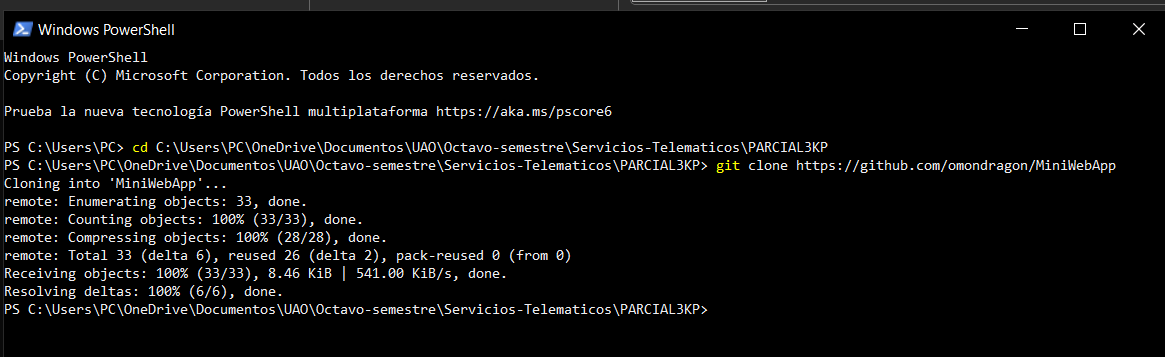
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Universidad Autónoma de Occidente | **UNIVERSIDAD AUTONOMA DE OCCIDENTE** | | | | | ***Valoración*** |
| **FACULTAD DE INGENIERIA** | | | **NOMBRE DE LA ASIGNATURA** | ***Servicios Telemáticos*** |
| **CODIGO:** | 2221343 | **NOMBRE:** | ***KAROL LIZETH PAYARES VIZCAINO*** | |
| **EXAMEN FINAL** | | | | | **FECHA SUSTENTACION: martes, noviembre 18 de 2025** | |

**Caso de Estudio: Despliegue Seguro, Monitoreo y Visualización de una Aplicación Web en la Nube**

La empresa ficticia CloudNova desea migrar su aplicación web de desarrollo a un entorno de producción seguro y monitoreado. Su objetivo es garantizar disponibilidad, seguridad y visibilidad del rendimiento mediante herramientas de código abierto y servicios en la nube. Usted ha sido contratado como ingeniero DevOps para liderar este proceso.

# Empaquetado y despliegue local con Docker (1.5 puntos)

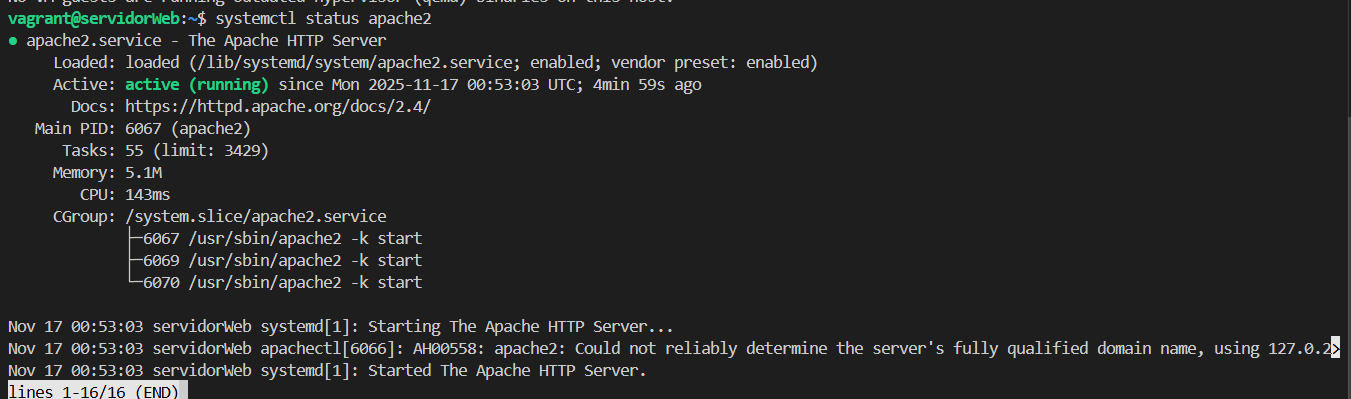
* Clone el repositorio [MiniWebApp en GitHub](https://github.com/omondragon/MiniWebApp) proporcionado en clase.



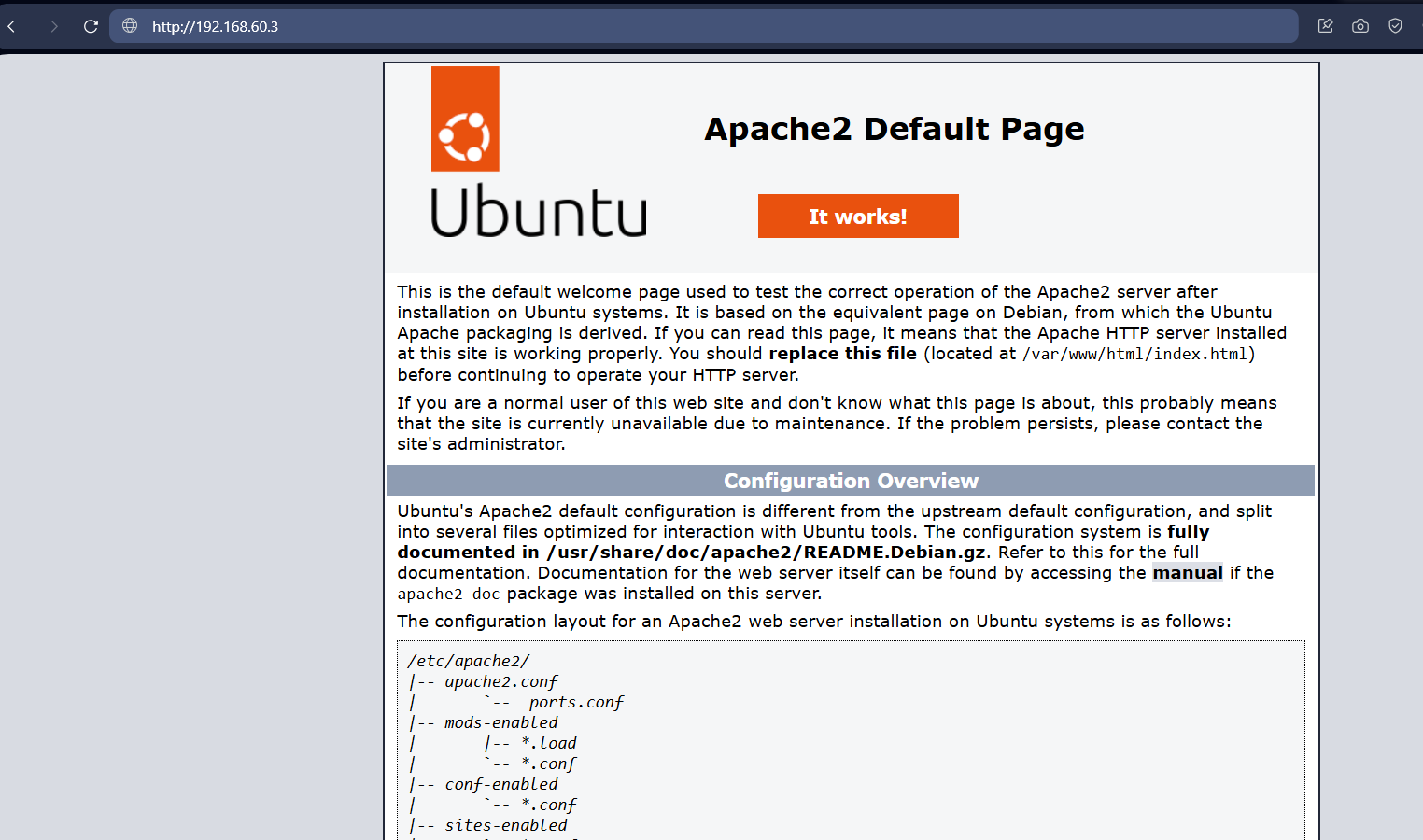
* git clone <https://github.com/omondragon/MiniWebApp>
* cd MiniWebApp
* vagrant up
* vagrant ssh servidorWeb
* Configure un servidor web (Apache o Nginx) para servir la aplicación mediante HTTPS, usando un certificado SSL válido.

**Configuración del servidor web con apache**

* sudo apt update
* sudo apt install apache2 -y
* systemctl status apache2 --no-pager



<http://192.168.60.3>



Verificar flask

vagrant@servidorWeb:~/webapp$ export FLASK\_APP=run.py

vagrant@servidorWeb:~/webapp$ python3 -m flask run --host=0.0.0.0

**Instalar OpenSSL**

* sudo apt list openssl --installed
* sudo apt install openssl -y
* sudo openssl version

**Desplegar la app de flask en apache**

* sudo apt update
* sudo apt install -y apache2 libapache2-mod-wsgi-py3

**Crear carpeta destino para la app**

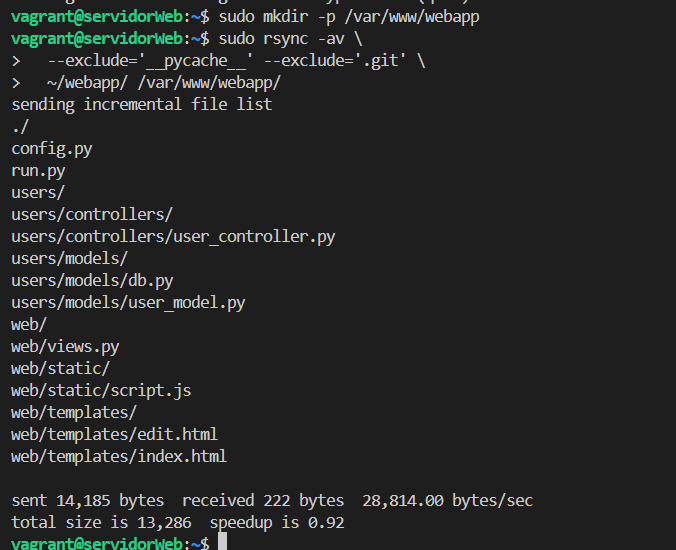
* sudo mkdir -p /var/www/webapp

**Copiar proyecto Flask hacia /var/www/webapp**

* sudo rsync -av \

--exclude='\_\_pycache\_\_' --exclude='.git' \

~/webapp/ /var/www/webapp/



**Dar permisos correctos**

* sudo chown -R www-data:www-data /var/www/webapp
* sudo chmod -R 755 /var/www/webapp

**Crear archivo WSGI**

* sudo tee /var/www/webapp/application.wsgi > /dev/null <<'WSGI'

#!/usr/bin/python3

import sys, os

# Asegura que Apache encuentre el paquete webapp

sys.path.insert(0, "/var/www/webapp")

# Tu aplicación Flask está en run.py y se llama "app"

from run import app as application

WSGI

* sudo chown www-data:www-data /var/www/webapp/application.wsgi
* sudo chmod 644 /var/www/webapp/application.wsgi

**Crear VirtualHost HTTP para la MiniWebApp**

* sudo tee /etc/apache2/sites-available/webapp.conf > /dev/null <<'APACHE'

<VirtualHost \*:80>

ServerName webapp.local

# Ejecuta Flask con mod\_wsgi

WSGIDaemonProcess webapp python-path=/var/www/webapp

WSGIProcessGroup webapp

WSGIScriptAlias / /var/www/webapp/application.wsgi

<Directory /var/www/webapp/>

Require all granted

</Directory>

# Archivos estáticos

Alias /static /var/www/webapp/web/static

<Directory /var/www/webapp/web/static>

Require all granted

</Directory>

ErrorLog ${APACHE\_LOG\_DIR}/webapp\_error.log

CustomLog ${APACHE\_LOG\_DIR}/webapp\_access.log combined

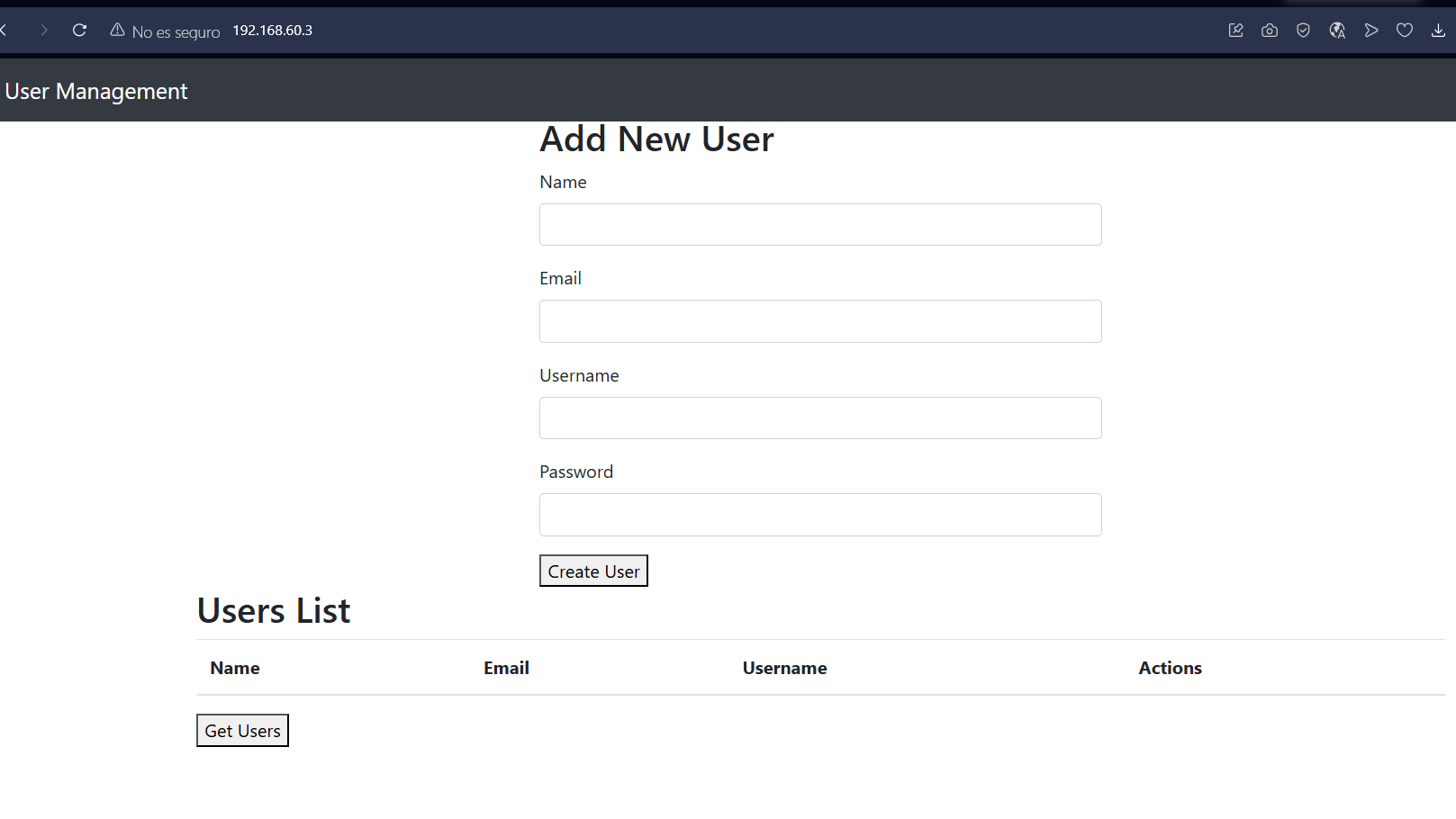
</VirtualHost>

APACHE

**Deshabilitar default site y habilitar sitio**

* sudo a2dissite 000-default.conf
* sudo a2ensite webapp.conf
* sudo systemctl reload apache2

<http://192.168.60.3>



**Habilitar SSL en Apache**

* sudo a2enmod ssl
* sudo systemctl restart apache2

Verifica que está activo:

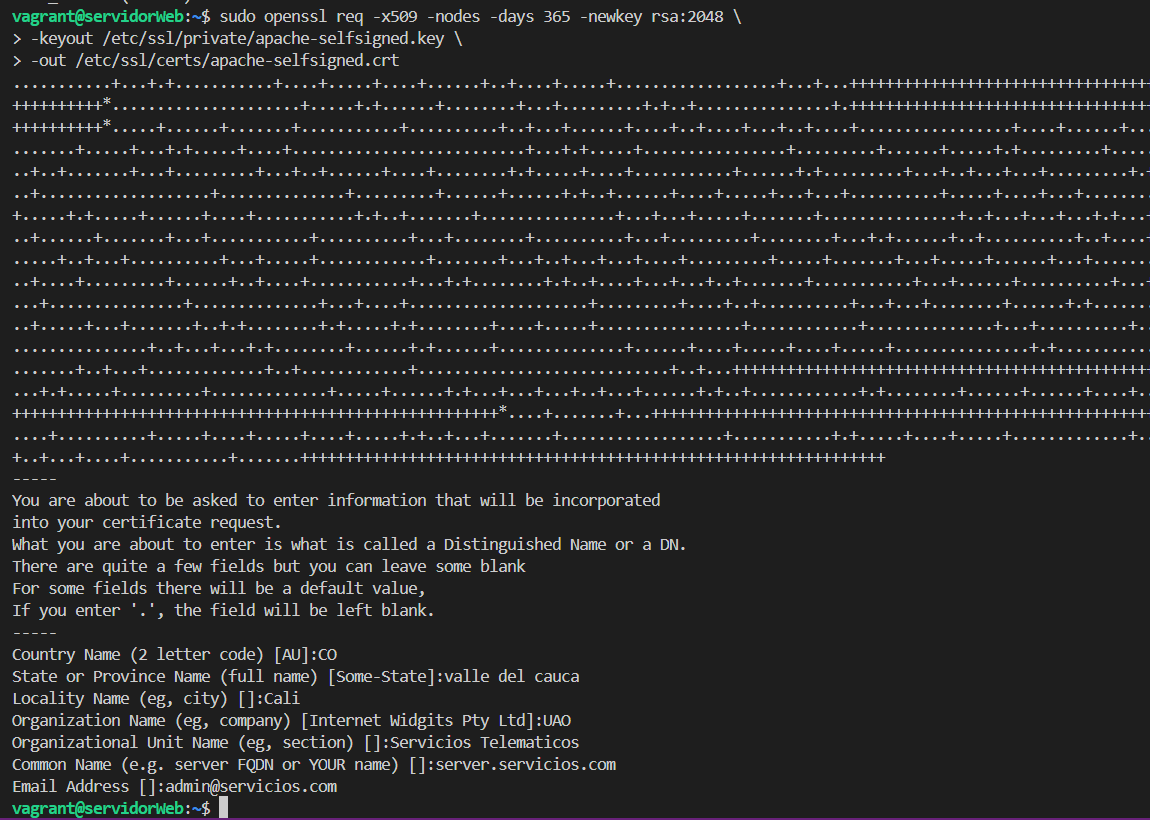
* sudo apache2ctl -M | grep ssl

**Crear certificado SSL autofirmado**

* sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

-keyout /etc/ssl/private/apache-selfsigned.key \

-out /etc/ssl/certs/apache-selfsigned.crt



* Asegure la redirección automática de HTTP a HTTPS.

Crear VirtualHost HTTPS (443)

* sudo tee /etc/apache2/sites-available/webapp-ssl.conf > /dev/null <<'SSL'

<VirtualHost \*:443>

ServerName webapp.local

SSLEngine on

SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt

SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key

# Ejecuta Flask por mod\_wsgi

WSGIDaemonProcess webapp python-path=/var/www/webapp

WSGIProcessGroup webapp

WSGIScriptAlias / /var/www/webapp/application.wsgi

<Directory /var/www/webapp/>

Require all granted

</Directory>

# Archivos estáticos

Alias /static /var/www/webapp/web/static

<Directory /var/www/webapp/web/static>

Require all granted

</Directory>

ErrorLog ${APACHE\_LOG\_DIR}/webapp\_ssl\_error.log

CustomLog ${APACHE\_LOG\_DIR}/webapp\_ssl\_access.log combined

</VirtualHost>

SSL

**Activar el sitio HTTPS**

* sudo a2ensite webapp-ssl.conf
* sudo apache2ctl configtest

ESTO DARA UN ERROR DE SINTAXIS YA QUE TENEMOS DOS CONFIGURACIONES PARECIDAS

SOLUCIONAR:

modificar **/etc/apache2/sites-available/webapp.conf**

* sudo tee /etc/apache2/sites-available/webapp.conf > /dev/null <<'APACHE'

<VirtualHost \*:80>

ServerName 192.168.60.3

Redirect permanent / https://192.168.60.3/

</VirtualHost>

APACHE

--------------------------------

* sudo tee /etc/apache2/sites-available/webapp-ssl.conf > /dev/null <<'SSL'

<VirtualHost \*:443>

ServerName 192.168.60.3

SSLEngine on

SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt

SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key

WSGIDaemonProcess webapp python-path=/var/www/webapp

WSGIProcessGroup webapp

WSGIScriptAlias / /var/www/webapp/application.wsgi

<Directory /var/www/webapp/>

Require all granted

</Directory>

Alias /static /var/www/webapp/web/static

<Directory /var/www/webapp/web/static>

Require all granted

</Directory>

ErrorLog ${APACHE\_LOG\_DIR}/webapp\_ssl\_error.log

CustomLog ${APACHE\_LOG\_DIR}/webapp\_ssl\_access.log combined

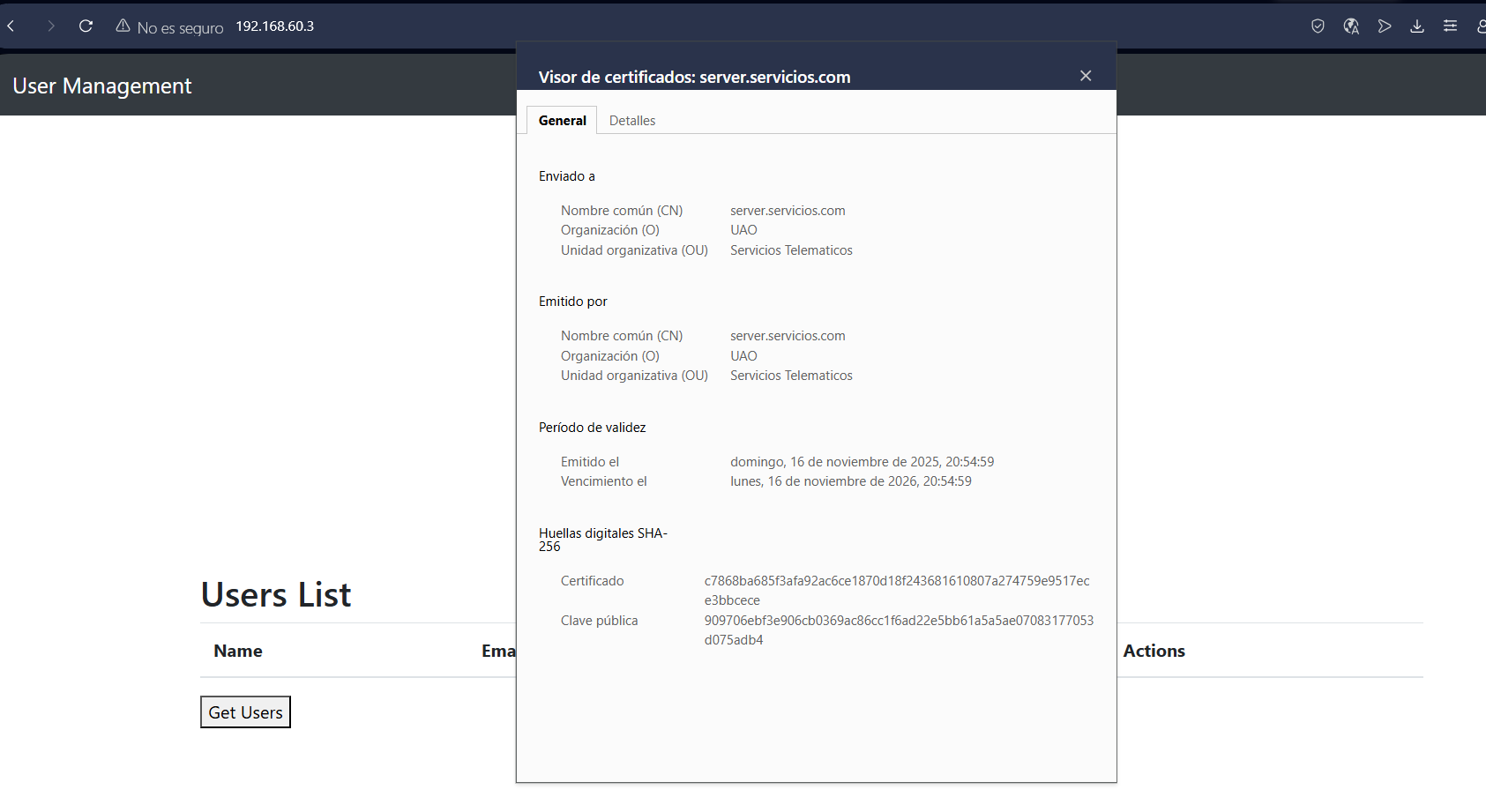
</VirtualHost>

SSL

* sudo apache2ctl configtest
* sudo systemctl reload apache2

curl -I <http://192.168.60.3>

curl -k <https://192.168.60.3>



Texto

El contenido generado por IA puede ser incorrecto.

<https://192.168.60.3>

* Cree un Dockerfile y un docker-compose.yml para orquestar la aplicación.

INTALAR LO NECESARIO

* sudo apt-get update
* sudo apt-get install ca-certificates curl -y
* sudo install -m 0755 -d /etc/apt/keyrings
* sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
* sudo chmod a+r /etc/apt/keyrings/docker.asc
* echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \

* sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
* sudo apt-get update
* sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
* sudo usermod -aG docker $USER
* EXIT
* VAGRANT SSH
* mkdir docker-webapp
* cd docker-webapp
* mkdir app
* cp -r /var/www/webapp/\* app/
* sudo tee requirements.txt > /dev/null <<'EOF'

flask

flask\_sqlalchemy

pymysql

gunicorn

EOF

* sudo tee app/wsgi.py > /dev/null <<'EOF'

from web.views import app

if \_\_name\_\_ == "\_\_main\_\_":

app.run()

EOF

Ajustar conexión MySQL para usar PyMySQL

* sudo sed -i 's/mysql+mysqldb/mysql+pymysql/g' app/config.py
* sudo sed -i 's/mysql:\/\/\([^"]\*\)/mysql+pymysql:\/\/\1/g' app/config.py
* sudo tee Dockerfile > /dev/null <<'EOF'

FROM python:3.10-slim

WORKDIR /app

COPY requirements.txt /app/

RUN pip install --no-cache-dir -r requirements.txt

COPY app/ /app/

EXPOSE 5000

CMD ["gunicorn", "-w", "4", "-b", "0.0.0.0:5000", "wsgi:app"]

EOF

sudo tee docker-compose.yml > /dev/null <<'EOF'

services:

web:

build: .

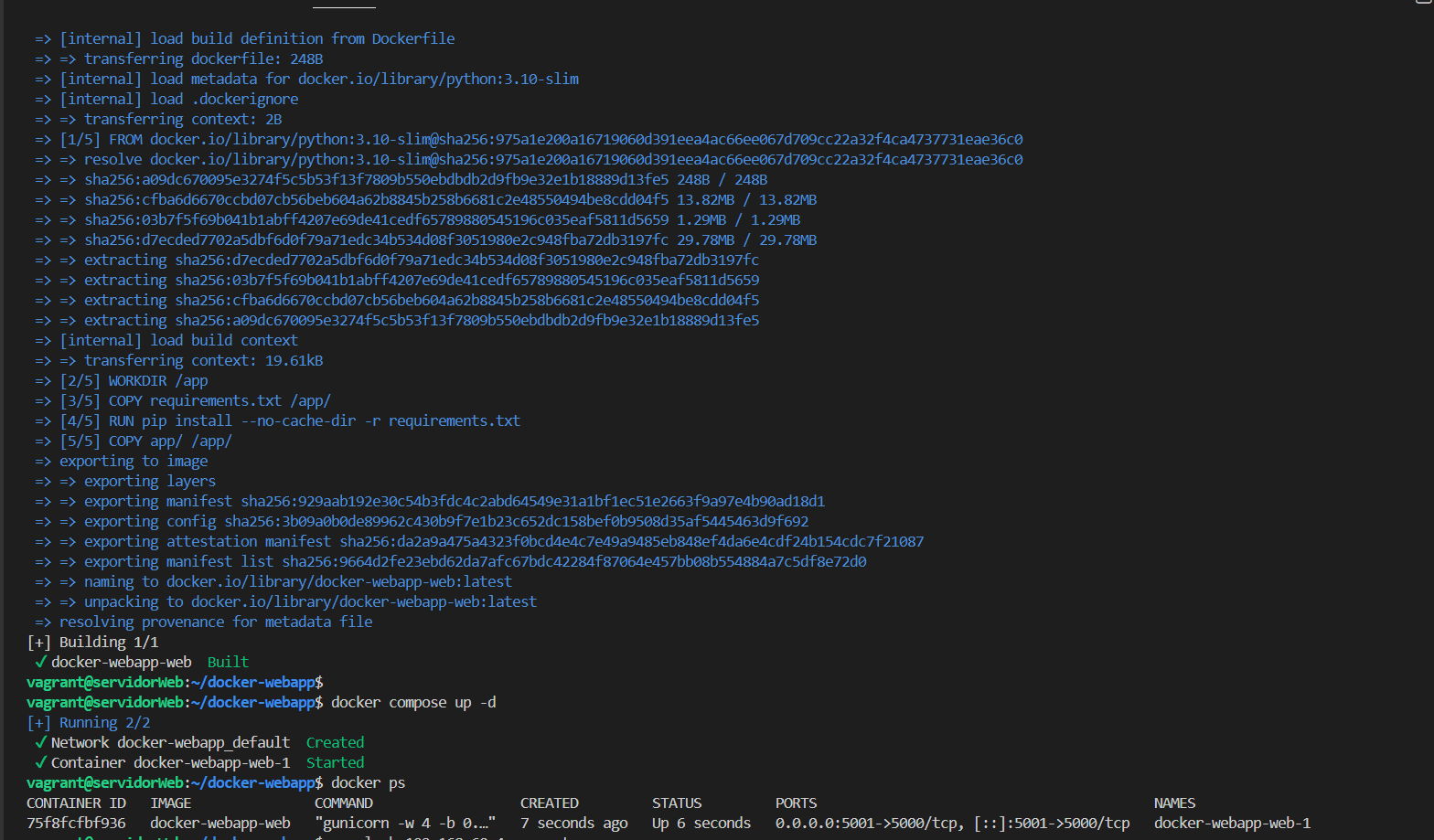
ports:

- "5001:5000"

restart: always

EOF

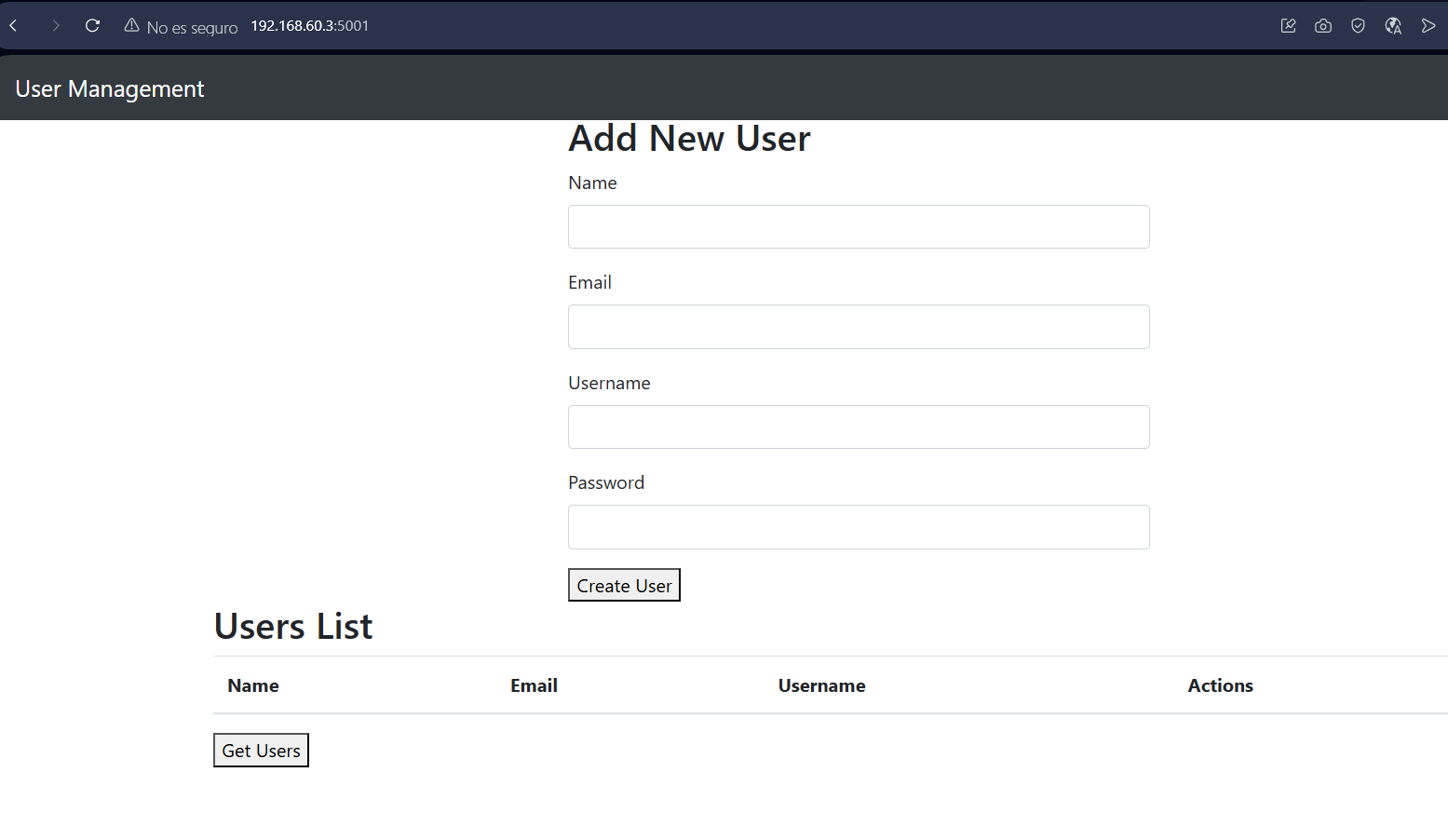
* docker compose down
* docker compose build --no-cache
* docker compose up -d
* docker ps



“Configuración de Apache con SSL y redirección HTTP → HTTPS en Docker”

* mkdir docker-apache-ssl
* cd docker-apache-ssl
* mkdir html
* cp /var/www/webapp/web/templates/index.html html/
* sudo tee Dockerfile > /dev/null <<'EOF'
* FROM ubuntu:22.04
* RUN apt update && apt install -y apache2 openssl
* RUN mkdir /etc/apache2/ssl && \
* openssl req -x509 -nodes -days 365 \
* -subj "/C=CO/ST=Valle/L=Cali/O=UAO/OU=Telematicos/CN=webapp.local" \
* -newkey rsa:2048 \
* -keyout /etc/apache2/ssl/apache.key \
* -out /etc/apache2/ssl/apache.crt
* RUN a2enmod ssl
* RUN a2enmod rewrite
* COPY html/ /var/www/html/
* RUN tee /etc/apache2/sites-available/ssl.conf > /dev/null <<EOF2
* <VirtualHost \*:443>
* DocumentRoot /var/www/html/
* SSLEngine on
* SSLCertificateFile /etc/apache2/ssl/apache.crt
* SSLCertificateKeyFile /etc/apache2/ssl/apache.key
* </VirtualHost>
* EOF2
* RUN a2ensite ssl.conf && a2dissite 000-default.conf
* EXPOSE 80 443
* CMD ["apache2ctl", "-D", "FOREGROUND"]
* EOF
* docker build -t miniwebapp-apache-ssl .
* docker run -d --name web-ssl -p 8080:80 -p 8443:443 miniwebapp-apache-ssl
* docker exec -it web-ssl bash
* sudo tee /etc/apache2/sites-available/000-default.conf > /dev/null <<'EOF'
* <VirtualHost \*:80>
* ServerName 192.168.60.3
* Redirect permanent / https://192.168.60.3:8443/
* </VirtualHost>
* EOF
* a2ensite 000-default.conf
* service apache2 reload
* exit
* Verifique el funcionamiento accediendo desde un navegador local.

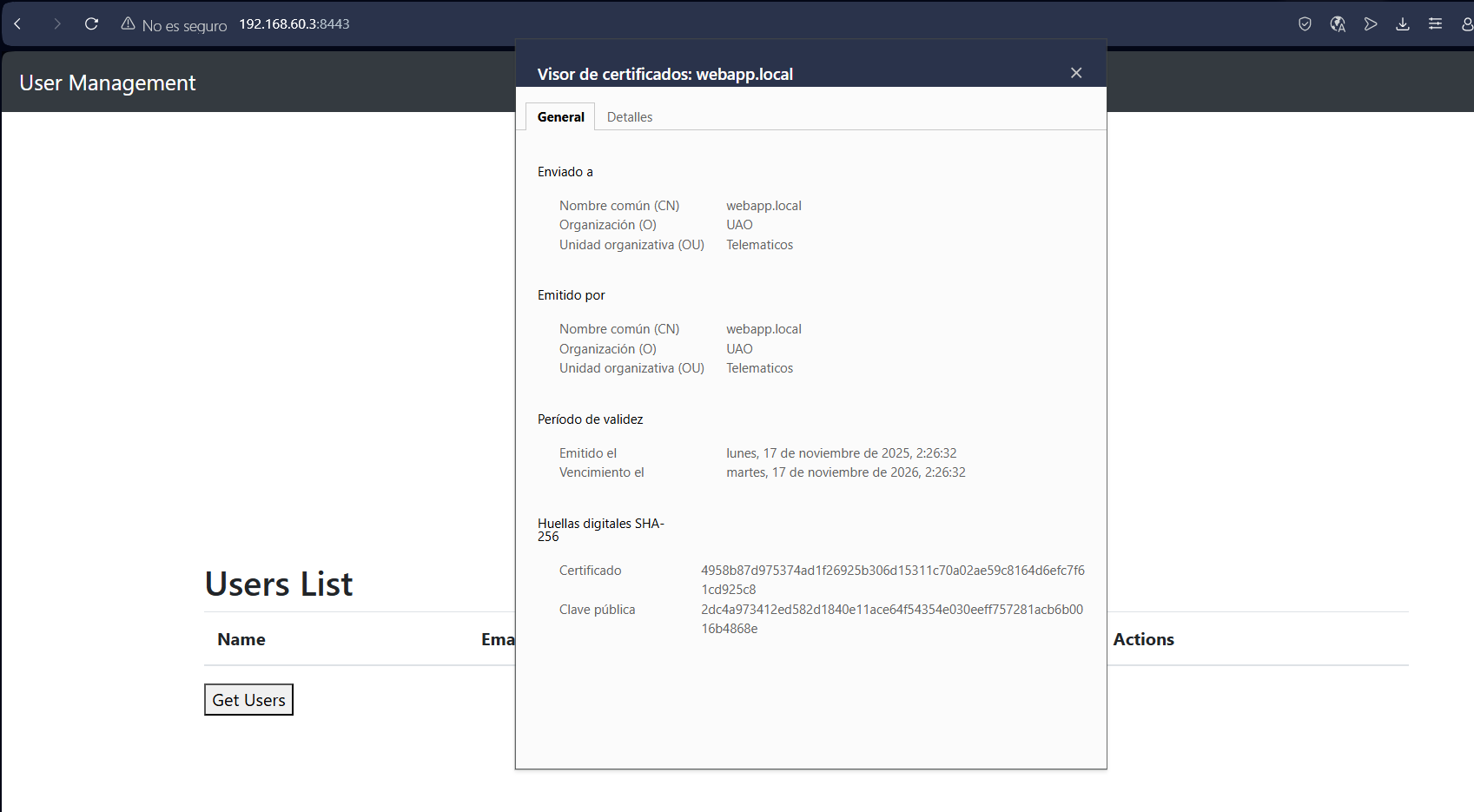
<http://192.168.60.3:5001>



Con apache

<http://192.168.60.3:8080>

<https://192.168.60.3:8443>



#### **Recursos sugeridos (Sin garantía ni soporte)**

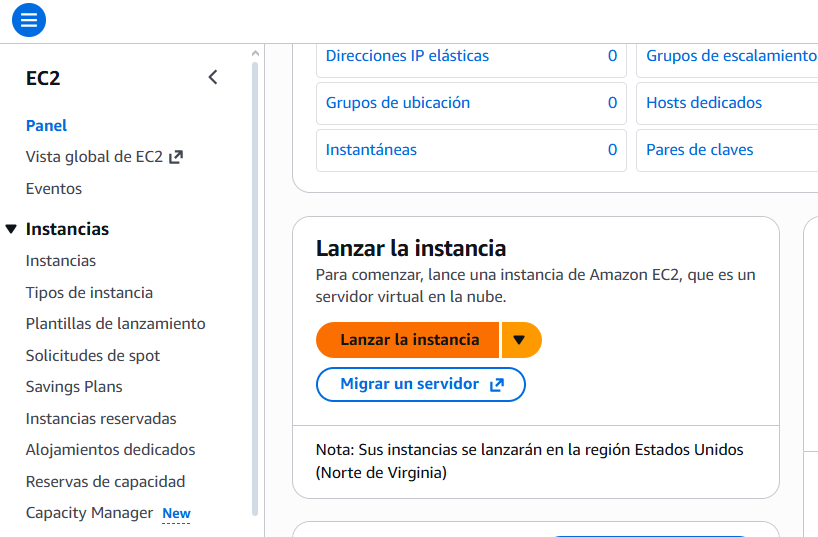
[Asegurar Apache con SSL en Docker](https://medium.com/@tshenolomos/secure-apache-with-ssl-in-docker-9efd86329129)

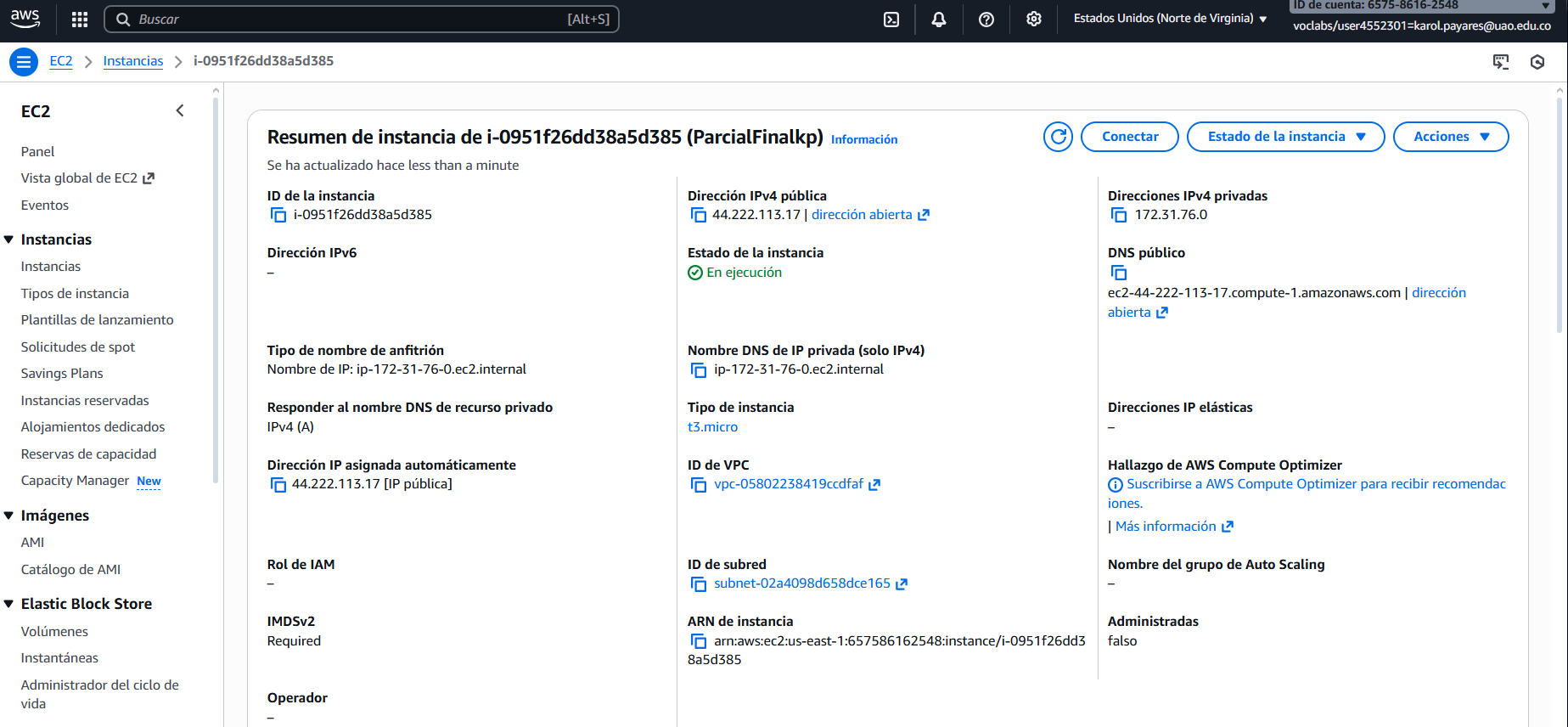
# Despliegue en la nube con AWS EC2 (1.0 punto)

* Inicie una instancia EC2 en AWS y configure las reglas de seguridad para

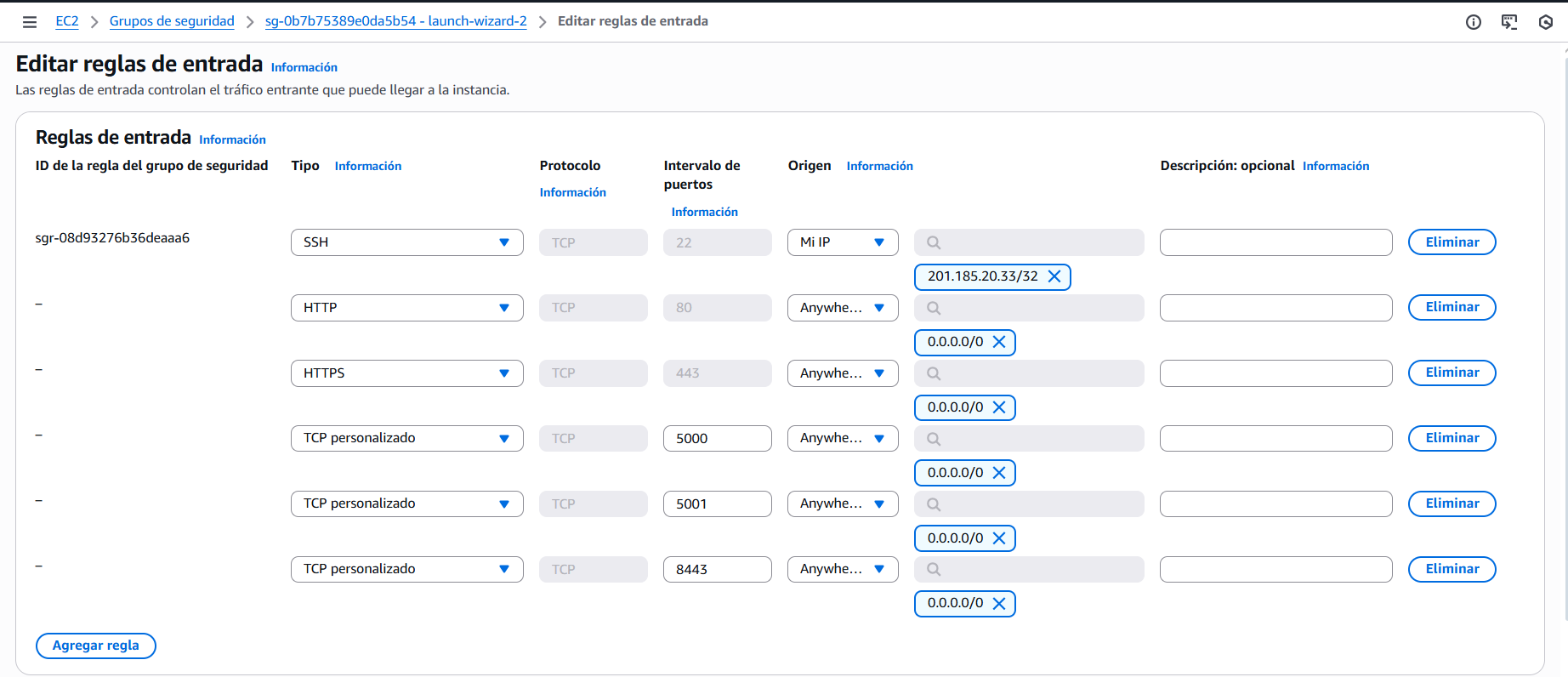
Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.



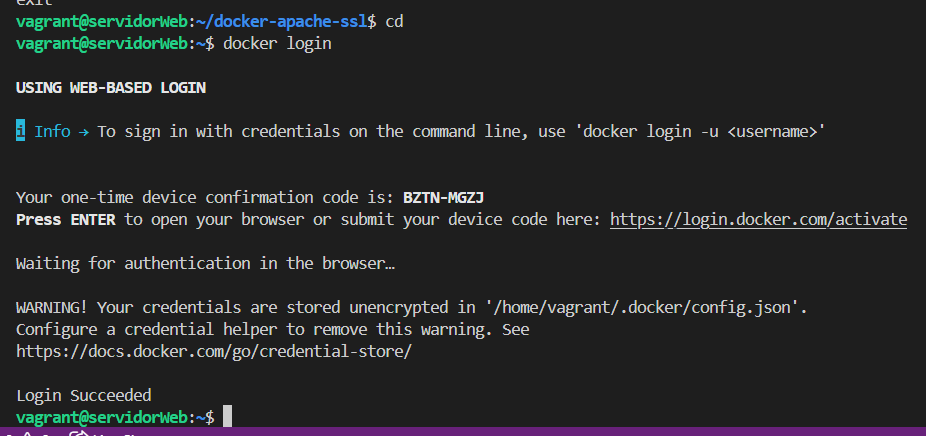
* permitir tráfico HTTP/HTTPS.



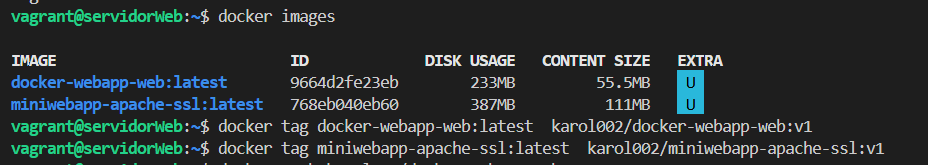
Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

* Instale Docker y ejecute el contenedor de la aplicación usando Docker Compose.
* docker login
* docker images
* docker tag docker-webapp-web:latest karol002/docker-webapp-web:v1
* docker tag miniwebapp-apache-ssl:latest karol002/miniwebapp-apache-ssl:v1
* docker push karol002/docker-webapp-web:v1
* docker push karol002/ miniwebapp-apache-ssl:v1







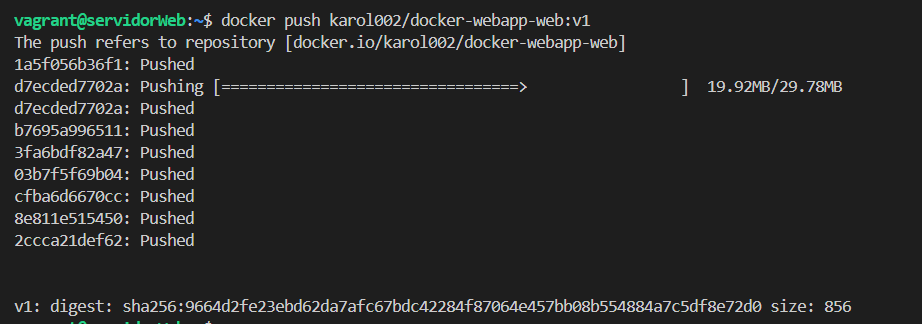
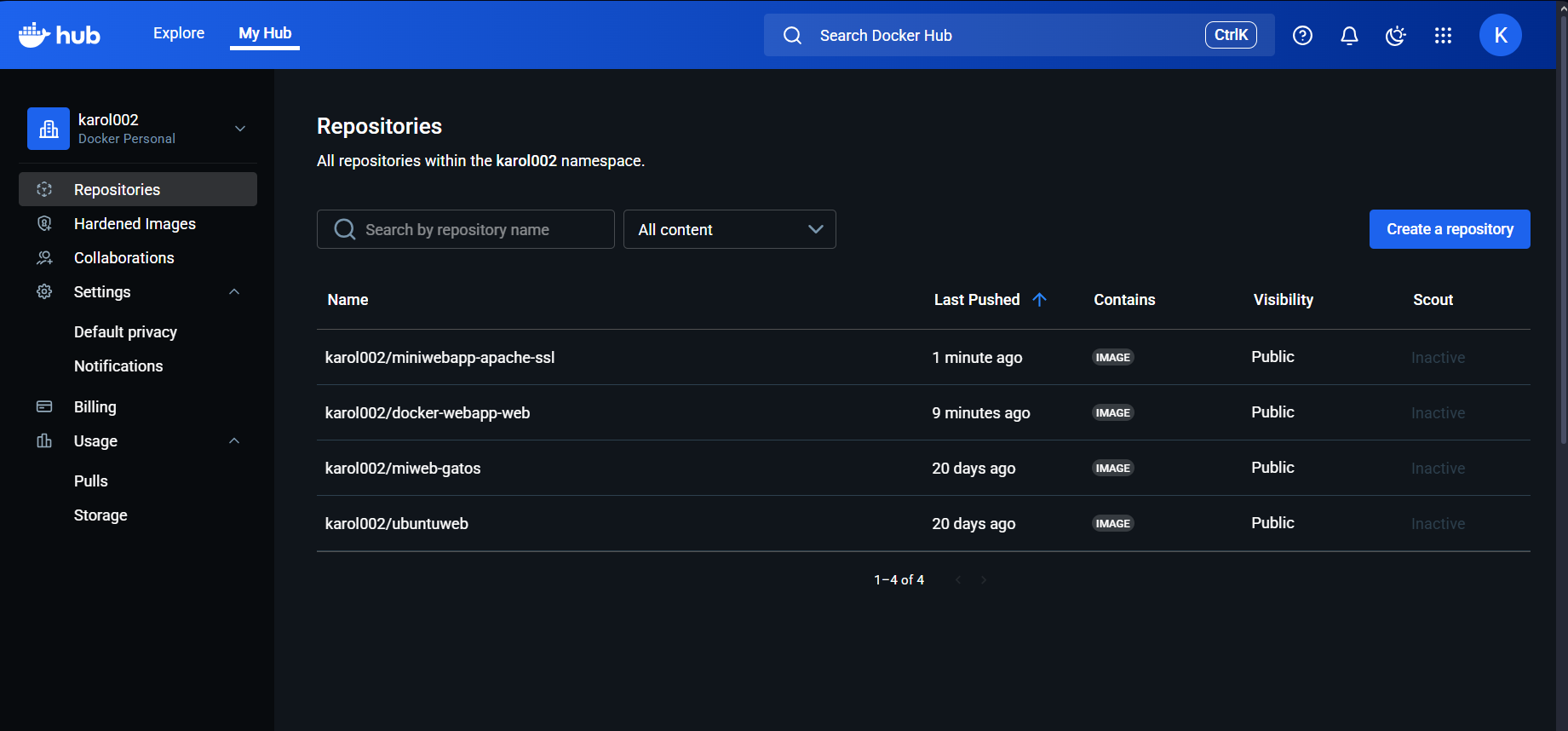
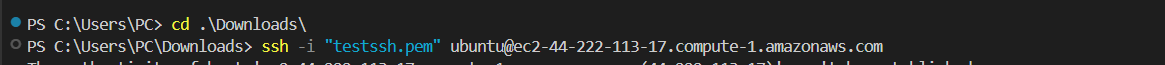


Imagen que contiene computadora, tabla

El contenido generado por IA puede ser incorrecto.



Poner Docker en la vm de aws



* sudo apt install -y docker.io
* sudo systemctl enable docker
* sudo systemctl start Docker
* sudo apt install -y docker-compose
* sudo usermod -aG docker $USER
* exit
* ssh a la maquina de aws

Texto

El contenido generado por IA puede ser incorrecto.

docker pull karol002/miniwebapp-apache-ssl:v1

Texto

El contenido generado por IA puede ser incorrecto.

* mkdir miniwebaws
* cd miniwebaws
* sudo vim docker-compose.yml

version: "3.9"

services:

miniweb:

image: karol002/miniwebapp-apache-ssl:v1

container\_name: miniweb-aws

ports:

- "80:80"

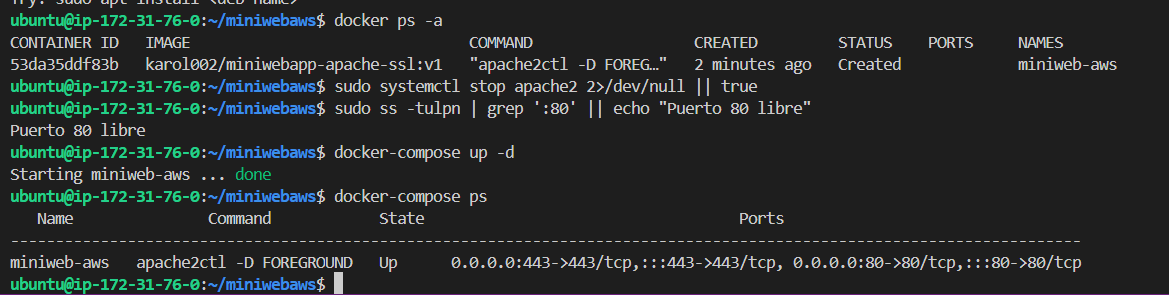
- "443:443"

restart: always

* docker-compose up -d

si sale error

* sudo systemctl stop apache2 2>/dev/null || true
* sudo ss -tulpn | grep ':80' || echo "Puerto 80 libre"
* docker-compose ps



* Verifique el acceso remoto a la aplicación a través de la IP pública.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

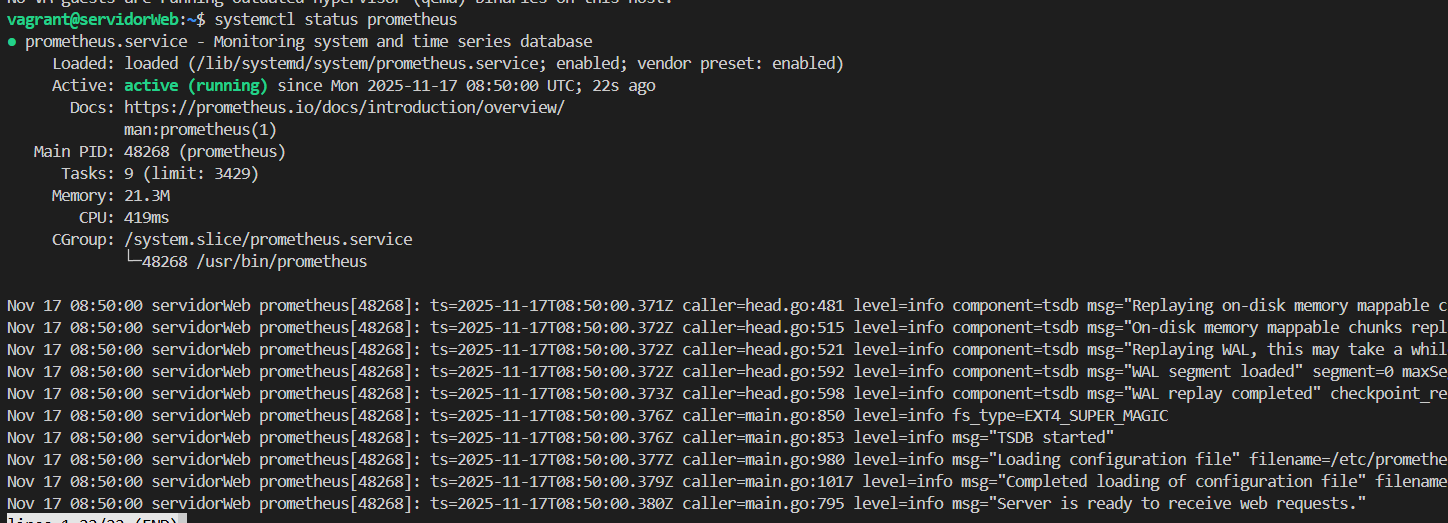
El contenido generado por IA puede ser incorrecto.

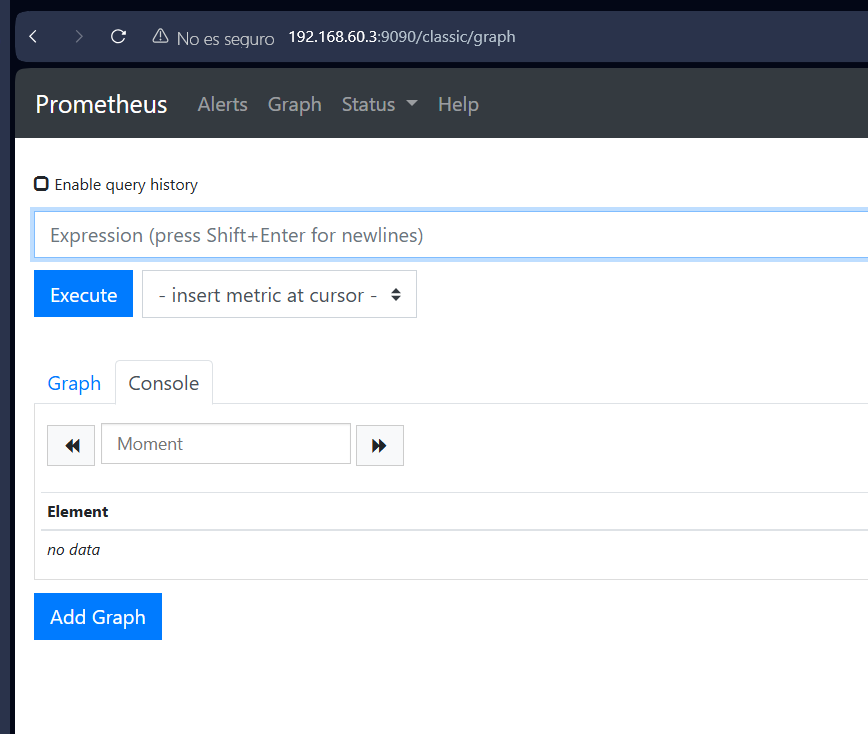
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

# Monitoreo con Prometheus y Node Exporter (1.5 puntos)

* Instale y configure Prometheus
* sudo apt update
* sudo apt install -y prometheus prometheus-node-exporter
* systemctl status prometheus





* Configure prometheus.yml para recolectar métricas locales y de Node Exporter.

sudo cp /etc/prometheus/prometheus.yml /etc/prometheus/prometheus.yml.bak

cat << 'EOF' | sudo tee /etc/prometheus/prometheus.yml

global:

scrape\_interval: 15s

evaluation\_interval: 15s

# Cargar reglas de alertas

rule\_files:

- /etc/prometheus/alerts.yml

scrape\_configs:

# Prometheus se scrapea a sí mismo

- job\_name: 'prometheus'

static\_configs:

- targets: ['localhost:9090']

# Node Exporter de esta misma máquina

- job\_name: 'node-exporter'

static\_configs:

- targets: ['localhost:9100']

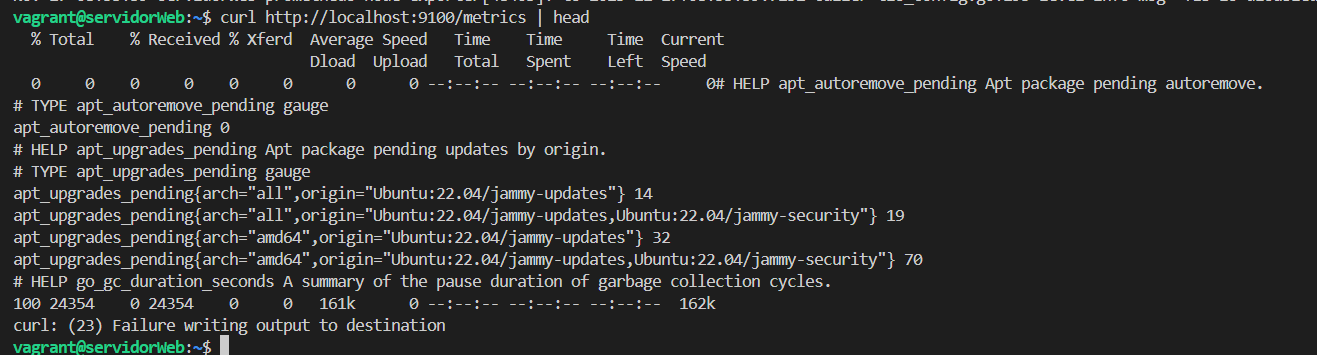
EOF

* Instale Node Exporter para recopilar métricas del sistema (CPU, memoria, disco).

$systemctl status prometheus-node-exporterTexto

El contenido generado por IA puede ser incorrecto.

curl http://localhost:9100/metrics | head



<http://192.168.60.3:9100/metrics>

Texto

El contenido generado por IA puede ser incorrecto.

* Documente tres métricas específicas y explique su utilidad en el monitoreo de un sistema Linux.

### 1)node\_cpu\_seconds\_total

* **Qué es:**  
  Contador de segundos acumulados que la CPU pasa en cada modo (user, system, idle, etc.).
* **Por qué sirve (CloudNova):**  
  A partir de esta métrica se calcula el **porcentaje de uso de CPU** (por ejemplo usando rate o irate). Si el uso de CPU se mantiene alto (por encima de 80%) durante varios minutos, indica que la aplicación de CloudNova está bajo mucha carga o hay procesos mal optimizados. Permite decidir si hay que escalar recursos o investigar procesos específicos.

Ejemplo de consulta para uso de CPU:

100 - (avg by (instance) (irate(node\_cpu\_seconds\_total{mode="idle"}[5m])) \* 100)

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

#### 2) node\_memory\_MemAvailable\_bytes

* **Qué es:**  
  Cantidad de memoria RAM disponible para nuevas aplicaciones (incluye caché reutilizable).
* **Por qué sirve (CloudNova):**  
  Indica si el servidor se está quedando sin memoria. Si este valor es bajo de forma sostenida, el sistema puede empezar a usar swap y la aplicación web de CloudNova se vuelve lenta o incluso puede fallar. Es clave para planear aumento de RAM o redistribución de servicios.

Ejemplo para porcentaje de memoria disponible:

(node\_memory\_MemAvailable\_bytes / node\_memory\_MemTotal\_bytes) \* 100

Gráfico

El contenido generado por IA puede ser incorrecto.

#### 3) node\_filesystem\_avail\_bytes{mountpoint="/"}

* **Qué es:**  
  Espacio disponible en bytes en la partición raíz / del sistema de archivos.
* **Por qué sirve (CloudNova):**  
  Si el disco se llena, la aplicación no podrá escribir logs, archivos subidos por usuarios ni datos críticos. Servicios como la base de datos o el propio sistema operativo pueden fallar. Lo habitual es configurar alertas cuando queda menos del 10–15% libre para poder actuar antes de que el sistema se bloquee.

Ejemplo de porcentaje libre en /:

(node\_filesystem\_avail\_bytes{mountpoint="/"}

/ node\_filesystem\_size\_bytes{mountpoint="/"}) \* 100



* Configure alertas básicas en Prometheus (por ejemplo, CPU > 80%).

#### cat << 'EOF' | sudo tee /etc/prometheus/alerts.yml

#### groups:

#### - name: cloudnova-alerts

#### rules:

#### - alert: HighCPUUsage

#### expr: 100 - (avg by (instance) (irate(node\_cpu\_seconds\_total{mode="idle"}[5m])) \* 100) > 80

#### for: 2m

#### labels:

#### severity: warning

#### annotations:

#### summary: "CPU alta en {{ $labels.instance }}"

#### description: "Uso de CPU mayor al 80% durante más de 2 minutos. Revisar la carga de la aplicación CloudNova."

#### EOF

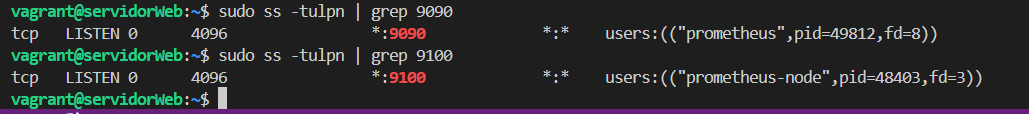
#### **Recursos sugeridos (Sin garantía ni soporte)**

[Guía para instalar Prometheus en Ubuntu](https://www.cherryservers.com/blog/install-prometheus-ubuntu)

[Guía para instalar y ejecutar Node Exporter](https://prometheus.io/docs/guides/node-exporter/#installing-and-running-the-node-exporter)

PRUEBAS DE FUCIONAMIENTO:

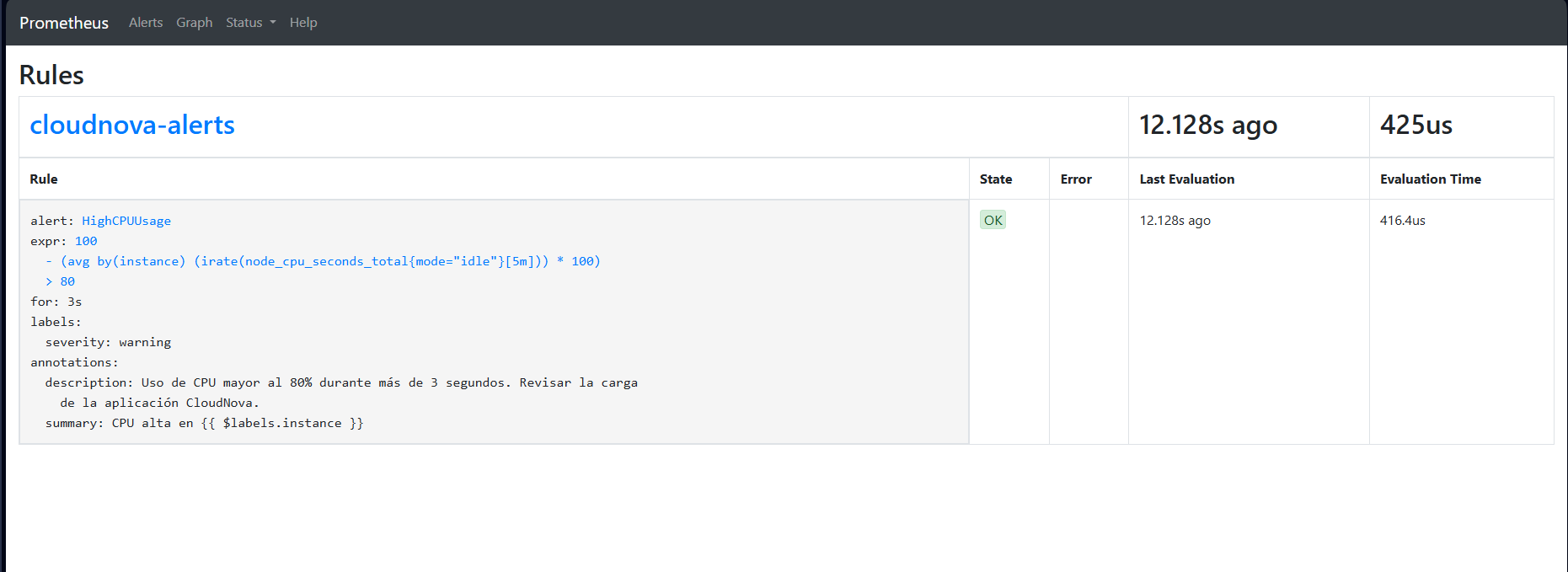
* sudo systemctl status prometheus
* sudo systemctl status prometheus-node-exporter
* sudo ss -tulpn | grep 9090
* sudo ss -tulpn | grep 9100





sudo promtool check rules /etc/prometheus/alerts.yml

sudo systemctl restart prometheus y status



Probar con stress

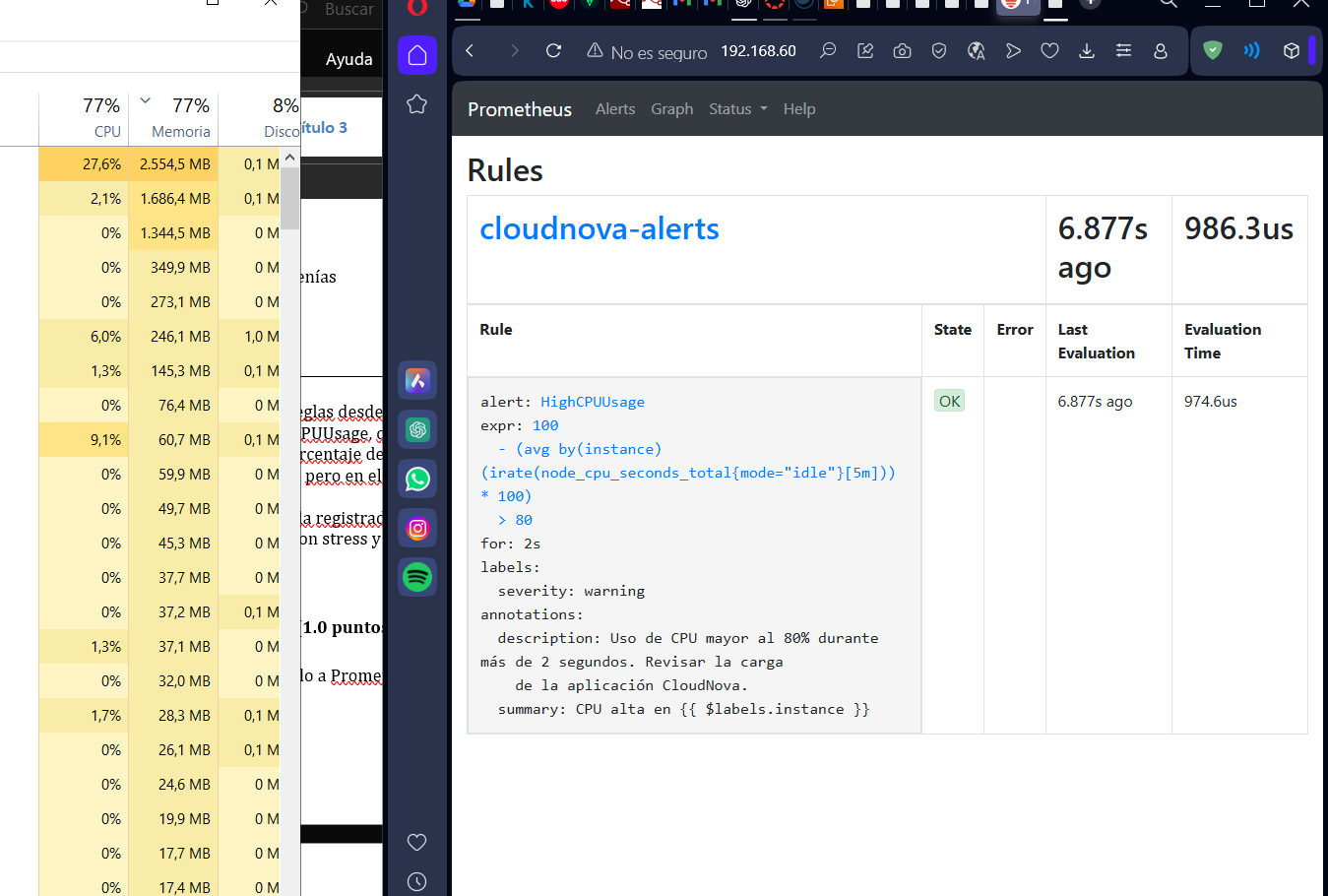
sudo apt update

sudo apt install -y stress # si no lo tenías

# Usar 2 CPUs durante 20 segundos

stress --cpu 2 --timeout 20

Configuré Prometheus para cargar reglas desde /etc/prometheus/alerts.yml.  
En este archivo definí la alerta HighCPUUsage, que usa la métrica node\_cpu\_seconds\_total de Node Exporter para calcular el porcentaje de CPU.  
Para la demo la configuré con for: 3s, pero en el informe la planteo con 2 minutos, que sería más razonable en producción.  
Aquí en la pestaña **Rules** se ve la regla registrada, y en la pestaña **Alerts** se ve cómo pasa a estado **Firing** cuando genero carga con stress y vuelve a **Inactive** cuando la CPU baja.



# Visualización con Grafana (1.0 puntos)

* Instale Grafana y conéctelo a Prometheus como fuente de datos.
* sudo apt update && sudo apt upgrade -y
* sudo apt install -y gnupg2 curl software-properties-common wget
* sudo mkdir -p /etc/apt/keyrings/

wget -q -O - https://apt.grafana.com/gpg.key | \

gpg --dearmor | sudo tee /etc/apt/keyrings/grafana.gpg > /dev/null

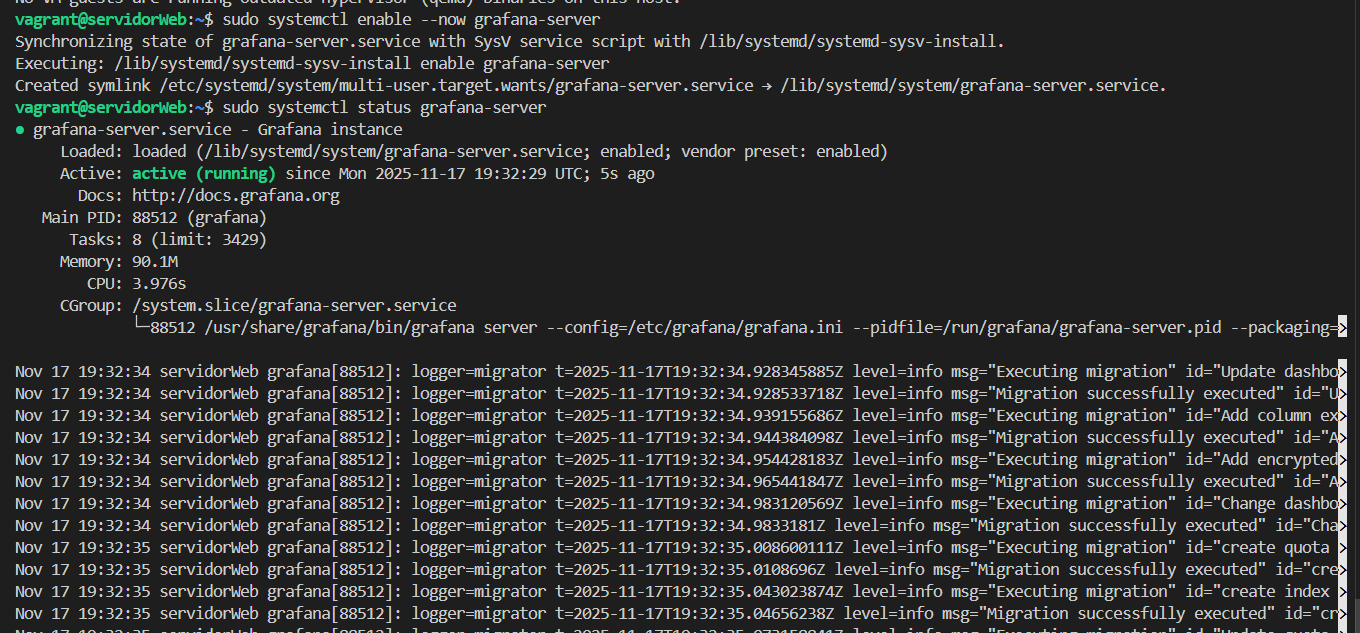
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable main" | \

sudo tee /etc/apt/sources.list.d/grafana.list

* sudo apt update
* sudo apt install -y grafana

Habilitar y arrancar el servicio

* sudo systemctl enable --now grafana-server
* sudo systemctl status grafana-server

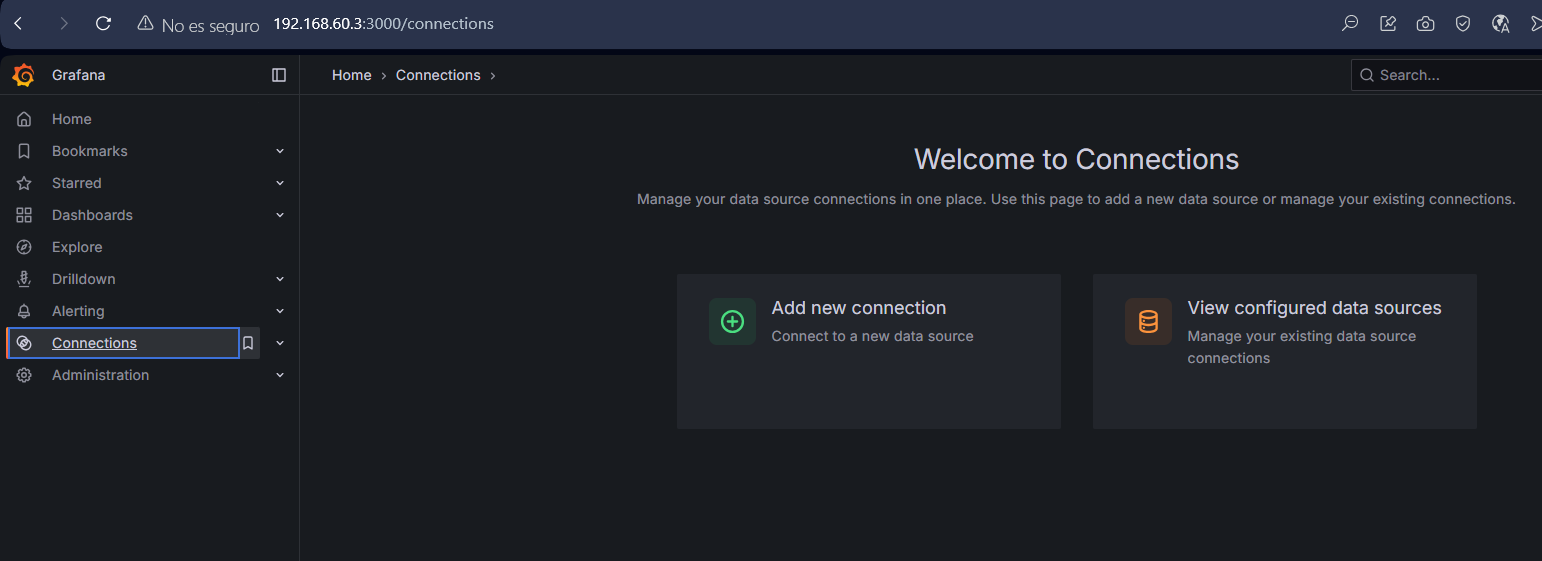


Entrar a

<http://192.168.60.3:3000/>

admin:admin

CONECTAR CON PROMETHEUS



Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

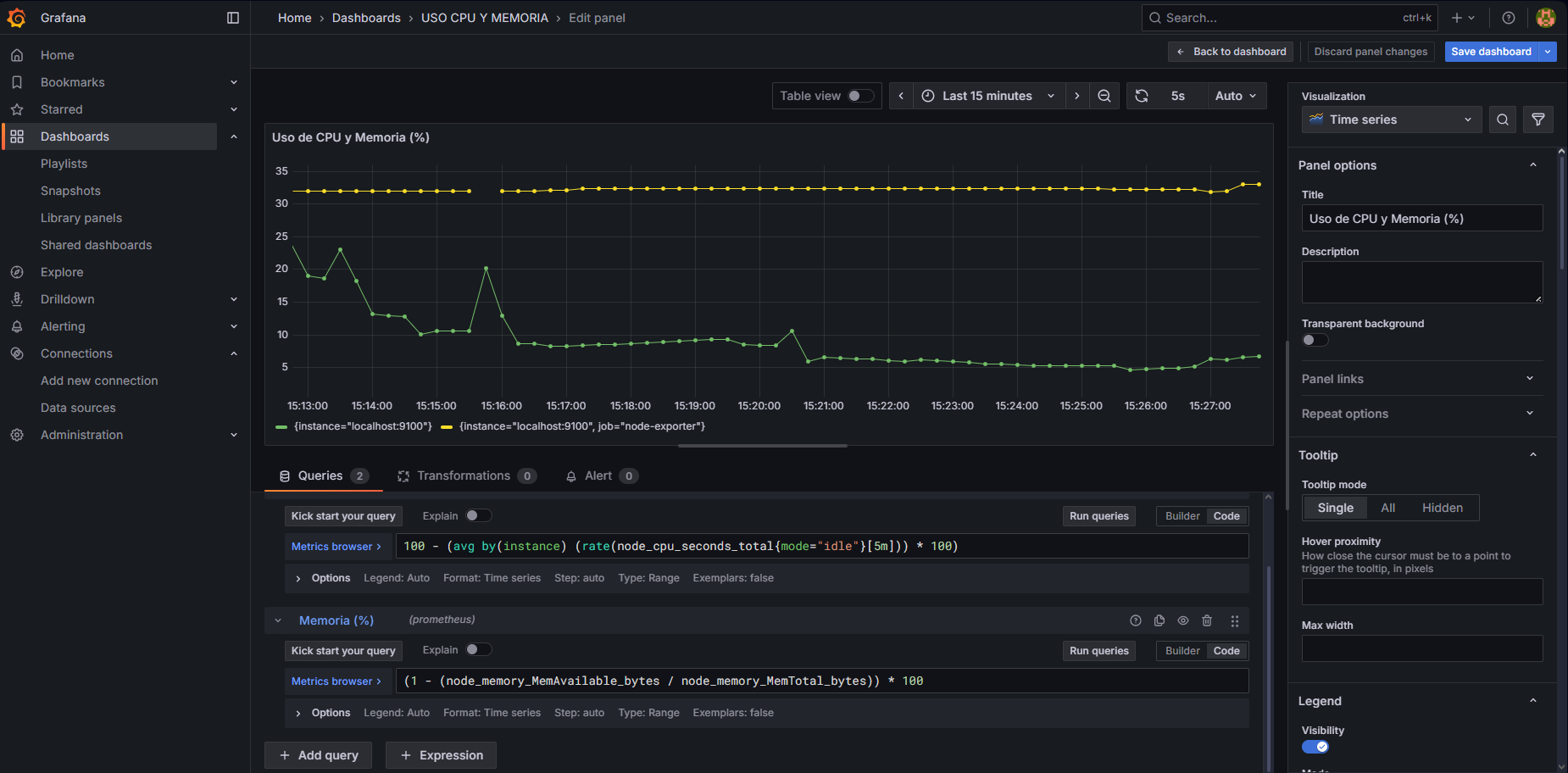
El contenido generado por IA puede ser incorrecto. SAVE AND TEST

Interfaz de usuario gráfica

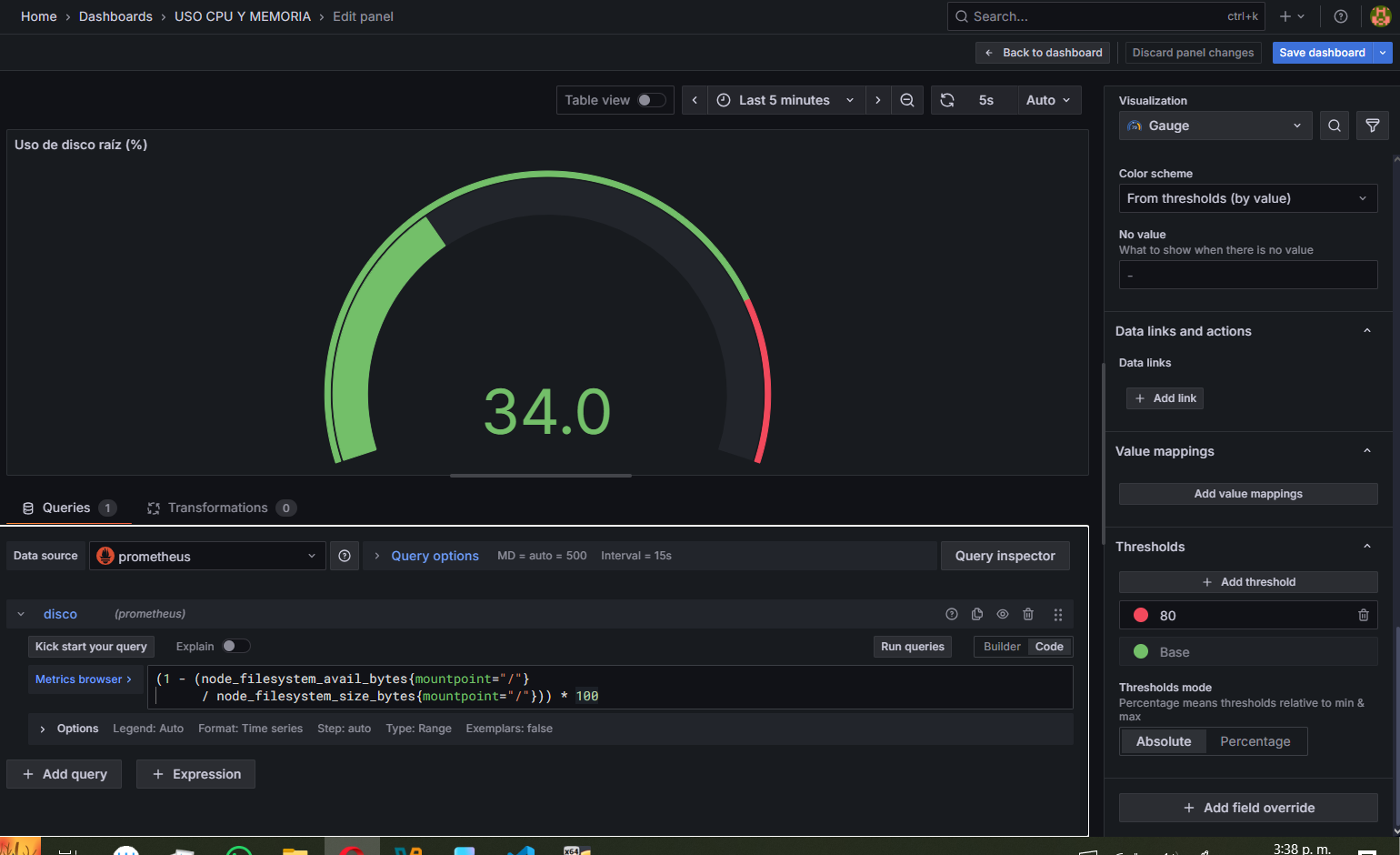
El contenido generado por IA puede ser incorrecto.

* Cree un dashboard con al menos dos paneles: uno de uso de CPU/memoria y un gauge de espacio en disco.

Panel 1: cpu y memoria



Panel 2





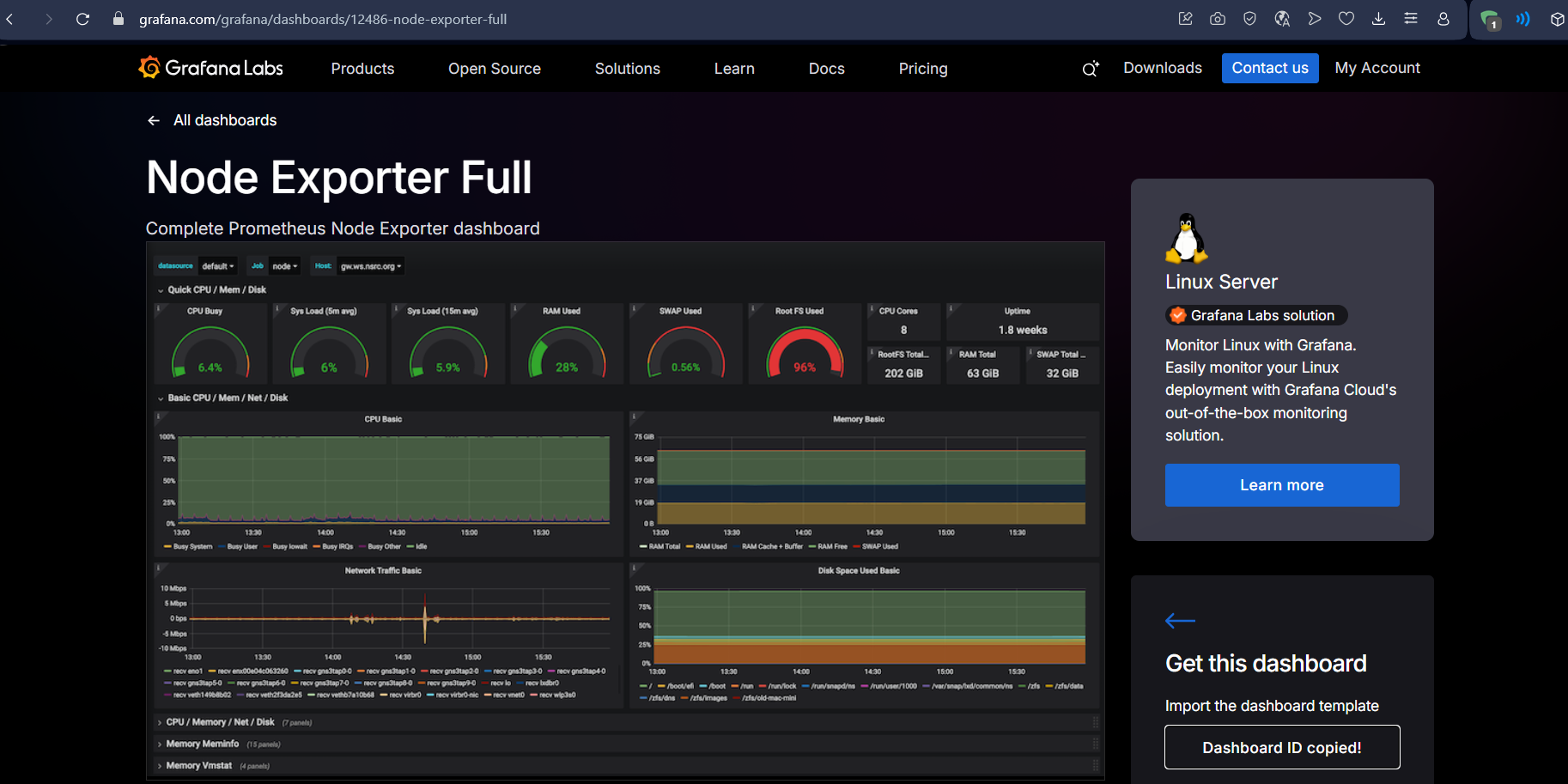


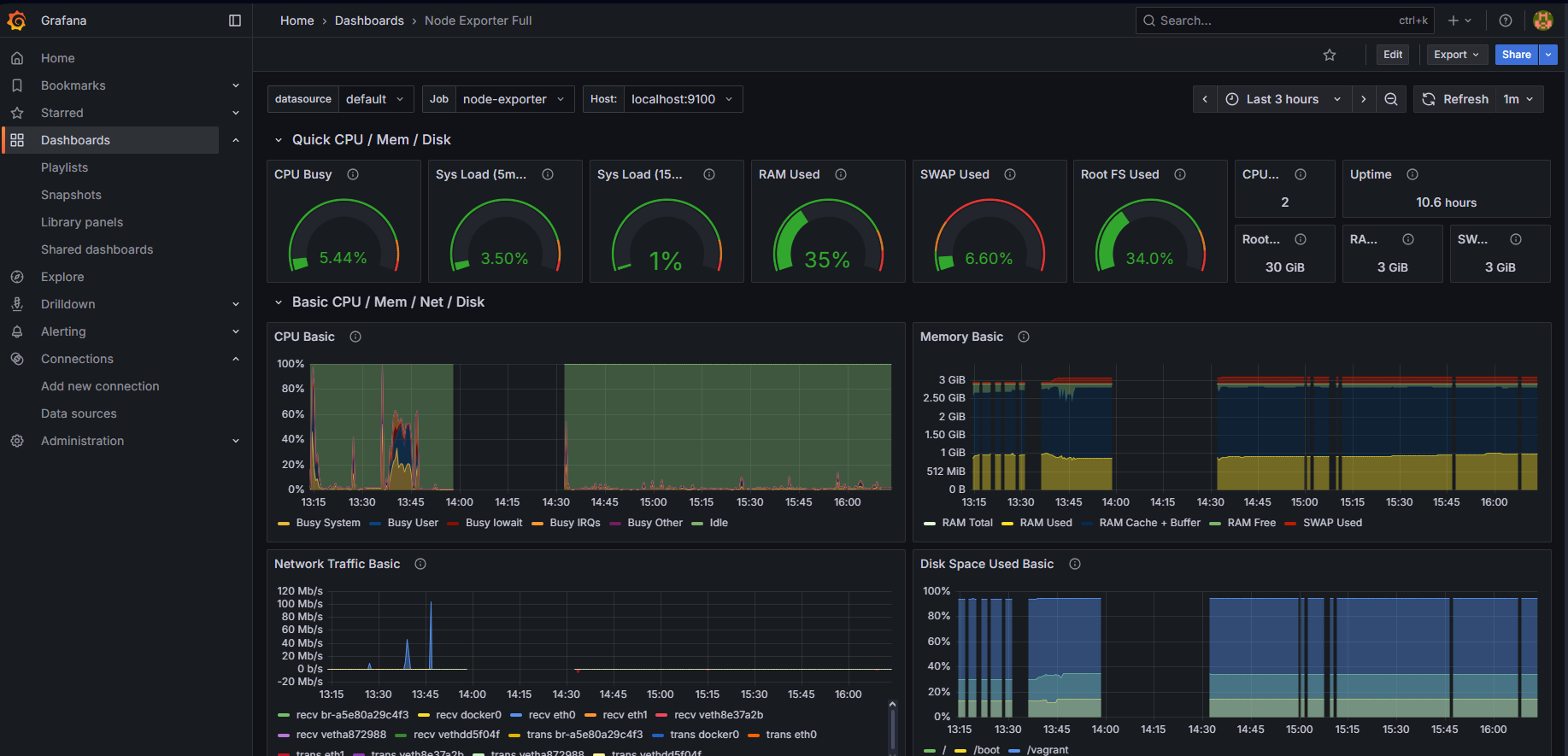
* Importe un panel preconfigurado desde la biblioteca oficial de Grafana y verifique su funcionamiento.

Paneles seleccionados

<https://grafana.com/grafana/dashboards/12486-node-exporter-full>

<https://grafana.com/grafana/dashboards/6014-host-stats-0-16-0>





#### **Recurso sugerido**

[Guía para integrar Grafana con Prometheus](https://grafana.com/docs/grafana/latest/getting-started/get-started-grafana-prometheus/)

# Entrega de Resultados

Cree un repositorio público en GitHub con todos los archivos de configuración, scripts, dashboards y un README.md explicativo. Incluya evidencias del despliegue (capturas o video corto). Comparta el enlace del repositorio en el sitio del curso para su evaluación.

Conclusión Técnica

Responda brevemente en su README.md:  
• ¿Qué aprendió al integrar Docker, AWS y Prometheus?  
• ¿Qué fue lo más desafiante y cómo lo resolvería en un entorno real?  
• ¿Qué beneficio aporta la observabilidad en el ciclo DevOps?

**Evaluación**

|  |  |  |
| --- | --- | --- |
| **Valor** | **Descripción** | **Puntaje Obtenido** |
| 1.5 | Aplicación en Docker |  |
| 1.0 | Despliegue en AWS |  |
| 1.5 | Prometheus + Node Exporter |  |
| 1.0 | Grafana + Prometheus |  |

Para guardar todo en github…

git clone <https://github.com/Neksun/Servicios-Telematicos-kp.git>

mkdir -p ~/Servicios-Telematicos-kp/mipracticas/Parcial3

cd ~/Servicios-Telematicos-kp/mipracticas/Parcial3

cp -r ~/docker-webapp ~/Servicios-Telematicos-kp/mipracticas/Parcial3/

cp -r ~/docker-apache-ssl ~/Servicios-Telematicos-kp/mipracticas/Parcial3/

cp -r ~/webapp ~/Servicios-Telematicos-kp/mipracticas/Parcial3/

mkdir -p ~/Servicios-Telematicos-kp/mipracticas/Parcial3/prometheus

sudo cp /etc/prometheus/prometheus.yml ~/Servicios-Telematicos-kp/mipracticas/Parcial3/prometheus/

sudo cp /etc/prometheus/alerts.yml ~/Servicios-Telematicos-kp/mipracticas/Parcial3/prometheus/

mkdir -p ~/Servicios-Telematicos-kp/mipracticas/Parcial3/grafana

cp \*.json ~/Servicios-Telematicos-kp/mipracticas/Parcial3/grafana/ 2>/dev/null || true

ls ~/Servicios-Telematicos-kp/mipracticas/Parcial3

cd ~/Servicios-Telematicos-kp/mipracticas/Parcial3

git add .

git commit -m "Añadiendo todos los archivos del Parcial 3 (Docker, Apache SSL, Prometheus, Grafana)"

git push origin main

neksun: ghp\_DYVtWgrUKT6maPrTYP6RpCpBKdCe1Y477TeE

mipracticas/Parcial3

<https://github.com/Neksun/Servicios-Telematicos-kp/tree/main/mipracticas/Parcial3>