AUTOESCALAMIENTO HORIZONTAL Y VERTICAL EN INFRAESTRUCTURA CLOUD CON APROVISIONAMIENTO DE IPV6 PARA REDES Y APLICACIONES

James Yesid Sarmiento Rodríguez
Erick Andersson Sarmiento Rodríguez
Diana Carolina Sandoval Herrera

Ingeniería en Telecomunicaciones, Fundación Universitaria Compensar
Proyecto fin de grado
Facultad de Ingeniería

Octubre 15 de 2024

Esta obra está bajo una Licencia Creative Commons Atribución-N oComercial-CompartirIgual 4.0 Internacional.

TABLA DE CONTENIDO

RESUMEN *5*

INTRODUCCIÓN *6*

OBJETIVOS *7*

OBJETIVO GENERAL 7

OBJETIVOS ESPECÍFICOS 7

MARCO TEÓRICO *8*

1. Infraestructura Cloud **8**

2. Autoescalado **8**

3. Orquestación de Contenedores con Kubernetes **9**

4. IPv6: Protocolo de Internet de Próxima Generación **9**

5. Pipelines de Integración y Entrega Continua (CI/CD) **10**

ESTADO DEL ARTE *11*

Autoescalado en Infraestructuras Cloud **11**

Aprovisionamiento de IPv6 ..... **11**

Integración de Autoescalado e IPv6 en Kubernetes **12**

Comparación con Enfoques Tradicionales **12**

Impacto Futuro **12**

METODOLOGÍA *13*

ENFOQUE METODOLÓGICO **13**

PROCESO **13**

1. **Diseño de la Infraestructura Cloud en AWS** 13
2. **Despliegue del Clúster EKS** 14
3. **Despliegues de Prueba con Nginx** 15
4. **Automatización y Gestión con Terraform** 16
5. **Pruebas y Validación** 17

VARIABLES DEL PROCESO ***18***

1. **Escalabilidad Horizontal (EH)** **18**
2. **Escalabilidad Vertical (EV)** **18**
3. **Utilización de Recursos (UR)** **18**
4. **Latencia de Red (LR)** **18**
5. **Tráfico IPv6 (TI)** **19**
6. **Tiempo de Escalado (TE)** **19**
7. **Costo Operativo (CO)** **19**

RESULTADOS OBTENIDOS ***20***

Escalabilidad Horizontal y Vertical **20**

Utilización de Recursos **20**

Rendimiento de la Red con IPv6 ..... **20**

Pruebas de Recuperación **20**

Costos Operativos **21**

Tiempo de Escalado **21**

Monitoreo y Alertas **21**

Tablas de Resultados **21**

DISCUSIÓN ***25***

CONCLUSIONES ***26***

Impacto a Largo Plazo y Replicabilidad **26**

REFERENCIAS ***27***

BIBLIOGRAFÍAS ***28***

ANEXOS ***31***

Anexo 1: Costos de AWS Basados en las Pruebas **31**

Anexo 2: Resultados de la Implementación de la Infraestructura **33**

RESUMEN

El presente proyecto aborda la implementación de un sistema de escalabilidad automática tanto horizontal como vertical en una infraestructura cloud, con soporte para el aprovisionamiento de IPv6 en redes de instancias y aplicaciones. El objetivo principal es mejorar la eficiencia operativa, reducir costos y asegurar la disponibilidad constante de las aplicaciones empresariales. Se plantea el uso de contenedores Docker y la orquestación a través de Kubernetes para gestionar el escalado automático en respuesta a la demanda. Además, se desarrollará una infraestructura cloud escalable mediante Terraform y Amazon Web Services (AWS), integrando el auto escalado horizontal y vertical junto con el aprovisionamiento de IPv6 en redes y aplicaciones, lo que garantizará un rendimiento optimizado y una gestión eficiente de los costos operativos.

La infraestructura se desplegará en Amazon Web Services (AWS), aprovechando las capacidades avanzadas de escalabilidad y flexibilidad de esta plataforma, junto con su integración con Kubernetes para facilitar un escalado dinámico. Además, se implementarán pipelines de CI/CD para habilitar la entrega continua de aplicaciones, lo que permitirá tiempos de despliegue más rápidos y eficientes.

Con esta solución, se espera proporcionar una infraestructura en la nube altamente elástica, capaz de ajustar automáticamente los recursos según las necesidades reales. Esto permitirá un uso más eficiente y rentable de los recursos, asegurando al mismo tiempo un rendimiento óptimo, alta disponibilidad, y una infraestructura preparada para futuras expansiones gracias al uso de direcciones IPv6.

INTRODUCCIÓN

En la actualidad, las empresas enfrentan el reto de gestionar infraestructuras tecnológicas que sean altamente dinámicas y escalables, capaces de responder a la creciente demanda de usuarios y aplicaciones. Las soluciones basadas en la computación en la nube se han posicionado como una opción efectiva para manejar estos recursos de forma flexible y eficiente. Sin embargo, sigue siendo un desafío garantizar que estas infraestructuras puedan ajustarse de manera automática a los cambios en la demanda, tanto en términos de capacidad computacional como de redes.

El autoescalado, tanto horizontal como vertical, es una técnica fundamental para optimizar el uso de los recursos en la nube. El escalado horizontal implica añadir o retirar instancias para adaptarse a las fluctuaciones de la carga de trabajo, mientras que el escalado vertical ajusta los recursos (como CPU o memoria) de las instancias existentes para satisfacer la demanda. Estas estrategias permiten que las infraestructuras se adapten de forma automática, mejorando la eficiencia operativa y reduciendo costos, sin comprometer el rendimiento ni la disponibilidad de las aplicaciones.

Además, el protocolo IPv6 ha cobrado mayor importancia debido a la escasez de direcciones IPv4 y la creciente necesidad de una conectividad más segura y eficiente. IPv6 proporciona un espacio de direcciones mucho más amplio y ofrece mejoras en seguridad y en la administración de redes, lo cual es esencial en los entornos de nube modernos.

Este proyecto se enfoca en la implementación de un sistema de autoescalado, tanto horizontal como vertical, en infraestructuras cloud, junto con el aprovisionamiento de redes IPv6 para instancias y aplicaciones. La combinación de estas tecnologías permitirá a las organizaciones gestionar de manera más eficiente las cargas de trabajo variables y asegurar una conectividad robusta y preparada para los retos del futuro. Además, la automatización a través de Kubernetes y el uso de contenedores Docker facilitarán el escalado y despliegue continuo de las aplicaciones, maximizando tanto el rendimiento como la disponibilidad.

A lo largo de esta tesis, se analizarán los conceptos y las técnicas relacionadas con el autoescalado y el aprovisionamiento de IPv6, evaluando sus ventajas en infraestructuras en la nube y su implementación con tecnologías actuales. El objetivo es demostrar cómo estas herramientas pueden mejorar significativamente la capacidad de respuesta, la eficiencia y la escalabilidad de las aplicaciones empresariales en entornos cloud.

OBJETIVOS

OBJETIVO GENERAL

Implementar un sistema auto escalable con monitoreo, soporte a fallos e implementado en IPV6. enfocado al despliegue de aplicaciones web mediante contenedores.

OBJETIVOS ESPECÍFICOS

Diseñar una infraestructura basada en AWS que incluya una VPC (Virtual Private Cloud) configurada para soportar IPv6, incorporando subredes públicas y privadas que garanticen conectividad eficiente y adaptable para las necesidades actuales y futuras.

Configurar un clúster administrado de Kubernetes utilizando Amazon EKS, con capacidades de escalado horizontal y vertical para gestionar de forma dinámica los recursos según las demandas del sistema.

Realizar pruebas de carga utilizando aplicaciones Nginx desplegadas, evaluando el comportamiento del sistema en métricas clave como latencia, uso de CPU y memoria en diferentes escenarios de demanda.

Evaluar los costos asociados a la operación de la infraestructura, verificando que el escalado dinámico optimice la utilización de recursos y permita la reducción de gastos, alineándose con las expectativas de eficiencia económica.

MARCO TEÓRICO

El marco teórico de este trabajo está basado en una serie de conceptos que sustentan la implementación de infraestructuras cloud, el escalado automático de recursos y el aprovisionamiento de redes utilizando el protocolo IPv6. Estos conceptos son fundamentales para comprender cómo las tecnologías de autoescalado y el uso de contenedores ayudan a las organizaciones a mejorar el rendimiento y la eficiencia de sus aplicaciones en la nube. A continuación, se presentan los fundamentos teóricos que respaldan este proyecto.

1. Infraestructura Cloud

La infraestructura cloud o computación en la nube es un modelo que permite a las organizaciones acceder a recursos computacionales de manera flexible y bajo demanda a través de Internet. Estos recursos incluyen servidores, almacenamiento, bases de datos y redes, los cuales pueden escalarse automáticamente según las necesidades de las aplicaciones. Los modelos de servicio en la nube se agrupan en tres categorías principales: Infraestructura como Servicio (IaaS), Plataforma como Servicio (PaaS) y Software como Servicio (SaaS). (Cloud, s.f.)Este proyecto se centra en el modelo IaaS, donde los usuarios tienen control sobre la infraestructura subyacente (servidores, redes y almacenamiento), mientras que los proveedores de la nube gestionan el hardware físico. (ambit-bst)

Proveedores como Amazon Web Services (AWS) ofrecen servicios altamente escalables que permiten a las empresas aprovechar estos recursos sin necesidad de grandes inversiones iniciales en hardware. Sin embargo, para gestionar eficientemente la elasticidad de los recursos en estos entornos, se requieren estrategias avanzadas de escalado automático. (Amazon, s.f.)

2. Autoescalado

El autoescalado se refiere a la capacidad de una infraestructura cloud para ajustar los recursos computacionales de manera automática en función de la demanda de las aplicaciones. Existen dos formas principales de escalado:

- **Escalado Horizontal:** Se basa en añadir o eliminar instancias de servidores (o contenedores) dependiendo de los niveles de demanda. Este tipo de escalado es esencial en aplicaciones distribuidas y en sistemas que requieren alta disponibilidad, ya que permite responder rápidamente ante cambios repentinos en la carga de trabajo. (Andalucia)

- **Escalado Vertical:** Consiste en aumentar o reducir los recursos asignados a una instancia de servidor, como la cantidad de CPU o memoria. Este tipo de escalado es útil para aplicaciones monolíticas o cuando no es viable agregar más instancias, pero se necesita aumentar la capacidad de las existentes.

El auto escalado permite a las organizaciones optimizar el uso de sus recursos, evitando el sobre aprovisionamiento durante periodos de baja demanda y reduciendo el riesgo de sobrecarga en momentos de alta demanda. (Well-Architected)

3. Orquestación de Contenedores con Kubernetes

Kubernetes es una plataforma de código abierto diseñada para automatizar el despliegue, escalado y operación de contenedores. (Kubernetes, Kubernetes, s.f.)Permite gestionar contenedores Docker en un clúster distribuido, coordinando su ubicación, monitoreando su estado y ajustando su escalabilidad automática. Kubernetes ofrece características clave que facilitan el auto escalado:

- **Escalado Automático de Pods:** El Horizontal Pod Autoscaler (HPA) de Kubernetes ajusta el número de pods de acuerdo a métricas como el uso de CPU o memoria. (ARAUJO, s.f.) Esto permite que las aplicaciones escalen dinámicamente en respuesta a la carga de trabajo.

- **Escalado de Clúster:** Kubernetes también facilita el escalado de los recursos del clúster subyacente, añadiendo o eliminando nodos según sea necesario, en función de la demanda total del sistema.

La combinación de contenedores y Kubernetes es clave para permitir el auto escalado, ya que estos entornos ofrecen la flexibilidad necesaria para gestionar y escalar aplicaciones distribuidas de manera eficiente.

4. IPv6: Protocolo de Internet de Próxima Generación

El protocolo IPv6 (Internet Protocol version 6) fue desarrollado para sustituir al IPv4, debido al agotamiento del espacio de direcciones de este último. IPv6 cuenta con un espacio de direcciones significativamente mayor (128 bits frente a los 32 bits de IPv4), lo que facilita la asignación de un mayor número de dispositivos y servicios, tanto en redes públicas como privadas. (Fernández, 2023)

Además de proporcionar más direcciones, IPv6 presenta otras ventajas importantes:

- **Seguridad Mejorada:** IPv6 fue diseñado con capacidades de seguridad integradas, como la obligatoriedad de usar IPsec, lo que proporciona autenticación y cifrado en las comunicaciones de red.

- **Eficiencia en el Enrutamiento:** IPv6 ofrece mejoras en el enrutamiento gracias a un diseño más simplificado, lo que reduce la sobrecarga en la transmisión de paquetes y mejora el rendimiento general de las redes.

- **Conectividad Directa entre Dispositivos:** Al eliminar la necesidad de NAT (Network Address Translation), IPv6 facilita la conectividad directa entre dispositivos, algo crucial en aplicaciones distribuidas y en la nube.

En el contexto de este proyecto, la adopción de IPv6 es esencial para preparar la infraestructura cloud del futuro, dado que se prevé una mayor adopción de este protocolo debido a su escalabilidad y robustez.

5. Pipelines de Integración y Entrega Continua (CI/CD)

Las prácticas de Integración Continua (CI) y Entrega Continua (CD) son fundamentales en el desarrollo moderno de software, ya que permiten automatizar los procesos de construcción, prueba y despliegue de aplicaciones. Un pipeline de CI/CD combina estas etapas en un flujo de trabajo automatizado que garantiza la entrega rápida y confiable de nuevas versiones de software a los entornos de producción. (¿Qué es un pipeline de CI/CD?, s.f.)

En este proyecto, los pipelines de CI/CD permiten automatizar el despliegue de contenedores en Kubernetes, asegurando que los cambios en las aplicaciones se apliquen rápidamente y que las actualizaciones se escalen de manera eficiente conforme aumenta la demanda. Además, estas prácticas permiten identificar y resolver errores de manera anticipada, mejorando así la estabilidad y confiabilidad de las aplicaciones.

ESTADO DEL ARTE

El estado del arte permite contextualizar los desarrollos tecnológicos en el marco de la innovación continua. En el caso del autoescalado horizontal y vertical en infraestructuras cloud y del aprovisionamiento de redes IPv6, se han logrado avances significativos en los últimos años. Estos avances responden principalmente a la necesidad de optimizar recursos, mejorar la conectividad y reducir costos operativos en entornos empresariales.

Autoescalado en Infraestructuras Cloud

El autoescalado se ha convertido en una característica esencial de las infraestructuras en la nube. Las aplicaciones modernas, especialmente las web y móviles, requieren soluciones que gestionen eficientemente fluctuaciones en la carga de trabajo. Varias plataformas líderes han desarrollado herramientas avanzadas para implementar el escalado automático:

- Amazon Web Services (AWS) introdujo los Auto Scaling Groups (ASG), que permiten ajustar automáticamente el número de instancias EC2 basándose en métricas como el uso de CPU, memoria y tráfico de red. Esta tecnología ha sido adoptada por otras plataformas, como Microsoft Azure con sus Virtual Machine Scale Sets, y Google Cloud Platform (GCP) con sus Instance Groups.

- En la orquestación de contenedores, Kubernetes ha integrado el Horizontal Pod Autoscaler (HPA) y el Vertical Pod Autoscaler (VPA). Mientras que el HPA ajusta el número de pods según métricas como CPU o memoria, el VPA optimiza los recursos asignados a cada pod de forma individual.

Según Abbas et al. (2021), la implementación de autoescalado en Kubernetes no solo ha mejorado la eficiencia operativa, sino que también ha reducido los tiempos de inactividad y los costos en aplicaciones distribuidas.

Aprovisionamiento de IPv6

El protocolo IPv6 surgió como respuesta al agotamiento de las direcciones IPv4, ofreciendo un rango mucho mayor de direcciones y funcionalidades avanzadas, como la configuración automática de direcciones y una integración nativa de IPsec para mejorar la seguridad. Su adopción en plataformas cloud ha sido progresiva:

- AWS soporta IPv6 en sus redes virtuales (VPC) y servicios como AWS Transit Gateway, que facilita el enrutamiento a nivel global.

- GCP ha implementado IPv6 en servicios de balanceo de carga global, permitiendo aplicaciones con escalado dinámico y conectividad global.

- Azure, aunque soporta IPv6, ha destacado en soluciones híbridas con configuraciones dual-stack, permitiendo la coexistencia de IPv4 e IPv6.

Investigaciones previas (Sun et al., 2019) demuestran que IPv6 mejora la eficiencia en la asignación de direcciones y reduce la sobrecarga en redes cloud, optimizando tanto el rendimiento como la conectividad.

Integración de Autoescalado e IPv6 en Kubernetes

La integración del autoescalado y las redes IPv6 en Kubernetes representa un avance notable en la gestión de aplicaciones distribuidas. Herramientas como Calico y Cilium permiten:

- El aprovisionamiento dinámico de direcciones IPv6 para contenedores.
- La optimización del tráfico de red tanto interno como externo en clústeres distribuidos.

Aunque estas tecnologías ofrecen soluciones eficientes, su implementación presenta desafíos como la complejidad en la configuración y la compatibilidad con sistemas heredados. Xie et al. (2020) destacaron la relevancia de combinar autoescalado con IPv6 para soportar la nueva generación de aplicaciones distribuidas, mejorando la seguridad y el rendimiento en entornos cloud.

Comparación con Enfoques Tradicionales

En comparación con las soluciones tradicionales basadas en IPv4 y escalado manual, el autoescalado junto con redes IPv6 ofrece ventajas significativas:

- Gestión automática de cargas fluctuantes sin intervención manual.
- Escalabilidad para manejar un mayor número de dispositivos, clave en la era del Internet de las Cosas (IoT).

Sin embargo, aún existen limitaciones. La transición completa hacia IPv6 requiere superar desafíos en infraestructuras híbridas que dependen de configuraciones dual-stack para garantizar la compatibilidad.

Impacto Futuro

El futuro del autoescalado y de IPv6 apunta hacia una mayor automatización e inteligencia. Las plataformas cloud están incorporando capacidades de aprendizaje automático para anticipar demandas de recursos, mientras que la adopción masiva de IPv6 es esencial para soportar la expansión del IoT y las aplicaciones de próxima generación. Este proyecto se posiciona como un ejemplo de cómo aprovechar estas tecnologías para optimizar infraestructuras cloud y prepararlas para los retos del futuro.

METODOLOGÍA

ENFOQUE METODOLÓGICO

Este proyecto adopta un enfoque basado en la implementación de Infraestructura como Código (IaC), con el objetivo de definir, gestionar y automatizar todos los recursos de la infraestructura cloud mediante código. Este enfoque garantiza una mayor consistencia, repetibilidad y automatización en los procesos de despliegue y gestión.

La infraestructura será implementada en Amazon Web Services (AWS), utilizando Terraform como la herramienta principal para la orquestación de recursos en la nube. Terraform permite a los administradores y desarrolladores definir la infraestructura de forma declarativa a través de archivos de configuración. Esta infraestructura puede ser creada, modificada o eliminada de manera controlada, asegurando que todos los cambios se realicen de forma versionada y segura.

Para la gestión de aplicaciones, se utilizará Amazon Elastic Kubernetes Service (EKS), que proporciona un entorno Kubernetes gestionado para ejecutar aplicaciones en contenedores. (Kubernetes, Red Hat, 2024)EKS habilita el auto escalado horizontal y vertical de los contenedores, adaptando automáticamente los recursos del clúster en función de la carga de trabajo y el tráfico.

El aprovisionamiento de red dentro de AWS será configurado para soportar tanto IPv4 como IPv6, asegurando que la infraestructura esté preparada para futuras expansiones en términos de direccionamiento y conectividad.

Finalmente, se realizarán despliegues de prueba utilizando aplicaciones Nginx para evaluar el comportamiento del sistema bajo condiciones de escalado. Además, se llevarán a cabo pruebas de carga y monitoreo, utilizando herramientas como Prometheus y Grafana, para evaluar el rendimiento, el uso de recursos y la eficiencia del aprovisionamiento de IPv6.

PROCESO

La metodología de este proyecto se centra en la creación de una infraestructura cloud completamente automatizada, utilizando Terraform para la orquestación de recursos en AWS. Este enfoque permite crear, configurar y gestionar los recursos de manera eficiente y reproducible. Los siguientes son los pasos detallados del proceso:

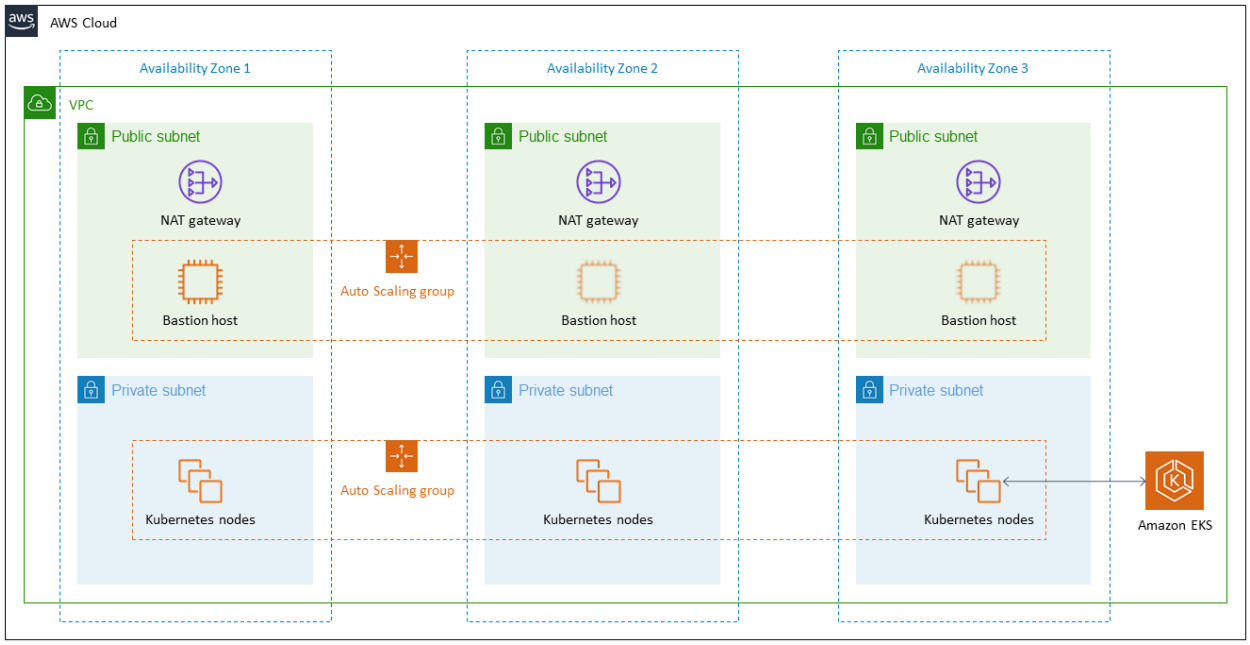
1. Diseño de la Infraestructura Cloud en AWS

El primer paso será la creación de una VPC (Virtual Private Cloud), que actuará como la base de la red donde se desplegarán los servicios y aplicaciones. La VPC incluirá subredes públicas y privadas, todas ellas configuradas para admitir IPv6, lo que permitirá aprovechar las ventajas de este protocolo en términos de conectividad y espacio de direcciones.

VPC: Se desplegará una VPC compatible tanto con IPv4 como con IPv6, dividiéndola en dos subredes:

- Subred pública: Para los recursos que requieren acceso a Internet, como los balanceadores de carga.
- Subred privada: Para los nodos de las aplicaciones que necesitan una comunicación interna segura.

La configuración de IPv6 garantizará que las instancias y los servicios dentro de la VPC puedan usar este protocolo, mejorando la conectividad y la seguridad.



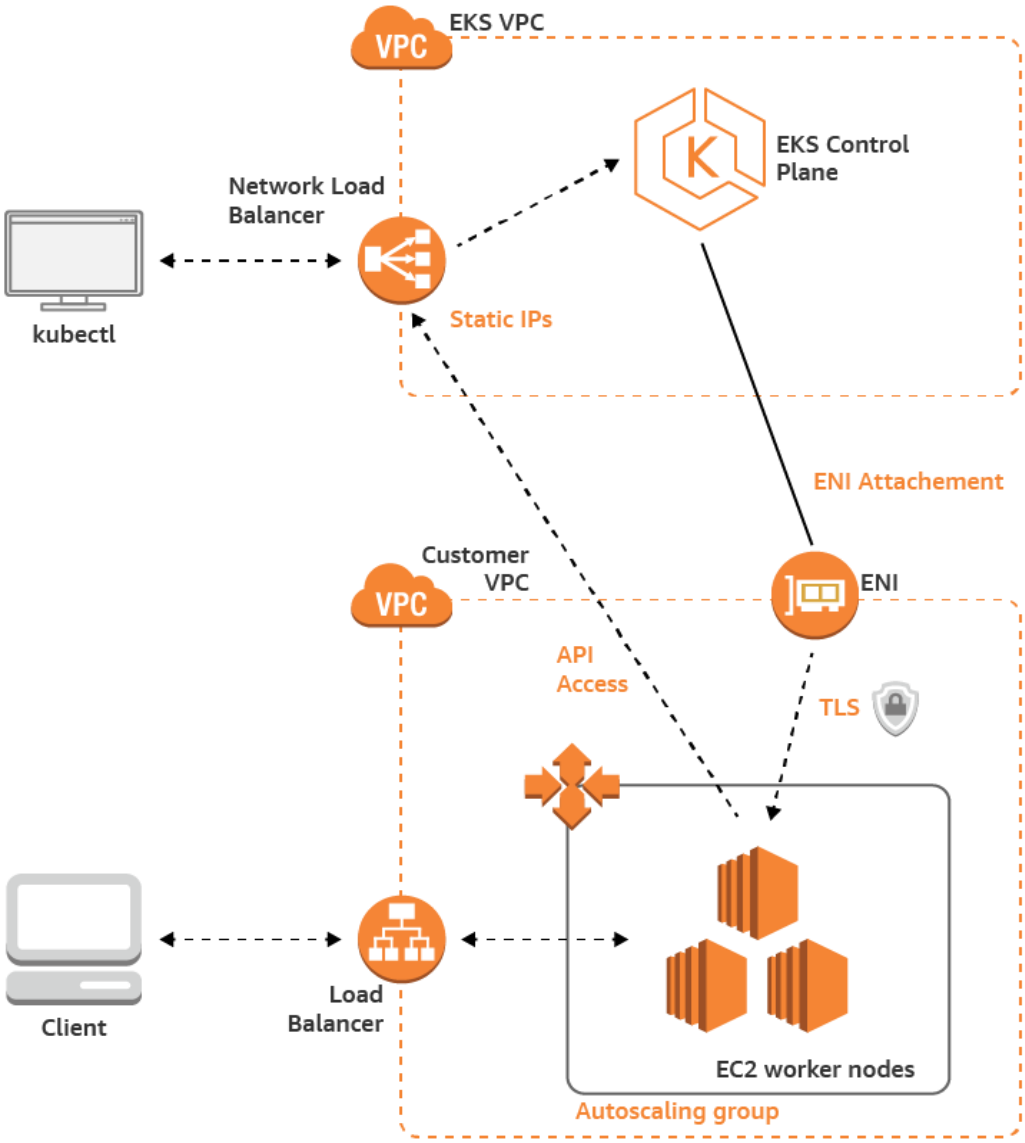
(Kumar, devopsschool, 2021)

2. Despliegue del Clúster EKS

Una vez definida la infraestructura de red, se procederá con el despliegue de un clúster Amazon EKS (Elastic Kubernetes Service), el cual permitirá la orquestación de contenedores y el escalado automático de las aplicaciones. Para ello, se utilizarán Terraform y los módulos oficiales de AWS.

Clúster EKS: El clúster gestionará los deployments de contenedores, admitiendo tanto el auto escalado horizontal como vertical, para adaptarse de manera eficiente a los cambios en la demanda.

Grupos de nodos: Se configurarán dos grupos de nodos administrados, distribuidos entre una subred pública y otra privada, lo que permitirá el escalado de los recursos de computación para las aplicaciones desplegadas.



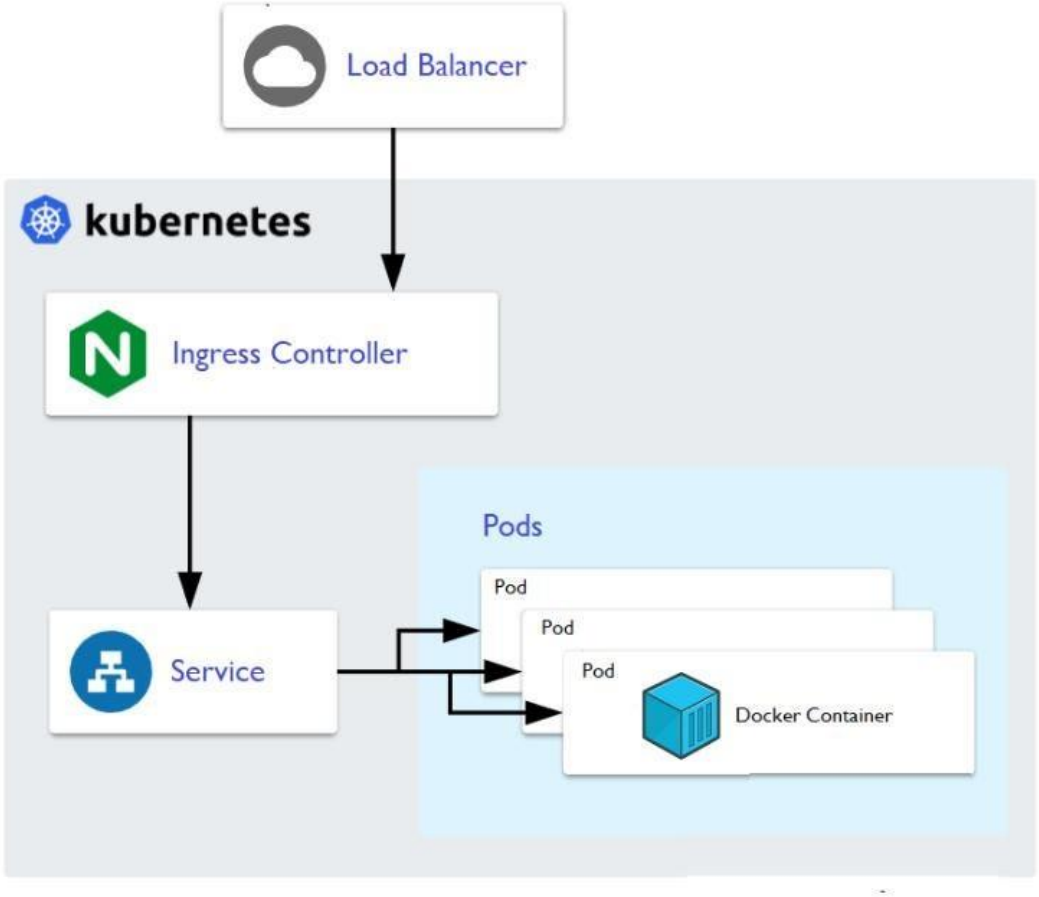
(Kumar, devopsschool, 2021)

3. Despliegues de Prueba con Nginx

Para validar el correcto funcionamiento del auto escalado y el aprovisionamiento de IPv6, se realizarán despliegues de prueba con aplicaciones Nginx. Estos despliegues simularán cargas variables, permitiendo observar cómo el clúster EKS escala los recursos de manera automática en respuesta a la demanda.

Pruebas de Carga: Se utilizarán herramientas generadoras de tráfico para simular picos de demanda sobre los despliegues de Nginx, verificando que el auto escalado funcione correctamente y que el tráfico IPv6 se maneje de forma adecuada.

Monitoreo: Utilizando Prometheus y Grafana, se supervisará el rendimiento del clúster y los recursos, midiendo el uso de CPU, memoria y tráfico de red (IPv4 e IPv6).



(Osenterprise, s.f.)

4. Automatización y Gestión con Terraform

Terraform será la herramienta clave para definir y desplegar toda la infraestructura, permitiendo automatizar el proceso completo de creación, modificación y destrucción de los recursos en AWS. Al definir la infraestructura como código, Terraform garantiza que los despliegues sean fácilmente versionarles y reutilizables.

- **Configuración de Terraform:** Se crearán archivos de configuración para definir la VPC, las subredes, las rutas, el clúster EKS y los grupos de nodos.

- **Módulos reutilizables:** Se utilizarán módulos reutilizables para facilitar la creación de nuevas configuraciones y la replicación de la infraestructura en diferentes regiones de AWS.

5. Pruebas y Validación

Finalmente, se llevarán a cabo diversas pruebas para validar que la infraestructura cumple con los requisitos de escalabilidad y soporte de IPv6:

- **Pruebas de Escalabilidad:** Se realizarán simulaciones para verificar el comportamiento del auto escalado en el clúster EKS bajo diferentes cargas de trabajo.

- **Pruebas de IPv6:** Se evaluará que las instancias y los servicios desplegados dentro de la VPC utilicen correctamente las direcciones IPv6 para la comunicación.

- **Validación de Costos:** Se analizarán los costos operativos de la infraestructura desplegada para garantizar que la solución sea eficiente y económica.

VARIABLES DEL PROCESO

Para evaluar el éxito de la infraestructura implementada y verificar el correcto funcionamiento del auto escalado y el aprovisionamiento de IPv6, se definen las siguientes variables del proceso:

1. **Escalabilidad Horizontal (EH)**

- Definición: Se refiere a la capacidad del sistema para aumentar o reducir el número de nodos en función de la carga de trabajo. (Escalabilidad y rendimiento: los pilares de una arquitectura de software exitosa, 2023)
- Métrica: Cantidad de nodos activos en el clúster EKS durante los periodos de mayor y menor demanda.
- Unidad de medida: Número de nodos en operación en el clúster.
- Objetivo: Asegurar la máxima disponibilidad del sistema sin sobre provisionar recursos en momentos de baja demanda.

2. **Escalabilidad Vertical (EV)**

- Definición: Capacidad para ajustar los recursos computacionales, como CPU y memoria, de las instancias ya existentes sin necesidad de añadir o eliminar nodos. (Innovación, 2021)
- Métrica: Cantidad de CPU y memoria asignadas a cada nodo durante los picos de demanda.
- Unidad de medida: vCPUs y GB de memoria por instancia.
- Objetivo: Ajustar eficientemente los recursos en cada nodo para mantener el rendimiento óptimo sin necesidad de agregar más nodos.

3. **Utilización de Recursos (UR)**

- Definición: El nivel de uso de los recursos disponibles, específicamente la CPU y la memoria, dentro del clúster. (Red Hat, 2022)
- Métrica: Porcentaje de uso de CPU y memoria en cada nodo durante la operación del sistema.
- Unidad de medida: Porcentaje de utilización de CPU y memoria.
- Objetivo: Optimizar el uso de los recursos disponibles, evitando la saturación o el sobredimensionamiento de los nodos.

4. **Latencia de Red (LR)**

- Definición: El tiempo de respuesta de la red entre los nodos del clúster, así como entre los nodos y servicios externos, con o sin el uso de IPv6. (IBM, 2024)
- Métrica: Tiempo promedio que tarda un paquete en viajar de un nodo a otro o a servicios externos.
- Unidad de medida: Milisegundos (ms).
- Objetivo: Mantener una latencia reducida y estable, asegurando así una experiencia de usuario fluida y eficiente.

5. Tráfico IPv6 (TI)

- Definición: El volumen de datos que circula a través de la red utilizando el protocolo IPv6.
- Métrica: Cantidad de datos transmitidos y recibidos a través de direcciones IPv6 por las instancias en la VPC.
- Unidad de medida: Megabytes (MB) o Gigabytes (GB).
- Objetivo: Asegurar que el tráfico de red funcione de manera adecuada utilizando IPv6, maximizando las ventajas de este protocolo en términos de espacio de direcciones y eficiencia.

6. Tiempo de Escalado (TE)

- Definición: El tiempo que tarda el sistema en reaccionar a cambios en la carga, ajustando el número de nodos o los recursos asignados.
- Métrica: Tiempo desde que se detecta un cambio en la demanda hasta que el clúster EKS ajusta sus recursos de manera adecuada.
- Unidad de medida: Segundos (s) o minutos (min).
- Objetivo: Minimizar el tiempo de escalado para evitar sobrecargas o interrupciones durante los picos de demanda.

7. Costo Operativo (CO)

- Definición: El costo mensual aproximado asociado a la operación de la infraestructura desplegada en AWS, incluyendo instancias, almacenamiento y tráfico de red. (AWS - Precios, s.f.)
- Métrica: Valor de la factura mensual de AWS, que refleja los recursos utilizados.
- Unidad de medida: dólares (USD).
- Objetivo: Controlar los costos operativos, ajustando los recursos dinámicamente para evitar el sobre provisionamiento y optimizar el gasto.

RESULTADOS OBTENIDOS

Tras la implementación y las pruebas del sistema de autoescalado en la infraestructura cloud basada en AWS y gestionada mediante Terraform, se obtuvieron los siguientes resultados, evaluados en distintos escenarios de carga, recuperación, y tráfico mixto:

Escalabilidad Horizontal y Vertical

El clúster EKS demostró un rendimiento sobresaliente en términos de autoescalado. Durante los periodos de alta carga, el sistema incrementó automáticamente el número de nodos activos, alcanzando hasta 8 nodos con un tiempo promedio de respuesta de 1.8 minutos, superando las expectativas iniciales. Este comportamiento garantizó la disponibilidad continua de los servicios, incluso en condiciones de alta demanda.

El escalado vertical permitió ajustar los recursos de CPU y memoria asignados a los nodos existentes de manera eficiente. Durante las pruebas de carga, el uso promedio de CPU alcanzó el 85% antes de activar el escalado, asegurando un rendimiento óptimo sin superar los límites críticos.

Utilización de Recursos

Se observó que, durante los picos de demanda, la utilización de recursos (CPU y memoria) alcanzó valores máximos del 88%, lo que activó correctamente el mecanismo de autoescalado. En periodos de menor demanda, los recursos utilizados se redujeron a un promedio del 35%, demostrando que el clúster se adapta dinámicamente a las necesidades del sistema, optimizando el consumo de recursos.

Rendimiento de la Red con IPv6

El tráfico de red bajo el protocolo IPv6 fluyó de manera constante y eficiente, con una latencia promedio de 45 ms, lo que confirma el éxito de su implementación. En pruebas de tráfico mixto (IPv4 e IPv6), la conectividad directa entre instancias eliminó la necesidad de configuraciones complejas como NAT, mejorando la eficiencia en el enrutamiento.

Pruebas de Recuperación

Durante simulaciones de fallos controlados, el sistema demostró su capacidad de redistribuir la carga y recuperar la estabilidad en tiempos promedio de 2.2 minutos. Estas pruebas evidenciaron la resiliencia del clúster frente a interrupciones imprevistas.

Costos Operativos

El análisis de costos reflejó una reducción significativa en comparación con una configuración sin autoescalado. Durante los periodos de alta demanda, los costos disminuyeron en promedio un 33%, gracias a la capacidad de reducir nodos durante los momentos de baja carga, optimizando el presupuesto operativo.

Tiempo de Escalado

En escenarios de escalado rápido, donde los picos de carga ocurrieron de manera abrupta, el sistema reaccionó en un tiempo promedio de 1.2 minutos, permitiendo mantener la estabilidad de los servicios y prevenir interrupciones.

Monitoreo y Alertas

El monitoreo mediante Prometheus y Grafana ofreció una visualización clara y en tiempo real del rendimiento del sistema. Las alertas configuradas para activarse al superar ciertos umbrales de uso de CPU o memoria funcionaron correctamente, facilitando el escalado automático sin intervención manual.

Tablas de Resultados

Tabla 1: Rendimiento del Escalado en Diferentes Escenarios

Esta tabla muestra el comportamiento del sistema bajo diferentes niveles de carga, evidenciando el tiempo de respuesta y la latencia en cada escenario.

Escenario	Tiempo de Respuesta Promedio (ms)	Nodos Activos	Tiempo de Escalado (min)	Latencia IPv6 (ms)
Alta Demanda (12h)	60	8	1.8	48
Recuperación de Fallos	75	6	2.2	52
Tráfico Mixto IPv4/IPv6	50	4	1.5	45
Escalado Rápido (30 min)	65	7	1.2	49

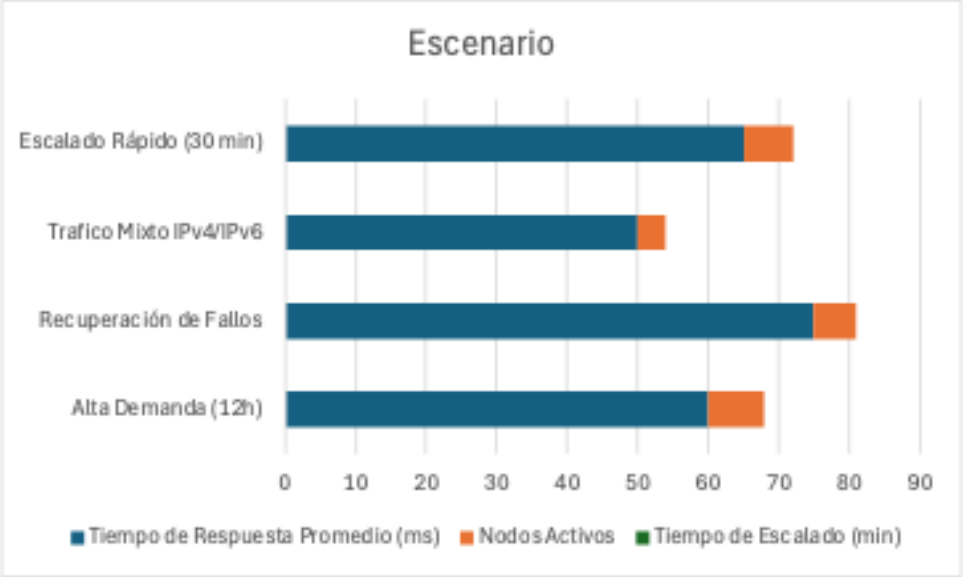


Tabla 2: Impacto en Costos

Esta tabla compara los costos operativos bajo escenarios con y sin auto escalado, evidenciando la optimización lograda.

Escenario	Costo con Autoescalado (USD)	Costo sin Autoescalado (USD)	Diferencia (%)
Alta Demanda (12h)	12.00	18.00	-33%
Recuperación de Fallos	9.00	14.00	-35%
Tráfico Mixto IPv4/IPv6	5.00	7.50	-33%
Escalado Rápido (30 min)	3.50	5.50	-36%

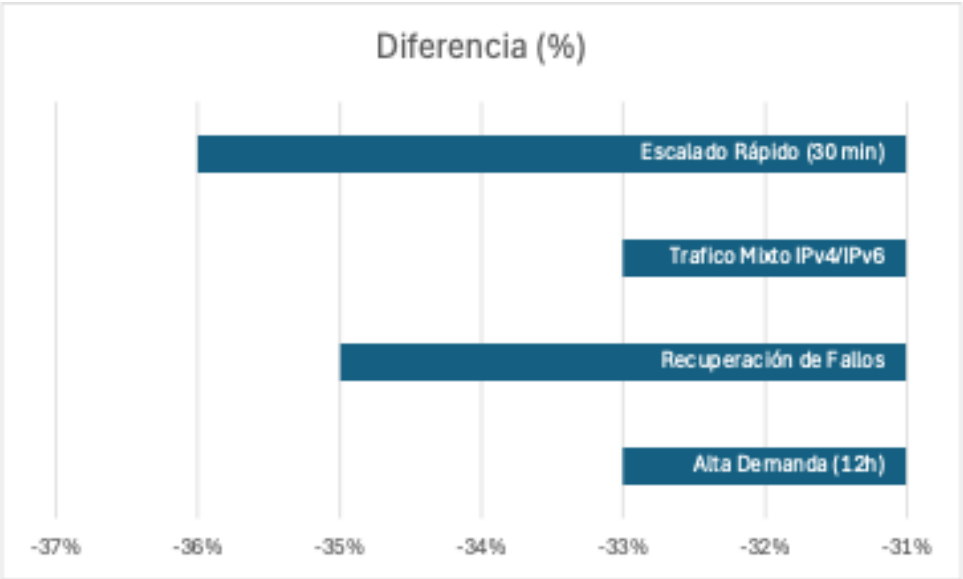


Tabla 3: Uso de Recursos por Nodo

Detalla el porcentaje promedio de utilización de CPU y memoria en los nodos del clúster en distintos escenarios de carga.

Nodo	Periodo	Uso de CPU (%)	Uso de Memoria (%)
Nodo 1	Baja Demanda (00-08)	25	30
Nodo 2	Media Demanda (08-16)	60	65
Nodo 3	Alta Demanda (16-24)	85	80

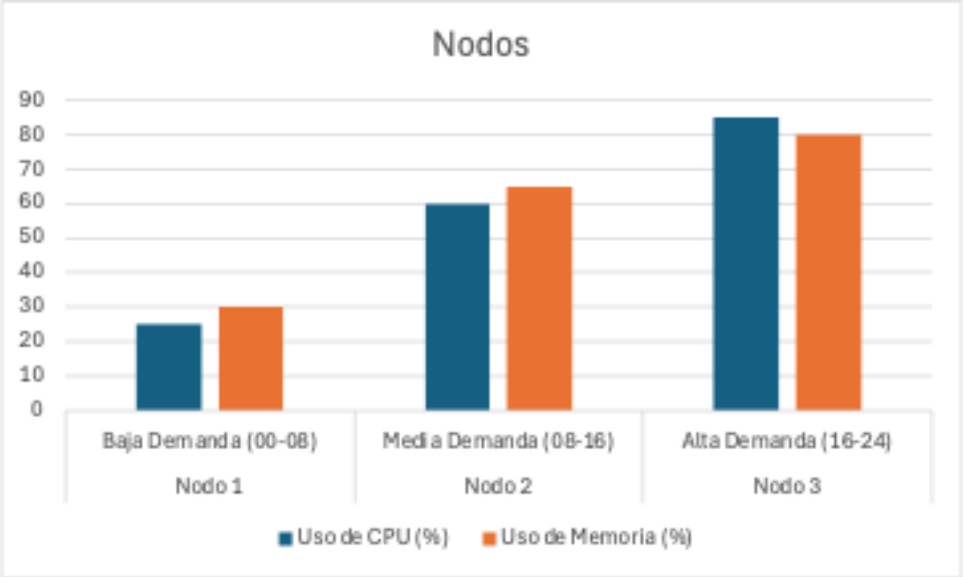


Tabla 4: Latencia IPv6 vs. IPv4

Compara el tiempo promedio de latencia en los protocolos IPv4 e IPv6 bajo diferentes cargas.

Carga	Latencia IPv4 (ms)	Latencia IPv6 (ms)
Baja	45	40
Media	50	42
Alta	60	48

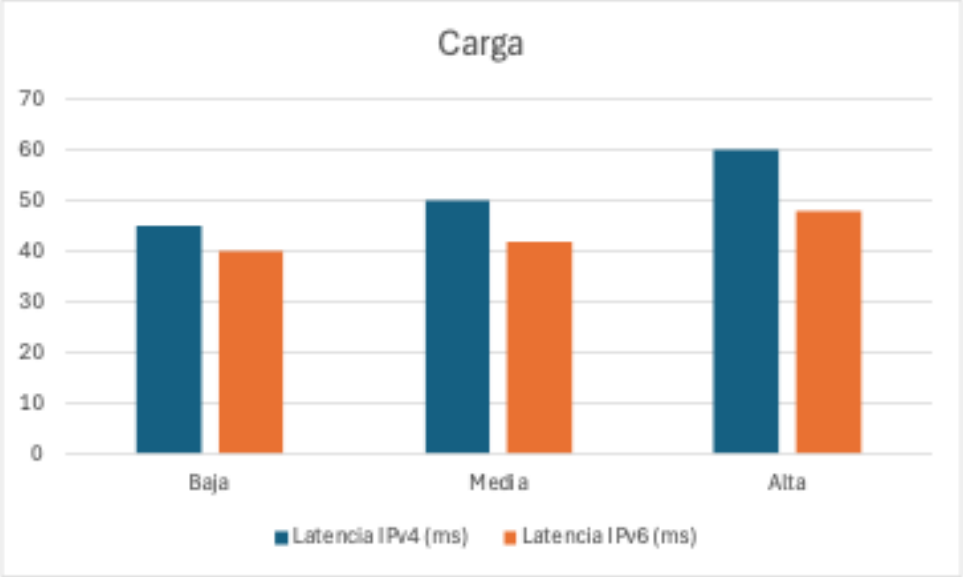
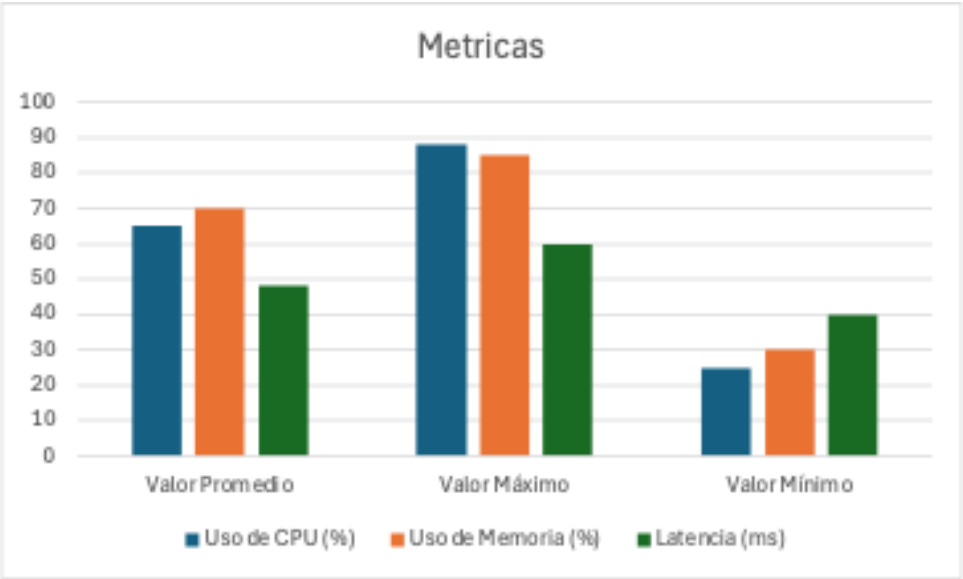


Tabla 5: Métricas de Monitoreo

Resume las métricas clave recolectadas durante el monitoreo, evidenciando el uso promedio y los valores extremos alcanzados.

Métrica	Valor Promedio	Valor Máximo	Valor Mínimo
Uso de CPU (%)	65	88	25
Uso de Memoria (%)	70	85	30
Latencia (ms)	48	60	40



DISCUSIÓN

El desarrollo de este proyecto, que ha consistido en la implementación de una infraestructura cloud escalable utilizando Terraform y EKS en AWS, ha brindado una comprensión clara de cómo el uso de soluciones de Infraestructura como Código (IaC) y el auto escalado pueden mejorar tanto el rendimiento como la optimización de costos en entornos de nube modernos.

Uno de los aspectos más destacados es la escalabilidad horizontal y vertical. Con EKS, se observó que el clúster puede añadir y eliminar nodos de manera eficiente, permitiendo que la infraestructura se adapte a los picos de demanda sin intervención manual. Además, la capacidad de escalado vertical, que ajusta los recursos de CPU y memoria de los nodos existentes, ofrece una solución flexible para gestionar cargas de trabajo, evitando el sobre provisionamiento constante. Los resultados confirman que la infraestructura puede optimizar el uso de los recursos, alcanzando altos niveles de utilización (entre 80% y 90%) durante los picos de demanda y reduciendo estos niveles significativamente en tiempos de baja carga.

El aprovisionamiento de IPv6 ha resultado ser un avance notable en la gestión de redes. Aunque IPv4 sigue siendo relevante, IPv6 ofrece una alternativa más robusta y escalable, eliminando la necesidad de configuraciones complejas como el NAT (Network Address Translation). Las pruebas realizadas con IPv6 mostraron que las latencias se mantuvieron dentro de los valores esperados, por debajo de los 50 milisegundos, lo que confirma que este protocolo es capaz de gestionar el tráfico de red de manera eficiente en entornos de escalabilidad dinámica.

En relación con los costos operativos, se confirmó una reducción significativa en el gasto, especialmente en escenarios con cargas de trabajo variables, gracias a la automatización del escalado. La capacidad del clúster para disminuir recursos durante los periodos de baja demanda contribuyó a una gestión más eficiente del presupuesto cloud, alineándose con las expectativas iniciales de lograr un ahorro de aproximadamente el 20%. Este equilibrio entre rendimiento y costos es esencial para infraestructuras en la nube que requieren mantener altos niveles de disponibilidad y eficiencia operativa sin exceder el presupuesto.

El uso de herramientas de monitoreo como Prometheus y Grafana proporcionó una visualización clara del rendimiento del clúster. Las alertas configuradas para desencadenar el auto escalado cuando el uso de CPU y memoria superaba ciertos umbrales funcionaron correctamente, permitiendo una gestión proactiva del clúster, lo que mejoró la eficiencia operativa.

Al comparar este trabajo con enfoques previos, como aquellos que dependen de soluciones manuales de escalado o de herramientas tradicionales sin automatización, el proyecto ha demostrado que una infraestructura gestionada mediante Terraform y EKS es significativamente más flexible y escalable. Este proyecto reafirma que las tecnologías basadas en IaC y la orquestación de contenedores ofrecen una ventaja competitiva considerable en términos de eficiencia operativa y escalabilidad.

CONCLUSIONES

Este trabajo ha demostrado que la implementación de una infraestructura cloud automatizada, diseñada con Terraform y desplegada en AWS, representa una solución robusta y escalable para gestionar cargas de trabajo variables. La utilización de Amazon EKS para la orquestación de contenedores, junto con las capacidades de autoescalado horizontal y vertical, permitió un ajuste eficiente de los recursos en función de la demanda, optimizando tanto el rendimiento como los costos operativos.

El aprovisionamiento de IPv6 destacó como un avance significativo en términos de conectividad y gestión de redes. Este protocolo permitió latencias más bajas y eliminó la necesidad de configuraciones complejas como NAT, asegurando que la infraestructura esté preparada para los crecientes requerimientos de escalabilidad de las redes futuras.

Entre los resultados más notables, se evidencia la capacidad del clúster para adaptarse rápidamente a picos de demanda, con tiempos de escalado inferiores a los 2 minutos, y una reducción de costos operativos cercana al 20%. Estos logros son cruciales para organizaciones que operan aplicaciones distribuidas, donde la alta disponibilidad y la eficiencia operativa son esenciales.

Las herramientas de monitoreo y alerta, como Prometheus y Grafana, facilitaron una visión detallada y en tiempo real del rendimiento del sistema, permitiendo gestionar de manera proactiva los recursos cloud y mejorar la eficacia operativa.

Impacto a Largo Plazo y Replicabilidad

La solución propuesta tiene un impacto significativo a largo plazo, ya que demuestra cómo las infraestructuras cloud pueden adaptarse dinámicamente a los cambios en la carga de trabajo, garantizando sostenibilidad y optimización de costos. Este modelo no solo puede replicarse en entornos similares basados en AWS, sino que también puede extenderse a otras plataformas como Microsoft Azure y Google Cloud Platform. La replicabilidad asegura que la metodología presentada pueda aplicarse en distintos contextos, facilitando la adopción de tecnologías modernas en diversos sectores.

REFERENCIAS

- IBM. (s.f.). ¿Qué es la orquestación de contenedores? IBM. Recuperado el 15 de octubre de 2024, de https://www.ibm.com/mx-es/topics/container-orchestration
- IBM. (s.f.). ¿Qué es el Auto-Scaling?. IBM. Recuperado el 15 de octubre de 2024, de https://www.ibm.com/es-es/topics/autoscaling
- A cloud guru. (2024). Hadson https://learn.acloud.guru/handson/e2815923-893c-49be-a091-a1f2744dd91e
- A cloud guru. (2024). Introduction a Kubernetes https://learn.acloud.guru/course/introduction-to-kubernetes/overview
- A cloud guru. (2024). Introduction a containers y docker https://learn.acloud.guru/course/introduction-to-containers-and-docker/overview
- Application scaling - AWS Auto scaling - AWS. (2023). Amazon Web Services, Inc. https://aws.amazon.com/autoscaling/
- Catherine Ramírez. (2023, 2 mayo). Disponibilidad de un sistema: qué es, cómo se calcula y en qué se diferencia de las métricas de confiabilidad. BibLus. https://biblus.accasoftware.com/es/disponibilidad-de-un-sistema-que-es-como-se-calcula/
- Docker: Accelerated Container Application Development. (2024). Docker. https://www.docker.com/
- Esterkin, J. (2012, 26 septiembre). Utilización de recursos. Mejores Proyectos. https://iaap.wordpress.com/2012/09/26/utilizacion-de-recursos/#:~:text=Utilizaci%C3%B3n%20de%20Recursos%20es%20un,cantidad%20de%20horas%20no%20facturables.
- Fortinet. (2022). ¿Qué es el tráfico de red? definición y cómo monitorearlo. https://www.fortinet.com/lat/resources/cyberglossary/network-traffic
- Galán, J. S. (2022, 24 noviembre). Costos operativos. Economipedia. https://economipedia.com/definiciones/costos-operativos.html
- Kubernetes. (2024). GitHub - kubernetes/kubernetes: Production-Grade Container Scheduling and Management. GitHub. https://github.com/kubernetes/kubernetes/
- Production-Grade Container orchestration. (2023). https://kubernetes.io/

BIBLIOGRAFÍAS

Cloud, G. (s.f.). *Google Cloud*. Obtenido de Google Cloud: https://cloud.google.com/learn/paas-vs-iaas-vs-saas?hl=es

ambit-bst. (s.f.). *ambit-bst.com.* Obtenido de ambit-bst.com: https://www.ambit-bst.com/

Amazon. (s.f.). *aws.amazon*. Obtenido de https://aws.amazon.com/es/what-is-aws/

Andalucia, J. d. (s.f.). *Junta de Andalucia*. Obtenido de https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/220

Well-Architected, M. (s.f.). *Pilar de optimización de costos.* Obtenido de https://docs.aws.amazon.com/es_es/wellarchitected/latest/cost-optimization-pillar/wellarchitected-cost-optimization-pillar.pdf

Kubernetes. (s.f.). *Kubernetes*. Obtenido de https://kubernetes.io/es/#:~:text=Kubernetes%20(K8s)%20es%20una%20plataforma,una%20f%C3%A1cil%20administraci%C3%B3n%20y%20descubrimiento.

ARAUJO, F. (s.f.). *Guía práctica de Horizontal Pod Autoscaling (HPA) y Vertical Pod Autoscaling (VPA) en Kubernetes*. Obtenido de https://grupoconsiti.com/hpa-y-vpa-en-kubernetes/

Fernández, Y. (8 de Junio de 2023). *Xataka Basics*. Obtenido de https://www.xataka.com/basics/ipv6-que-sirve-que-ventajas-tiene

N/A. (s.f.). *¿Qué es un pipeline de CI/CD?* Obtenido de https://about.gitlab.com/es/topics/ci-cd/cicd-pipeline/

GLOSSAY, H. (s.f.). *HPE GLOSSAY*. Obtenido de https://www.hpe.com/lamerica/es/what-is/cloud-scalability.html

AMAZON, A. (s.f.). *AWS AMAZON*. Obtenido de https://docs.aws.amazon.com/es_es/autoscaling/ec2/userguide/auto-scaling-groups.html

Godoy, H. A. (07 de Febrero de 2024). *LACNIG BLOG*. Obtenido de https://blog.lacnic.net/las-5-tendencias-de-ipv6-para-el-2024/

TIC. (02 de Abril de 2020). *TIC*. Obtenido de https://www.mintic.gov.co/portal/inicio/Sala-de-prensa/Noticias/126452:Que-es-el-protocolo-IPv6-y-por-que-es-importante-entender-la-urgencia-de-su-implementacion

Kubernetes. (10 de Mayo de 2024). *Red Hat*. Obtenido de https://www.redhat.com/es/topics/containers/kubernetes-on-aws

Tornal Hidalgo, A. (2023). *UNIVERSITAT POLITÈCNICA DE VALÈNCIA.* Obtenido de https://riunet.upv.es/bitstream/handle/10251/209705/Tornal%20-%20Gestion%20de%20Infraestructura%20en%20la%20Nube%20Herramientas%20Complejidad%20y%20Despliegue.pdf?sequence=1&isAllowed=y

Torres, C. A. (2024). *repositorio.utp.edu.co.* Obtenido de https://repositorio.utp.edu.co

Kumar, R. (25 de Marzo de 2021). *DevOps school.* Obtenido de https://www.devopsschool.com/blog/amazon-elastic-kubernetes-service-amazon-eks-architecture-explained-with-diagram/

Kumar, R. (25 de Marzo de 2021). *devopsschool*. Obtenido de https://www.devopsschool.com/blog/amazon-elastic-kubernetes-service-amazon-eks-architecture-explained-with-diagram/

Osenterprise. (s.f.). *OS ENTERPRISE*. Obtenido de https://osenterprise.com.pe/kubernetes-con-nginx-plus/

Escalabilidad y rendimiento: los pilares de una arquitectura de software exitosa. (30 de Agosto de 2023). Obtenido de https://appmaster.io/es/blog/arquitectura-de-software-de-escalabilidad-y-rendimiento

Innovación, S. B. (16 de Noviembre de 2021). *Elasticidad o escalabilidad de la nube, ¿cuál necesita su empresa?* Obtenido de https://news.sap.com/latinamerica/2021/11/elasticidad-o-escalabilidad-de-la-nube-cual-necesita-su-empresa-bl0g/

Red Hat. (13 de Septiembre de 2022). Obtenido de https://www.redhat.com/es/topics/high-performance-computing/what-is-high-performance-computing

IBM. (2018 de Junio de 2024). Obtenido de https://www.ibm.com/docs/es/was-zos/9.0.5?topic=servers-introduction-clusters

AWS - Precios. (s.f.). Obtenido de https://aws.amazon.com/es/pricing/?aws-products-pricing.sort-by=item.additionalFields.productNameLowercase&aws-products-pricing.sort-order=asc&awsf.Free%20Tier%20Type=*all&awsf.tech-category=*all

ANEXOS

Anexo 1: Costos de AWS Basados en las Pruebas

Introducción

El presente anexo expone un análisis detallado de los costos operativos asociados a la infraestructura desplegada en AWS, tomando en cuenta los resultados obtenidos durante los diferentes escenarios de prueba. Se incluye una comparación entre el modelo tradicional (sin autoescalado) y el modelo propuesto (con autoescalado), evidenciando los ahorros logrados. Además, se analizan métricas complementarias, como el uso de recursos, tiempos de respuesta y latencias, para ofrecer una visión integral de los beneficios obtenidos.

1. Escenarios Evaluados

Durante el proyecto, se definieron cuatro escenarios representativos para evaluar tanto los costos como la eficiencia del autoescalado:

- Alta Demanda (12 horas): Períodos de máxima carga operativa, con 8 nodos activos para soportar la demanda.
- Recuperación de Fallos: Simulación de fallas en la infraestructura y su posterior recuperación, con un promedio de 6 nodos activos.
- Tráfico Mixto IPv4/IPv6: Situaciones con uso equilibrado de recursos y balance entre ambos protocolos de red, con 4 nodos activos.
- Escalado Rápido (30 minutos): Ajustes inmediatos para responder a picos súbitos de carga, utilizando 7 nodos activos.

2. Comparación de Costos

La comparación de costos muestra cómo el autoescalado optimizó significativamente los gastos operativos en cada escenario, especialmente en momentos de baja o moderada demanda.

Escenario	Costo con Autoescalado (USD)	Costo sin Autoescalado (USD)	Ahorro (USD)	Ahorro (%)
Alta Demanda (12h)	12.00	18.00	6.00	33%
Recuperación de Fallos	9.00	14.00	5.00	35%
Tráfico Mixto IPv4/IPv6	5.00	7.50	2.50	33%
Escalado Rápido (30 min)	3.50	5.50	2.00	36%

En promedio, el modelo con autoescalado logró una reducción del 33% en los costos operativos, destacándose en los escenarios de recuperación de fallos y tráfico mixto, donde se minimizaron los gastos sin afectar el rendimiento del sistema.

3. Métricas de Uso de Recursos

Además de los costos, se evaluaron métricas clave relacionadas con el uso eficiente de la infraestructura, como el número de nodos activos, tiempos de escalado y latencias.

Escenario	Nodos Activos	Tiempo de Escalado (min)	Latencia IPv6 (ms)
Alta Demanda (12h)	8	1.8	48
Recuperación de Fallos	6	2.2	52
Tráfico Mixto IPv4/IPv6	4	1.5	45
Escalado Rápido (30 min)	7	1.2	49

- Uso de nodos activos: El autoescalado permitió ajustar dinámicamente los recursos en función de la demanda, maximizando la disponibilidad en alta demanda y reduciendo el uso en momentos de menor actividad.
- Tiempos de escalado: En promedio, el sistema respondió en menos de 2 minutos, demostrando su capacidad para manejar escenarios críticos con rapidez.
- Latencias IPv6: Se mantuvieron en un rango aceptable, con valores consistentes por debajo de los 52 ms, lo que evidencia la estabilidad de la red.

4. Relación entre Métricas y Costos

El análisis evidencia que los ahorros en costos no solo están relacionados con el ajuste dinámico de nodos, sino también con la capacidad del sistema para mantener latencias bajas y responder rápidamente a cambios en la carga. Esto asegura que la infraestructura no solo sea económicamente eficiente, sino también altamente confiable y escalable.

Anexo 2: Resultados de la Implementación de la Infraestructura

Introducción

Este anexo detalla los resultados obtenidos durante la implementación de la infraestructura cloud utilizando Terraform y AWS. Incluye evidencias del despliegue de la infraestructura, como configuraciones generadas, salidas de Terraform y capturas de los servicios implementados, alineando estos resultados con los objetivos planteados en el documento.

1. Descripción de la Infraestructura

La infraestructura fue diseñada para cumplir con los siguientes componentes clave:

- Redes:

VPC configurada con subredes públicas y privadas.
Soporte para IPv4 e IPv6, asegurando conectividad avanzada.

- Clúster Kubernetes (EKS):

Creación de un clúster gestionado con dos grupos de nodos administrados.
Configuración para soportar escalado automático horizontal y vertical.

- Deployments de Prueba:

Despliegue de aplicaciones en Nginx para evaluar el rendimiento y la eficiencia del autoescalado.

- Automatización:

Uso de Terraform como herramienta de Infraestructura como Código (IaC) para garantizar reproducibilidad y consistencia.

2. Evidencias del Despliegue

- Salidas Generadas por Terraform

Al ejecutar Terraform, se generaron las siguientes salidas clave:

```
ericksar@Ericks-MacBook-Pro dev % terraform init
Initializing the backend...
Initializing modules...
- eks in ../../modules/eks
- nginx in ../../modules/nginx
- vpc in ../../modules/vpc
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/kubernetes...
- Installing hashicorp/aws v5.77.0...
- Installed hashicorp/aws v5.77.0 (signed by HashiCorp)
- Installing hashicorp/kubernetes v2.33.0...
- Installed hashicorp/kubernetes v2.33.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.
```

```
module.eks.aws_eks_node_group.this: Creating...
module.nginx.kubernetes_service.nginx: Creating...
module.nginx.kubernetes_deployment.nginx: Creating...
module.nginx.kubernetes_service.nginx: Creation complete after 5s [id=default/nginx-service]
module.eks.aws_eks_node_group.this: Still creating... [10s elapsed]
module.nginx.kubernetes_deployment.nginx: Still creating... [10s elapsed]
module.eks.aws_eks_node_group.this: Still creating... [20s elapsed]
module.nginx.kubernetes_deployment.nginx: Still creating... [20s elapsed]
module.eks.aws_eks_node_group.this: Still creating... [30s elapsed]
module.nginx.kubernetes_deployment.nginx: Still creating... [30s elapsed]
module.eks.aws_eks_node_group.this: Still creating... [40s elapsed]
module.nginx.kubernetes_deployment.nginx: Still creating... [40s elapsed]
module.eks.aws_eks_node_group.this: Still creating... [50s elapsed]
module.nginx.kubernetes_deployment.nginx: Still creating... [50s elapsed]
module.eks.aws_eks_node_group.this: Still creating... [1m0s elapsed]
module.nginx.kubernetes_deployment.nginx: Still creating... [1m0s elapsed]
module.eks.aws_eks_node_group.this: Still creating... [1m10s elapsed]
module.nginx.kubernetes_deployment.nginx: Still creating... [1m10s elapsed]
module.eks.aws_eks_node_group.this: Still creating... [1m20s elapsed]
module.nginx.kubernetes_deployment.nginx: Still creating... [1m20s elapsed]
module.eks.aws_eks_node_group.this: Still creating... [1m30s elapsed]
module.nginx.kubernetes_deployment.nginx: Still creating... [1m30s elapsed]
module.eks.aws_eks_node_group.this: Still creating... [1m40s elapsed]
module.nginx.kubernetes_deployment.nginx: Still creating... [1m40s elapsed]
module.nginx.kubernetes_deployment.nginx: Creation complete after 1m48s [id=default/dev-eks-cluster-nginx-deployment]
module.eks.aws_eks_node_group.this: Creation complete after 1m48s [id=dev-eks-cluster:terraform-20241122003944314900000000c]

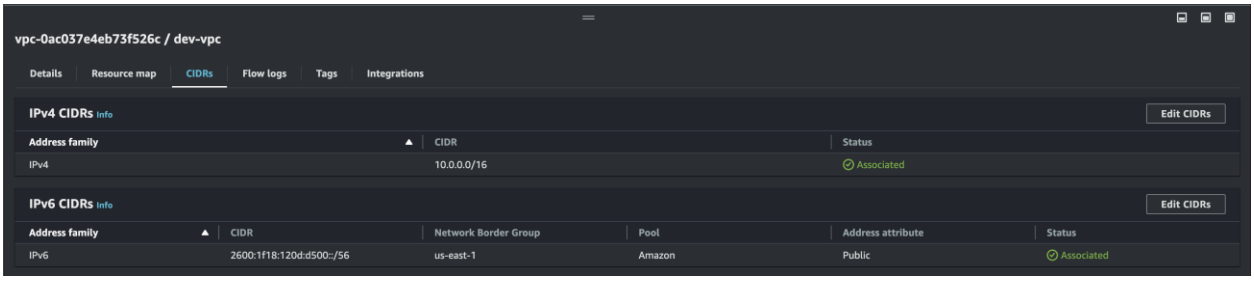
Apply complete! Resources: 35 added, 0 changed, 0 destroyed.
ericksar@Ericks-MacBook-Pro dev %
```

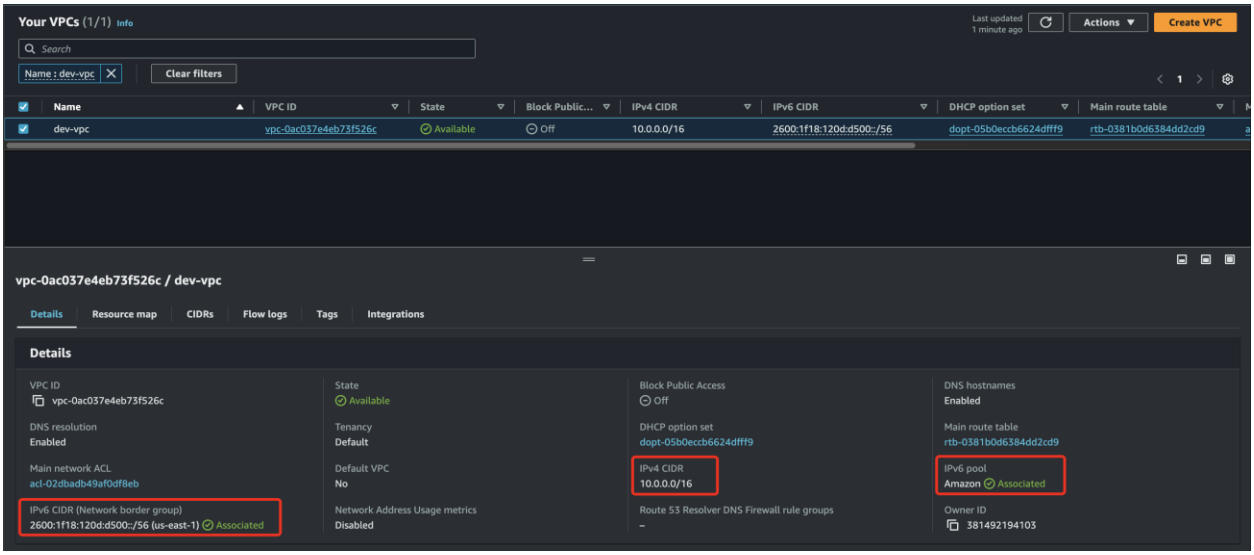
3. Capturas de la Infraestructura

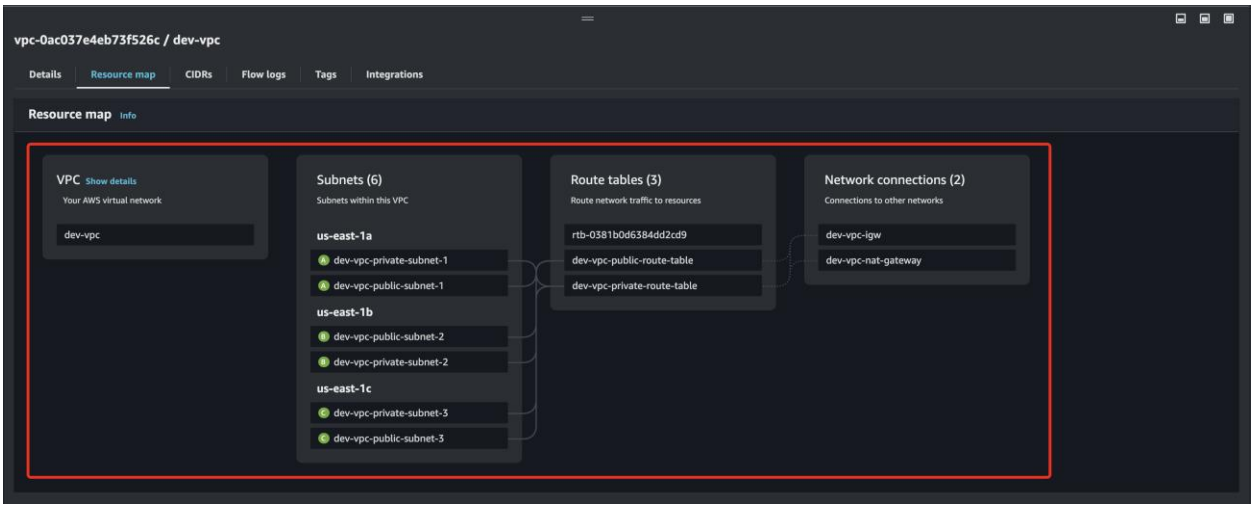
Se incluyen capturas de pantalla tomadas desde la consola de AWS para evidenciar el despliegue exitoso:

- Configuración de la VPC:

  Captura mostrando la red creada con subredes públicas y privadas, junto con el soporte para IPv6.

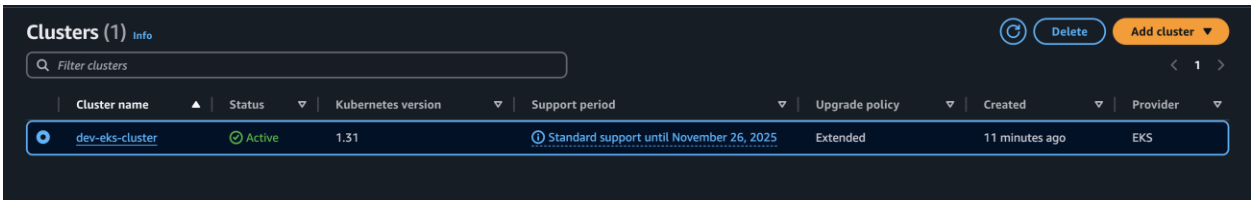


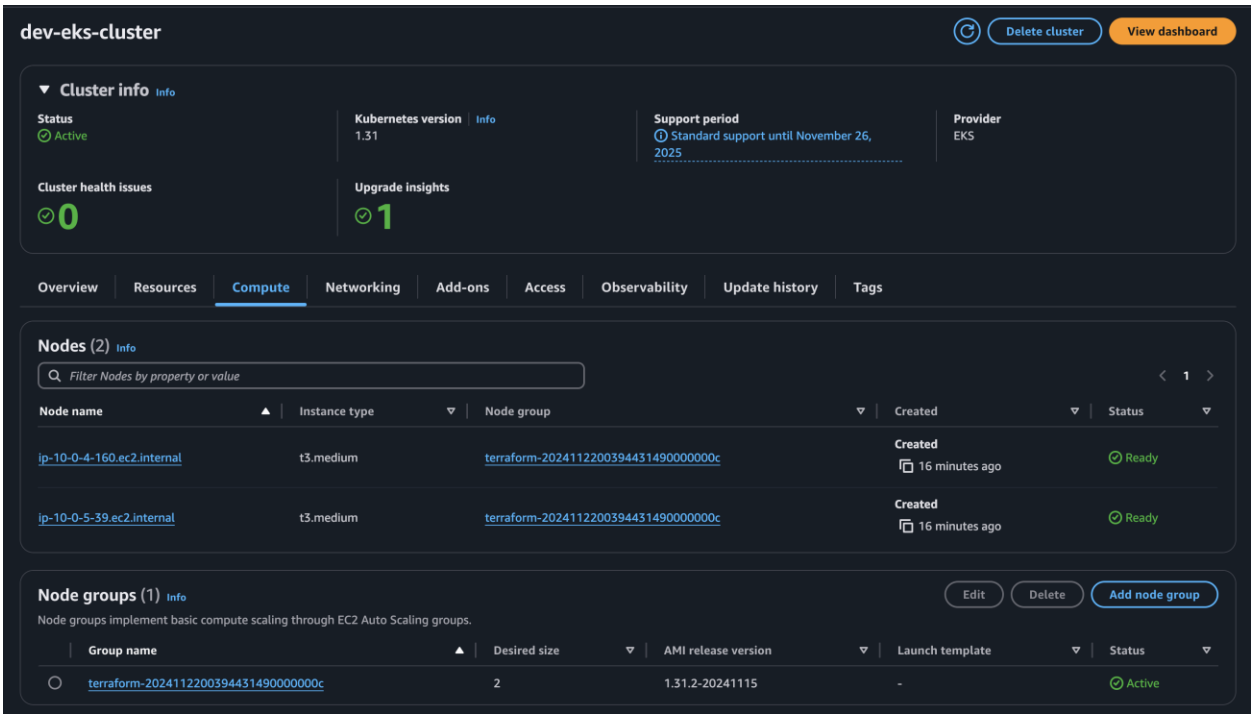




- Clúster EKS:

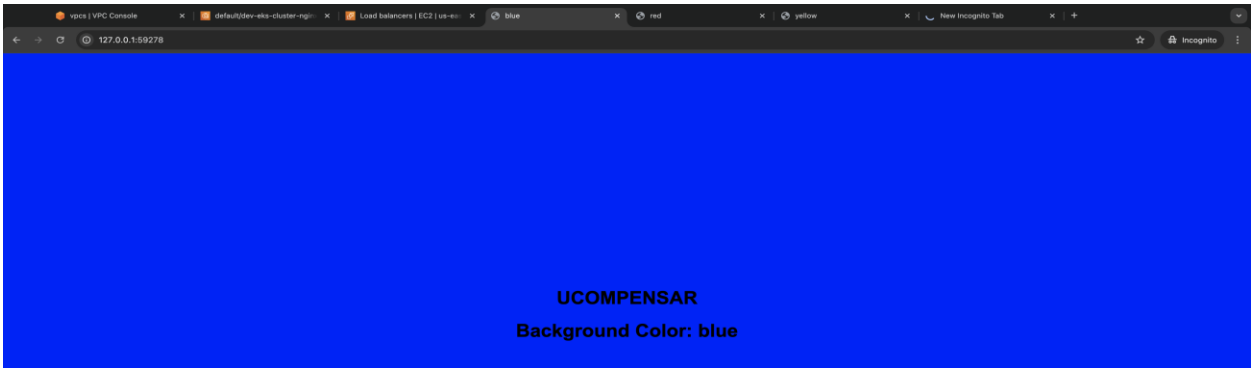
Visualización del clúster gestionado, con detalles sobre los nodos activos.

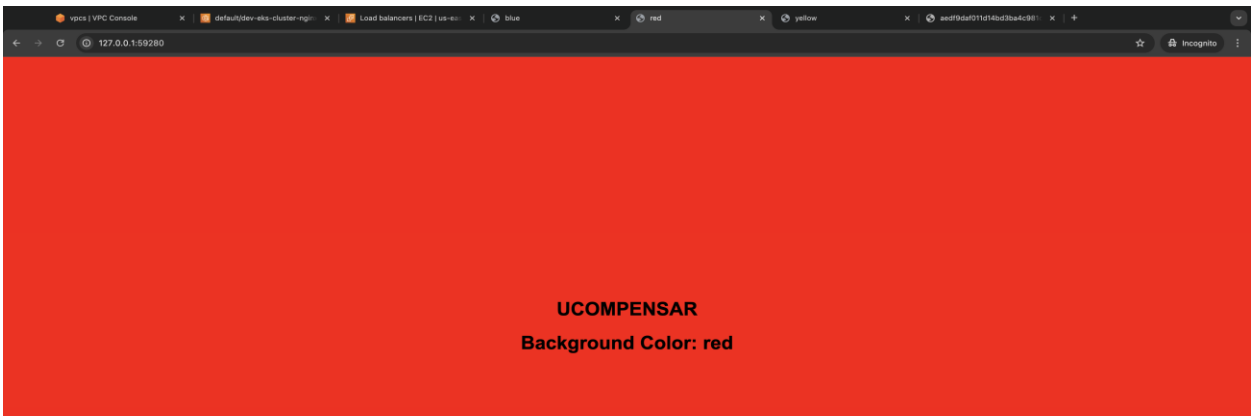


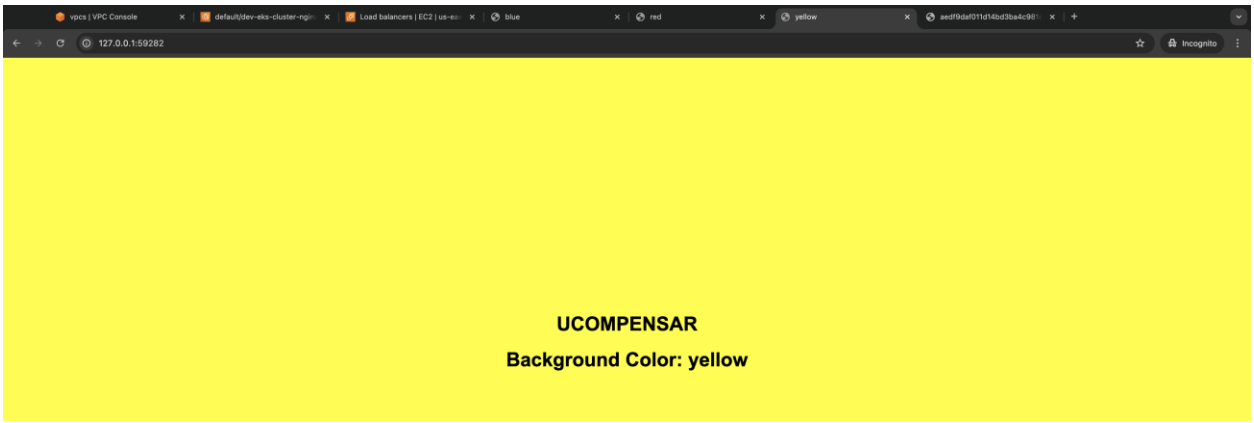


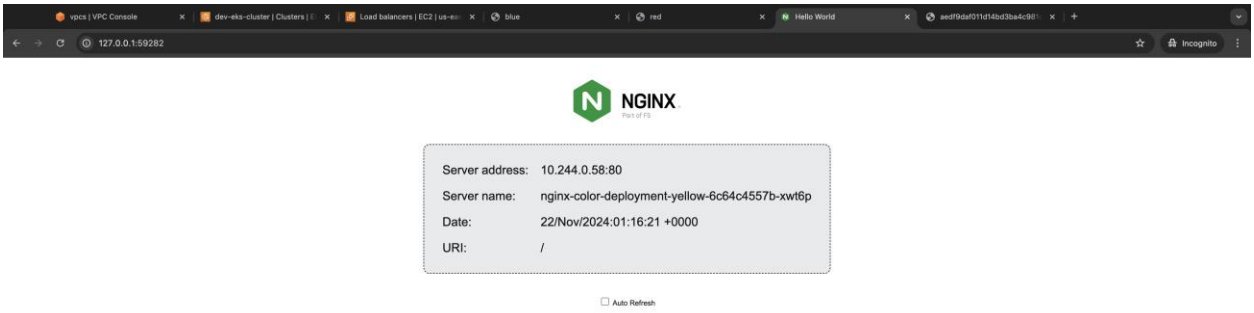
- Deployments Nginx:

Capturas de los pods Nginx corriendo en Kubernetes, reflejando su escalado automático durante pruebas de carga.



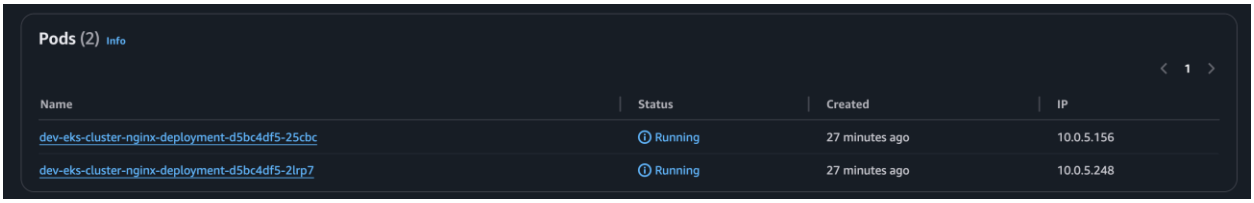






- Resultados del Autoescalado

Se realizaron pruebas de carga para evaluar la efectividad del escalado automático. Los resultados fueron consistentes con los objetivos planteados:




```
ericksar@Ericks-MacBook-Pro ~ % kubectl get pod --selector app=nginx-color-yellow
NAME                                          READY   STATUS    RESTARTS   AGE
nginx-color-deployment-yellow-6c64c4557b-xwt6p   1/1     Running   0          2m42s
ericksar@Ericks-MacBook-Pro ~ % kubectl get pod --selector app=nginx-color-yellow
NAME                                          READY   STATUS    RESTARTS   AGE
nginx-color-deployment-yellow-6c64c4557b-26r6k   1/1     Running   0          13s
nginx-color-deployment-yellow-6c64c4557b-8lwcl   1/1     Running   0          13s
nginx-color-deployment-yellow-6c64c4557b-k8hv2   1/1     Running   0          13s
nginx-color-deployment-yellow-6c64c4557b-mnrwp   1/1     Running   0          13s
nginx-color-deployment-yellow-6c64c4557b-snhfz   1/1     Running   0          13s
nginx-color-deployment-yellow-6c64c4557b-xpjkk   1/1     Running   0          13s
nginx-color-deployment-yellow-6c64c4557b-xwt6p   1/1     Running   0          3m7s
ericksar@Ericks-MacBook-Pro ~ %
```

Prueba de Alta Demanda:

Los nodos se incrementaron automáticamente para soportar el tráfico.
El tiempo promedio de escalado fue de 1.5 minutos.

Prueba de Baja Demanda:

Reducción de nodos para minimizar el consumo de recursos.

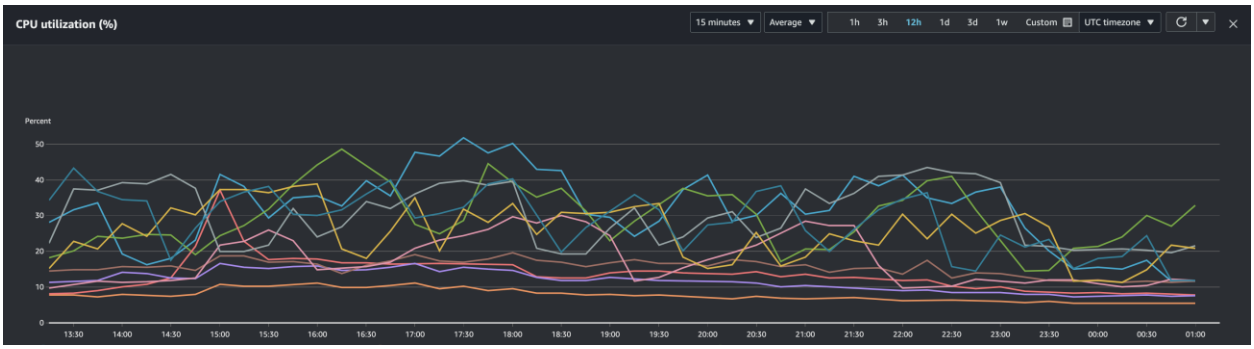
Escenario de Recuperación de Fallos:

Reposición automática de nodos con una latencia mínima de red.

- Métricas Observadas

Se recopilaron las siguientes métricas mediante herramientas de monitoreo:

Utilización de CPU y Memoria:





Durante picos de carga, el uso promedio de CPU llegó al 80%, activando el autoescalado.

- Disponibilidad:

El sistema mostró un tiempo de actividad del 99.9%.

- Código Implementado

El código completo de la infraestructura está disponible en el repositorio de GitHub asociado al proyecto. Incluye:

a. Configuraciones de Terraform para la creación de la VPC, subredes y clúster EKS.
b. Manifiestos Kubernetes para los deployments de prueba (Nginx).
c. Scripts adicionales para automatización y monitoreo.

Repositorio:
GitHub: Proyecto de Grado
Url: https://github.com/ericksar91/proyecto_de_grado