
astoian@uoc-cloud:~$ sudo apt-get install terraform
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libllvm17t64 python3-netifaces
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  git git-man liberror-perl
Paquetes sugeridos:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
Se instalarán los siguientes paquetes NUEVOS:
  git git-man liberror-perl terraform
0 actualizados, 4 nuevos se instalarán, 0 para eliminar y 62 no actualizados.
Se necesita descargar 32,4 MB de archivos.
Se utilizarán 115 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] y
Des:1 https://apt.releases.hashicorp.com noble/main amd64 terraform amd64 1.11.4-1 [27,6 MB]
Des:2 http://es.archive.ubuntu.com/ubuntu noble/main amd64 liberror-perl all 0.17029-2 [25,6 kB]
Des:3 http://es.archive.ubuntu.com/ubuntu noble-updates/main amd64 git-man all 1:2.43.0-1ubuntu7.2 [1.100 kB]
Des:4 http://es.archive.ubuntu.com/ubuntu noble-updates/main amd64 git amd64 1:2.43.0-1ubuntu7.2 [3.679 kB]
Descargados 32,4 MB en 1s (25,9 MB/s)
Seleccionando el paquete liberror-perl previamente no seleccionado.
(Leyendo la base de datos ... 210053 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../liberror-perl_0.17029-2_all.deb ...
Desempaquetando liberror-perl (0.17029-2) ...
Seleccionando el paquete git-man previamente no seleccionado.
Preparando para desempaquetar .../git-man_1%3a2.43.0-1ubuntu7.2_all.deb ...
Desempaquetando git-man (1:2.43.0-1ubuntu7.2) ...
Seleccionando el paquete git previamente no seleccionado.
Preparando para desempaquetar .../git_1%3a2.43.0-1ubuntu7.2_amd64.deb ...
Desempaquetando git (1:2.43.0-1ubuntu7.2) ...
Seleccionando el paquete terraform previamente no seleccionado.
Preparando para desempaquetar .../terraform_1.11.4-1_amd64.deb ...
Desempaquetando terraform (1.11.4-1) ...
Configurando liberror-perl (0.17029-2) ...
Configurando git-man (1:2.43.0-1ubuntu7.2) ...
Configurando git (1:2.43.0-1ubuntu7.2) ...
Configurando terraform (1.11.4-1) ...
Procesando disparadores para man-db (2.12.0-4build2) ...
astoian@uoc-cloud:~$
```

```
astoian@uoc-cloud:~$ terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
  metadata      Metadata related commands
  modules       Show all declared modules in a working directory
  output        Show output values from your root module
  providers     Show the providers required for this configuration
  refresh       Update the state to match remote systems
  show          Show the current state or a saved plan
  state         Advanced state management
  taint         Mark a resource instance as not fully functional
  test          Execute integration tests for Terraform modules
  untaint       Remove the 'tainted' state from a resource instance
  version       Show the current Terraform version
  workspace     Workspace management

Global options (use these before the subcommand, if any):
  -chdir=DIR    Switch to a different working directory before executing the
                given subcommand.
  -help         Show this help output, or the help for a specified subcommand.
  -version      An alias for the "version" subcommand.
astoian@uoc-cloud:~$ terraform -v
Terraform v1.11.4
on linux_amd64
astoian@uoc-cloud:~$
```

Como podemos observar la versión instalación ha finalizado correctamente y ya disponemos de terraform en nuestro servidor.

Previamente antes de continuar con la instalación de OpenNebula necesitamos instalar las herramientas de virtualización para las maquinas virtuales y los contenedores necesarios. Los sistemas de virtualización son **KVM** (Kernel-based Virtual Machine) para la virtualización de máquinas virtuales y **LXD** (Linux Container Daeamon) sistema basado en LXC nativo de Ubuntu para instalar en nuestro sistema para la virtualización de contenedores.

Comenzamos con la instalación de KVM

Simplemente tenemos que ejecutar dos comandos para ello.

```
astoian@uoc-cloud:~$ sudo apt-get update
Obj:1 http://es.archive.ubuntu.com/ubuntu noble InRelease
Obj:2 https://apt.releases.hashicorp.com noble InRelease
Des:3 http://es.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Obj:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Obj:5 http://es.archive.ubuntu.com/ubuntu noble-backports InRelease
Descargados 126 kB en 0s (323 kB/s)
Leyendo lista de paquetes... Hecho
astoian@uoc-cloud:~$ sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils -y
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Nota, seleccionando «qemu-system-x86» en lugar de «qemu-kvm»
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libllvm17t64 python3-netifaces
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  dmeventd ipxe-qemu ipxe-qemu-256k-compat-efi-roms libaio1t64 libcacard0 libdaxctl1 libdevmapper-event1.02.1 libfdt1
  libiscsi7 libjack-jackd2-0 liblvm2cmd2.03 libndctl6 libnss-mymachines libnss-systemd libpam-systemd libpmem1
  libpmemobj1 librados2 librbd1 librdmacm1t64 libsdl2-2.0-0 libslirp0 libspice-server1 libsystemd-shared libsystemd0
  libtpms0 libudev1 liburing2 libusbredirparser1t64 libvirglrenderer1 libvirt-daemon libvirt-daemon-config-network
  libvirt-daemon-config-nwfilter libvirt-daemon-driver-qemu libvirt-daemon-system-systemd libvirt-l10n libvirt0
  libxml2-utils lvm2 mdevctl ovmf qemu-block-extra qemu-system-common qemu-system-data qemu-system-gui
  qemu-system-modules-opengl qemu-system-modules-spice qemu-utils seabios swtpm swtpm-tools systemd systemd-container
  systemd-dev systemd-oomd systemd-resolved systemd-sysv systemd-timesyncd thin-provisioning-tools udev
Paquetes sugeridos:
  ifupdown jackd2 libvirt-clients-qemu libvirt-login-shell libvirt-daemon-driver-storage-gluster
  libvirt-daemon-driver-storage-iscsi-direct libvirt-daemon-driver-storage-rbd libvirt-daemon-driver-storage-zfs
  libvirt-daemon-driver-lxc libvirt-daemon-driver-vbox libvirt-daemon-driver-xen numad passt auditd nfs-common
  open-iscsi pm-utils systemtap zfsutils samba vde2 trousers systemd-homed systemd-userdbd systemd-boot libqrencode4
Se instalarán los siguientes paquetes NUEVOS:
  bridge-utils dmeventd ipxe-qemu ipxe-qemu-256k-compat-efi-roms libaio1t64 libcacard0 libdaxctl1
  libdevmapper-event1.02.1 libfdt1 libiscsi7 libjack-jackd2-0 liblvm2cmd2.03 libndctl6 libnss-mymachines libpmem1
  libpmemobj1 librados2 librbd1 librdmacm1t64 libsdl2-2.0-0 libslirp0 libspice-server1 libtpms0 liburing2
  libusbredirparser1t64 libvirglrenderer1 libvirt-clients libvirt-daemon libvirt-daemon-config-network
  libvirt-daemon-config-nwfilter libvirt-daemon-driver-qemu libvirt-daemon-system libvirt-daemon-system-systemd
  libvirt-l10n libvirt0 libxml2-utils lvm2 mdevctl ovmf qemu-block-extra qemu-system-common qemu-system-data
  qemu-system-gui qemu-system-modules-opengl qemu-system-modules-spice qemu-system-x86 qemu-utils seabios swtpm
  swtpm-tools systemd-container thin-provisioning-tools
Se actualizarán los siguientes paquetes:
  libnss-systemd libpam-systemd libsystemd-shared libsystemd0 libudev1 systemd systemd-dev systemd-oomd
  systemd-resolved systemd-sysv systemd-timesyncd udev
12 actualizados, 52 nuevos se instalarán, 0 para eliminar y 45 no actualizados.
```

Una vez instalado comprobamos la instalación y añadimos el usuario root y local a los grupos **libvirt** y **kvm** siendo los únicos que pueden gestionar maquinas virtuales.

```
astoian@uoc-cloud:~$ grep 'libvirt:.*$' /etc/group | cut -d: -f4
astoian,root
astoian@uoc-cloud:~$ grep 'kvm:.*$' /etc/group | cut -d: -f4
astoian,root
astoian@uoc-cloud:~$
astoian@uoc-cloud:~$
astoian@uoc-cloud:~$
astoian@uoc-cloud:~$
astoian@uoc-cloud:~$
astoian@uoc-cloud:~$ sudo virsh list --all
 Id   Name   State
--------------------

astoian@uoc-cloud:~$ sudo systemctl status libvirtd
● libvirtd.service - libvirt legacy monolithic daemon
     Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-04-12 12:55:49 CEST; 10s ago
TriggeredBy: ● libvirtd.socket
             ● libvirtd-admin.socket
             ● libvirtd-ro.socket
       Docs: man:libvirtd(8)
             https://libvirt.org/
   Main PID: 12539 (libvirtd)
      Tasks: 22 (limit: 32768)
     Memory: 11.4M (peak: 12.7M)
        CPU: 619ms
     CGroup: /system.slice/libvirtd.service
             ├─ 7367 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro --dhcp-script=/usr/lib/libvirt/libvirt_leaseshelper
             ├─ 7368 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro --dhcp-script=/usr/lib/libvirt/libvirt_leaseshelper
             └─12539 /usr/sbin/libvirtd --timeout 120

abr 12 12:55:49 uoc-cloud systemd[1]: Starting libvirtd.service - libvirt legacy monolithic daemon...
abr 12 12:55:49 uoc-cloud systemd[1]: Started libvirtd.service - libvirt legacy monolithic daemon.
abr 12 12:55:49 uoc-cloud dnsmasq[7367]: read /etc/hosts - 8 names
abr 12 12:55:49 uoc-cloud dnsmasq[7367]: read /var/lib/libvirt/dnsmasq/default.addnhosts - 0 names
abr 12 12:55:49 uoc-cloud dnsmasq-dhcp[7367]: read /var/lib/libvirt/dnsmasq/default.hostsfile
astoian@uoc-cloud:~$
```

Una vez instalado KVM se ha de instalar LXD siendo la herramienta que nos permitirá crear contenedores en Linux. Se ha optado por LXD siendo una herramienta que ofrece una capa superior de abstracción con funcionalidades avanzadas facilitando la gestión de contenedores en entornos de trabajo.

Para hacerlo hacemos uso del comando *snap* que nos permitirá instalar y configurar paquetes de instalación de forma mas sencilla que la forma convencional.

```
astoian@uoc-cloud:~$ sudo apt-get update
Obj:1 https://apt.releases.hashicorp.com noble InRelease
Obj:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu noble InRelease
Des:4 http://es.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Obj:5 http://es.archive.ubuntu.com/ubuntu noble-backports InRelease
Descargados 126 kB en 0s (297 kB/s)
Leyendo lista de paquetes... Hecho
astoian@uoc-cloud:~$
astoian@uoc-cloud:~$
astoian@uoc-cloud:~$
astoian@uoc-cloud:~$ sudo snap install lxd
lxd (5.21/stable) 5.21.3-c5ae129 from Canonical✓ installed
astoian@uoc-cloud:~$
```

```
astoian@uoc-cloud:~$ sudo lxd init
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (powerflex, zfs, btrfs, ceph, dir, lvm) [default=zfs]:
Create a new ZFS pool? (yes/no) [default=yes]:
Would you like to use an existing empty block device (e.g. a disk or partition)? (yes/no) [default=no]:
Size in GiB of the new loop device (1GiB minimum) [default=30GiB]:
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
Would you like the LXD server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
astoian@uoc-cloud:~$
```

Se realiza una preconfiguración por defecto, al ejecutar un contenedor y se nos olvida establecer algún parámetro por defecto se ejecuta lxd init para establecer parámetros por defecto de los contenedores.

OpenNebula, al orquestar contenedores a través de LXD, va a basarse en esa configuración base de LXD como el almacenamiento , redes, etc.. Por tanto, si no has configurado previamente un pool y una red con lxd init, OpenNebula puede encontrar un entorno incompleto y podrías toparte con errores o necesitar más pasos de configuración.

- OpenNebula: Sabemos la importancia y función de OpenNebula en el proyecto, sin embargo para nuestro escenario se va a realizar la instalación de la versión **Community *Stable 6.10.0*** siento una versión menos testeada y dependiendo de la comunidad de OpenNebula, la única comparación con el resto de las versiones es el soporte que obtendremos de los ingenieros de la plataforma y la investigación y pruebas realizadas previas a su lanzamiento. Para realizar la instalación de la plataforma simplemente podemos hacerlo siguiendo los pasos de la documentación oficial.

```
astoian@uoc-cloud:~$ sudo apt-get update
[sudo] contraseña para astoian:
Obj:1 https://apt.releases.hashicorp.com noble InRelease
Obj:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu noble InRelease
Des:4 http://es.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Obj:5 http://es.archive.ubuntu.com/ubuntu noble-backports InRelease
Descargados 126 kB en 1s (109 kB/s)
Leyendo lista de paquetes... Hecho
astoian@uoc-cloud:~$ sudo apt-get -y install gnupg wget apt-transport-https
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
gnupg ya está en su versión más reciente (2.4.4-2ubuntu17.2).
wget ya está en su versión más reciente (1.21.4-1ubuntu4.1).
fijado wget como instalado manualmente.
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libllvm17t64 python3-netifaces
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes NUEVOS:
  apt-transport-https
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 45 no actualizados.
Se necesita descargar 3.974 B de archivos.
Se utilizarán 35,8 kB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu noble/universe amd64 apt-transport-https all 2.7.14build2 [3.974 B]
Descargados 3.974 B en 0s (46,8 kB/s)
Seleccionando el paquete apt-transport-https previamente no seleccionado.
(Leyendo la base de datos ... 212411 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../apt-transport-https_2.7.14build2_all.deb ...
Desempaquetando apt-transport-https (2.7.14build2) ...
Configurando apt-transport-https (2.7.14build2) ...
```

```
root@uoc-cloud: /home/asto
astoian@uoc-cloud:~$ sudo su
root@uoc-cloud:/home/astoian#
root@uoc-cloud:/home/astoian#
root@uoc-cloud:/home/astoian#  echo "deb [signed-by=/etc/apt/keyrings/opennebula.gpg] https://downloads.opennebula.io/repo/6.10/U
buntu/24.04 stable opennebula" > /etc/apt/sources.list.d/opennebula.list
root@uoc-cloud:/home/astoian# apt-get update
Obj:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu noble InRelease
Obj:3 https://apt.releases.hashicorp.com noble InRelease
Obj:4 http://es.archive.ubuntu.com/ubuntu noble-updates InRelease
Ign:5 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable InRelease
Des:6 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable Release [2.560 B]
Des:7 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable Release.gpg [833 B]
Obj:8 http://es.archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:7 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable Release.gpg
Leyendo lista de paquetes... Hecho
W: Error de GPG: https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable Release: Las firmas siguientes no se pudieron veri
ficar porque su clave pública no está disponible: NO_PUBKEY 05A05927906DC27C
E: El repositorio «https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable Release» no está firmado.
N: No se puede actualizar de un repositorio como este de forma segura y por tanto está deshabilitado por omisión.
N: Vea la página de manual apt-secure(8) para los detalles sobre la creación de repositorios y la configuración de usuarios.
root@uoc-cloud:/home/astoian# wget -q -O- https://downloads.opennebula.io/repo/repo2.key | gpg --dearmor --yes --output /etc/apt/
keyrings/opennebula.gpg
root@uoc-cloud:/home/astoian# apt-get update
Obj:1 https://apt.releases.hashicorp.com noble InRelease
Obj:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:3 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable InRelease
Des:4 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable Release [2.560 B]
Obj:5 http://es.archive.ubuntu.com/ubuntu noble InRelease
Des:6 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable Release.gpg [833 B]
Obj:7 http://es.archive.ubuntu.com/ubuntu noble-updates InRelease
Obj:8 http://es.archive.ubuntu.com/ubuntu noble-backports InRelease
Des:9 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable/opennebula i386 Packages [2.663 B]
Des:10 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable/opennebula amd64 Packages [4.883 B]
Descargados 10,9 kB en 1s (17,6 kB/s)
Leyendo lista de paquetes... Hecho
root@uoc-cloud:/home/astoian#
```

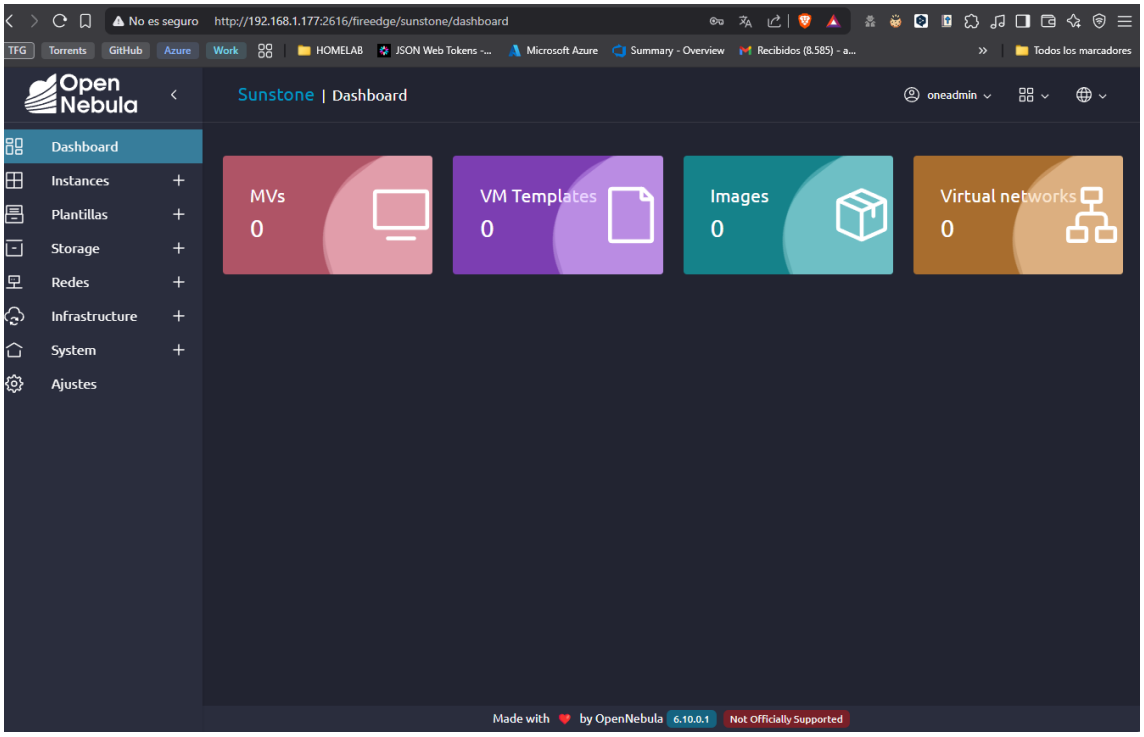
Es importante añadir el repositorio de descarga a la librería de Ubuntu, sin embargo tras realizarlo y hacer una actualización de repositorios nos puede salir un error de firma no coincide por lo tanto se ha de descargar la clave GPG de esta y podemos realizar la actualización de repositorio.

Para instalar OpenNebula se necesitan instalar diversos servicios

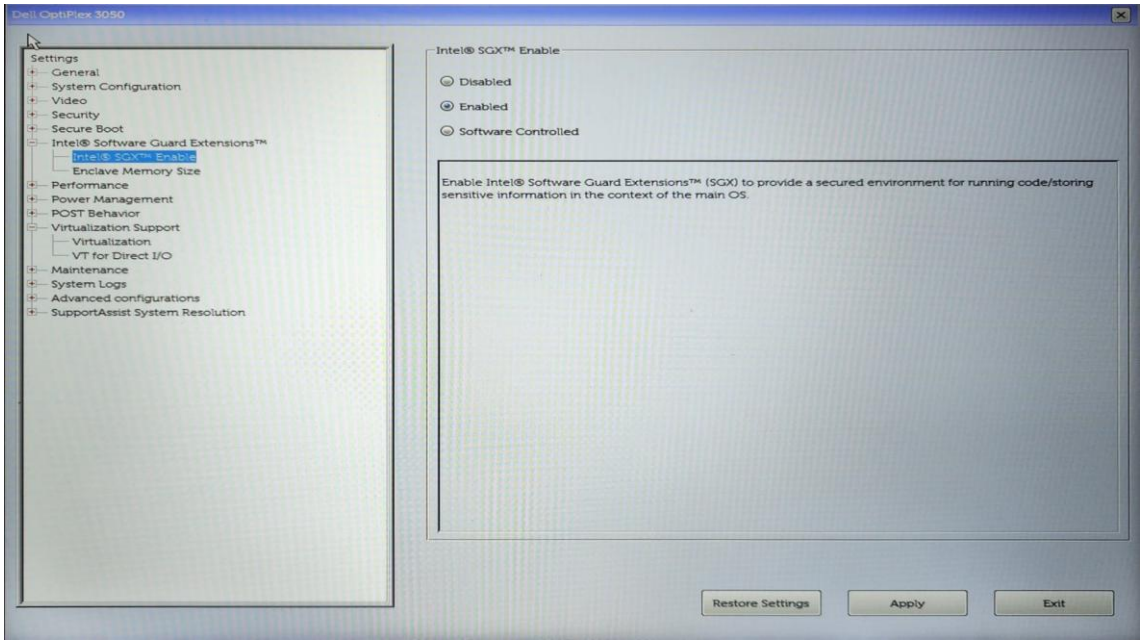
- ✓ opennebula – daeamon principal y sheduler siendo este el núcleo de OpenNebula.
- ✓ opennebula-fireedge - Interfaz grafica web avanzada
- ✓ opennebula-gate – funciones conectividad interna entre maquinas y contenedores.
- ✓ opennebula-flow – incluye funciones de escalabilidad
- ✓ opennebula-provision – herramientas de aprovisionamiento
- ✓ opennebula-node-kvm – herramientas para uso de cómputo para máquinas virtuales
- ✓ opennebula-node-lxc - herramientas para el despliegue y uso de contenedores LXD.

```
root@uoc-cloud:/home/astoian# apt-get -y install opennebula opennebula-fireedge opennebula-gate opennebula-flow opennebula-provision  opennebula-node-kvm
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libllvm17t64 python3-netifaces
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  ansible ansible-core augeas-lenses augeas-tools comerr-dev cppzmq-dev curl fonts-lato genisoimage icu-devtools ipset
  iputils-arping javascript-common krb5-multidev libaugeas0 libbsd-dev libfreerdp-client2-2t64 libfreerdp2-2t64 libgssrpc4t64
  libicu-dev libipset13 libjs-jquery libkadm5clnt-mit12 libkadm5srv-mit12 libkdb5-10t64 libkrb5-dev libmd-dev libmysqlclient21
  libnbd-bin libnbd0 libnode109 libnorm-dev libpgm-dev libruby libruby3.2 libsodium-dev libssh2-1t64 libwinpr2-2t64 libxml2-dev
  libzmq3-dev mysql-common node-acorn node-busboy node-cjs-module-lexer node-undici node-xtend nodejs nodejs-doc
  opennebula-common opennebula-common-onecfg opennebula-guacd opennebula-libs opennebula-provision-data opennebula-rubygems
  opennebula-tools python3-argcomplete python3-dnspython python3-jmespath python3-kerberos python3-libcloud python3-lockfile
  python3-ntlm-auth python3-packaging python3-passlib python3-requests-ntlm python3-resolvelib python3-selinux
  python3-simplejson python3-winrm python3-xmltodict rake ruby ruby-net-telnet ruby-rubygems ruby-sdbm ruby-sqlite3
  ruby-webrick ruby-xmlrpc ruby3.2 rubygems-integration sqlite3 vlan
Paquetes sugeridos:
  cowsay sshpass augeas-doc doc-base wodim cdrkit-doc apache2 | lighttpd | httpd krb5-doc freerdp2-x11 krb5-user icu-doc
  libnorm-doc pkg-config npm mysql-server gnuplot-nox python3-trio python3-aioquic python3-h2 python3-httpx python3-httpcore
  python-lockfile-doc ri ruby-dev bundler sqlite3-doc
Se instalarán los siguientes paquetes NUEVOS:
  ansible ansible-core augeas-lenses augeas-tools comerr-dev cppzmq-dev curl fonts-lato genisoimage icu-devtools ipset
  iputils-arping javascript-common krb5-multidev libaugeas0 libbsd-dev libfreerdp-client2-2t64 libfreerdp2-2t64 libgssrpc4t64
  libicu-dev libipset13 libjs-jquery libkadm5clnt-mit12 libkadm5srv-mit12 libkdb5-10t64 libkrb5-dev libmd-dev libmysqlclient21
  libnbd-bin libnbd0 libnode109 libnorm-dev libpgm-dev libruby libruby3.2 libsodium-dev libssh2-1t64 libwinpr2-2t64 libxml2-dev
  libzmq3-dev mysql-common node-acorn node-busboy node-cjs-module-lexer node-undici node-xtend nodejs nodejs-doc opennebula
  opennebula-common opennebula-common-onecfg opennebula-fireedge opennebula-flow opennebula-gate opennebula-guacd
  opennebula-libs opennebula-node-kvm opennebula-provision opennebula-provision-data opennebula-rubygems opennebula-tools
  python3-argcomplete python3-dnspython python3-jmespath python3-kerberos python3-libcloud python3-lockfile python3-ntlm-auth
  python3-packaging python3-passlib python3-requests-ntlm python3-resolvelib python3-selinux python3-simplejson python3-winrm
  python3-xmltodict rake ruby ruby-net-telnet ruby-rubygems ruby-sdbm ruby-sqlite3 ruby-webrick ruby-xmlrpc ruby3.2
  rubygems-integration sqlite3 vlan
0 actualizados, 88 nuevos se instalarán, 0 para eliminar y 45 no actualizados.
Se necesita descargar 149 MB de archivos.
Se utilizarán 1.140 MB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu noble/main amd64 fonts-lato all 2.015-1 [2.781 kB]
Des:2 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable/opennebula amd64 opennebula-common-onecfg all 6.10.0.1-1 [4.972 B]
```

Una vez creado el usuario de este e iniciado los servicios siguiendo las indicaciones de la documentación oficial ya podemos acceder al panel de control de OpenNebula. http://192.168.1.177:2616/fireedge/sunstone/dashboard



- Confidential Computing (opcional en laboratorio): En el escenario dispuesto para el proyecto, es importante destacar que el hardware lo soporta (Intel SGX). Para habilitar esta característica se ha entrado a la BIOS del equipo y se ha habilitado esta propiedad como podemos observar en la imagen.



- Ansible: Versión, plugins o colecciones relevantes (para la configuración de hosts).

  Tal como se indica en la documentación oficial, los módulos de openNebula forman parte de la colección community.general.

  Sera necesario instalar las dependencias de Python3.

  Se ejecuta en el servidor el comando de instalación del paquete pythone3-pyone.

```
astoian@uoc-cloud:~$ sudo apt install python3-pyone
[sudo] contraseña para astoian:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libllvm17t64 python3-netifaces
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  python3-bs4 python3-cssselect python3-dict2xml python3-html5lib python3-lxml python3-soupsieve python3-webencodings
Paquetes sugeridos:
  python3-genshi python-lxml-doc
Se instalarán los siguientes paquetes NUEVOS:
  python3-bs4 python3-cssselect python3-dict2xml python3-html5lib python3-lxml python3-pyone python3-soupsieve python3-webencodings
0 actualizados, 8 nuevos se instalarán, 0 para eliminar y 45 no actualizados.
Se necesita descargar 1.855 kB de archivos.
Se utilizarán 12,3 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://es.archive.ubuntu.com/ubuntu noble/main amd64 python3-soupsieve all 2.5-1 [33,0 kB]
Des:2 http://es.archive.ubuntu.com/ubuntu noble/main amd64 python3-bs4 all 4.12.3-1 [109 kB]
Des:3 http://es.archive.ubuntu.com/ubuntu noble/universe amd64 python3-dict2xml all 1.7.5-1 [11,9 kB]
Des:4 http://es.archive.ubuntu.com/ubuntu noble/main amd64 python3-webencodings all 0.5.1-5 [11,5 kB]
Des:5 http://es.archive.ubuntu.com/ubuntu noble/main amd64 python3-html5lib all 1.1-6 [88,8 kB]
Des:6 http://es.archive.ubuntu.com/ubuntu noble/main amd64 python3-lxml amd64 5.2.1-1 [1.243 kB]
Des:7 http://es.archive.ubuntu.com/ubuntu noble/main amd64 python3-cssselect all 1.2.0-2 [18,5 kB]
Des:8 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable/opennebula amd64 python3-pyone all 6.10.0.1-1 [339 kB]
Descargados 1.855 kB en 1s (3.164 kB/s)
Seleccionando el paquete python3-soupsieve previamente no seleccionado.
(Leyendo la base de datos ... 347187 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../0-python3-soupsieve_2.5-1_all.deb ...
Desempaquetando python3-soupsieve (2.5-1) ...
Seleccionando el paquete python3-bs4 previamente no seleccionado.
Preparando para desempaquetar .../1-python3-bs4_4.12.3-1_all.deb ...
Desempaquetando python3-bs4 (4.12.3-1) ...
Seleccionando el paquete python3-dict2xml previamente no seleccionado.
Preparando para desempaquetar .../2-python3-dict2xml_1.7.5-1_all.deb ...
Desempaquetando python3-dict2xml (1.7.5-1) ...
Seleccionando el paquete python3-webencodings previamente no seleccionado.
Preparando para desempaquetar .../3-python3-webencodings_0.5.1-5_all.deb ...
Desempaquetando python3-webencodings (0.5.1-5) ...
Seleccionando el paquete python3-html5lib previamente no seleccionado.
Preparando para desempaquetar .../4-python3-html5lib_1.1-6_all.deb ...
Desempaquetando python3-html5lib (1.1-6) ...
Seleccionando el paquete python3-lxml:amd64 previamente no seleccionado.
Preparando para desempaquetar .../5-python3-lxml_5.2.1-1_amd64.deb ...
Desempaquetando python3-lxml:amd64 (5.2.1-1) ...
Seleccionando el paquete python3-cssselect previamente no seleccionado.
Preparando para desempaquetar .../6-python3-cssselect_1.2.0-2_all.deb ...
Desempaquetando python3-cssselect (1.2.0-2) ...
Seleccionando el paquete python3-pyone previamente no seleccionado.
```

Y Realizamos la instalación de ansible versión 10.5.0

```
astoian@uoc-cloud:~$ ansible-galaxy collection install community.general
Starting galaxy collection install process
Nothing to do. All requested collections are already installed. If you want to reinstall them, consider using `--force`.
astoian@uoc-cloud:~$
```

- Monitorización: Instalación de Prometheus y Grafana se realizará en el apartado de monitorización, donde vamos a mostrar simplemente el resultado final de la implementación.

Todos los recursos que se crearan en OpenNebula será utilizando Terraform.

6.2. Estructura de ficheros Terraform

La infraestructura declarada para desplegar el entorno de maquinas virtuales con Confidenctial Computing sobre OpenNebula se organiza en cuatro ficheros principales. La segmentación de los archivos se ha separado por funciones y conceptos de uno, es decir cada fichero terraform desempeña una función en la construcción de la infraestructura. Esta metodología llamada *separation of concerns* ayuda a la trazabilidad de cambio y la reutilización del código.

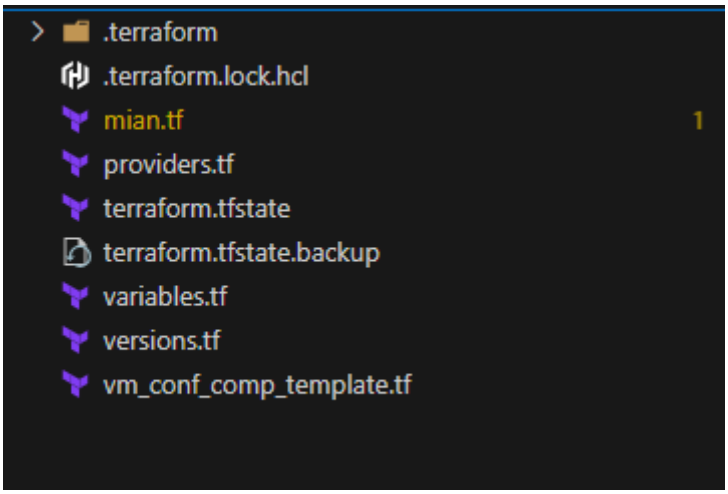
El primer archivo terraform necesario para realizar el despliegue es **providers.tf**

Este contiene la definición del provider OpenNebula responsable de la traducción de los recursos declarativos al API de frontend de OpenNebula.

El segundo archivo necesario es **versions.tf** en este archivo de definen las versiones de las herramientas empleadas para garantizar la creación de infraestructura sin que implique versiones deprecadas o fallos relacionados a versiones anteriores o deprecadas.

En tercer lugar, tenemos el archivo **vm_conf_comp_template.tf**, este archivo se utiliza para definir una plantilla o template de base para el despliegue de máquinas virtuales Confidencial Computing con soporte de Intel SGX.

Por último, tenemos el fichero **main.tf** que actúa como punto de orquestación que consume de módulos y crea los recursos concretos. Este fichero crea una red virtual, grupos de seguridad aplicados a la red virtual, una imagen de Ubuntu, y máquinas virtuales.



6.3. Creación de plantillas en OpenNebula

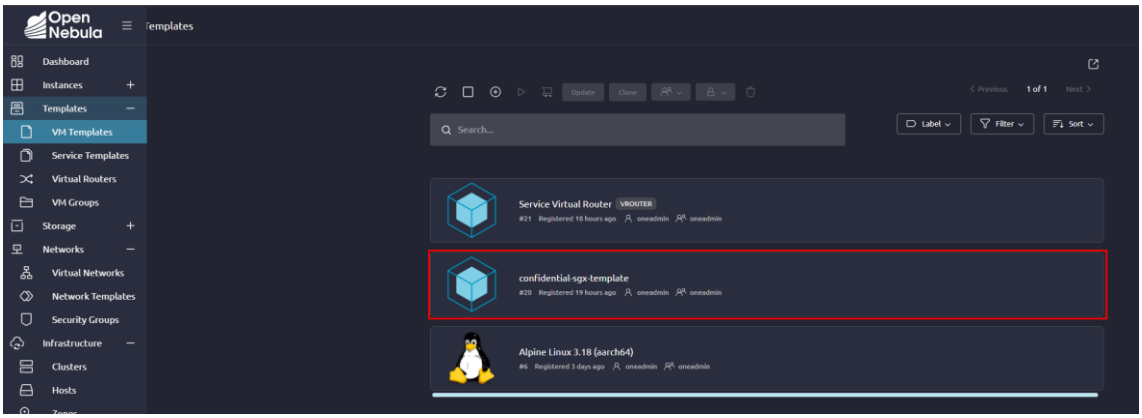
Para la creación de plantillas de maquinas virtuales con propiedades de confidencial computing, utilizando tecnologías soportadas por el hardware del servidor de virtualización y OpenNebula siendo en nuestro caso Intel SGX.

Se ha realizado la creación de esta plantilla que describe la configuración mínima que ha de tener cada maquina virtual antes de ser instanciada.

```
resource "opennebula_template" "confidential_vm_template" {
  name        = "confidential-sgx-template"
  permissions = "660"
  group       = "oneadmin"
  #-- Recursos de cómputo --
  memory = 1024 # MiB
  cpu    = 1    # fracción de core
  vcpu   = 1    # núcleos virtuales asignados
  #-- Contextualización --
  context = {
    NETWORK      = "YES"
    SET_HOSTNAME = "$NAME"
    START_SCRIPT = "echo -e '123456\n123456' | passwd root"
  }
  #-- Parámetros de arranque --
  os {
    arch = "x86_64"
    boot = "disk0"
  }
  #-- Modelo de CPU --
  cpumodel {
    model = "host-passthrough" # expone las instrucciones SGX del host
  }
```

```
  #-- Consola gráfica --
  graphics {
    type   = "VNC"
    listen = "0.0.0.0"
    keymap = "es"
  }

  disk {
    image_id = opennebula_image.ubuntu_image.id
    size     = 10000             # capacidad del disco virtual en MB
    target   = "vda"
    driver   = "qcow2"
  }


  #-- Interfaz de red --
  nic {
    model           = "virtio"
    network_id      = opennebula_virtual_network.private_net.id
    security_groups = [opennebula_security_group.basic_sg.id]
  }
  #-- Habilitar SGX en libvirt / KVM --
  raw {
    type = "kvm"
    data = "<cpu mode='host-passthrough'><feature policy='require'
name='sgx'/></cpu>"
  }

}
```

Con esta plantilla se puede instanciar maquinas virtuales que cumplan los requisitos y estándares de confidential computing.



> onevm instantiate confidential-sgx-template --name sgx-vm

Es importante destacar que para instanciar las maquinas virtuales con esta plantilla se ha de realizar mediante comando debido a que por interfaz web no es posible debido a un bug de la versión 6.10.

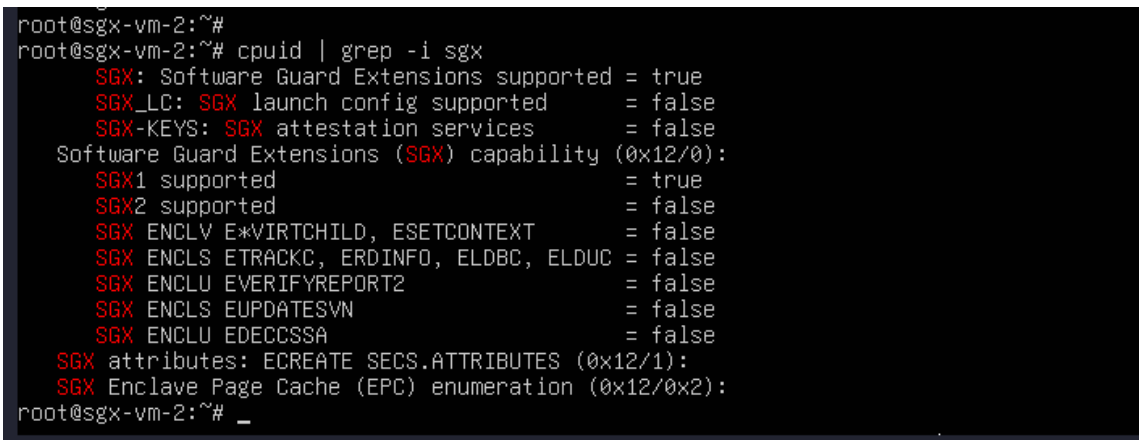
6.4. Habilitación de Confidential VMs

La función de confidencial computing, la cual ha de estar soportada por el hardware del servidor físico. Por lo tanto, aprovechar la tecnología Intel SGX de la cual dispone el servidor el objetivo consiste en que la VM haga uso de esta tecnología de procesamiento y computación.

Para habilitar esta opción en la maquina virtual se ha diseñado la plantilla expuesta en el apartado anterior donde utilizamos la opción de **host-passthrough** para garantizar que el huésped vea exactamente las extensiones de CPU del servidor físico. Además de utilizar el bloque RAW del KVM donde se indica la flag de SGX, la cual se indica que si el host no soporta esta opción la maquina virtual no arranca evitando de esta forma las ejecuciones inseguras.

Habilitando esta opción en la maquina virtual conseguimos un aislamiento en tiempo de ejecución por lo tanto se cifran los datos procesados y solo son accesible dentro del host. Por otra parte, conseguimos la integridad de datos por lo canto cualquier modificación en memoria genera excepciones, además de garantizar aun aislamiento completo del procesamiento de dichos datos. Esto es útil si se requiere demostración clientes o terceros garantizando seguridad de procesos.

Para comprobar que la máquina virtual desplegada utilizando la plantilla mencionada en el apartado anterior se ejecuta un comando para obtener información del procesador.



HABILITAR LUKS (*Linux Unified Key Setup*)

Mediante la extensión Intel SGX, se ha garantizado la computación confidencial de los datos en uso, es decir mientras están en memoria de la vm. No obstante, esta protección termina cuando los datos son liberados de la memoria RAM, dichos datos continúan expuestos en el disco de la VM. Para cubrir esta brecha de seguridad se recure al cifrado en reposo o (***data-at-rest***). En el ecosistema Linux el estándar es LUKS o (Linux Unified Key Setup), integrado en dm-crypt y soportado por KVM y LIBVIRT. La función de Luks es añadir una cabecera con múltiples key-short y cifra todos los bloques del disco con algoritmos simétricos por defecto siendo AES-XTS el mas común. Es por ello si la clave el archivo que se intenta leer es simplemente ruido o datos inconexos o aleatorios.

El objetivo de esta integración es determinar como combinar Confidential VMs con cifrado de disco en OpenNebula. Siendo que vamos a proteger los datos en todo momento, tanto los datos en uso siendo protegidos por Intel SGX en memoria y cpu. Y por otra parte los datos en reposo van a permanecer cifrados gracias a LUKS incluso dentro del datastore de OpenNebula. La entrega de claves se haga de forma transparente y automática mediante el subsistema de secretos de libvirt, sin intervención del usuario final.

Para hacer uso del código IaC se han re realizar una serie de configuraciones previas.

Lo primero es descargar la imagen de Ubuntu del Marketplace que vamos a utilizar porque el cifrado de la VM se comienza desde la imagen utilizada en la VM.

1.1 Descarga la qcow2 del Marketplace

>wget -O ubuntu24.qcow2
https://marketplace.opennebula.io/appliance/f4cc1890-f430-013c-b669-7875a4a4f528/download/0

```
root@uoc-cloud:/var/img_mkp# # 1.1 Descarga la qcow2 del Marketplace
wget -O ubuntu24.qcow2 \
   https://marketplace.opennebula.io/appliance/f4cc1890-f430-013c-b669-7875a4a4f528/download/0
--2025-05-15 13:11:40--  https://marketplace.opennebula.io/appliance/f4cc1890-f430-013c-b669-7875a4a4f
528/download/0
Resolviendo marketplace.opennebula.io (marketplace.opennebula.io)... 2600:3c01::f03c:94ff:fe7a:6861, 2
3.239.5.185
Conectando con marketplace.opennebula.io (marketplace.opennebula.io)[2600:3c01::f03c:94ff:fe7a:6861]:4
43... conectado.
Petición HTTP enviada, esperando respuesta... 302 Found
Ubicación: https://d24fmfybwxpuhu.cloudfront.net/ubuntu2404-6.10.0-3-20250127.qcow2 [siguiente]
--2025-05-15 13:11:41--  https://d24fmfybwxpuhu.cloudfront.net/ubuntu2404-6.10.0-3-20250127.qcow2
Resolviendo d24fmfybwxpuhu.cloudfront.net (d24fmfybwxpuhu.cloudfront.net)... 52.85.187.63, 52.85.187.5
, 52.85.187.196, ...
Conectando con d24fmfybwxpuhu.cloudfront.net (d24fmfybwxpuhu.cloudfront.net)[52.85.187.63]:443... cone
ctado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 586454528 (559M) [application/octet-stream]
Guardando como: ‘ubuntu24.qcow2’

ubuntu24.qcow2            100%[===================================>] 559,29M  73,6MB/s    en 7,7s

2025-05-15 13:11:49 (72,6 MB/s) - ‘ubuntu24.qcow2’ guardado [586454528/586454528]
```

Una vez tengamos la imagen de Ubuntu que vamos a utilizar se genera la clave y el UUID del secreto que vamos a utilizar en la creación de la imagen.

Se genera clave y se guarda en el archivo luks

>openssl rand -base64 10 | tr -d '=' > passphrase.luks

Se asigna una variable con un identificador.

>UUID=$(uuidgen)

Una vez almacenado el uuid y la cave vamos a convertir la imagen QCOW2 en un contenedor LUKS, con este comando creamos una imagen cifrada.

> qemu-img convert --object secret,id=sec0,file=passphrase.luks -O luks -o key-secret=sec0 ubuntu24.qcow2 /var/tmp /ubuntu24.luks

Por último se crea y se indica el secreto a Libvirt y para todos los nodos KVM necesarios el cual vamos a utilizar.

cat > secret.xml <<EOF

<secret ephemeral='no' private='yes'>

 <uuid>$UUID</uuid>

 <description>ubuntu24-luks</description>

</secret>

EOF

```
oneadmin@uoc-cloud:/var/tmp$ cat secret.xml
<secret ephemeral='no' private='yes'>
  <uuid>e1f493ef-2d1c-43b0-954e-de1fa08f7d7f</uuid>
  <description>ubuntu24-luks</description>
</secret>
oneadmin@uoc-cloud:/var/tmp$
```

Se realiza el registro del secreto en libvirt, indicándolo con el siguiente comando donde lo guarda en el almacén local de libvirt (por defecto /etc/libvirt/secrets/).

>virsh secret-define secret.xml

```
oneadmin@uoc-cloud:/var/tmp$ virsh secret-define secret.xml
Secret e1f493ef-2d1c-43b0-954e-de1fa08f7d7f created
```

Por ultimo se carga la clave real en el secreto y con ello tenemos ya listo el secreto libvirt podrá inyectar la clave dentro del XML de la VM al arrancar QEMU, usando un canal privado para que no aparezca en logs ni procesos.

> virsh secret-set-value e1f493ef-2d1c-43b0-954e-de1fa08f7d7f --file passphrase.luks --plain

```
oneadmin@uoc-cloud:/var/tmp$ virsh secret-set-value e1f493ef-2d1c-43b0-954e-de1fa08f7d7f --file passph
rase.luks --plain
Secret value set
```

Por último, se realizan cambios en los archivos terraform para implementar esta imagen y plantilla de VM con LUKS.

```
resource "opennebula_image" "ubuntu_image_luks" {
  name         = "Ubuntu 24.04 (LUKS)"
  description  = "Terraform image (LUKS-encrypted)"
  datastore_id = 1
  persistent   = false
  lock         = "UNLOCK"
  path         = "/var/tmp/ubuntu24.luks" # ruta del fichero cifrado
  dev_prefix   = "vd"
  driver       = "raw" # LUKS se expone 'raw'
  type         = "OS"  # OS, CDROM, DATABLOCK
  permissions  = "660"
  group        = "oneadmin"


  tags = {
    environment = "prod"
  }
```

```
  template_section {
    name = "Ubuntu_Dis"
    elements = {
      OS          = "Linux",
      VER         = "24.04"
      LUKS_SECRET = "e1f493ef-2d1c-43b0-954e-de1fa08f7d7f" # o pon aquí
directamente ${UUID}
    }
  }
}
```

Por ultimo adjuntamos la template VM que utilizara LUKS.

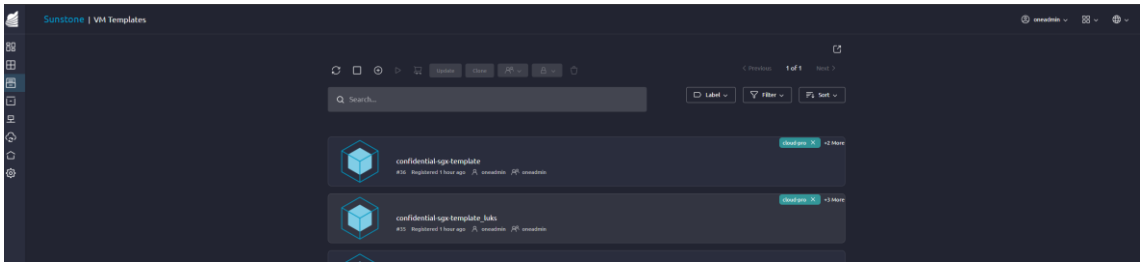
```
resource "opennebula_template" "confidential_vm_templat_luks" {
  name        = "confidential-sgx-template_luks"
  permissions = "660"
  group       = "oneadmin"
  #-- Recursos de cómputo --
  memory = 1024 # MiB
  cpu    = 1    # fracción de core
  vcpu   = 1    # núcleos virtuales asignados

  #-- Contextualización --
  context = {
    NETWORK         = "YES"
    USERNAME        = "admin"
    PASSWORD_BASE64 = "YWRtaW4x"
    SSH_PUBLIC_KEY  = "$USER[SSH_PUBLIC_KEY]"
    SET_HOSTNAME    = "$NAME"
    START_SCRIPT    = "echo -e '123456\n123456' | passwd root |sudo apt-
get update -y && sudo apt-get install sshpass -y"
  }
  #-- Parámetros de arranque --
  os {
    arch = "x86_64"
    boot = "disk0"
  }
  #-- Modelo de CPU --
  cpumodel {
    model = "host-passthrough" # expone las instrucciones SGX del host
  }
  #-- Consola gráfica --
  graphics {
    type   = "VNC"
    listen = "0.0.0.0"
    keymap = "es"
  }
```

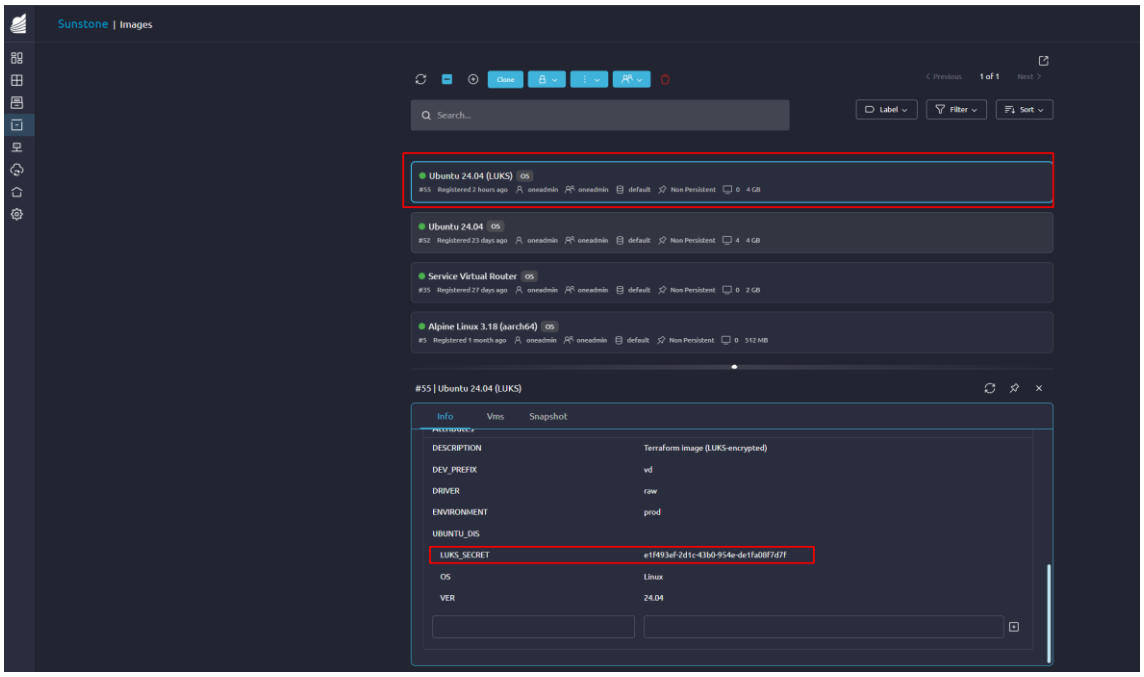
```
  disk {
    image_id = opennebula_image.ubuntu_image.id # imagen a utilizar
creada an el fichero main.tf
    size     = 10000                            # capacidad del disco
virtual en MB
    target   = "vda"
    driver   = "raw"
  }
  #-- Interfaz de red --
  nic {
    model           = "virtio"
    network_id      = opennebula_virtual_network.private_net.id
    security_groups = [opennebula_security_group.basic_sg.id]
  }
  #-- Habilitar SGX en libvirt / KVM --
  raw {
    type = "kvm"
    data = "<cpu mode='host-passthrough'><feature policy='require'
name='sgx'/></cpu>"
  }
  tags = {
    environment     = "uoclab"
    deployment_mode = "terraform"
    image           = "ubuntu 24.04"
    LABELS          = "ubuntu,terraform,vm_luks,cloud-pro"
  }
}
```

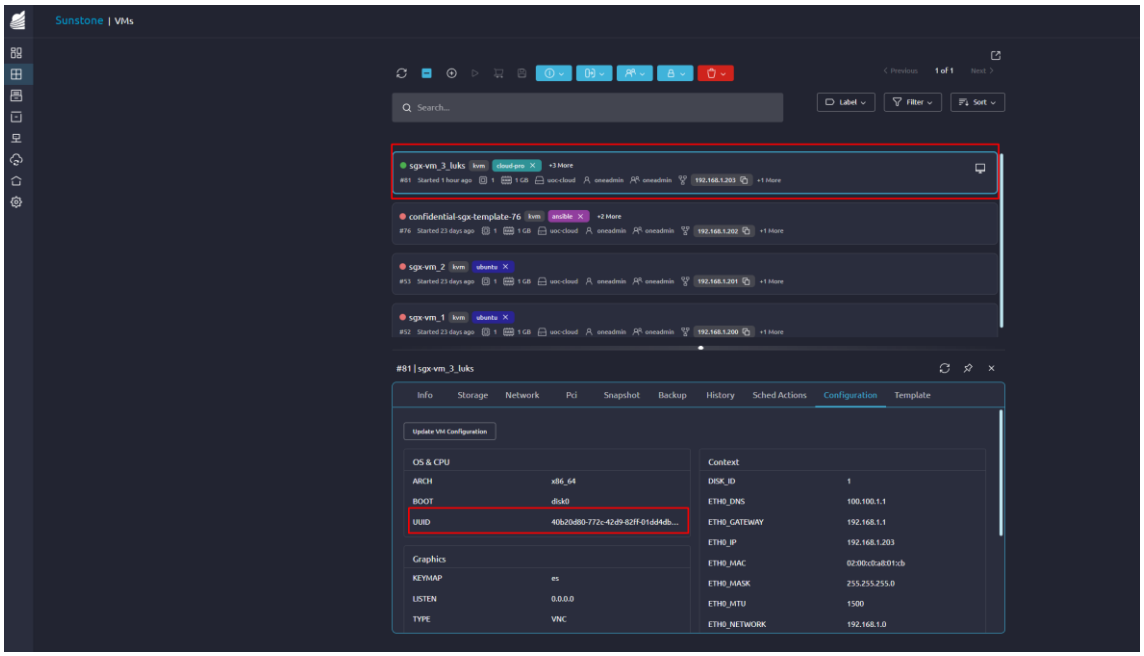
Execution del Código IaC

```
        # (6 unchanged blocks hidden)
    }

Plan: 0 to add, 2 to change, 0 to destroy.
opennebula_template.confidential_vm_template: Modifying... [id=36]
opennebula_template.confidential_vm_templat_luks: Modifying... [id=35]
opennebula_template.confidential_vm_templat_luks: Modifications complete after 0s [id=35]
opennebula_template.confidential_vm_template: Modifications complete after 0s [id=36]

Warning: Argument is deprecated

  with opennebula_virtual_network.private_net,
  on mian.tf line 99, in resource "opennebula_virtual_network" "private_net":
  99: resource "opennebula_virtual_network" "private_net" {

use virtual network address range resource instead

(and one more similar warning elsewhere)

Apply complete! Resources: 0 added, 2 changed, 0 destroyed.
PS C:\Users\alexandru.stoian\OneDrive - Plain Concepts\UOC\TFG\TerrraformFileOpenNebula>
```

Como podemos observar en estas capturas se ha habilitado LUKS







6.5. Automatización con Ansible

Con el objetivo de optimizar y estandarizar las configuraciones de las maquinas virtuales desplegadas en el entorno de OpenNebula se ha desarrollado un conjunto de plantillas o playbooks de Ansible que nos permite automatizar tanto el aprovisionamiento de nuevas maquinas como la configuracion inicial del sistema.

Para ello se utiliza un inventario dinámico basado en el plugin de la comunidad (**community.general.opennebula**), el cual permite la comunicación con la API de OpenNebula.

Como podemos ver en el archivo **inventory_opennebula.yml** se define la conexión con el servidor de OpenNebula facilitando las credenciales de conexión y nombre de host a identificar.

```
plugin: community.general.opennebula
api_url: http://192.168.1.177:2633/RPC2
api_username: oneadmin
api_password: oneadmin1
hostname: v4_first_ip
```

La ejecucion de playbook se realiza con el siguiente comando:
> ansible-inventory -i inventory_opennebula.yml –graph
> ansible-inventory -i inventory_opennebula.yml –list

```
alex@ZG8-5CG21323CW:~/.ansible/collections/ansible_custom$ ansible-inventory -i inventory_opennebula.yml --graph
@all:
  |--@ungrouped:
  |  |--sgx-vm_1
  |  |--sgx-vm_2
alex@ZG8-5CG21323CW:~/.ansible/collections/ansible_custom$ ansible-inventory -i inventory_opennebula.yml --graph
@all:
  |--@ungrouped:
  |  |--sgx-vm_1
  |  |--sgx-vm_2
alex@ZG8-5CG21323CW:~/.ansible/collections/ansible_custom$ ansible-inventory -i inventory_opennebula.yml --list
{
    "_meta": {
        "hostvars": {
            "sgx-vm_1": {
                "HYPERVISOR": "kvm",
                "LABELS": [],
                "LOGO": "images/logos/ubuntu.png",
                "SCHED_REQUIREMENTS": "(HYPERVISOR=kvm)",
                "TAG": "ubuntu",
                "ansible_host": "192.168.1.200",
                "host": "uoc-cloud",
                "id": 45,
                "name": "sgx-vm_1",
                "v4_first_ip": "192.168.1.200",
                "v6_first_ip": false
            },
            "sgx-vm_2": {
                "HYPERVISOR": "kvm",
                "LABELS": [],
                "LOGO": "images/logos/ubuntu.png",
                "SCHED_REQUIREMENTS": "(HYPERVISOR=kvm)",
                "ansible_host": "192.168.1.201",
                "host": "uoc-cloud",
                "id": 44,
                "name": "sgx-vm_2",
                "v4_first_ip": "192.168.1.201",
                "v6_first_ip": false
            }
        }
    },
    "all": {
        "children": [
            "ungrouped"
        ]
    },
    "ungrouped": {
        "hosts": [
            "sgx-vm_1",
            "sgx-vm_2"
        ]
    }
}
alex@ZG8-5CG21323CW:~/.ansible/collections/ansible_custom$
```

Otra playbook permite realizar una configuración más específica sobre las VMs ya desplegadas. Entre las tareas de configuración se destacan tareas de actualización y configuración como:

- Actualización de la cache de paquetes APT
- Instalación de paquetes esenciales para la administración de sistemas como git, curl y htop.
- Creación de un usuario administrador adicional con permisos de sudo y acceso a SSH mediante clave pública.
- Configuración de e la zona horaria del sistema a Europa/Madrid
- Instalación y configuración del sistema de actualización automáticas *unattended-upgrades*.
- Reinicio del servicio SSH para aplicar los cambios.

```
- name: Configurar Ubuntu en OpenNebula
  hosts: sgx-vm_* # ← 'ungrouped', o un patrón como 'sgx-vm_*'
  become: yes
  vars:
    # credenciales SSH
    ansible_user: root                    # usuario dentro de la VM
    ansible_ssh_private_key_file: ~/.ssh/id_rsa
    # si sudo pide password
    ansible_become_password: 123456
    ansible_python_interpreter: /usr/bin/python3

  tasks:
    - name: Actualizar caché APT
      ansible.builtin.apt:
        update_cache: yes
        cache_valid_time: 3600

    - name: Instalar paquetes base
      ansible.builtin.apt:
        name:
          - git
          - curl
          - htop
        state: present

    - name: Crear usuario administrador adicional
      ansible.builtin.user:
        name: adminuser
        groups: sudo
        shell: /bin/bash
        create_home: yes

    - name: Configurar acceso SSH para usuario administrador adicional
```

```
    ansible.posix.authorized_key:
      user: adminuser
      state: present
      key: "{{ lookup('file', '~/.ssh/id_rsa.pub') }}"

  - name: Configurar zona horaria del servidor
    community.general.timezone:
      name: Europe/Madrid

  - name: Habilitar actualizaciones automáticas del sistema
    ansible.builtin.apt:
      name: unattended-upgrades
      state: present

  - name: Copiar configuración de unattended-upgrades
    ansible.builtin.template:
      src: 50unattended-upgrades.j2
      dest: /etc/apt/apt.conf.d/50unattended-upgrades

  - name: Reiniciar servicio SSH
    ansible.builtin.service:
      name: ssh
      state: restarted
```

La ejecución del playbook de esta plantilla se realiza con este comando.

> ansible-playbook -i inventory_opennebula.yml playbooks/configurar_ubuntu.yml

```
alex@ZG8-5CG21323CW:~/.ansible/collections/ansible_custom$ ansible-playbook -i inventory_opennebula.yml playbooks/configurar_ubuntu.yml

PLAY [Configurar Ubuntu en OpenNebula] ***************************************************************************************************

TASK [Gathering Facts] *******************************************************************************************************************
ok: [sgx-vm_1]
ok: [sgx-vm_2]

TASK [Actualizar caché APT] **************************************************************************************************************
ok: [sgx-vm_2]
ok: [sgx-vm_1]

TASK [Instalar paquetes base] ************************************************************************************************************
ok: [sgx-vm_2]
ok: [sgx-vm_1]

TASK [Crear usuario administrador adicional] *********************************************************************************************
ok: [sgx-vm_2]
ok: [sgx-vm_1]

TASK [Configurar acceso SSH para usuario administrador adicional] ************************************************************************
ok: [sgx-vm_2]
ok: [sgx-vm_1]

TASK [Configurar zona horaria del servidor] **********************************************************************************************
ok: [sgx-vm_2]
ok: [sgx-vm_1]

TASK [Habilitar actualizaciones automáticas del sistema] *********************************************************************************
ok: [sgx-vm_2]
ok: [sgx-vm_1]

TASK [Copiar configuración de unattended-upgrades] ***************************************************************************************
changed: [sgx-vm_2]
changed: [sgx-vm_1]

TASK [Reiniciar servicio SSH] ************************************************************************************************************
changed: [sgx-vm_2]
changed: [sgx-vm_1]

PLAY RECAP *******************************************************************************************************************************
sgx-vm_1                   : ok=9    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
sgx-vm_2                   : ok=9    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

alex@ZG8-5CG21323CW:~/.ansible/collections/ansible_custom$
```

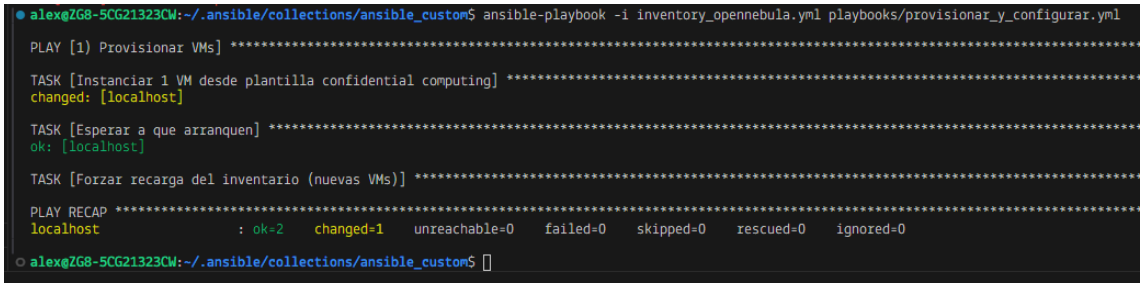
Por último se ha creado este playbook de aprovisionamiento y configuración de nuevas VMs (**provisionar_y_configurar.yml**) , en este archivo se automatiza el despliegue de maquinas virtuales a partir de una plantilla previamente creada con Terraform. Se utiliza el módulo one_vm de la comunidad para instancias una nueva VM. Posteriormente se espera que la maquina arranque completamente y se fuerza la recarga del inventario de máquinas dinámico para incluir la nueva instancia en el flujo de configuración.

Una vez este la nueva maquina configurada se aplica una segunda fase de configuración de la VM con las etiquetas definidas anteriormente , instalando herramientas básicas de configuración y actualizando el caché de paquetes APT.

```
- name: 1) Provisionar VMs
  hosts: localhost
  gather_facts: no
  collections:
    - community.general
  tasks:
    - name: Instanciar 1 VM desde plantilla confidential computing
      community.general.one_vm:
        api_url: http://192.168.1.177:2633/RPC2
        api_username: oneadmin
        api_password: oneadmin1
        template_id: 32
        count: 1
        labels:
          - ubuntu
          - web
          - ansible
      register: new_vms
      run_once: true # una unica ejecucion

    - name: Esperar a que arranquen
      community.general.one_vm:
        instance_ids: "{{ new_vms.instances_ids }}"
        state: running
        api_url: http://192.168.1.177:2633/RPC2
      register: ready
      retries: 20
      delay: 15
      until: (ready.instances | selectattr('state', 'equalto', 'ACTIVE')
| list | length) == (new_vms.instances_ids | length)

    - name: Forzar recarga del inventario (nuevas VMs)
      meta: refresh_inventory
      run_once: true
```

```yaml
# ← NUEVO PLAY, MISMO NIVEL DE RAÍZ
- name: 2) Configurar las nuevas VMs
  hosts: ubuntu:&web:&ansible        # intersección de etiquetas
  become: yes
  vars:
    ansible_user: root
    ansible_ssh_private_key_file: ~/.ssh/id_rsa
    ansible_python_interpreter: /usr/bin/python3
  tasks:
    - name: Actualizar caché APT
      ansible.builtin.apt:
        update_cache: yes
        cache_valid_time: 3600

    - name: Instalar paquetes base
      ansible.builtin.apt:
        name:
          - git
          - curl
          - htop
        state: present
```

> ansible-playbook -i inventory_opennebula.yml playbooks/provisionar_y_configurar.yml



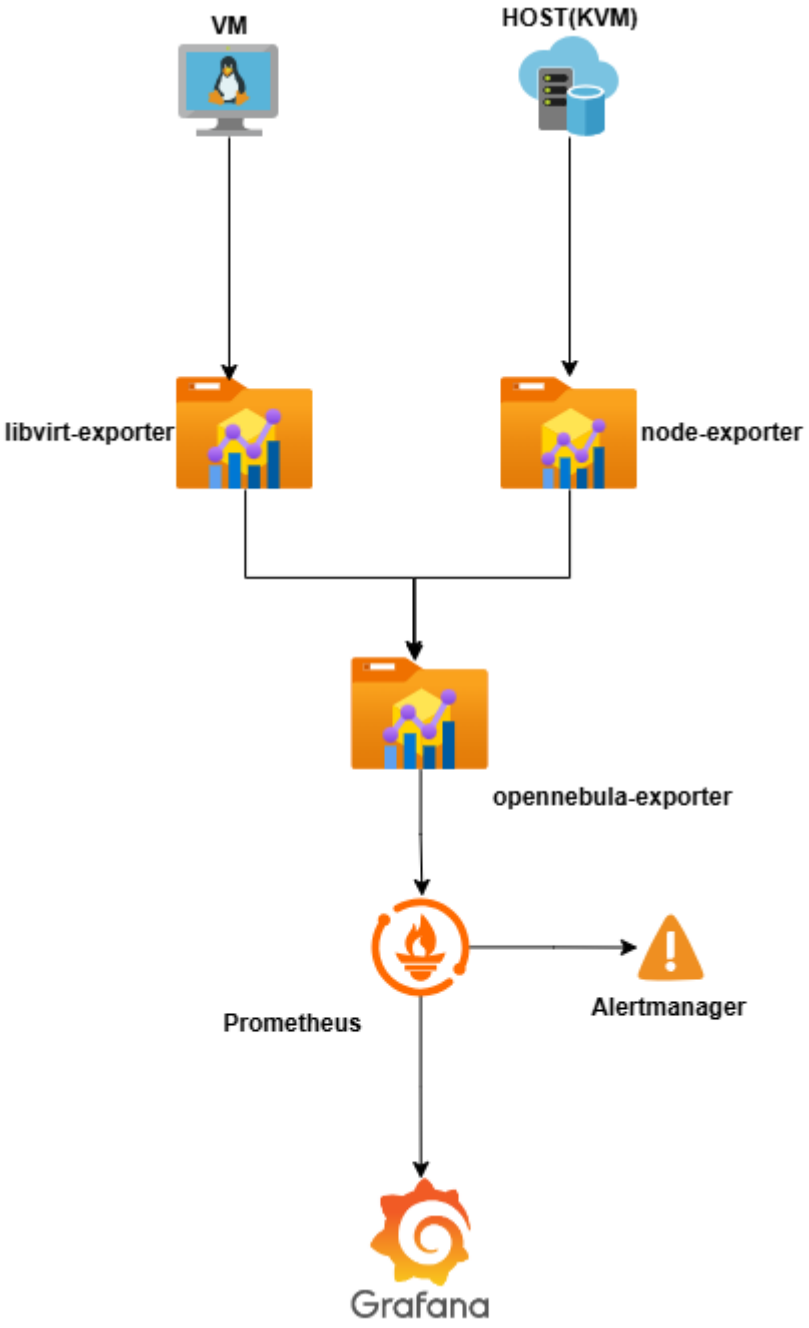
6.6. Monitorización con Prometheus y Grafana

La implementación de un sistema de monitorización es fundamental para hacer una trazabilidad del sistema y las maquinas virtuales que esta operativas en nuestro entorno OpenNebula. Tal como hemos comentado anteriormente se hace uso de prometheus para obtener las métricas del servidor OpenNebula y la visualización de estas hacemos uso de Grafana.

Comenzamos con la implementación implementando en nuestro servidor los paquetes necesarios. Los paquetes necesarios y instalados en nuestro servidor es incluyen Prometheus, Alertmanager, opennebula-exporter, node-exporter y libvirt-exporter ya configurados como servicios systemd.

- **Prometheus** para recoger, almacenar y consultar métricas en tiempo real.
- **Alertmanager** cuya función es la de recibir alertas desde Prometheus y envia notificaciones de caídas de servicios o eventos.
- **Opennebula-exporter** es el exportador oficial de OpenNebula cuya función es la de extraer métricas haciendo uso de comandos estándar y los expone a Prometheus.
- **Node-exporter** es el exportador sistema operativo base cuya función es la de exponer métricas del sistema base como la CPU, RAM, Disco, red….
- **Libvirt-exporter** es el exportador para el sistema de virtualización KVM, útil para obtener métricas de las máquinas virtuales desplegadas.

A continuación, se hace una representación del flujo de cada uno de estos paquetes.



Fuente: Elaboración propia

El comando de instalación de los paquetes necesarios son los siguientes.

sudo apt-get -y install opennebula-prometheus opennebula-prometheus-kvm

```
astoian@uoc-cloud:~$ sudo apt-get -y install opennebula-prometheus opennebula-prometheus-kvm
[sudo] contraseña para astoian:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libllvm17t64 python3-netifaces
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes NUEVOS:
  opennebula-prometheus opennebula-prometheus-kvm
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 1 no actualizados.
Se necesita descargar 128 MB de archivos.
Se utilizarán 356 MB de espacio de disco adicional después de esta operación.
Des:1 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable/opennebula amd64 opennebula-prometheus amd
64 6.10.0.1-1 [118 MB]
Des:2 https://downloads.opennebula.io/repo/6.10/Ubuntu/24.04 stable/opennebula amd64 opennebula-prometheus-kvm
 amd64 6.10.0.1-1 [9.727 kB]
Descargados 128 MB en 5s (25,6 MB/s)
Seleccionando el paquete opennebula-prometheus previamente no seleccionado.
(Leyendo la base de datos ... 347956 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../opennebula-prometheus_6.10.0.1-1_amd64.deb ...
Desempaquetando opennebula-prometheus (6.10.0.1-1) ...
Seleccionando el paquete opennebula-prometheus-kvm previamente no seleccionado.
Preparando para desempaquetar .../opennebula-prometheus-kvm_6.10.0.1-1_amd64.deb ...
Desempaquetando opennebula-prometheus-kvm (6.10.0.1-1) ...
Configurando opennebula-prometheus (6.10.0.1-1) ...
Configurando opennebula-prometheus-kvm (6.10.0.1-1) ...
astoian@uoc-cloud:~$
```

Una vez instalado listos los host visualizando el id del host necesario *onehost list*

```
oneadmin@uoc-cloud:/home/astoian$ onehost list
  ID NAME                                              CLUSTER    TVM      ALL
OCATED_CPU      ALLOCATED_MEM STAT
  16 uoc-cloud                                         uoc-cloud    3    300 /
 400 (75%)   3G / 15.4G (19%) on
oneadmin@uoc-cloud:/home/astoian$
```

Con la información obtenida del comando anterior se realiza la configuración de los parámetros y endpoints/*scrape targets* de prometheus localizados en /etc/one/prometheus/prometheus.yml

Este es el resultado de la modificación del fichero, donde se ha integrado los *scrape targets* necesarios.

```
  GNU nano 7.2                                    /etc/one/prometheus/prometheus.yml
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets: ['localhost:9093']

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files: ['rules.yml']

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: prometheus

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
    - targets: ['localhost:9090']

  - job_name: opennebula_exporter
    static_configs:
    - targets:
      - 127.0.0.1:9925

  - job_name: node_exporter
    static_configs:
    - targets:
      - 127.0.0.1:9100
    - targets:
      - uoc-cloud:9100
      labels:
        one_host_id: '16'

  - job_name: libvirt_exporter
    static_configs:
    - targets:
      - uoc-cloud:9926
      labels:
        one_host_id: '16'
```

Una vez realizados los cambios es necesario reiniciar y ponemos en marcha los servicios aplicando los cambios

Servicio Frontend

systemctl enable --now opennebula-prometheus.service

Servicios de Exportadores del host y forntend

systemctl enable --now opennebula-exporter.service opennebula-node-exporter.service

Reiniciamos el exportador del host KVM

systemctl enable --now opennebula-node-exporter.service opennebula-libvirt-exporter.service

```
root@uoc-cloud:/home/astoian#
root@uoc-cloud:/home/astoian# systemctl enable --now opennebula-prometheus.service
Created symlink /etc/systemd/system/multi-user.target.wants/opennebula-prometheus.service → /usr/lib/systemd/system/opennebula-prometheus.
service.
root@uoc-cloud:/home/astoian# systemctl enable --now opennebula-exporter.service opennebula-node-exporter.service
Created symlink /etc/systemd/system/multi-user.target.wants/opennebula-exporter.service → /usr/lib/systemd/system/opennebula-exporter.serv
ice.
Created symlink /etc/systemd/system/multi-user.target.wants/opennebula-node-exporter.service → /usr/lib/systemd/system/opennebula-node-exp
orter.service.
root@uoc-cloud:/home/astoian# systemctl enable --now opennebula-node-exporter.service opennebula-libvirt-exporter.service
Created symlink /etc/systemd/system/multi-user.target.wants/opennebula-libvirt-exporter.service → /usr/lib/systemd/system/opennebula-libvi
rt-exporter.service.
root@uoc-cloud:/home/astoian#
```

Una vez reiniciados comprobamos si los exporters están funcionando con el comando *> ss -tapn | grep 'LISTEN.*\(9925\|9100\|9090\)'*

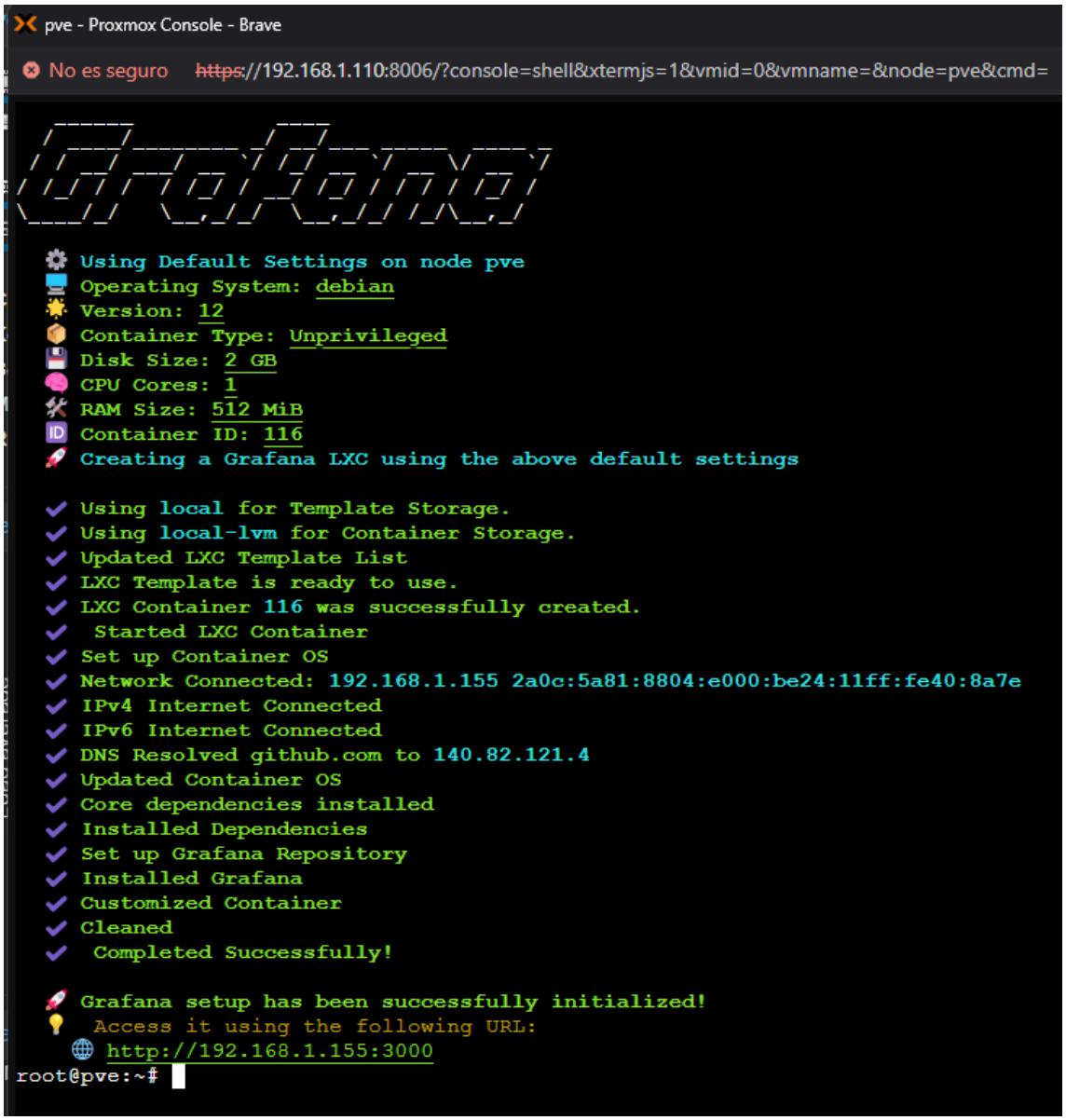
```
root@uoc-cloud:/home/astoian# ss -tapn | grep 'LISTEN.*\(9925\|9100\|9090\)'
LISTEN    0      100                0.0.0.0:9925               0.0.0.0:*     users:(("ruby",pid=1609322,fd=7))

LISTEN    0      4096                     *:9090                     *:*     users:(("prometheus",pid=1608236,fd=7))

LISTEN    0      4096                     *:9100                     *:*     users:(("node_exporter",pid=1609323,fd=3))

root@uoc-cloud:/home/astoian#
```

Podemos comprobar si se reciben métricas con el comando
> curl localhost:9926/metrics

```
oneadmin@uoc-cloud:/home/astoian$ curl localhost:9926/metrics
# TYPE opennebula_libvirt_requests_total counter
# HELP opennebula_libvirt_requests_total The total number of HTTP requests handled by the Rack application.
opennebula_libvirt_requests_total{code="200",method="get",path="/metrics"} 1.0
# TYPE opennebula_libvirt_request_duration_seconds histogram
# HELP opennebula_libvirt_request_duration_seconds The HTTP response duration of the Rack application.
opennebula_libvirt_request_duration_seconds_bucket{method="get",path="/metrics",le="0.005"} 1.0
opennebula_libvirt_request_duration_seconds_bucket{method="get",path="/metrics",le="0.01"} 1.0
opennebula_libvirt_request_duration_seconds_bucket{method="get",path="/metrics",le="0.025"} 1.0
opennebula_libvirt_request_duration_seconds_bucket{method="get",path="/metrics",le="0.05"} 1.0
opennebula_libvirt_request_duration_seconds_bucket{method="get",path="/metrics",le="0.1"} 1.0
opennebula_libvirt_request_duration_seconds_bucket{method="get",path="/metrics",le="0.25"} 1.0
opennebula_libvirt_request_duration_seconds_bucket{method="get",path="/metrics",le="0.5"} 1.0
opennebula_libvirt_request_duration_seconds_bucket{method="get",path="/metrics",le="1"} 1.0
opennebula_libvirt_request_duration_seconds_bucket{method="get",path="/metrics",le="2.5"} 1.0
```

Para mejorar la visualización y monitorización del entorno OpenNebula se ha realizado la instalación de Grafana utilizando un *helper script* desarrollado por la comunidad. Este script automatiza gran parte de los procesos de instalación y configuración inicial. De esta forma disponemos de nuestro servicio de visualización de métricas de forma fácil rápida y sencilla.

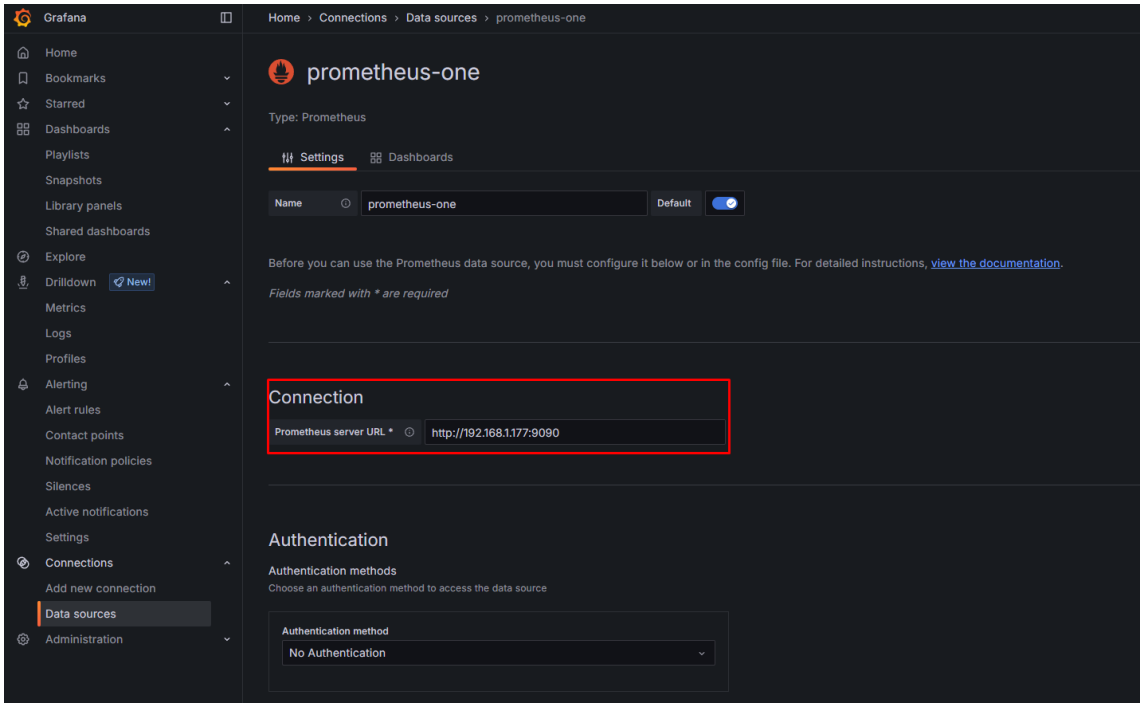
Se ha implementado Grafana en un contenedor LXC en otro servidor para mantener separados los servicios.



Una vez finalizado el proceso guiado de instalación de Grafana podemos acceder a la web de configuracion de Grafana.

http://192.168.1.155:3000

Accediendo a la web de Grafana podemos configurar el Data Source indicando la IP del servidor de OpenNebula donde tenemos configurado el servicio de Prometheus tal y como podemos visualizar en la imagen.

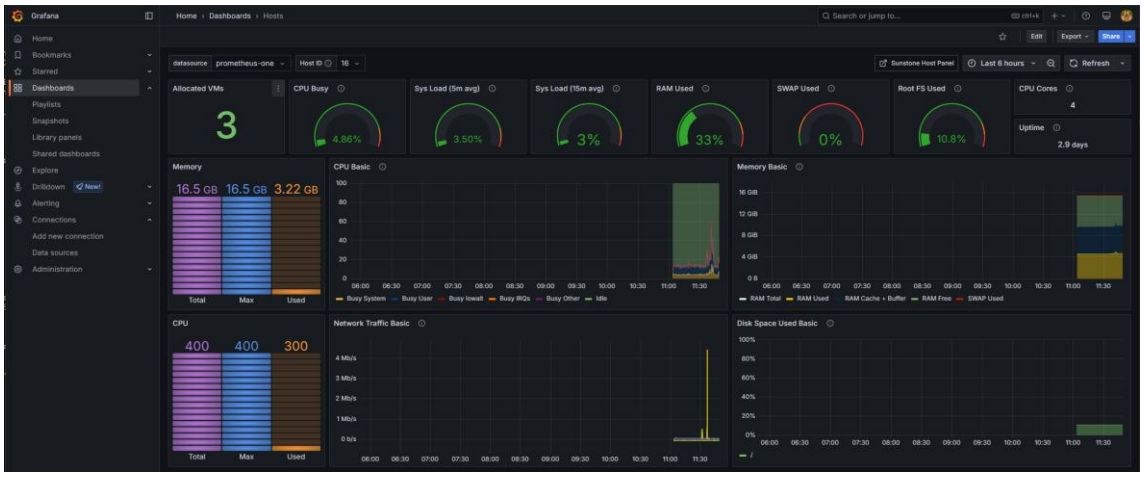


La configuración de los ***dashboards***(paneles) proporcionados por la comunidad de OpenNebula los podemos localizar en el siguiente directorio /usr/share/one/grafana/dashboards/

```
oneadmin@uoc-cloud:/home/astoian$ ls -lha /usr/share/one/grafana/dashboards/
total 132K
drwxr-xr-x 2 root root 4,0K abr 22 10:29 .
drwxr-xr-x 3 root root 4,0K abr 22 10:29 ..
-rw-r--r-- 1 root root  60K feb 21 01:10 hosts.json
-rw-r--r-- 1 root root  29K feb 21 01:10 opennebula.json
-rw-r--r-- 1 root root  32K feb 21 01:10 vms.json
oneadmin@uoc-cloud:/home/astoian$
```

Simplemente con importar estos dashboards podemos disponer de la información tanto de las VM, Host(KVM) y Servidor OpenNebula.

Finalmente podemos ver los paneles de métricas.



7. Pruebas y validación

7.1. Metodología de pruebas

En este apartado se describe las pruebas o comprobación realizadas para configura que el clúster de virtualización operado por OpenNebula ofrece un rendimiento aceptable. Importante mencionar que dicho clúster configurado con varias maquinas virtuales Ubuntu 24.04 y soporte Intel SGX funciona correctamente. Conociendo las limitaciones del hardware físico las pruebas han sido adaptadas a ello, debido a que si se excede con las pruebas el equipo se bloquea.

El alcande de las pruebas es validar el funcionamiento, la seguridad y el coste del rendimiento del entorno onpremise.

El entorno esta compuesto por un nodo físico con CPU Intel(R) Core(TM) i5-7500 CPU @ 3.40GHz, con 16 GB de RAM y un disco duro SSD 512GB. Un Hipervisor KVM/QEMU gestionado por OpenNebula 6.10.3 , además de disponer de una plantilla Ubuntu 24.04 1Gb de RAM y 10Gb de disco.

Ya definidas las características físicas del entorno se ha realizado un instanciado tres VMs idénticas, se ha realizado la batería de pruebas en las tres instancias y se ha comprobado los resultados de cada vm con sgx habilitado en las tres VMs.

7.2. Validación de seguridad y confidencialidad

Se ha realizado una comprobación de soporte SGX tanto del sistema huésped como del anfitrión.

```
root@sgx-vm-1:~# cpuid -1 -l 0x12 | grep SGX
   Software Guard Extensions (SGX) capability (0x12/0):
      SGX1 supported                         = true
      SGX2 supported                         = false
      SGX ENCLV E*VIRTCHILD, ESETCONTEXT     = false
      SGX ENCLS ETRACKC, ERDINFO, ELDBC, ELDUC = false
      SGX ENCLU EVERIFYREPORT2               = false
      SGX ENCLS EUPDATESVN                   = false
      SGX ENCLU EDECCSSA                     = false
root@sgx-vm-1:~#
```

```
astoian@uoc-cloud:~$ cpuid -1 -l 0x12 | grep SGX
   Software Guard Extensions (SGX) capability (0x12/0):
      SGX1 supported                         = true
      SGX2 supported                         = false
      SGX ENCLV E*VIRTCHILD, ESETCONTEXT     = false
      SGX ENCLS ETRACKC, ERDINFO, ELDBC, ELDUC = false
      SGX ENCLU EVERIFYREPORT2               = false
      SGX ENCLS EUPDATESVN                   = false
      SGX ENCLU EDECCSSA                     = false
astoian@uoc-cloud:~$
```

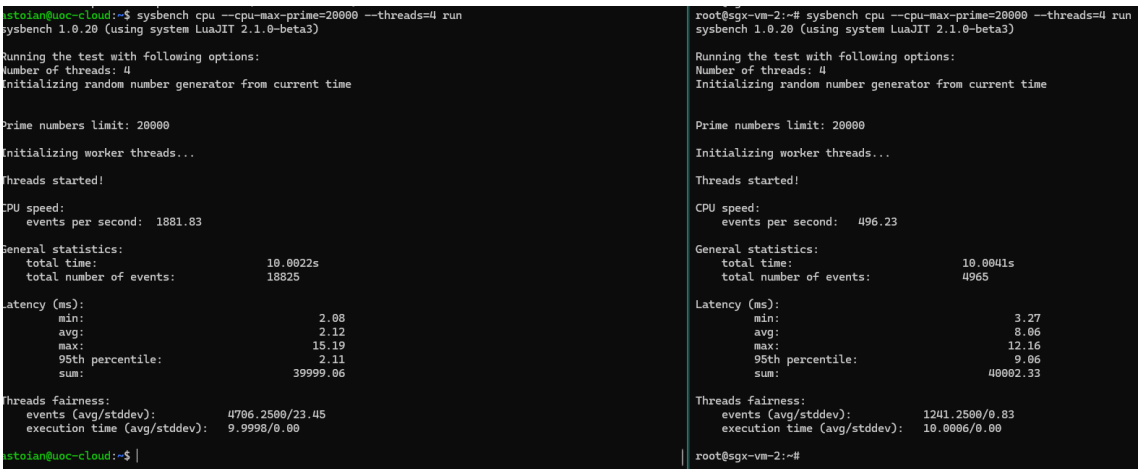
En las imágenes podemos ver que tanto la VM como el sistema soporta Intel SGXv1. Esto confirma que la tecnología Intel SGX v1 este habilitado en ambas maquinas. El resto de las capacidades aparecen como false lo cual es normal en entornos virtualizados donde no se han habilitado extensiones avanzadas de SGX. No obstante, esto no impide la ejecución de entornos simples.

7.3. Evaluación de rendimiento

Para evaluar el impacto de la computación confidencial y loe sistemas disponibles en nuestro entorno se ha realizado pruebas básicas en cuatro ámbitos clave, siendo una de ellos el procesamiento (CPU), rendimiento de memoria RAM , rendimiento del disco y rendimiento de la red Las herramientas utilizadas son **sysbench**, **dd** e **iperf3**, todas ampliamente conocidas y con resultados interpretables de forma sencilla.

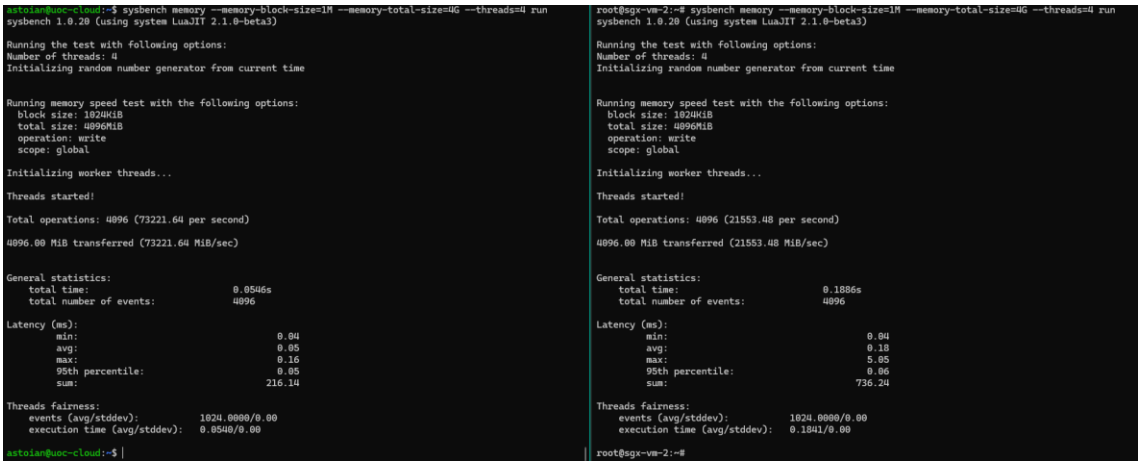
- Se usa sysbanch para calcular números primos usando los cuatro hilos de la CPU hasta un valor maximo, midiendo el tiempo requerido.

  > sysbench cpu --cpu-max-prime=20000 --threads=4 run



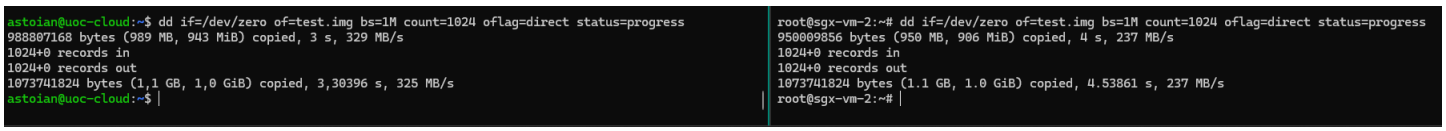
- Test de memoria RAM, se utiliza la herramienta sysbench para medir la tasa de transferencia de la memoria RAM. Esta prueba lee y escribe 4Gb de datos en bloques de 1Mb, simulando una carga de memoria intensiva multihilo.

>sysbench memory --memory-block-size=1M --memory-total-size=4G --threads=4 run



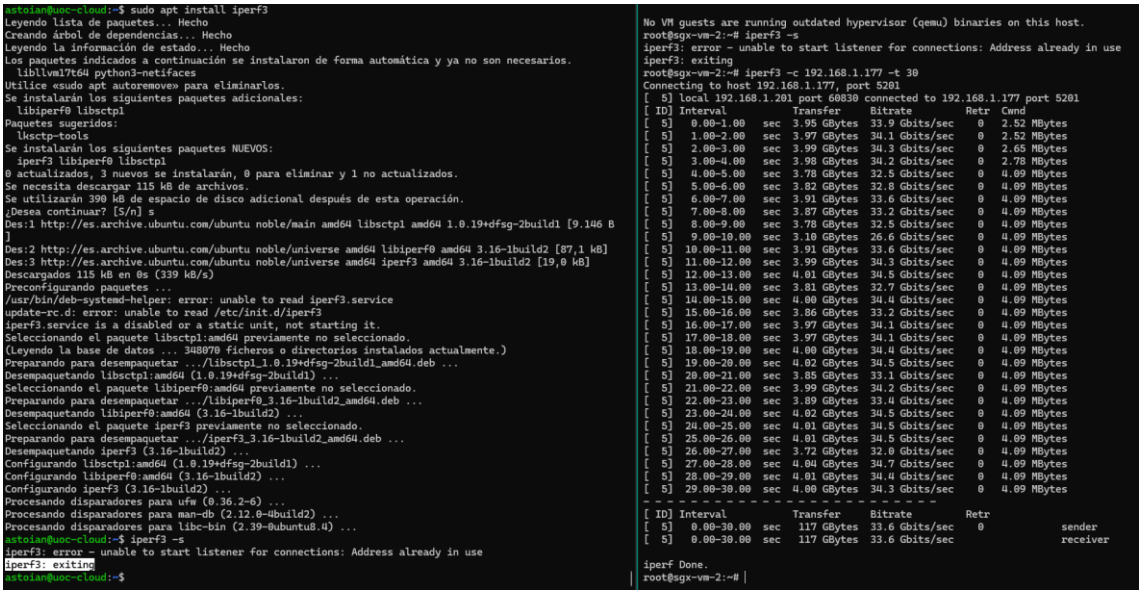
- Test del disco duro, para ello se hace uso de la herramienta dd que mide la velocidad de escritura y lectura secuencial en disco. Para ello se crea un archivo de 1Gb escribiendo datos sin pasar por la cache del sistema, para medir el rendimiento real del disco, en ambos casos.

>dd if=/dev/zero of=test.img bs=1M count=1024 oflag=direct status=progress



- Test de conectividad RED, se ha utilizado la herramienta iperf3 para comprobar el ancho de banda de la red entre la VM y el sistema central.

  Ejecución en el servidor: iperf3 -s
  Ejecución en la VM: iperf3 -c 192.168.1.177 -t 30



  Las pruebas se ejecutan se han ejecutado en varias ocasiones y se promedia el resultado para minimizar variaciones.

7.4. Resultados y análisis

Concluyo el análisis de los resultados de las pruebas anteriores mencionando que los resultados más relevantes obtenidas de las pruebas realizadas tanto en la VM como el sistema local con la tecnología Intel SGX garantizando la computación confidencial.

Respecto a la computación confidencial se confirma que está habilitada la opción soportada por el hardware del sistema tanto el host local y la máquina virtual, disponiendo de Intel SGX v1, habilitado.

Respecto al rendimiento de procesamiento en la VM sin SGX se ha registrado un rendimiento de 1881 eventos/segundos completando la prueba aproximadamente en 10 segundos. Sin embargo, en la VM con SGX habilitado el rendimiento es bajo llegando a valores de 496 eventos/segundo lo cual representa una reducción considerable atribuida a la sobrecarga añadida por la virtualización y compatibilidad del SGX.

Por otra parte, respecto a los resultados obtenidos del rendimiento de la memoria RAM, tenemos que destacar que en la VM sin SGX se ha logrado valores de 72321 Mb/s mientras que en la VM con SGX se ha obtenido un descenso a 21553 Mb/s. A pesar de la reducción el rendimiento sigue siendo adecuado para una carga estándar.

Las pruebas relevantes al rendimiento del disco duro, en la VM sin SGX se alcanzó velocidades de escritura lectura de 325Mb/s y en la VM con SGX se ha reducido esta velocidad aproximadamente un 27%, llegando a valores de 237Mb/s.

Para finalizar las pruebas de conectividad relacionado a la RED de las VMs. Indican que la tasa de transferencias llega a superar los 20Gb/s estando en la misma red reflejando un funcionamiento adecuado para estos sistemas. Sin haber diferencias entre ellas.

Concluyo indicando que las pruebas demuestran que el entorno basado en OpenNebula con SGX es operativo y funcional. Aunque se observa cierta penalización en rendimiento en CPU, memoria y disco, estos resultados se consideran aceptables dado el valor añadido en términos de seguridad y confidencialidad que aporta SGX. La red, por su parte, se mantiene sin impactos medibles.

Anexo

El **código utilizado en el TFG** se encuentra en este **botón**, se puede acceder para su revisión o análisis.

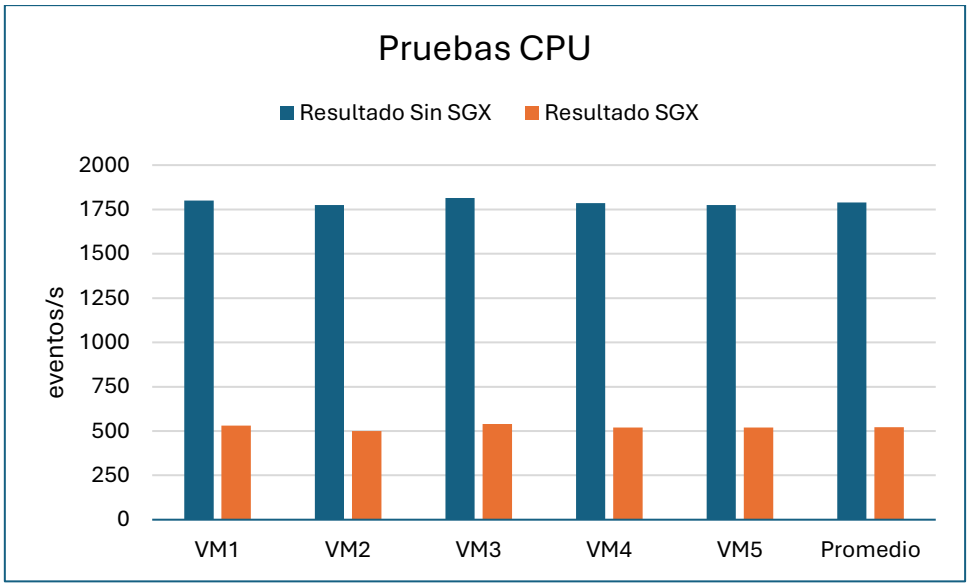
****Si no funciona Link Código.**

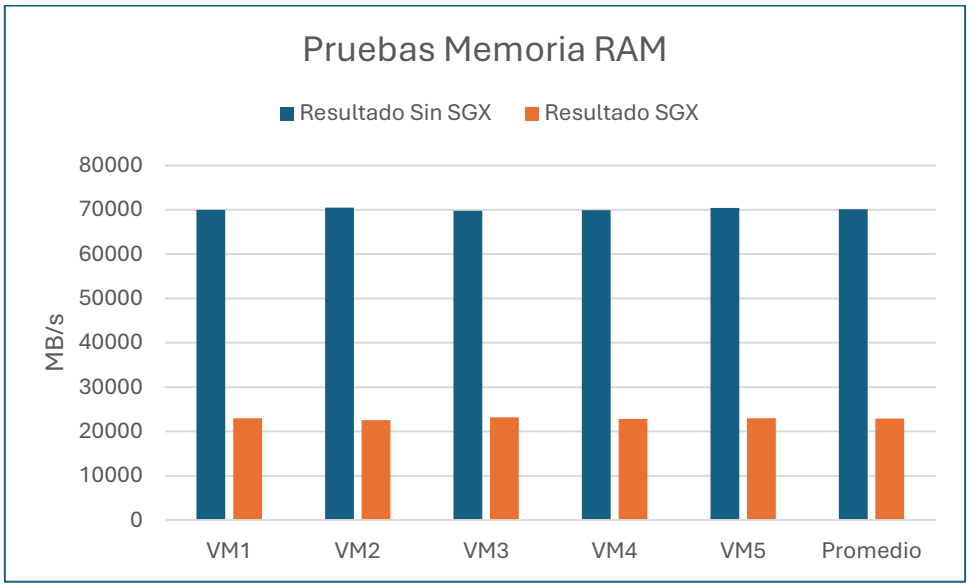
Se han realizado repetidas pruebas , con cinco máquinas virtuales desplegadas en el entorno. Los resultados obtenidos tras la ejecución de estas cinco maquinas virtuales se han expuesto en esta tabla. Dicha tabla resume los valores obtenidos para las métricas de rendimiento evaluadas en las VM con SGX comparándolas con VM sin SGX habilitado.

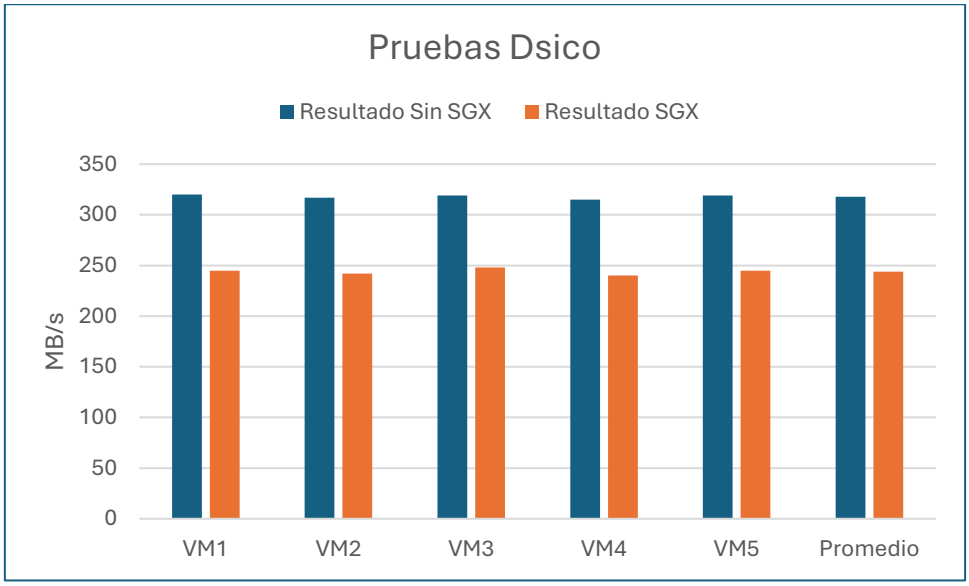
Prueba	Herramienta	Máquina	Resultado Sin SGX	Resultado SGX
CPU	sysbench	VM1	1800 proceso/seg	530 proceso /seg
CPU	sysbench	VM2	1775 procesos/seg	500 proceso /seg
CPU	sysbench	VM3	1815 procesos/seg	540 proceso /seg
CPU	sysbench	VM4	1785 procesos/seg	520 proceso /seg
CPU	sysbench	VM5	1775 procesos/seg	520 proceso /seg
CPU	sysbench	Promedio	1790 procesos/seg	522 proceso /seg
Memoria	sysbench	VM1	70000 Mb/s	23000 Mb/s
Memoria	sysbench	VM2	70500 Mb/s	22500 Mb/s
Memoria	sysbench	VM3	69800 Mb/s	23200 Mb/s
Memoria	sysbench	VM4	69900 Mb/s	22800 Mb/s
Memoria	sysbench	VM5	70400 Mb/s	22950 Mb/s
Memoria	sysbench	Promedio	70120 Mb/s	22890 Mb/s
Disco	dd	VM1	320 Mb/s	245 Mb/s
Disco	dd	VM2	317 Mb/s	242 Mb/s
Disco	dd	VM3	319 Mb/s	248 Mb/s
Disco	dd	VM4	315 Mb/s	240 Mb/s
Disco	dd	VM5	319 Mb/s	245 Mb/s
Disco	dd	Promedio	318 Mb/s	244 Mb/s
Red	iperf3	VM1	33,40 Gbps	33,20 Gbps
Red	iperf3	VM2	33,50 Gbps	33,30 Gbps
Red	iperf3	VM3	33,20 Gbps	33,40 Gbps
Red	iperf3	VM4	33,30 Gbps	33,30 Gbps
Red	iperf3	VM5	33,40 Gbps	33,20 Gbps
Red	iperf3	Promedio	33,40 Gbps	33,30 Gbps

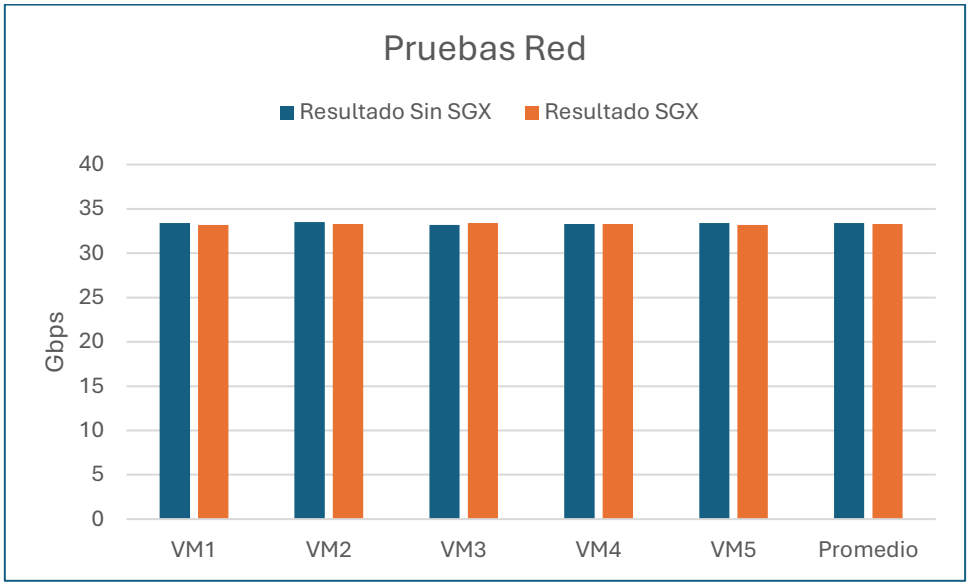
Se muestran los gráficos que permiten visualizar de forma mas clara las diferencias de rendimiento entre ambas configuraciones de cada una de las pruebas realizadas. Realizando un calculo promedio de todas las pruebas realizadas como se puede visualizar tanto en la tabla como en los gráficos.

Estos resultados facilitan un análisis del impacto que implica utilizar la tecnología Intel SGX en los recursos de cómputo, memoria, almacenamiento y red del entorno OneCloud.






Pruebas Dsico
Resultado Sin SGX
Resultado SGX
MB/s
350
300
250
200
150
100
50
0
VM1
VM2
VM3
VM4
VM5
Promedio


Pruebas Red
Resultado Sin SGX
Resultado SGX
Gbps
40
35
30
25
20
15
10
5
0
VM1
VM2
VM3
VM4
VM5
Promedio

Glosario

Administrador de la infraestructura (SysAdmin/DevOps): Profesional encargado de diseñar, desplegar y mantener la infraestructura; crea los ficheros de IaC, aplica automatización y vela por la disponibilidad y seguridad del sistema.

Administrador de seguridad (SecOps): Define políticas, supervisa accesos y realiza auditorías para garantizar la confidencialidad, integridad y trazabilidad de los datos.

Amazon Web Services (AWS): Plataforma de servicios en la nube que, entre otras funciones, ofrece AWS Nitro Enclaves para cargas de trabajo confidenciales.

Ansible: Herramienta de automatización y gestión de configuración basada en *playbooks* YAML; ejecuta tareas a través de SSH sin requerir agentes en los nodos.

ARM Template / Azure Resource Manager (ARM): Plantillas declarativas en JSON que describen y aprovisionan recursos en Microsoft Azure.

Bicep: Lenguaje de alto nivel que simplifica la sintaxis de las ARM Templates para Azure.

Chef: Solución de IaC "basada en *cookbooks*" escrita en Ruby para configurar y mantener servidores de forma continua.

CI/CD (Integración Continua / Despliegue Continuo): Práctica de incorporar cambios de código con pruebas automáticas y despliegue frecuente en entornos controlados.

Cloud híbrido: Modelo que combina recursos *on-premise* con servicios de la nube pública, gestionados de forma unificada.

Cloud privado: Infraestructura cloud dedicada a una organización, alojada en su propio CPD o en recursos exclusivos de un proveedor.

Cloud público: Servicios en la nube compartidos, ofrecidos por un proveedor y facturados bajo demanda.

Confidential Computing: Conjunto de tecnologías que protegen datos "en uso" mediante entornos de ejecución confidenciales (TEE), aislados incluso del SO o hipervisor.

Contenedor: Unidad ligera que empaqueta una aplicación y sus dependencias, garantizando ejecución consistente entre entornos.

Datastore: Repositorio de almacenamiento en OpenNebula que alberga imágenes base, discos y plantillas para VMs y contenedores.

DevSecOps: Enfoque que integra prácticas de seguridad a lo largo de todo el ciclo DevOps: planificación, desarrollo, despliegue y operación.

Grafana: Plataforma de visualización que crea paneles interactivos y alertas a partir de métricas recolectadas por Prometheus u otras fuentes.

Hypervisor: Capa de software o firmware que virtualiza hardware físico para ejecutar múltiples máquinas virtuales de forma aislada.

IaC (Infraestructura como Código): Práctica de definir y aprovisionar infraestructura mediante archivos de configuración versionables y repetibles, eliminando labores manuales.

Intel SGX (Software Guard Extensions): Extensiones de CPU de Intel que crean enclaves seguros cifrando la memoria asociada a una carga de trabajo confidencial.

Kubernetes: Sistema de código abierto para orquestar contenedores a gran escala, automatizando distribución, escalado y recuperación.

Máquina Virtual (VM): Instancia de sistema operativo que se ejecuta sobre un hypervisor con recursos virtualizados de CPU, memoria, red y almacenamiento.

OpenNebula: Plataforma *open-source* que orquesta máquinas virtuales y contenedores sobre hipervisores KVM, VMware o nubes públicas, facilitando el despliegue de nubes privadas e híbridas.

PaaS (Platform as a Service): Modelo de nube que suministra un entorno gestionado de desarrollo y despliegue de aplicaciones.

Prometheus: Sistema de monitorización y alertas que recolecta métricas en tiempo real mediante un modelo *pull* y lenguaje de consultas PromQL.

Puppet: Herramienta declarativa de IaC que describe el estado deseado de un sistema mediante *manifests*, operando en modo *master/agent* o *apply*.

SaltStack: Solución de orquestación y configuración rápida a gran escala, con arquitectura *master/minion* o sin servidor.

SaaS (Software as a Service): Aplicaciones completas entregadas a través de Internet, gestionadas íntegramente por el proveedor.

Terraform: Herramienta de IaC declarativa de HashiCorp capaz de gestionar múltiples proveedores; mantiene un estado de la infraestructura y aplica cambios de forma planificada (*plan/apply*).

VM confidencial: Máquina virtual cuya memoria está cifrada y aislada (por ejemplo, con Intel SGX o AMD SEV), impidiendo que el hipervisor acceda a datos en uso.

Zero Trust: Modelo de seguridad que parte de la premisa "nunca confíes, verifica siempre", aplicando validación estricta para cada acceso y comunicación.

Data-at-rest: Información digital que permanece guardada de forma permanente (discos, copias de seguridad, snapshots) sin ser transmitida ni procesada. Su protección se basa en cifrado de almacenamiento y controles de acceso físicos o lógicos.

Referencias bibliográficas

¿Qué es la infraestructura como código - Infrastructure as Code? (2024). Retrieved March 13, 2025, from Redhat.com website: https://www.redhat.com/es/topics/automation/what-is-infrastructure-as-code-iac

de, C. (2020, June 30). Infraestructura como código. Retrieved March 13, 2025, from Wikipedia.org website: https://es.wikipedia.org/wiki/Infraestructura_como_c%C3%B3digo

¿Qué es la infraestructura como código? (2025). Retrieved March 13, 2025, from F5, Inc. website: https://www.f5.com/es_es/glossary/infrastructure-as-code-iac#:~:text=La%20infraestructura%20como%20c%C3%B3digo%20(IaC,el%20desarrollo%20de%20CI%2FCD

¿Qué es la infraestructura como código? - Explicación de IaC - AWS. (2023). Retrieved March 13, 2025, from Amazon Web Services, Inc. website: https://aws.amazon.com/es/what-is/iac/

What is Terraform | Terraform | HashiCorp Developer. (2024). Retrieved March 13, 2025, from What is Terraform | Terraform | HashiCorp Developer website: https://developer.hashicorp.com/terraform/intro

Abdullahi, A. (2023, February 21). Las 8 principales herramientas de infraestructura como código (IaC) para 2025. Retrieved March 15, 2025, from Geekflare Spain website: https://geekflare.com/es/infrastructure-as-code-iac-tools/

de. (2024, March 13). Infrastructure as code (IaC). Retrieved March 15, 2025, from IONOS Digital Guide website: https://www.ionos.es/digitalguide/servidores/know-how/infrastructure-as-code/

Flores, F. (2021, May 31). Infraestructura como Código: Qué es y herramientas. Retrieved March 15, 2025, from OpenWebinars.net website: https://openwebinars.net/blog/infraestructura-como-codigo-que-es-y-herramientas/

IBM. (2024, December 26). What is Terraform? | IBM. Retrieved March 15, 2025, from Ibm.com website: https://www.ibm.com/es-es/topics/terraform

Fernandez, O. (2020, August 18). Terraform: Infraestructura como Código - Aprender BIG DATA. Retrieved March 15, 2025, from Aprender BIG DATA website: https://aprenderbigdata.com/terraform/

Raj, P. (2023, October 2). Terraform Architecture Overview — Structure and Workflow. Retrieved March 15, 2025, from Medium website: https://medium.com/@impradeep.techie/terraform-architecture-overview-structure-and-workflow-fdc60697941a

decide4AI. (2022, January 31). Terraform como herramienta para la automatización de infraestructuras. Retrieved March 15, 2025, from Decide website: https://decidesoluciones.es/terraform-automatizacion-de-infraestructuras/

admin. (2023, December). 6 razones por las que los desarrolladores deberían aprender Terraform - Distillery. Retrieved March 15, 2025, from Distillery website: https://distillery.com/es/blog/6-razones-por-las-que-los-desarrolladores-deberian-aprender-terraform/

Paniagua, A. P. (2015, November 4). Jugando con OpenNebula - Adictos al trabajo. Retrieved March 16, 2025, from Adictos al trabajo website: https://adictosaltrabajo.com/2015/11/04/tutorial-de-opennebula-jugando-con-opennebula/

Innovación Open Source. (2023, July 18). Retrieved March 16, 2025, from OpenNebula – Open Source Cloud & Edge Computing Platform website: https://opennebula.io/innovation/spain/

de, C. (2015, October 30). Opennebula. Retrieved March 16, 2025, from Wikipedia.org website: https://es.wikipedia.org/wiki/Opennebula

Primeros pasos en OpenNebula - TeideHPC. (2025). Retrieved March 16, 2025, from Iter.es website: https://doc.hpc.iter.es/2023.03/one/how_to_first_steps_one/

Use cacse

Aguado, V. (2023, November 30). 5 Advantages and Use Cases of OpenNebula - Tecsens -. Retrieved March 17, 2025, from Tecsens website: https://www.tecsens.com/en/5-advantages-and-use-cases-of-opennebula/

Pecho, M. P. (2021, February 6). OpenNebula Community Blog. Retrieved March 17, 2025, from OpenNebula – Open Source Cloud & Edge Computing Platform website: https://opennebula.io/blog/

Intel SGX (Software Guard Extensions)

Intel. (2025). *Intel® Software Guard Extensions*. [online] Available at: https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html

AMD SEV (Secure Encrypted Virtualization)

AMD. (2025). *AMD Secure Encrypted Virtualization (SEV)*. [online] Available at: https://www.amd.com/en/developer/sev.html

AWS Nitro Esclave

Amazon Web Services, Inc. (2021). *AWS Nitro System*. [online] Available at: https://aws.amazon.com/es/ec2/nitro/

Microsoft Azure Confidential Computing

ju-shim (2024). *Productos de computación confidencial de Azure*. [online] Microsoft.com. Available at: https://learn.microsoft.com/es-es/azure/confidential-computing/overview-azure-products

Confidential Computing

Revista Cloud. (2024). *Computación confidencial: Todo lo que necesitas saber sobre esta innovadora tecnología - Revista Cloud*. [online] Available at: https://revistacloud.com/computacion-confidencial-todo-lo-que-necesitas-saber-sobre-esta-innovadora-tecnologia/

to, C. (2023, March 16). privacy-enhancing computing technique. Retrieved March 17, 2025, from Wikipedia.org website: https://en.wikipedia.org/wiki/Confidential_computing

ju-shim. (2024, October 16). Introducción a la computación confidencial de Azure. Retrieved March 17, 2025, from Microsoft.com website: https://learn.microsoft.com/es-es/azure/confidential-computing/overview

Intel® Software Guard Extensions. Intel. Published 2025. Accessed March 17, 2025. https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html

Arantxa Herranz. Qué es Confidential Computing, la tecnología en la que IBM y AMD han anunciado un acuerdo de cooperación. Xataka.com. Published November 12, 2020. Accessed March 17, 2025. https://www.xataka.com/pro/que-confidential-computing-tecnologia-que-ibm-amd-han-anunciado-acuerdo-cooperacion

Security Guidance for Critical Areas of Focus in Cloud Computing v4.0. (2017). https://anskaffelser.no/sites/default/files/csa_security_guidance_v4.0.pdf

Amazon Web Services, Inc. (2025). *Reliance Steel and Aluminum Uses AWS Well-Architected Framework to Build Better in the Cloud (4:02)*. [online] Available at: https://aws.amazon.com/es/architecture/well-architected/?wa-lens-whitepapers.sort-by=item.additionalFields.sortDate&wa-lens-whitepapers.sort-order=desc&wa-guidance-whitepapers.sort-by=item.additionalFields.sortDate&wa-guidance-whitepapers.sort-order=desc

PageWriter-MSFT (2025). *Azure Well-Architected Framework - Microsoft Azure Well-Architected Framework*. [online] Microsoft.com. Available at: https://learn.microsoft.com/en-us/azure/well-architected/

Security Guidance for Critical Areas of Focus in Cloud Computing v4.0. (2017). https://anskaffelser.no/sites/default/files/csa_security_guidance_v4.0.pdf

PageWriter-MSFT (2025). *Azure Well-Architected Framework - Microsoft Azure Well-Architected Framework*. [online] Microsoft.com. Available at: https://learn.microsoft.com/en-us/azure/well-architected/

Amazon Web Services, Inc. (2021). *AWS Nitro System*. [online] Available at: https://aws.amazon.com/es/ec2/nitro/

Productos de computación confidencial de Azure. [online] Microsoft.com. Available at: https://learn.microsoft.com/es-es/azure/confidential-computing/overview-azure-products

Google Cloud. (2025). *Confidential Computing | Google Cloud*. [online] Available at: https://cloud.google.com/security/products/confidential-computing?hl=es_419

Ibm.com. (2025). *Confidential Computing | IBM*. [online] Available at: https://www.ibm.com/confidential-computing

Instalación:

Install Terraform | Terraform | HashiCorp Developer. (2024). *Install Terraform | Terraform | HashiCorp Developer*. [online] Available at: https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

Aleksic, M. (2024). *How to Install KVM on Ubuntu | phoenixNAP KB*. [online] Knowledge Base by phoenixNAP. Available at: https://phoenixnap.com/kb/ubuntu-install-kvm

Ubuntu Server. (2025). *LXD containers*. [online] Available at: https://documentation.ubuntu.com/server/how-to/containers/lxd-containers/#

Canonical (2025). *LXD*. [online] Snapcraft. Available at: https://snapcraft.io/lxd

Configuración:

Opennebula.io. (2024). *OpenNebula Repositories — OpenNebula 6.10.3 documentation*. [online] Available at: https://docs.opennebula.io/6.10/installation_and_configuration/frontend_installation/opennebula_repository_configuration.html

Opennebula.io. (2023). *Ansible — OpenNebula 6.4.7 documentation*. [online] Available at: https://docs.opennebula.io/6.4/integration_and_development/automation_tools_integration/ansible.html

Libvirt.org. (2020). *libvirt: Domain XML format*. [online] Available at: https://libvirt.org/formatdomain.html

OpenNebula Systems (2025). *Enable Intel SGX on Windows Guest*. [online] OpenNebula Community Forum. Available at: https://forum.opennebula.io/t/enable-intel-sgx-on-windows-guest/13751

Opennebula.io. (2024). *Virtual Machine Template — OpenNebula 6.10.3 documentation*. [online] Available at: https://docs.opennebula.io/6.10/management_and_operations/references/template.html#template-user-inputs

Terraform.io. (2025). *Terraform Registry*. [online] Available at: https://registry.terraform.io/providers/OpenNebula/opennebula/latest/docs/resources/template#context-1

Opennebula.io. (2024). *Installation and Configuration — OpenNebula 6.10.3 documentation*. [online] Available at: https://docs.opennebula.io/6.10/management_and_operations/monitor_alert/install.html#step-4-configure-prometheus-front-end

Opennebula.io. (2024). *Grafana Visualization — OpenNebula 6.10.3 documentation*. [online] Available at: https://docs.opennebula.io/6.10/management_and_operations/monitor_alert/grafana.html

Grafana Labs. (2025). *Download Grafana | Grafana Labs*. [online] Available at: https://grafana.com/grafana/download [Accessed 22 Apr. 2025].

Opennebula.io. (2024). *Images — OpenNebula 6.10.3 documentation*. [online] Available at: https://docs.opennebula.io/6.10/management_and_operations/storage_management/images.html

Opennebula.io. (2024). *Image Template — OpenNebula 6.10.3 documentation*. [online] Available at: https://docs.opennebula.io/6.10/management_and_operations/references/img_template.html?utm_source

SysDev Laboratories. (2023). *¿Qué es LUKS en Linux? Los fundamentos*. [online] Available at: https://www.sysdevlabs.com/es/articles/storage-technologies/what-is-luks-and-how-does-it-work/

Redhat.com. (2024). *Capítulo 8. Cifrado de dispositivos de bloque mediante LUKS | Red Hat Product Documentation*. [online] Available at: https://docs.redhat.com/es/documentation/red_hat_enterprise_linux/8/html/security_hardening/encrypting-block-devices-using-luks_security-hardening

Test command:

Canonical (2019). *Ubuntu Manpage: sysbench - A modular, cross-platform and multi-threaded benchmark tool.* [online] Ubuntu.com. Available at: https://manpages.ubuntu.com/manpages/trusty/man1/sysbench.1.html

does, W. (2016). *What does `dd if=/dev/zero of=/dev/sda` do*. [online] Unix & Linux Stack Exchange. Available at: https://unix.stackexchange.com/questions/275243/what-does-dd-if-dev-zero-of-dev-sda-do

GUEANT, V. (2015). *iPerf - iPerf3 and iPerf2 user documentation*. [online] Iperf.fr. Available at: https://iperf.fr/iperf-doc.php