

Práctica: Monitoreo de un clúster AWS EMR con Prometheus y Grafana

Índice

Instrucciones:	3
Parte 1: Configuración del clúster AWS EMR (3,00 puntos)	3
Parte 2: Configuración de JMX Exporter (3,00 puntos)	5
Parte 3: Despliegue de Prometheus y Grafana (2,00 puntos)	7
Parte 4: Visualización de métricas en Grafana (2,00 puntos)	14
Preguntas para reflexión	16

Instrucciones:

Parte 1: Configuración del clúster AWS EMR (3,00 puntos)

1. Crear un clúster EMR:

- Accede a la consola de AWS y crea un clúster EMR con las siguientes características:

- Nombre del clúster: EMR-Monitoring-iabdXX.

EMR-Monitoring-iabd12

- Aplicaciones: Hadoop, Spark y Hive.

Aplicaciones instaladas
Hadoop 3.2.1, Hive 3.1.2, Spark 3.2.0

- El resto de las configuraciones, las mismas que las adoptadas en otras prácticas sobre AWS EMR

- Número de instancias: 1 nodo maestro y 2 nodos core.

Capacidad
1 Primary (Principal) | 2 Principal | 0 Tarea

- Clave SSH: ls por defecto en el laboratorio.

The screenshot displays the AWS Management Console interface for an EMR cluster. At the top, a green notification bar states: "El clúster 'EMR-Monitoring-iabd12' se ha creado correctamente." Below this, the cluster name "EMR-Monitoring-iabd12" is prominently displayed. The console is divided into several sections: "Resumen" (Summary) on the left, "Aplicaciones" (Applications) in the middle, and "Administración de clústeres" (Cluster Management) on the right. The "Resumen" section includes details like the cluster ID, ARN, and configuration. The "Aplicaciones" section lists installed applications: Hadoop 3.2.1, Hive 3.1.2, and Spark 3.2.0. The "Administración de clústeres" section shows the S3 destination for logs and the public DNS of the master node. Below these, a horizontal navigation bar allows switching between various tabs: "Propiedades", "Acciones de arranque", "Instancias (hardware)", "Pasos", "Aplicaciones", "Configuraciones", "Monitorización", "Eventos", and "Etiquetas (0)". The "Propiedades" tab is active, showing "Registros de clúster" (Cluster Logs) and "Terminación del clúster y reemplazo de nodos" (Cluster Termination and Node Replacement). The "Registros de clúster" section shows the S3 location for logs and the status of logging. The "Terminación del clúster y reemplazo de nodos" section shows the termination protection status and the replacement of nodes in a failed state. At the bottom, the "Red y seguridad" (Network and Security) section is visible, showing the VPC, subnets, and security groups configured for the cluster.

- Espera a que el clúster esté en estado **Waiting**.

Estado y hora

Estado
✓ Esperando

2. Conectar al nodo maestro:

- Usa SSH para conectarte al nodo maestro del clúster:

```
ssh -i /ruta/a/tu/archivo.pem hadoop@<ip-nodo-maestro>
```

```
C:\Users\IABD12\Downloads>ssh -i labsuser.pem hadoop@44.222.84.153
Last login: Fri Apr 11 15:23:54 2025 from 195.57.144.137

#_
~\_ #####_
NN \#####\
NN \###|
NN \#/ _--
NN V~' '--->
NNN /
NN _./ _./
_/_/ _/_/
_/_/ '

Amazon Linux 2

AL2 End of Life is 2026-06-30.

A newer version of Amazon Linux is available!

Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

13 package(s) needed for security, out of 15 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRRRRRRRRRRR
E::::::::::::::::::::E M::::::::M M::::::::M R::::::::::::R
EE::::::::EEEEEEEEEE E M::::::::M M::::::::M R::::RRRRRR::::R
E:::E EEEEE M::::::::M M::::::::M RR:::R R:::R
E:::E M::::::::M M::M M::M M::M R::R R:::R
E::::EEEEEEEEEE M::::M M::M M::M M::M R::RRRRRR::::R
E::::::::::::E M::::M M::M M::M M::M R:::::::::RR
E::::EEEEEEEEEE M::::M M::M M::M M::M R::RRRRRR::::R
E:::E M::::M M::M M::M M::M R::R R:::R
E:::E EEEEE M::::M MMM M::M M::M R::R R:::R
EE::::::::EEEEEEEEEE M::::M M::M M::M R::R R:::R
E::::::::::::E M::::M M::M M::M RR:::R R:::R
EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRR RRRRRR

[hadoop@ip-172-31-1-66 ~]$
```

Parte 2: Configuración de JMX Exporter (3,00 puntos)

1. Instalar JMX Exporter:

- Explicar o dar un razonamiento extenso de que es y para que sirve JMX en este entorno

¿Qué es JMX Exporter y para qué sirve?

JMX (Java Management Extensions) es una tecnología que permite monitorizar y gestionar aplicaciones Java. En un clúster EMR, muchos componentes (Hadoop, Spark, Hive) son aplicaciones Java que exponen métricas a través de JMX.

JMX Exporter es un agente que:

- Recoge métricas JMX de aplicaciones Java
- Las convierte al formato de Prometheus
- Expone un endpoint HTTP que Prometheus puede raspar (scrape)

Es esencial para:

- Monitorizar el rendimiento del clúster
- Detectar cuellos de botella
- Planificar capacidad
- Solucionar problemas

- el JMX Exporter en el nodo maestro:

```
wget
https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.16.1/jmx_prometheus_javaagent-0.16.1.jar
```

```
[hadoop@ip-172-31-1-66 ~]$ wget https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.16.1/jmx_prometheus_javaagent-0.16.1.jar
--2025-04-11 15:27:19-- https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.16.1/jmx_prometheus_javaagent-0.16.1.jar
Resolving repo1.maven.org (repo1.maven.org)... 146.75.32.209, 2a04:4e42:77::209
Connecting to repo1.maven.org (repo1.maven.org)|146.75.32.209|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 469645 (459K) [application/java-archive]
Saving to: 'jmx_prometheus_javaagent-0.16.1.jar'

100%[=====] 469,645 --.-K/s in 0.005s

2025-04-11 15:27:19 (84.8 MB/s) - 'jmx_prometheus_javaagent-0.16.1.jar' saved [469645/469645]

[hadoop@ip-172-31-1-66 ~]$
```

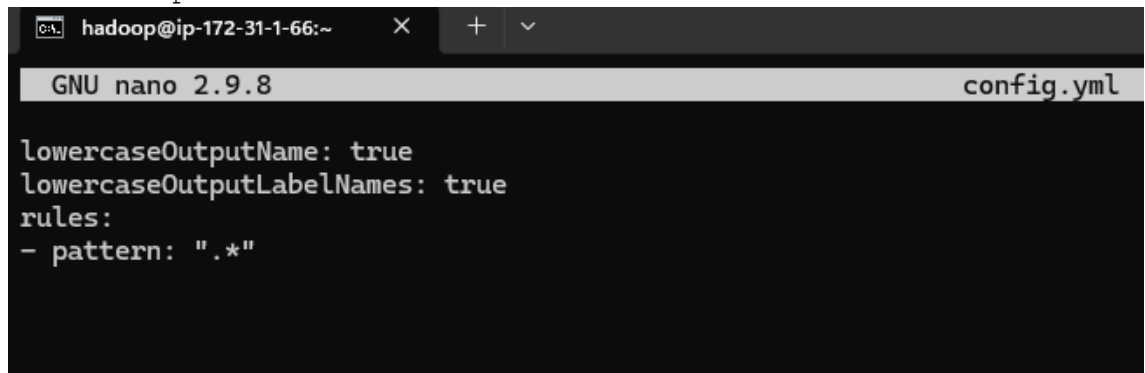
2. Crear el archivo de configuración:

- Crea un archivo config.yml para el JMX Exporter:

```
nano config.yml
```

- Agrega el siguiente contenido:

```
lowercaseOutputName: true
lowercaseOutputLabelNames: true
rules:
  - pattern: ".*"
```



```
hadoop@ip-172-31-1-66:~
GNU nano 2.9.8 config.yml

lowercaseOutputName: true
lowercaseOutputLabelNames: true
rules:
- pattern: ".*"
```

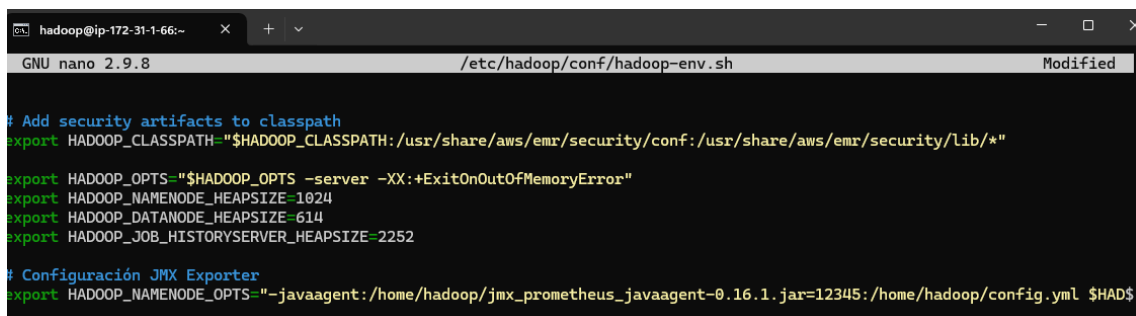
3. Configurar el NameNode para usar JMX Exporter:

- Edita el archivo de configuración del NameNode:

```
sudo nano /etc/hadoop/conf/hadoop-env.sh
```

- Agrega la siguiente línea:

```
export HADOOP_NAMENODE_OPTS="-
javaagent:/home/hadoop/jmx_prometheus_javaagent-
0.16.1.jar=12345:/home/hadoop/config.yml $HADOOP_NAMENODE_OPTS"
```



```
hadoop@ip-172-31-1-66:~
GNU nano 2.9.8 /etc/hadoop/conf/hadoop-env.sh Modified

# Add security artifacts to classpath
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/share/aws/emr/security/conf:/usr/share/aws/emr/security/lib/*"

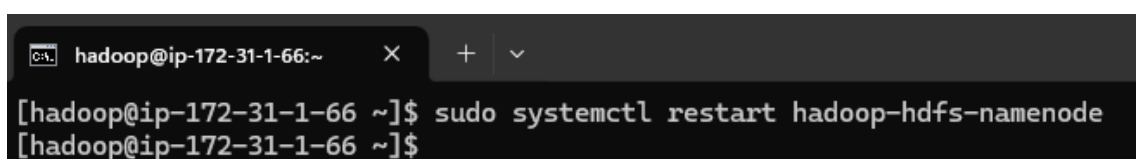
export HADOOP_OPTS="$HADOOP_OPTS -server -XX:+ExitOnOutOfMemoryError"
export HADOOP_NAMENODE_HEAPSIZE=1024
export HADOOP_DATANODE_HEAPSIZE=614
export HADOOP_JOB_HISTORYSERVER_HEAPSIZE=2252

# Configuración JMX Exporter
export HADOOP_NAMENODE_OPTS="-javaagent:/home/hadoop/jmx_prometheus_javaagent-0.16.1.jar=12345:/home/hadoop/config.yml $HAD$"
```

4. Reiniciar el NameNode:

- Reinicia el servicio del NameNode para aplicar los cambios:

```
sudo systemctl restart hadoop-hdfs-namenode
```

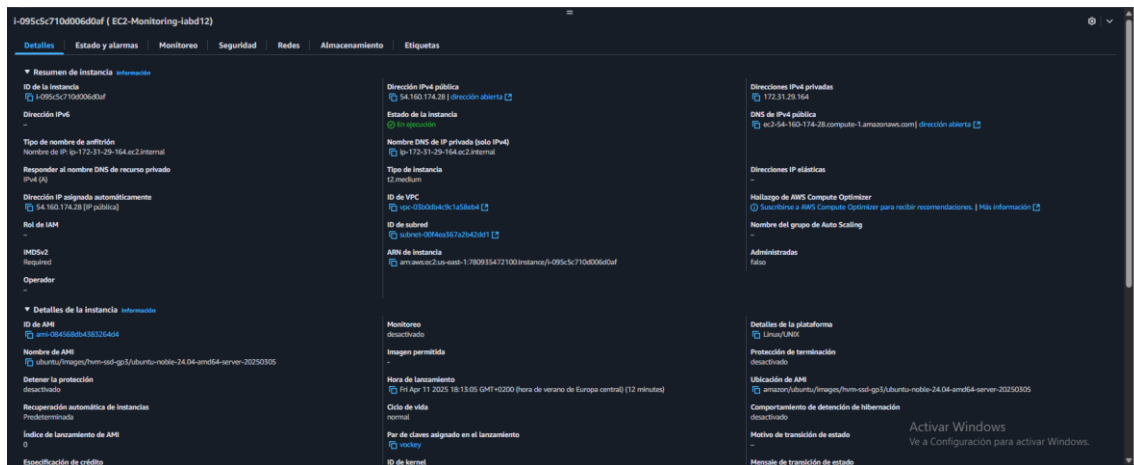


```
hadoop@ip-172-31-1-66:~
[hadoop@ip-172-31-1-66 ~]$ sudo systemctl restart hadoop-hdfs-namenode
[hadoop@ip-172-31-1-66 ~]$
```

Parte 3: Despliegue de Prometheus y Grafana (2,00 puntos)

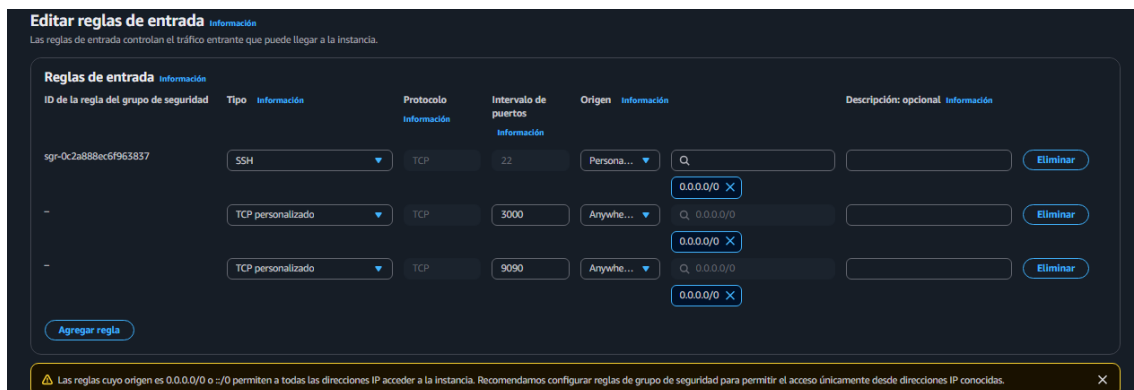
1. Crear una instancia EC2 para Prometheus y Grafana:

- Crea una instancia EC2 con Ubuntu en la misma VPC que el clúster EMR.



Permisos de red necesarios:

- SSH: en el puerto 22
- 3000: grafana
- 9090: prometheus



2. Instalar Prometheus:

- Conéctate a la instancia EC2 y sigue los pasos para instalar Prometheus:

```
C:\Users\IABD12\Downloads>ssh -i labsuser.pem ubuntu@18.207.126.141
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/pro

System information as of Fri Apr 11 16:34:21 UTC 2025

System load:  0.0               Processes:            119
Usage of /:   25.0% of 6.71GB   Users logged in:     0
Memory usage: 5%               IPv4 address for enX0: 172.31.25.118
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

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

Last login: Fri Apr 11 16:34:23 2025 from 195.57.144.137
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-25-118:~$
```

Instalamos prometheus mediante el siguiente comando.

```
wget https://github.com/prometheus/prometheus/releases/download/v2.30.3/prometheus-2.30.3.linux-amd64.tar.gz
tar -xzf prometheus-2.30.3.linux-amd64.tar.gz
cd prometheus-2.30.3.linux-amd64
./prometheus --config.file=prometheus.yml
```

```
ubuntu@ip-172-31-25-118:~$ wget https://github.com/prometheus/prometheus/releases/download/v2.30.3/prometheus-2.30.3.linux-amd64.tar.gz
--2025-04-11 16:35:17-- https://github.com/prometheus/prometheus/releases/download/v2.30.3/prometheus-2.30.3.linux-amd64.tar.gz
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/6838921/bc9e0970-09b3-4893-a66b-5fb918691ale?X-Amz-Date=20250411T163517Z&X-Amz-Expires=300&X-Amz-Signature=ead9aa589451466d1f6878fb8a84a08146isposition=attachment%3B%20filename%3Dprometheus-2.30.3.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream [following]
--2025-04-11 16:35:17-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/6838921/bc9e0970-09b3-4893-a66b-5production%2F20250411T163517Z&X-Amz-Date=20250411T163517Z&X-Amz-Expires=300&X-Amz-Signature=ead9aa589451466d1f6878fb8a84a08146isposition=attachment%3B%20filename%3Dprometheus-2.30.3.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream [following]
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.109.133, 185.199.110.133, 185.199.111.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 72638078 (69M) [application/octet-stream]
Saving to: 'prometheus-2.30.3.linux-amd64.tar.gz'

prometheus-2.30.3.linux-amd64.tar.gz 100%[=====]
2025-04-11 16:35:17 (112 MB/s) - 'prometheus-2.30.3.linux-amd64.tar.gz' saved [72638078/72638078]

ubuntu@ip-172-31-25-118:~$ tar -xzf prometheus-2.30.3.linux-amd64.tar.gz
ubuntu@ip-172-31-25-118:~$ cd prometheus-2.30.3.linux-amd64
ubuntu@ip-172-31-25-118:~/prometheus-2.30.3.linux-amd64$ |
```


3. Configurar Prometheus:

- Edita el archivo prometheus.yml para agregar el clúster EMR como objetivo:

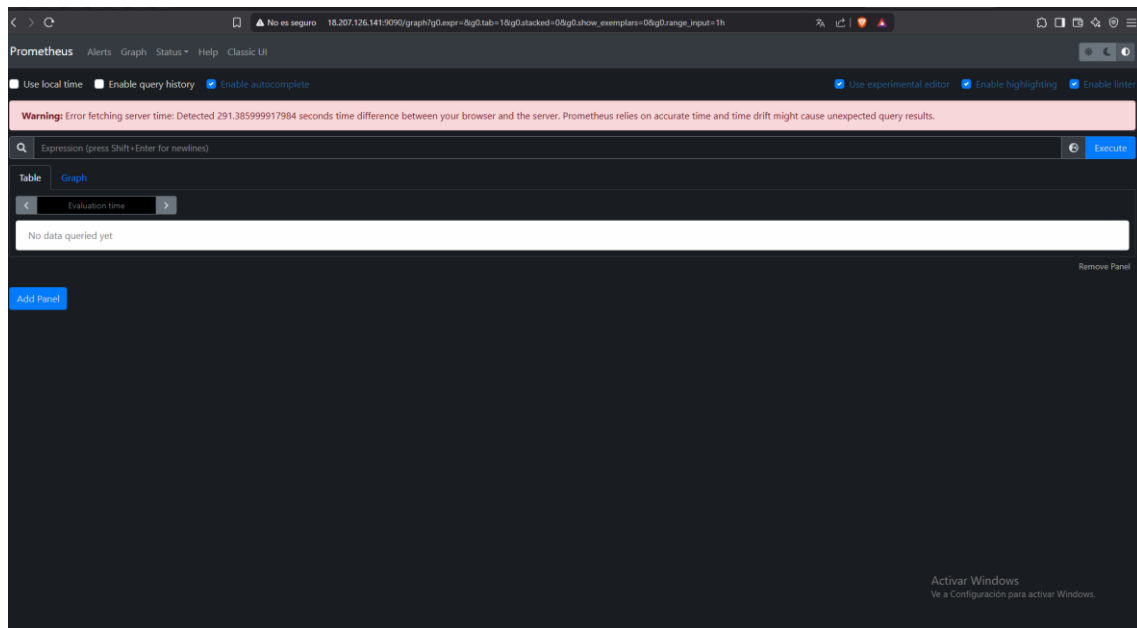
```
scrape_configs:  
  - job_name: 'emr-namenode'  
    static_configs:  
      - targets: ['<ip-nodo-maestro>:12345']
```

```
GNU nano 7.2 prometheus.yml  
# my global config  
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:  
        # - alertmanager:9093  
  
# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.  
rule_files:  
  # - "first_rules.yml"  
  # - "second_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: "emr-namenode"  
  
    # metrics_path defaults to '/metrics'  
    # scheme defaults to 'http'.  
  
    static_configs:  
      - targets: ["3.230.126.202:12345"]
```

Lanzamos el siguiente comando para comprobar que prometheus funcione:
./prometheus --config.file=prometheus.yml

```
ubuntu@ip-172-31-25-118:~/prometheus-2.30.3.linux-amd64$ ./prometheus --config.file=prometheus.yml &  
[1] 1309  
ubuntu@ip-172-31-25-118:~/prometheus-2.30.3.linux-amd64$ level=info ts=2025-04-11T16:37:15.300Z caller=main.go:438 msg="Starting Prometheus" version="(version=2.30.3, branch=HEAD, build_date=20250408-15:00:00, build_user=root@5cfff4, build_context=(go=go1.17.1, user=root@5cfff4, host_details=(Linux 6.8.0-1024-aws #26-Ubun)) fd_limits=(soft=1024, hard=1048576))"  
level=info ts=2025-04-11T16:37:15.300Z caller=main.go:443 build_context="(go=go1.17.1, user=root@5cfff4, build_context=(go=go1.17.1, user=root@5cfff4, host_details=(Linux 6.8.0-1024-aws #26-Ubun)) fd_limits=(soft=1024, hard=1048576))"  
level=info ts=2025-04-11T16:37:15.300Z caller=main.go:444 host_details="(Linux 6.8.0-1024-aws #26-Ubun)"  
level=info ts=2025-04-11T16:37:15.300Z caller=main.go:445 fd_limits="(soft=1024, hard=1048576)"
```

Y podemos comprobar que prometheus carga



4. Instalar Grafana:

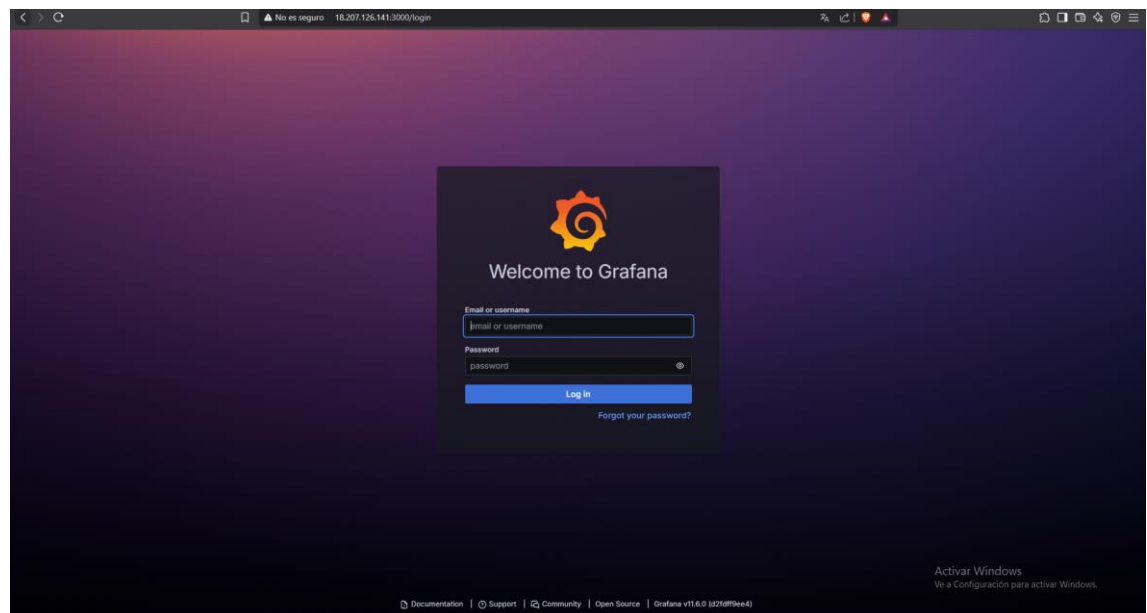
- Instala Grafana en la misma instancia EC2:

```
sudo apt-get install -y apt-transport-https
sudo apt-get install -y software-properties-common wget
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a
/etc/apt/sources.list.d/grafana.list
sudo apt-get update
sudo apt-get install grafana
sudo systemctl start grafana-server
sudo systemctl enable grafana-server
```

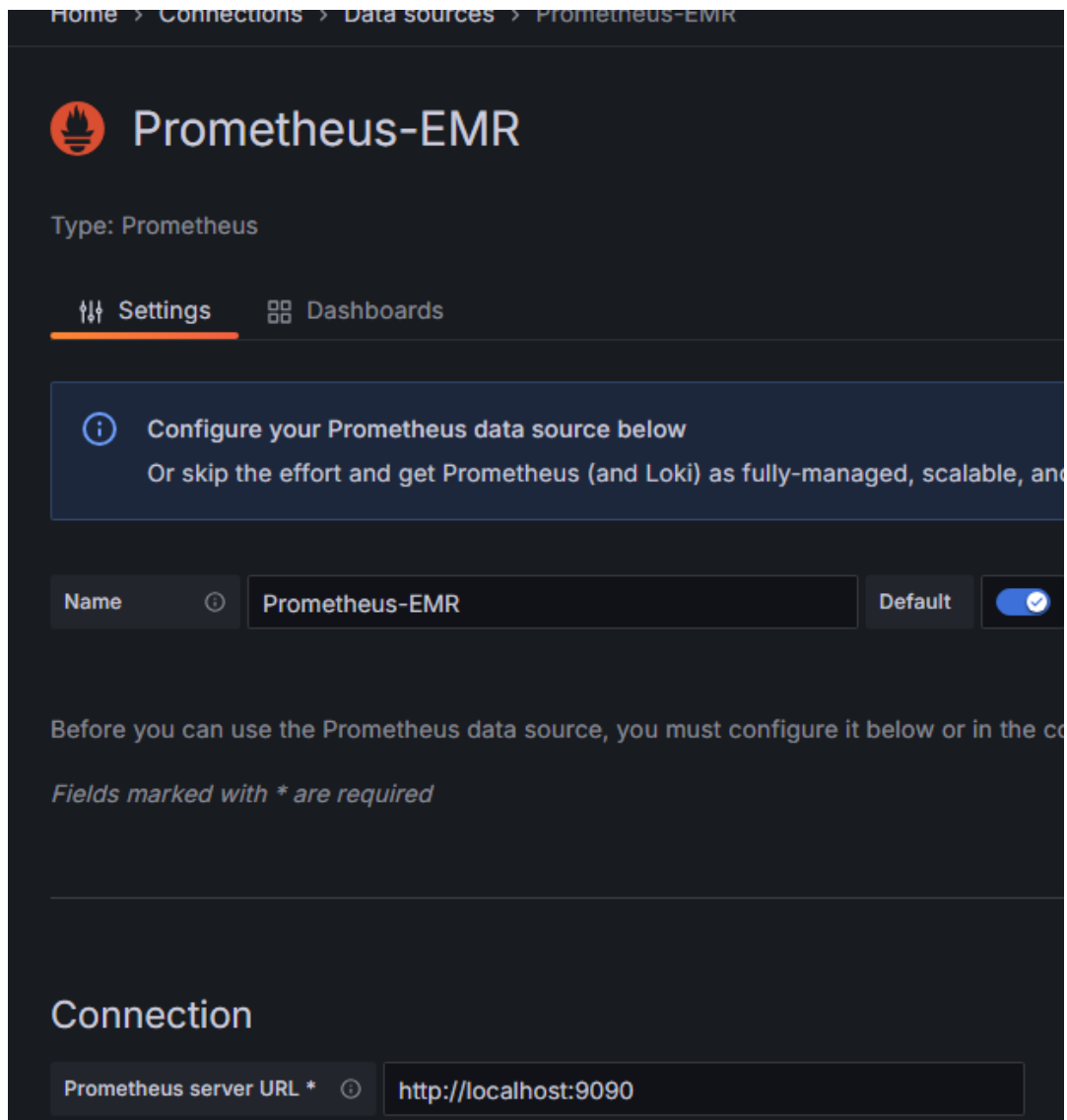
```
ubuntu@ip-172-31-25-118:~/prometheus-2.30.3.linux-amd64$ sudo apt-get update
sudo apt-get install -y apt-transport-https software-properties-common wget
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
sudo apt-get update
sudo apt-get install grafana
sudo systemctl start grafana-server
sudo systemctl enable grafana-server
```

5. Configurar Grafana:

- Accede a Grafana en `http://<ip-instancia-ec2>:3000`.

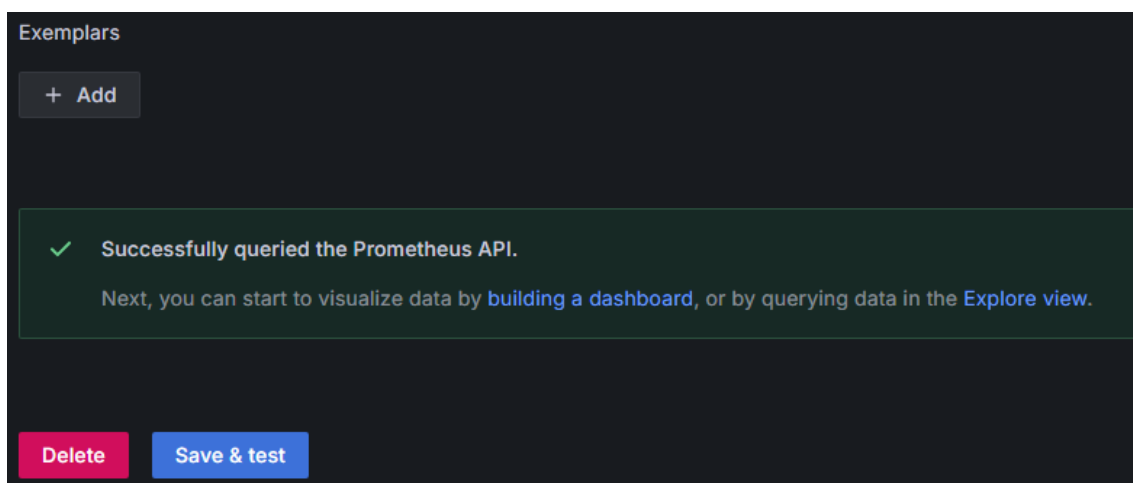


- Agrega Prometheus como fuente de datos:
 - URL: `http://localhost:9090`.



The screenshot shows the 'Prometheus-EMR' configuration page in Grafana. The breadcrumb trail at the top is 'Home > Connections > Data sources > Prometheus-EMR'. The page title is 'Prometheus-EMR' with a flame icon. Below the title, it says 'Type: Prometheus'. There are two tabs: 'Settings' (active) and 'Dashboards'. A blue information box contains the text: 'Configure your Prometheus data source below' and 'Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and...'. Below this, there is a form with a 'Name' field containing 'Prometheus-EMR', a 'Default' toggle switch which is turned on, and a 'Prometheus server URL' field containing 'http://localhost:9090'. A note at the bottom of the form says 'Fields marked with * are required'.

- Comprobamos que funcione la conexión de grafana a prometheus.



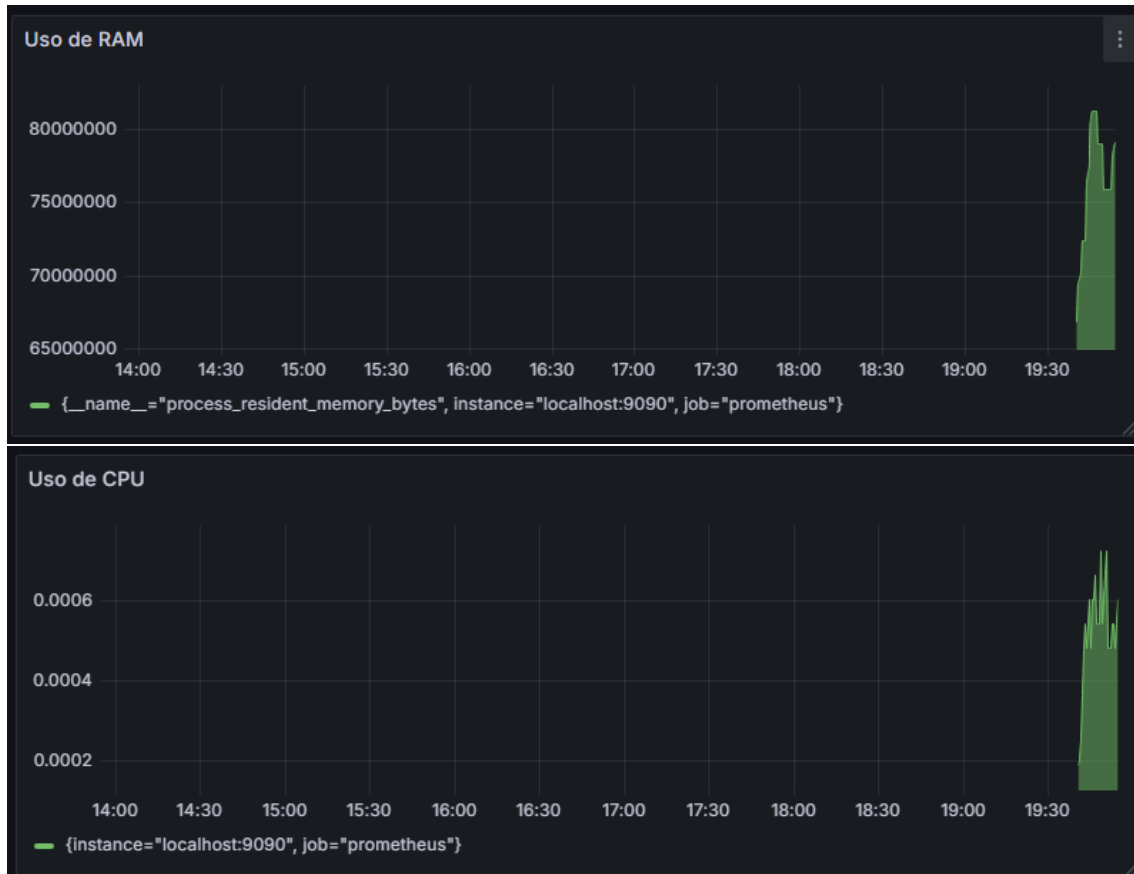
The screenshot shows the 'Exemplars' panel in Grafana. At the top, there is a '+ Add' button. Below it, a green success message reads: 'Successfully queried the Prometheus API.' followed by 'Next, you can start to visualize data by [building a dashboard](#), or by querying data in the [Explore view](#).' At the bottom, there are two buttons: 'Delete' (pink) and 'Save & test' (blue).

Parte 4: Visualización de métricas en Grafana (2,00 puntos)

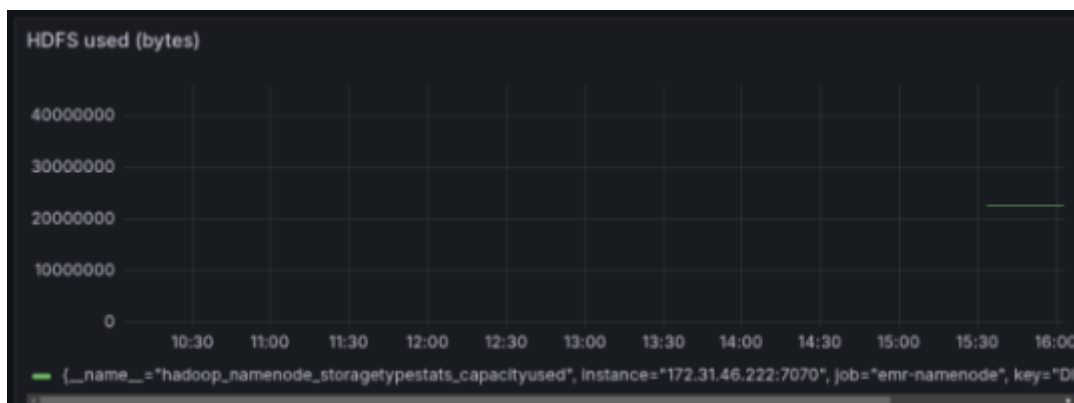
(PROBLEMA CON LA HORA)

1. Crear un dashboard en Grafana:

- Crea un nuevo dashboard y agrega paneles para monitorear métricas como:
 - Uso de CPU y RAM.



- Espacio utilizado en HDFS.



- Estado del NameNode.



2. Explorar métricas:

- Usa las métricas expuestas por JMX Exporter para crear gráficos en Grafana.

(Prometheus dejó de funcionar, trate de arreglarlo. Como podrás comprobar, no pude)

Preguntas para reflexión

¿Qué métricas consideras más importantes para monitorear en un clúster EMR? ¿Por qué?

Las métricas clave para monitorear en un clúster Amazon EMR (Elastic MapReduce) dependen del tipo de carga de trabajo (Spark, Hadoop, Hive, etc.), pero en general, las más importantes son:

- Métricas de recursos del cluster:
 - Uso de CPU y memoria: Para evitar cuellos de botella y garantizar que los nodos no estén sobresaturados.
 - Uso de disco y E/S: Importante para trabajos con alto volumen de datos (ej. Spark shuffles, HDFS).
 - Tráfico de red: Ayuda a detectar problemas de comunicación entre nodos (ej. NameNode y DataNodes).
- Métricas de YARN (si se usa Hadoop/Spark):
 - Contenedores asignados vs. disponibles: Para optimizar la utilización de recursos.
 - Tiempo de espera de aplicaciones (pending apps): Indica si el cluster está sobrecargado.
- Métricas de HDFS (si se usa Hadoop):
 - Espacio utilizado en HDFS: Evitar que se llene el almacenamiento.
 - Bloques corruptos o faltantes: Crítico para la integridad de los datos.
- Métricas específicas de Spark (si se usa):
 - Tiempo de ejecución de stages/tasks: Identificar etapas lentas.
 - Shuffle spill (lectura/escritura en disco): Puede indicar ineficiencias en operaciones de shuffle.
 - Número de ejecutores activos vs. inactivos.

¿Por qué?

Estas métricas permiten:

- Evitar costos innecesarios (ej. sobreaprovisionamiento).
- Detectar cuellos de botella antes de que afecten el rendimiento.
- Garantizar alta disponibilidad (ej. discos llenos, nodos inalcanzables).

2. ¿Cómo podrías mejorar la configuración de JMX Exporter para recopilar métricas más específicas?

JMX Exporter es una herramienta que extrae métricas JVM de aplicaciones Java (como Spark o HBase en EMR). Para mejorar su configuración:

1. Filtrar métricas relevantes:
 - En el archivo de configuración (config.yml), definir whitelist o blacklist para recoger solo métricas útiles (ej. métricas de GC, memoria heap, threads).

```
rules:
```



```
- pattern:
'java.lang<type=Memory><>(HeapMemoryUsage|NonHeapMemoryUsage) '
name: "jvm_memory_usage_$1"
```

- Evitar recoger todas las métricas JMX (puede generar demasiado ruido).
2. Etiquetado personalizado (labels):
- Añadir etiquetas como cluster_id, application_name o instance para facilitar el filtrado en Prometheus.

```
labels:
cluster: "emr-prod"
component: "spark"
```

3. Ajustar el intervalo de scraping:
- Configurar un balance entre granularidad y carga (ej. cada 15-30s).
4. Habilitar métricas específicas del framework:
- Para Spark, recoger métricas como spark_stage_failed_tasks o spark_driver_blockManager_memory_used.

3. ¿Qué ventajas tiene usar Prometheus y Grafana frente a otras herramientas de monitoreo?

- Prometheus:
 - ✓ Almacenamiento eficiente (TSDB optimizado para métricas temporales).
 - ✓ Lenguaje de consulta potente (PromQL): Permite consultas complejas y alertas.
 - ✓ Integración nativa con Kubernetes y EMR (service discovery).
 - ✓ Modelo pull (más escalable que push en algunos casos).
- Grafana:
 - ✓ Visualización avanzada: Dashboards interactivos y personalizables.
 - ✓ Soporte múltiples fuentes de datos (Prometheus, CloudWatch, etc.).
 - ✓ Alertas integradas (junto con Prometheus).

Ventajas vs. otras herramientas:

- CloudWatch:
 - Prometheus es más económico (no costos por métrica custom).
 - Mayor flexibilidad en consultas (PromQL vs. filtros básicos en CloudWatch).
- Datadog/New Relic:
 - Prometheus + Grafana es open-source (sin costos de licencia).
 - Mejor para métricas de infraestructura (no solo APM).
- Elastic Stack (ELK):
 - Prometheus está optimizado para métricas (no logs), con menor overhead.

Conclusión: La combinación Prometheus + Grafana es ideal para EMR por su flexibilidad, costo y capacidades de análisis en tiempo real.