

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

ÍNDICE

Objetivo de la práctica	3
Instrucciones:.....	3
Parte 1: Configuración del clúster AWS EMR	3
Parte 2: Configuración de JMX Exporter	5
Parte 3: Despliegue de Prometheus y Grafana	8
Parte 4: Visualización de métricas en Grafana	12
Preguntas para reflexión	14
Github con los archivos de configuración.....	16

Objetivo de la práctica

El objetivo de esta práctica es:

1. Configurar un clúster **AWS EMR**.
2. Exponer métricas del clúster utilizando **JMX Exporter**.
3. Recopilar y almacenar métricas utilizando **Prometheus**.
4. Visualizar las métricas en **Grafana**.

Al finalizar la práctica, habrás creado un sistema de monitoreo completo para un clúster EMR, utilizando herramientas de código abierto.

Instrucciones:

Parte 1: Configuración del clúster AWS EMR

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-iabd07.
 - Aplicaciones: Hadoop, Spark y Hive.
 - 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.
 - Clave SSH: ls por defecto en el laboratorio.

Iniciamos un laboratorio en AWS y vamos a EMR para crear el cluster

The screenshot displays the AWS Management Console interface for an Amazon EMR cluster. At the top, a green notification bar states: "El clúster 'EMR-Monitoring-iabd07' se ha creado correctamente." Below this, the cluster name "EMR-Monitoring-iabd07-" is shown, along with a refresh icon and a timestamp "Se ha actualizado hace menos de un minuto". Action buttons include "Terminar", "Clonar en AWS CLI", and "Clonar".

The main content area is divided into four columns:

- Resumen:** Includes "Información del clúster" with ID "j-157PIF1PNTCD2", "ARN del clúster" "arn:aws:elasticmapreduce:us-east-1:5003181884:69:cluster/j-157PIF1PNTCD2", "Configuración del clúster" with "Grupos de instancias", and "Capacidad" showing "1 Primary (Principal) | 2 Principal | 0 Tarea".
- Aplicaciones:** Shows "Versión de Amazon EMR" as "emr-6.6.0" and "Aplicaciones instaladas" as "Hadoop 3.2.1, Hive 3.1.2, Hue 4.10.0, Spark 3.2.0, Zeppelin 0.10.0".
- Administración de clústeres:** Includes "Destino del registro en Amazon S3" as "aws-emi-iabd07", "DNS público del nodo principal" as "ec2-34-239-104-228.compute-1.amazonaws.com", and links to "Conectarse al nodo principal mediante SSH" and "Conectarse al nodo principal mediante SSM".
- Estado y hora:** Shows "Estado" as "Comenzando", "Hora de creación" as "28 de marzo de 2025 16:25 (UTC+01:00)", and "Tiempo transcurrido" as "2 minutos, 26 segundos".

<div>Sistema operativo Información</div> <div>Versión de Amazon Linux 2.0.20250305.0</div>	<div>Registros de clúster Información</div> <div>Archivar los archivos de registro en Amazon S3 Activado</div> <div>Ubicación de Amazon S3 s3://aws-eme-lab077 🔗</div> <div>Cifrado para registros Desactivado</div>	<div>Terminación del clúster y reemplazo de nodos Información</div> <div>Editar</div> <div>Opción de terminación Terminar automáticamente el clúster después del tiempo de inactividad</div> <div>Protección contra la terminación Desactivado</div> <div>Tiempo de inactividad 4 horas</div> <div>Reemplazo de nodos en mal estado Activado</div>
<div>Red y seguridad Información</div> <div>Red</div> <div>Virtual Private Cloud (VPC) vpc-0a162a66603af98ef 🔗</div> <div>Subredes y zonas de disponibilidad (AZ) subnet-0218561b2da364b99 🔗 us-east-1b</div> <div>► Grupos de seguridad de EC2 (firewall)</div>	<div>Configuración de seguridad</div> <div>Configuración de seguridad Ninguna</div> <div>Par de claves de EC2 vockey</div>	<div>Permisos</div> <div>Rol de servicio para Amazon EMR LabRole 🔗</div> <div>Perfil de instancia EC2 EMR_EC2_DefaultRole</div> <div>Rol de escalamiento automático personalizado EMR_AutoScaling_DefaultRole 🔗</div>

El clúster "EMR-Monitoring-iab07-" se ha creado correctamente.

Notificaciones

EMR-Monitoring-iab07-

Se ha actualizado hace menos de un minuto

[Terminar](#)
[Clonar en AWS CLI](#)
[Clonar](#)

▼ Resumen

Información del clúster

ID del clúster
j-157PIF1PNTCD2

ARN del clúster
 arnaws:elasticmapreduce:us-east-1:5003181884:69:cluster/j-157PIF1PNTCD2

Configuración del clúster
Grupos de instancias

Capacidad
1 Primary (Principal) 2 Principal 0 Tarea

Aplicaciones

Versión de Amazon EMR
emr-6.6.0

Aplicaciones instaladas
Hadoop 3.2.1, Hive 3.1.2, Hue 4.10.0, Spark 3.2.0, Zeppelin 0.10.0

Administración de clústeres

Destino del registro en Amazon S3
[aws-emr-iab07](#)

IU de aplicación persistente
[Servidor de historial de Spark](#)

[Servidor de línea de tiempo de YARN](#)

[UI de Tez](#)

DNS público del nodo principal
 ec2-34-239-104-228.compute-1.amazonaws.com

[Conectarse al nodo principal mediante SSH](#)

[Conectarse al nodo principal mediante SSM](#)

Estado y hora

Estado
[Esperando](#)

Hora de creación
28 de marzo de 2025 16:25 (UTC+01:00)

Tiempo transcurrido
6 minutos, 58 segundos

TCP personalizado

TCP

12345

Anyw...

sg-0883c488dcb08f41d

0.0.0.0/0

Eliminar

TCP personalizado

TCP

7070

Anyw...

0.0.0.0/0

0.0.0.0/0

Eliminar

Agregar regla

- Usa SSH para conectarte al nodo maestro del clúster:

A continuación lanzamos en una terminal el siguiente comando dentro de C:\Users/IABD07:

```
PS C:\Users\IABD07> ssh -i labsuser.pem hadoop@ec2-54-196-138-53.compute-1.amazonaws.com
The authenticity of host 'ec2-54-196-138-53.compute-1.amazonaws.com (54.196.138.53)' can't be established.
ED25519 key fingerprint is SHA256:sinSrSGYsG4DxQNr/le9X2Qx1KPhjLec0Thv46FuE9w.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Parte 2: Configuración de JMX Exporter

1. Instalar JMX Exporter:

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

JMX es una tecnología de Java que proporciona una arquitectura para la gestión y supervisión de aplicaciones, sistemas y redes. JMX permite gestionar y monitorear aplicaciones Java en tiempo real.

El propósito de usar JMX en un entorno de monitoreo como Prometheus es que JMX proporciona una vía para acceder a las métricas internas de una aplicación Java. Por ejemplo, se pueden obtener métricas como el uso de memoria, la cantidad de hilos en ejecución, la tasa de solicitudes, la latencia de las operaciones, entre muchas otras. Estas métricas son fundamentales para el monitoreo y diagnóstico de sistemas, ya que permiten a los administradores detectar y solucionar problemas de rendimiento y eficiencia.

JMX Exporter es una herramienta que actúa como un puente entre JMX y Prometheus. Recoge las métricas expuestas por una aplicación Java a través de JMX y las expone en un formato que Prometheus puede recolectar y procesar.

- el JMX Exporter en el nodo maestro: usamos el siguiente comando:

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

```
[hadoop@ip-172-31-85-228 ~]$ wget https://repo1.maven.org/maven2/io/prometheus/jmx/jmx_prometheus_javaagent/0.16.1/jmx_prometheus_javaagent-0.16.1.jar
--2025-03-31 15:26:29-- 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 0.003s

2025-03-31 15:26:29 (133 MB/s) - 'jmx_prometheus_javaagent-0.16.1.jar' saved [469645/469645]
[hadoop@ip-172-31-85-228 ~]$ |
```

2. Crear el archivo de configuración:

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

```
nano config.yml
```

```
[hadoop@ip-172-31-85-228 ~]$ nano config.yml|
```

- Agrega el siguiente contenido:

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

```
hadoop@ip-172-31-85-228:~ x + v
GNU nano 2.9.8 config.yml
lowercaseOutputName: true
lowercaseOutputLabelNames: true
rules:
  - pattern: ".*"
```

3. Configurar el NameNode para usar JMX Exporter:

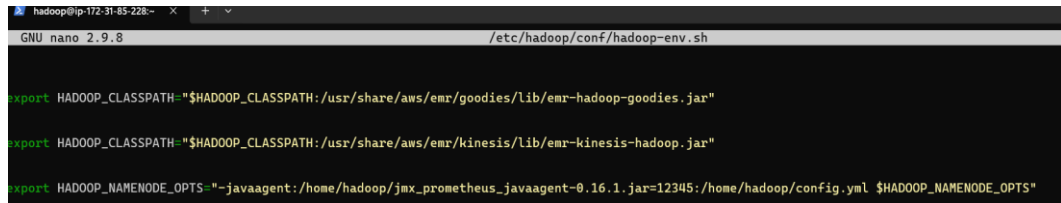
- Edita el archivo de configuración del NameNode:

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

```
[hadoop@ip-172-31-85-228 ~]$ 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"
```



```
GNU nano 2.9.8 /etc/hadoop/conf/hadoop-env.sh  
  
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/share/aws/emr/goodies/lib/emr-hadoop-goodies.jar"  
export HADOOP_CLASSPATH="$HADOOP_CLASSPATH:/usr/share/aws/emr/kinesis/lib/emr-kinesis-hadoop.jar"  
export HADOOP_NAMENODE_OPTS="-javaagent:/home/hadoop/jmx_prometheus_javaagent-0.16.1.jar=12345:/home/hadoop/config.yml $HADOOP_NAMENODE_OPTS"
```

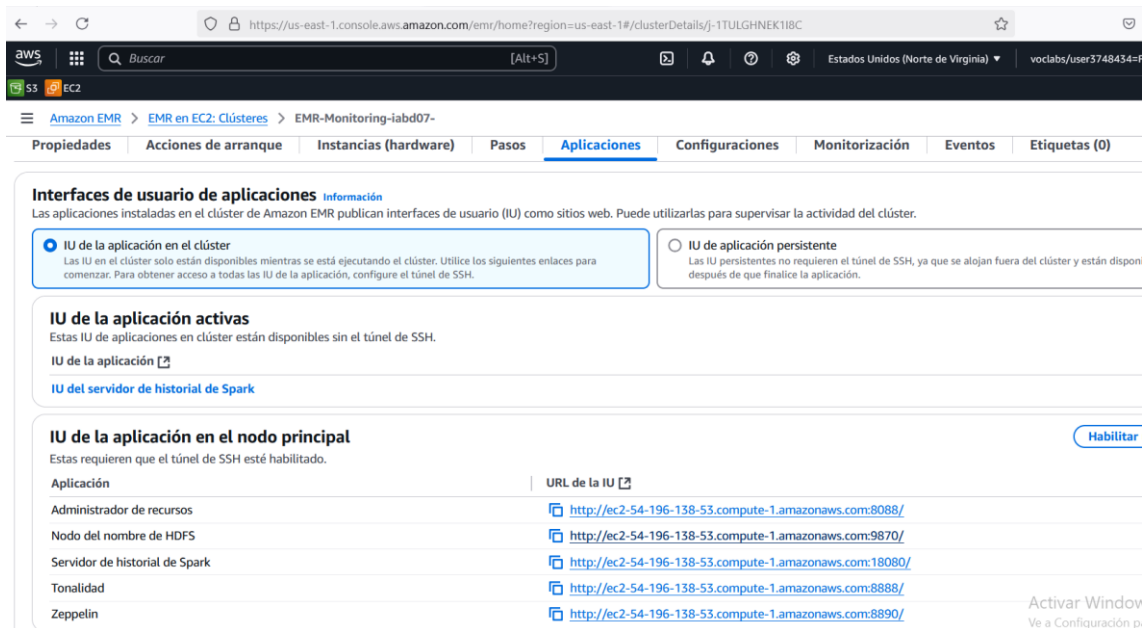
4. Reiniciar el NameNode:

- Reinicia el servicio del NameNode para aplicar los cambios:

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

```
[hadoop@ip-172-31-85-228 ~]$ sudo systemctl restart hadoop-hdfs-namenode  
[hadoop@ip-172-31-85-228 ~]$
```

En el cluster vamos a aplicaciones y al puerto 9870 que es del nodo del nombre de HDFS:



Interfases de usuario de aplicaciones [Información](#)

Las aplicaciones instaladas en el clúster de Amazon EMR publican interfaces de usuario (IU) como sitios web. Puede utilizarlas para supervisar la actividad del clúster.

☒ **IU de la aplicación en el clúster**
Las IU en el clúster solo están disponibles mientras se está ejecutando el clúster. Utilice los siguientes enlaces para comenzar. Para obtener acceso a todas las IU de la aplicación, configure el túnel de SSH.

☐ **IU de aplicación persistente**
Las IU persistentes no requieren el túnel de SSH, ya que se alojan fuera del clúster y están disponibles después de que finalice la aplicación.

IU de la aplicación activas
Estas IU de aplicaciones en clúster están disponibles sin el túnel de SSH.

IU de la aplicación [?](#)

[IU del servidor de historial de Spark](#)

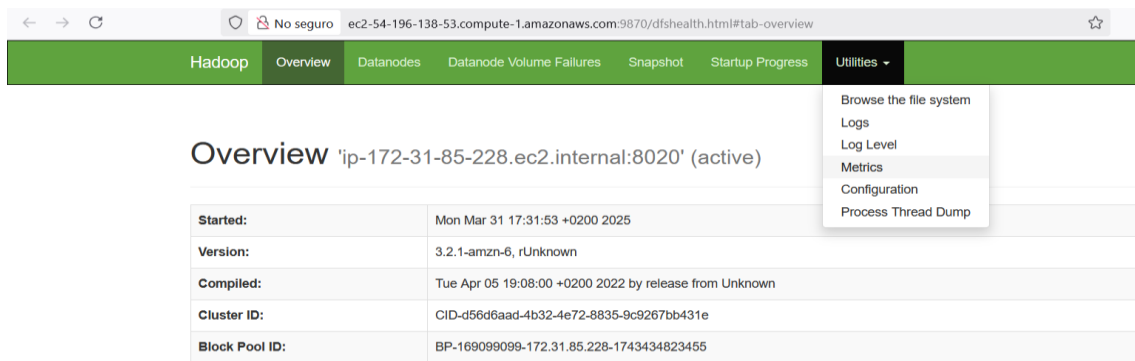
IU de la aplicación en el nodo principal [Habilitar](#)

Estas requieren que el túnel de SSH esté habilitado.

Aplicación	URL de la IU ?
Administrador de recursos	http://ec2-54-196-138-53.compute-1.amazonaws.com:8088/
Nodo del nombre de HDFS	http://ec2-54-196-138-53.compute-1.amazonaws.com:9870/
Servidor de historial de Spark	http://ec2-54-196-138-53.compute-1.amazonaws.com:18080/
Tonalidad	http://ec2-54-196-138-53.compute-1.amazonaws.com:8888/
Zeppelin	http://ec2-54-196-138-53.compute-1.amazonaws.com:8890/

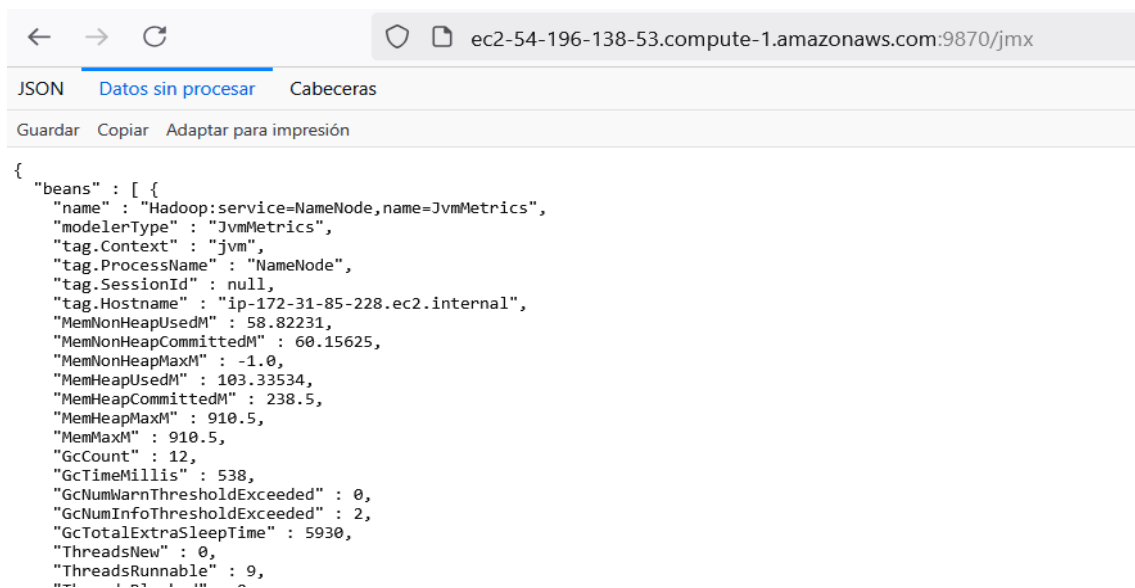
Activar Window
Ve a Configuración p.

Vamos a la pestaña de metrics



Overview 'ip-172-31-85-228.ec2.internal:8020' (active)

Started:	Mon Mar 31 17:31:53 +0200 2025
Version:	3.2.1-amzn-6, rUnknown
Compiled:	Tue Apr 05 19:08:00 +0200 2022 by release from Unknown
Cluster ID:	CID-d56d6aad-4b32-4e72-8835-9c9267bb431e
Block Pool ID:	BP-169099099-172.31.85.228-1743434823455

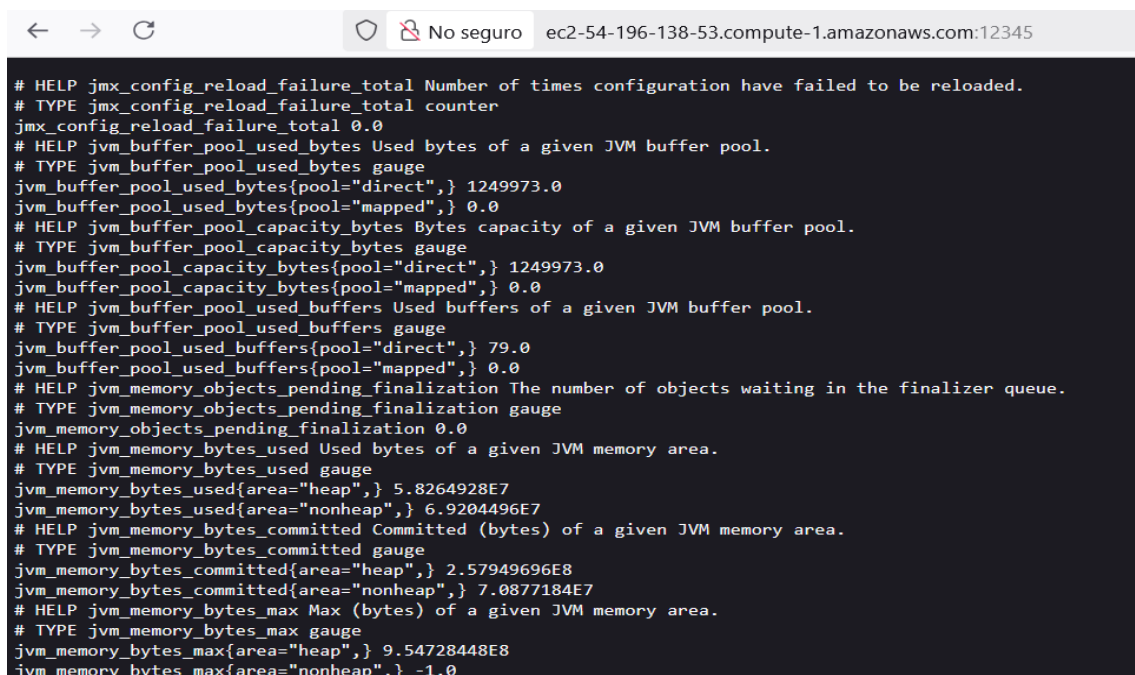


JSON Datos sin procesar Cabeceras

Guardar Copiar Adaptar para impresión

```
{
  "beans" : [ {
    "name" : "Hadoop:service=NameNode,name=JvmMetrics",
    "modelerType" : "JvmMetrics",
    "tag.Context" : "jvm",
    "tag.ProcessName" : "NameNode",
    "tag.SessionId" : null,
    "tag.Hostname" : "ip-172-31-85-228.ec2.internal",
    "MemNonHeapUsedM" : 58.82231,
    "MemNonHeapCommittedM" : 60.15625,
    "MemNonHeapMaxM" : -1.0,
    "MemHeapUsedM" : 103.33534,
    "MemHeapCommittedM" : 238.5,
    "MemHeapMaxM" : 910.5,
    "MemMaxM" : 910.5,
    "GcCount" : 12,
    "GcTimeMillis" : 538,
    "GcNumWarnThresholdExceeded" : 0,
    "GcNumInfoThresholdExceeded" : 2,
    "GcTotalExtraSleepTime" : 5930,
    "ThreadsNew" : 0,
    "ThreadsRunnable" : 9,
    "ThreadsTimedWaiting" : 0,
    "ThreadsWaiting" : 0,
    "Uptime" : 1000000000000000.0
  } ]
}
```

A continuación vamos al puerto 12345 con la ip-publica de la instancia de ec2 para comprobar que tenemos las métricas:



```
# HELP jmx_config_reload_failure_total Number of times configuration have failed to be reloaded.
# TYPE jmx_config_reload_failure_total counter
jmx_config_reload_failure_total 0.0
# HELP jvm_buffer_pool_used_bytes Used bytes of a given JVM buffer pool.
# TYPE jvm_buffer_pool_used_bytes gauge
jvm_buffer_pool_used_bytes{pool="direct",} 1249973.0
jvm_buffer_pool_used_bytes{pool="mapped",} 0.0
# HELP jvm_buffer_pool_capacity_bytes Bytes capacity of a given JVM buffer pool.
# TYPE jvm_buffer_pool_capacity_bytes gauge
jvm_buffer_pool_capacity_bytes{pool="direct",} 1249973.0
jvm_buffer_pool_capacity_bytes{pool="mapped",} 0.0
# HELP jvm_buffer_pool_used_buffers Used buffers of a given JVM buffer pool.
# TYPE jvm_buffer_pool_used_buffers gauge
jvm_buffer_pool_used_buffers{pool="direct",} 79.0
jvm_buffer_pool_used_buffers{pool="mapped",} 0.0
# HELP jvm_memory_objects_pending_finalization The number of objects waiting in the finalizer queue.
# TYPE jvm_memory_objects_pending_finalization gauge
jvm_memory_objects_pending_finalization 0.0
# HELP jvm_memory_bytes_used Used bytes of a given JVM memory area.
# TYPE jvm_memory_bytes_used gauge
jvm_memory_bytes_used{area="heap",} 5.8264928E7
jvm_memory_bytes_used{area="nonheap",} 6.9204496E7
# HELP jvm_memory_bytes_committed Committed (bytes) of a given JVM memory area.
# TYPE jvm_memory_bytes_committed gauge
jvm_memory_bytes_committed{area="heap",} 2.57949696E8
jvm_memory_bytes_committed{area="nonheap",} 7.0877184E7
# HELP jvm_memory_bytes_max Max (bytes) of a given JVM memory area.
# TYPE jvm_memory_bytes_max gauge
jvm_memory_bytes_max{area="heap",} 9.54728448E8
jvm_memory_bytes_max{area="nonheap",} -1.0
```

Parte 3: Despliegue de Prometheus y Grafana

1. Crear una instancia EC2 para Prometheus y Grafana:

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

Lanzar una instancia

[Información](#)

Amazon EC2 le permite crear máquinas virtuales, o instancias, que se ejecutan en la nube de AWS. Comience rápidamente siguiendo los sencillos pasos que se indican a continuación.

Nombre y etiquetas

[Información](#)

Nombre

InstanciaEC2_para_prometheus_y_grafana

[Agregar etiquetas adicionales](#)

▼ Imágenes de aplicaciones y sistemas operativos (Imagen de máquina de Amazon)

[Información](#)

Una AMI es una plantilla que contiene la configuración de software (sistema operativo, servidor de aplicaciones y aplicaciones) necesaria para lanzar la instancia. Busque o examine las AMI si no ve lo que busca a continuación.

Q. Busque en nuestro catálogo completo que incluye miles de imágenes de sistemas operativos y aplicaciones

Recientes

Mis AMI

Inicio rápido

Amazon macOS Ubuntu Windows RHEL Hat CentOS Linux Debian

▼ Resumen

Número de instancias [Información](#)

1

Imagen de software (AMI)

Amazon Linux 2023 AMI 2023.6.2...[más información](#)
ami-071226ecf16aa7d96

Tipo de servidor virtual (tipo de instancia)

t2.micro

Firewall (grupo de seguridad)

default

Almacenamiento (volúmenes)

Volúmenes: 1 (8 GiB)

[Cancelar](#)

[Lanzar instancia](#)

[Código de versión preliminar](#)

Resumen de instancia de i-07c1a43f1871b7c66 (InstanciaEC2_para_prometheus_y_grafana)

[Información](#)



[Conectar](#)

[Estado de la instancia](#)

[Acciones](#)

Se ha actualizado hace 2 minutos

ID de la instancia

i-07c1a43f1871b7c66

Dirección IPv6

—

Tipo de nombre de anfitrión

Nombre de IP: ip-172-31-85-32.ec2.internal

Responder al nombre DNS de recurso privado IPv4 (A)

Dirección IP asignada automáticamente

54.174.191.191 [IP pública]

Rol de IAM

—

IMDSv2

Required

Dirección IPv4 pública

54.174.191.191 | [dirección abierta](#)

Estado de la instancia

En ejecución

Nombre DNS de IP privada (solo IPv4)

ip-172-31-85-32.ec2.internal

Tipo de instancia

t2.micro

ID de VPC

vpc-0a162a66603af98ef

ID de subred

subnet-0218561b2da364b99

ARN de instancia

arn:aws:ec2:us-east-1:500318188469:instance/i-07c1a43f1871b7c66

Direcciones IPv4 privadas

172.31.85.32

DNS de IPv4 pública

ec2-54-174-191-191.compute-1.amazonaws.com | [dirección abierta](#)

Direcciones IP elásticas

—

Hallazgo de AWS Compute Optimizer

Suscribirse a AWS Compute Optimizer para recibir recomendaciones.

[Más información](#)

Nombre del grupo de Auto Scaling

—

Administradas

falso

[Activar Windows](#)

Ve a Configuración para activar Windows

2. Instalar Prometheus:

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

Como ya tenemos descargado anteriormente el labsuser.pem. A continuación lanzamos en una terminal el siguiente comando dentro de C:\Users\IABD07:

```
ssh -i labsuser.pem ubuntu@ec2-54-145-147-32.compute-1.amazonaws.com
```

```
PS C:\Users\IABD07> ssh -i labsuser.pem ubuntu@ec2-54-145-147-32.compute-1.amazonaws.com
Welcome to Ubuntu 22.04.5 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 Mon Mar 31 16:32:11 UTC 2025

System load:  0.0          Processes:    104
Usage of /:   40.0% of 7.57GB Users logged in: 0
Memory usage: 26%         IPv4 address for eth0: 172.31.85.32
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

7 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Fri Mar 28 17:02:58 2025 from 195.57.144.137
ubuntu@ip-172-31-85-32:~$
```



```
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-172-31-85-32:~$ 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
--2025-03-31 16:33:02-- 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-5fb918691a1e?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250331%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250331T163302Z&X-Amz-Expires=300&X-Amz-Signature=eac468dedc38ee349a7dbdbc4b60d2f1dc2384834f745dbd642f542abf1f8aac&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dprometheus-2.30.3.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream [following]
--2025-03-31 16:33:02-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/6838921/bc9e0970-09b3-4893-a66b-5fb918691a1e?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250331%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250331T163302Z&X-Amz-Expires=300&X-Amz-Signature=eac468dedc38ee349a7dbdbc4b60d2f1dc2384834f745dbd642f542abf1f8aac&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dprometheus-2.30.3.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.111.133|: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.1'

prometheus-2.30.3.linux-amd64 100%[=====] 69.27M 114MB/s in 0.6s
```

3. Configurar Prometheus:

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

```
global:
  scrape_interval: 15s

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

```
ubuntu@ip-172-31-85-32: ~/p  X + v
GNU nano 6.2 prometheus.yml *
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'emr-namenode'
    static_configs:
      - targets: ['ec2-44-208-167-114.compute-1.amazonaws.com:12345']
```

```
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ nano prometheus.yml
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ sudo systemctl restart prometheus
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ sudo systemctl status prometheus
● prometheus.service - Monitoring system and time series database
   Loaded: loaded (/lib/systemd/system/prometheus.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2025-03-31 16:36:21 UTC; 13s ago
     Docs: https://prometheus.io/docs/introduction/overview/
    Main PID: 2068 (prometheus)
      Tasks: 7 (limit: 1129)
     Memory: 28.9M
        CPU: 321ms
    CGroup: /system.slice/prometheus.service
            └─2068 /usr/bin/prometheus

Mar 31 16:36:22 ip-172-31-85-32 prometheus[2068]: ts=2025-03-31T16:36:22.502Z caller=head.go:598 level=info component=t
Mar 31 16:36:22 ip-172-31-85-32 prometheus[2068]: ts=2025-03-31T16:36:22.506Z caller=main.go:850 level=info fs_type=EXT
Mar 31 16:36:22 ip-172-31-85-32 prometheus[2068]: ts=2025-03-31T16:36:22.506Z caller=main.go:853 level=info msg="TSDB s
Mar 31 16:36:22 ip-172-31-85-32 prometheus[2068]: ts=2025-03-31T16:36:22.506Z caller=main.go:980 level=info msg="Loadin
Mar 31 16:36:22 ip-172-31-85-32 prometheus[2068]: ts=2025-03-31T16:36:22.509Z caller=main.go:1017 level=info msg="Compl
Mar 31 16:36:22 ip-172-31-85-32 prometheus[2068]: ts=2025-03-31T16:36:22.509Z caller=main.go:795 level=info msg="Server
Mar 31 16:36:28 ip-172-31-85-32 prometheus[2068]: ts=2025-03-31T16:36:28.501Z caller=compact.go:509 level=info componen
Mar 31 16:36:28 ip-172-31-85-32 prometheus[2068]: ts=2025-03-31T16:36:28.504Z caller=head.go:805 level=info component=t
Mar 31 16:36:28 ip-172-31-85-32 prometheus[2068]: ts=2025-03-31T16:36:28.504Z caller=checkpoint.go:97 level=info compon
Mar 31 16:36:28 ip-172-31-85-32 prometheus[2068]: ts=2025-03-31T16:36:28.534Z caller=head.go:974 level=info component=t
```

4. Instalar Grafana:

- Instala Grafana en la misma instancia EC2:

```
sudo apt-get install -y apt-transport-https
```

```
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ sudo apt-get install -y apt-transport-https
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 7 not upgraded.
Need to get 1510 B of archives.
After this operation, 170 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-transport-https all 2.4.13 [1510 B]
Fetched 1510 B in 0s (88.5 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 77910 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.4.13_all.deb ...
Unpacking apt-transport-https (2.4.13) ...
Setting up apt-transport-https (2.4.13) ...
Scanning processes...
```

```
sudo apt-get install -y software-properties-common wget
```

```
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ sudo apt-get install -y software-properties-common wget
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package software-properties-common wget
```

```
wget -q -O - https://packages.grafana.com/gpg.key |
sudo apt-keyadd -
```

*No me funciona con esos comandos. Buscando en chatgpt utilice este:

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo tee
/etc/apt/trusted.gpg.d/grafana.asc
```

```
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ wget -q -O - https://packages.grafana.com/gpg.key | sudo tee /etc/apt/trusted.gpg.d/grafana.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGTnhmkBDADUE+SzjRRyitIm1siGxiHLInn6K04C4GfEuV+PNzqxvYO+1r
mcLlGDU0ugo8ohXruAOC77Kwc4keVGNU89BeHvrYbIftz/yxEneupScbGnbDMIyC
k4HU0etRtV9/59Gj5YjNqnsZCr+e5D/JfrHUJTtWKLv88A9eHKxskrlZr7Un7j3i
kf3NCh10h27k9WfK8ThAcMMTferlW4iTTbOk+5fapShtXTu2BaxU3lkzF5G7VuiAHU
```

```
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee
/etc/apt/sources.list.d/grafana.list
```

```
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ echo "deb https://packages.grafana.com/oss/deb stable main" | su
do tee /etc/apt/sources.list.d/grafana.list
deb https://packages.grafana.com/oss/deb stable main
```

```
sudo apt-get update
```

```
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 https://packages.grafana.com/oss/deb stable InRelease [7661 B]
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2426 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [401 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [3185 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [564 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1196 kB]
Get:11 https://packages.grafana.com/oss/deb stable/main amd64 Packages [374 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2181 kB]
Get:13 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [970 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [208 kB]
Fetched 11.9 MB in 8s (1425 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ |
```

```
sudo apt-get install grafana
```

```
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ sudo apt-get install grafana
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
grafana is already the newest version (11.6.0).
```

```
sudo systemctl start grafana-server
```

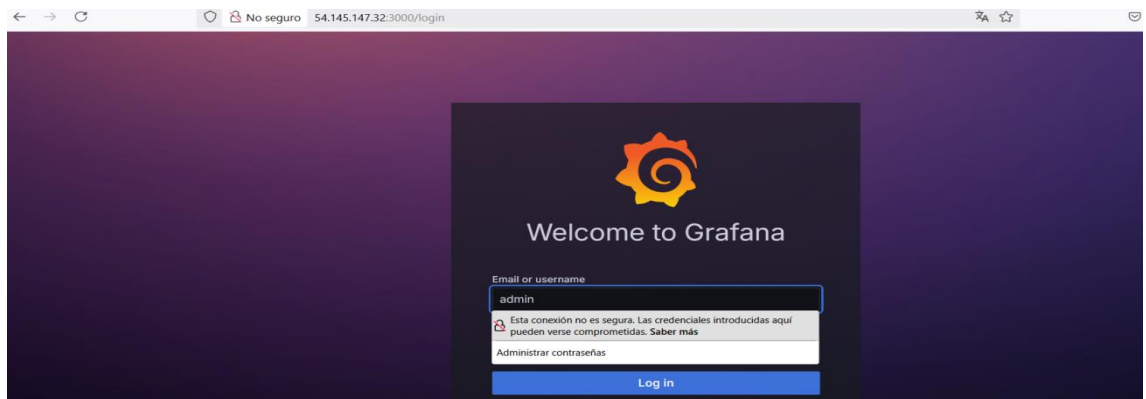
```
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ sudo systemctl start grafana-server
```

```
sudo systemctl enable grafana-server
```

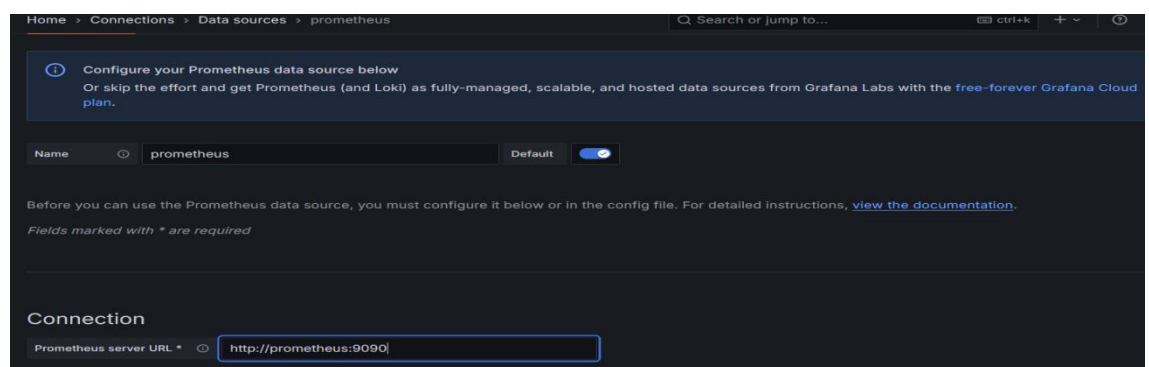
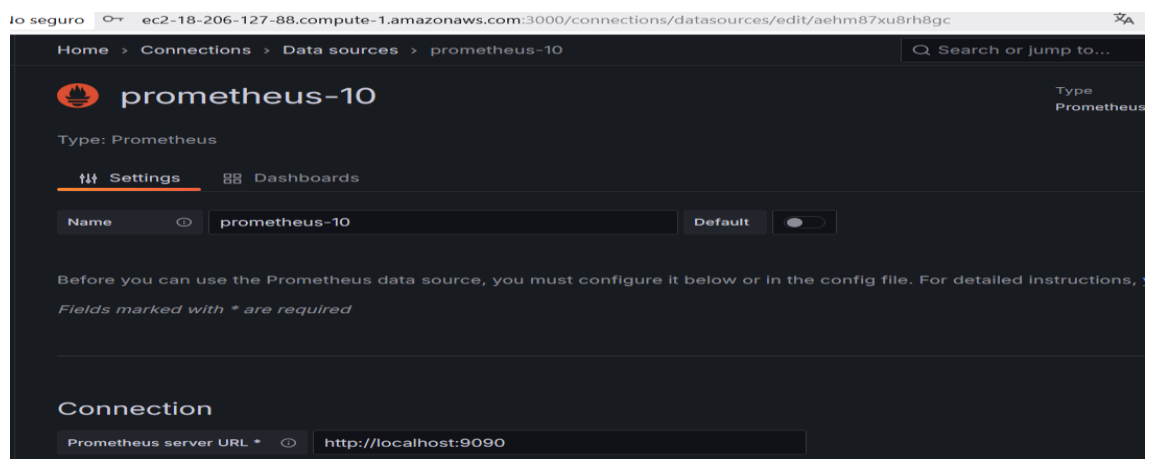
```
ubuntu@ip-172-31-85-32:~/prometheus-2.30.3.linux-amd64$ sudo systemctl enable grafana-server
Synchronizing state of grafana-server.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable grafana-server
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.service → /lib/systemd/system/grafana-server.service.
```

5. Configurar Grafana:

- Accede a Grafana en <http://<ip-instancia-ec2>:3000>.
- <http://ec2-18-206-127-88.compute-1.amazonaws.com:3000/>



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



Parte 4: Visualización de métricas en Grafana

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:

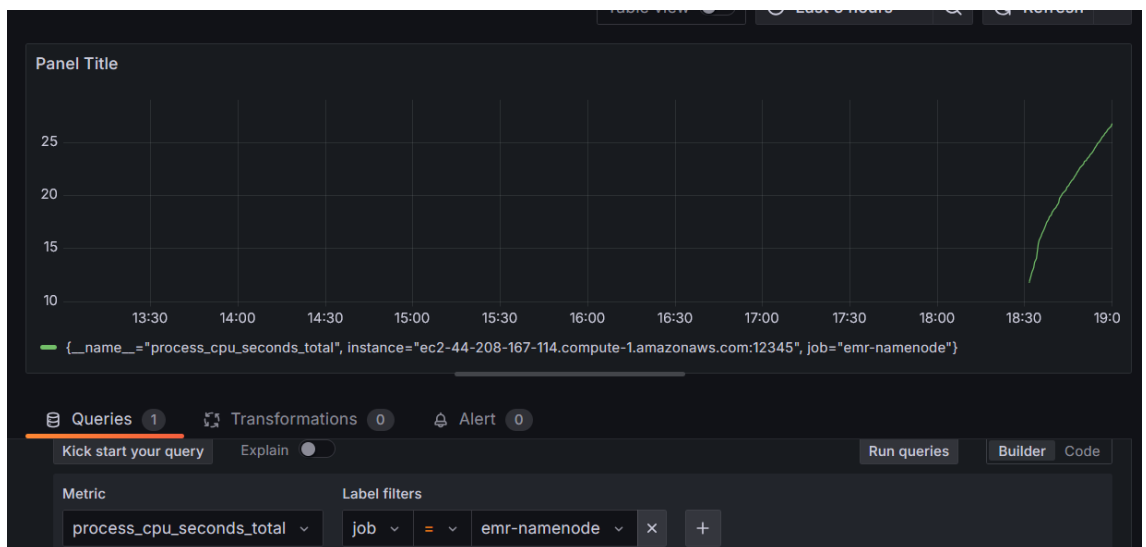
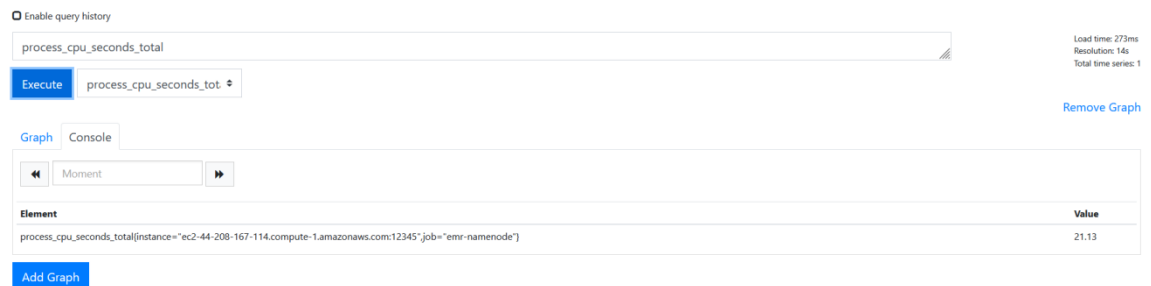
- Usa las métricas expuestas por JMX Exporter para crear gráficos en Grafana.

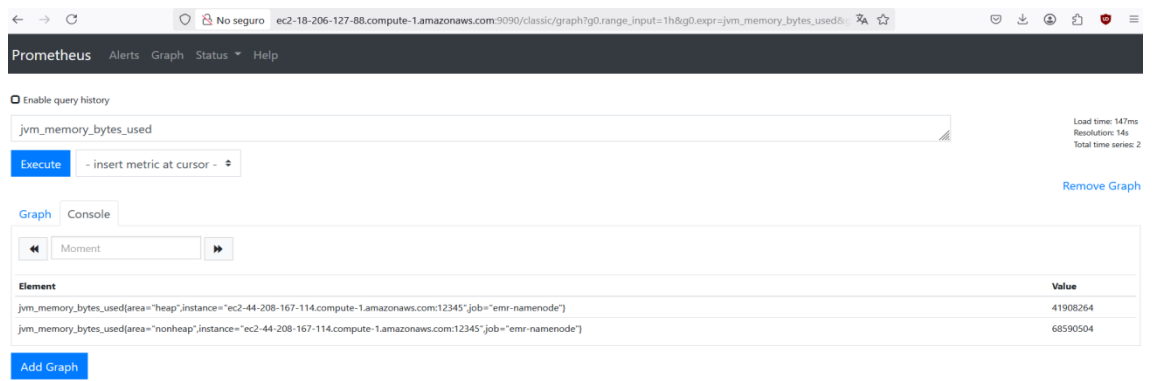
3. Explorar métricas:

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

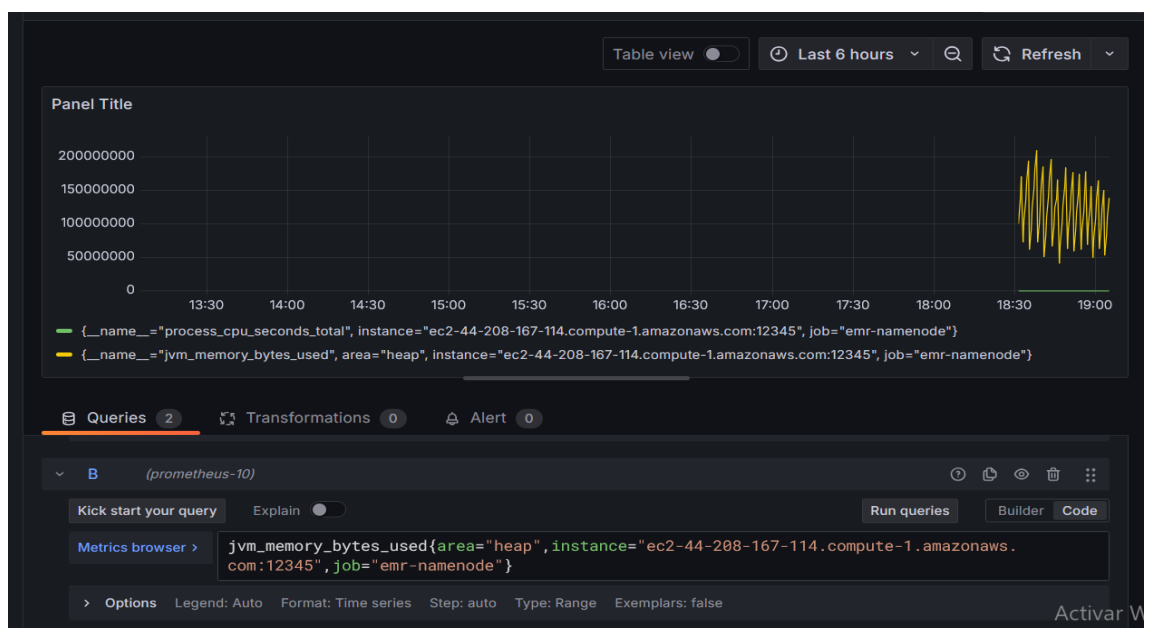
Voy a mostrar las métricas en prometheus y luego en grafana con sus gráficas

Uso de CPU y RAM:

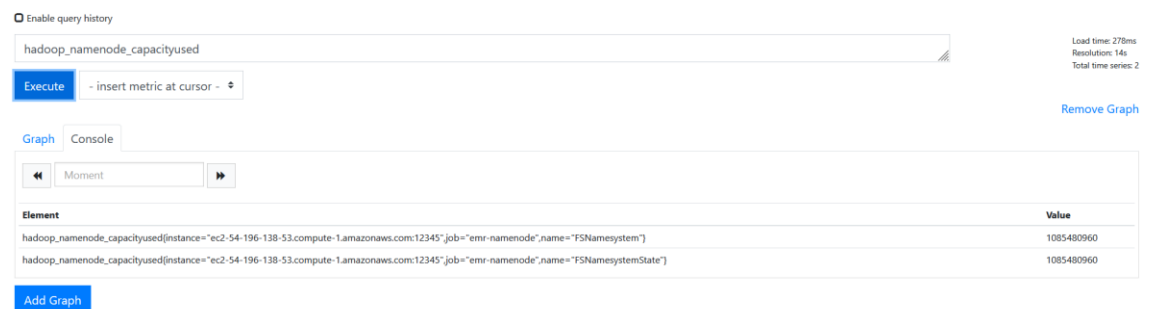


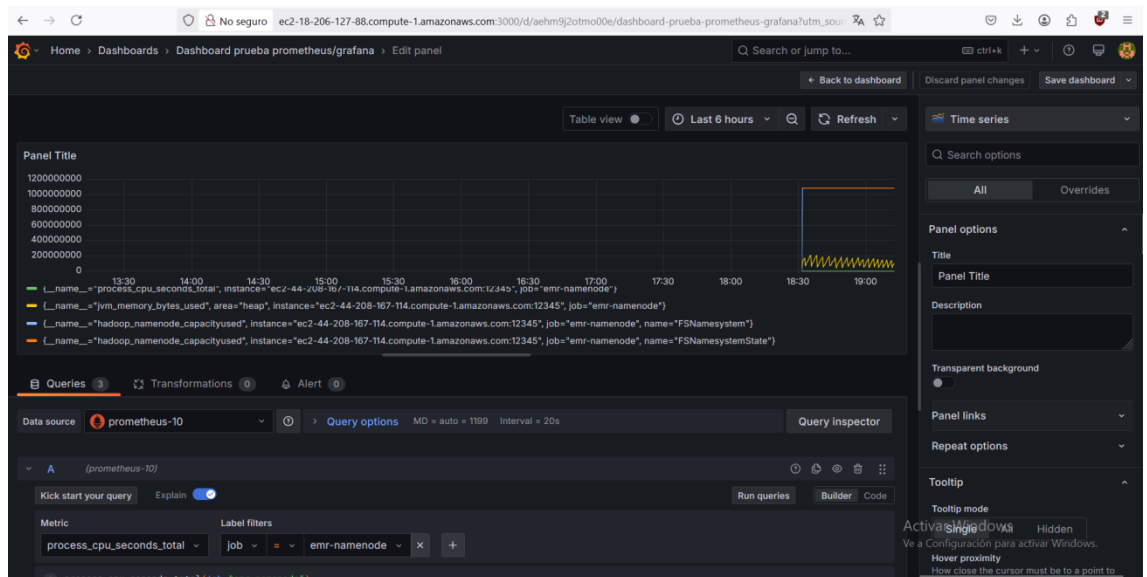


Seria la “heap” y para sacar la grafica en grafana la he buscado en la pestaña code, ya que en builder me sacaba ambas métricas



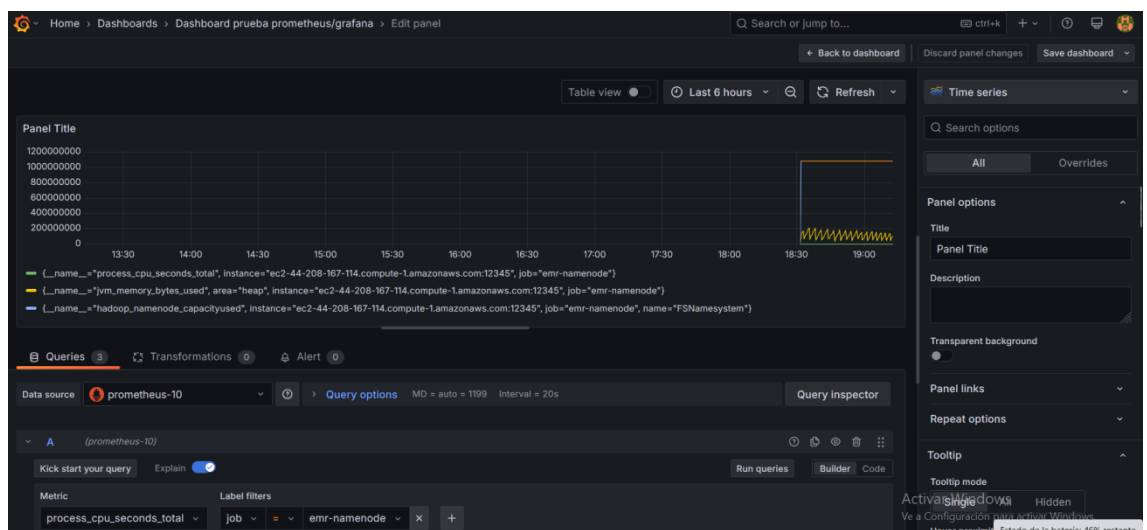
Espacio utilizado en HDFS:





Estado del NameNode: (No he encontrado esta)

Dashboard creado con nombre: Dashboard prueba promethetus/grafana:



Preguntas para reflexión

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

En un clúster **Amazon EMR (Elastic MapReduce)**, hay varias métricas clave que se deben monitorear para asegurar el buen funcionamiento del sistema. Algunas de las más importantes son:

- **Uso de CPU:** Es vital monitorear la carga de la CPU de los nodos del clúster para detectar posibles cuellos de botella. Si los nodos están sobrecargados, las tareas pueden tardar más en completarse o incluso fallar.
- **Uso de memoria:** La memoria es crítica para la ejecución de trabajos en un clúster EMR. Monitorear el uso de la memoria ayuda a identificar posibles fugas de memoria o procesos que consumen más memoria de la esperada.

- **Latencia de red:** La latencia entre nodos afecta el rendimiento global del clúster, especialmente en aplicaciones distribuidas. Se debe monitorear tanto la latencia interna del clúster como la latencia hacia otras redes o servicios.
- **Estado de los nodos y máquinas virtuales:** Asegurarse de que los nodos estén en un estado saludable es fundamental para evitar problemas en la ejecución de trabajos. Si un nodo se cae o está inactivo, puede causar fallos en los trabajos.
- **Tasa de fallos de trabajos:** Es importante monitorear los trabajos que fallan o tienen tiempos de ejecución elevados, ya que esto podría indicar problemas en el clúster o en la configuración de los trabajos.
- **Uso de disco:** El almacenamiento en disco es un componente clave en EMR, especialmente cuando se están procesando grandes volúmenes de datos. Monitorear el uso del disco y las operaciones de I/O es esencial para evitar problemas de rendimiento.
- **Métricas de YARN:** Si usas **YARN (Yet Another Resource Negotiator)** como el gestor de recursos en EMR, es importante monitorear métricas relacionadas con los recursos asignados, la cantidad de contenedores en ejecución, la cantidad de contenedores fallidos, etc.

Monitorear estas métricas permite detectar problemas antes de que afecten gravemente el rendimiento o la disponibilidad del clúster

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

JMX Exporter es una herramienta muy útil para exportar métricas de Java Management Extensions (JMX) a **Prometheus**. Para mejorar su configuración y recopilar métricas más específicas, puedes hacer lo siguiente:

- **Configurar filtros de métricas:** En la configuración de JMX Exporter, puedes especificar qué métricas deseas exportar utilizando filtros. Esto te permite centrarte solo en las métricas más relevantes para tu caso de uso, reduciendo la sobrecarga de datos innecesarios.
- **Agrupar métricas por categorías:** Puedes agrupar las métricas en diferentes "módulos" según las áreas de monitoreo que te interesen (por ejemplo, rendimiento, recursos, garbage collection, etc.). Esto hace que sea más fácil centrarse en un conjunto de métricas en particular cuando hay un gran volumen de datos.
- **Ajustar el intervalo de recolección:** Puedes ajustar el intervalo de recolección de métricas para que sea más frecuente o menos frecuente, dependiendo de la necesidad de los datos. Para ciertos servicios, puede ser útil tener métricas más detalladas, mientras que en otros, un intervalo mayor podría ser suficiente.
- **Configurar las métricas de JVM:** JMX Exporter puede extraer métricas sobre el estado de la JVM (memoria, hilos, recolección de basura, etc.). Asegúrate de habilitar estas métricas y configurarlas adecuadamente, ya que son esenciales para entender el rendimiento de las aplicaciones que se ejecutan en el clúster EMR.
- **Utilizar etiquetas (labels):** Las etiquetas o "labels" son útiles para agregar contexto a las métricas exportadas, como el tipo de nodo, la aplicación, la versión, etc. Esto te permite segmentar y filtrar métricas de manera más granular.
- **Activar métricas avanzadas:** Dependiendo de la configuración de tu aplicación, puedes habilitar métricas más avanzadas, como las métricas de memoria nativa, métricas de hilos de ejecución, estadísticas de discos, entre otros. Estas métricas adicionales te proporcionarán información detallada sobre el rendimiento de los servicios.

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

Prometheus y Grafana son dos herramientas muy populares para la recolección y visualización de métricas. Algunas de las ventajas clave frente a otras herramientas de monitoreo son:

- **Escalabilidad:** Prometheus está diseñado para escalar de manera eficiente y manejar grandes volúmenes de métricas en tiempo real. Esto lo convierte en una excelente opción para clústeres grandes y sistemas distribuidos.
- **Modelo de datos basado en series temporales:** Prometheus usa un modelo de datos de series temporales, lo que lo hace especialmente adecuado para monitorear métricas que cambian a lo largo del tiempo. Además, se puede almacenar de forma eficiente con un bajo costo de almacenamiento.

- **Consulta flexible:** Prometheus tiene su propio lenguaje de consulta, **PromQL**, que permite hacer consultas muy flexibles y complejas. Esto es útil cuando necesitas extraer métricas específicas o realizar análisis detallados.
- **Integración con Grafana:** Grafana es una plataforma de visualización que se integra perfectamente con Prometheus, lo que te permite crear dashboards personalizados para monitorear tus métricas. La integración entre ambas herramientas es fluida y ampliamente soportada.
- **Alertas personalizables:** Prometheus soporta alertas basadas en las métricas recopiladas, y puedes configurarlas fácilmente para que te notifiquen en caso de que alguna métrica alcance un umbral crítico. Esto te permite responder rápidamente ante problemas antes de que impacten gravemente en el rendimiento.
- **Ecosistema y comunidad activa:** Prometheus y Grafana cuentan con un ecosistema grande y una comunidad activa que proporciona integración con muchas otras herramientas y servicios. Esto hace que sea más fácil implementar soluciones avanzadas de monitoreo y solución de problemas.
- **Manejo de métricas distribuidas:** Estas herramientas están diseñadas para trabajar en entornos distribuidos y microservicios, lo que las hace muy adecuadas para entornos de big data como el clúster EMR, donde los servicios están dispersos a través de múltiples nodos.
- **Facilidad de implementación:** Prometheus y Grafana son fáciles de desplegar y configurar, y hay muchas guías y documentación disponible para ayudar en su implementación. Además, son herramientas open-source, lo que permite su personalización y adaptación a necesidades específicas.

En resumen, la combinación de Prometheus y Grafana ofrece una solución poderosa, flexible y escalable para monitorear clústeres grandes y entornos distribuidos como EMR. Su popularidad, integración con otros sistemas, y la comunidad activa que los respalda, son grandes ventajas frente a otras herramientas de monitoreo.

Github con los archivos de configuración:

<https://github.com/iabd07/Monitoreo-de-un-cl-ster-AWS-EMR-con-Prometheus-y-Grafana/tree/main>