Reporte del Reto Técnico - Night Watch

1. Introducción

Este documento describe el desarrollo del reto técnico propuesto, en el cual se construyó una infraestructura básica pero funcional en AWS utilizando Terraform, enfocada en el monitoreo con Grafana y Prometheus, almacenamiento de métricas en S3 y conexión entre subredes. Se documentan las actividades realizadas, los incidentes enfrentados (especialmente de conectividad), las soluciones aplicadas y las configuraciones relevantes.

2. Actividades Realizadas

Creación de infraestructura base en AWS mediante Terraform:

Creación de una VPC con los recursos necesarios para acceder a Internet a través de Instancias públicas y privadas:

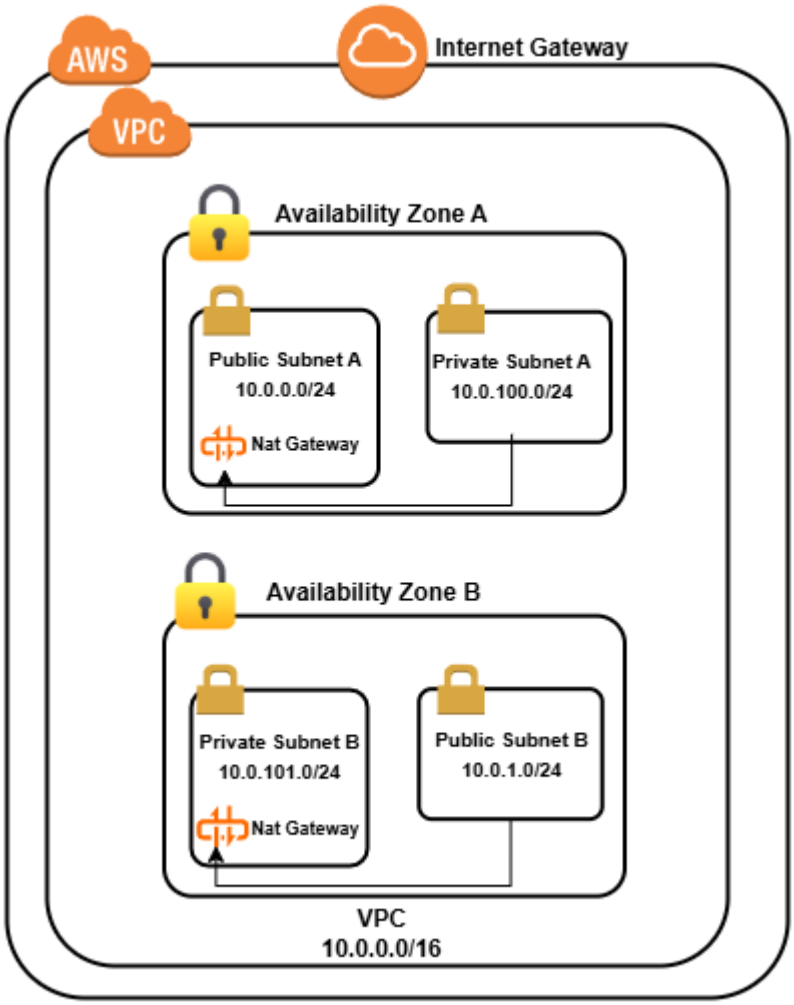
VPC con bloque CIDR `10.0.0.0/16`

Dos subredes públicas y dos privadas, distribuidas en zonas de disponibilidad distintas.

Internet Gateway y NAT Gateway configurados.

Tablas de ruteo para conectar adecuadamente las subredes públicas y privadas.

Arquitectura de Red en AWS con VPC y Subredes Distribuidas:



Public Subnet A/B: pueden comunicarse directamente con Internet gracias al **Internet Gateway**.

Private Subnet A/B: su ruta `0.0.0.0/0` va al NAT Gateway correspondiente.

Los **NAT Gateways** están correctamente colocados en **subredes públicas**.

Despliegue de instancias EC2:

EC2 pública: utilizada como bastión, sistema de logs y servidor para monitoreo (Grafana, Prometheus, K3s).

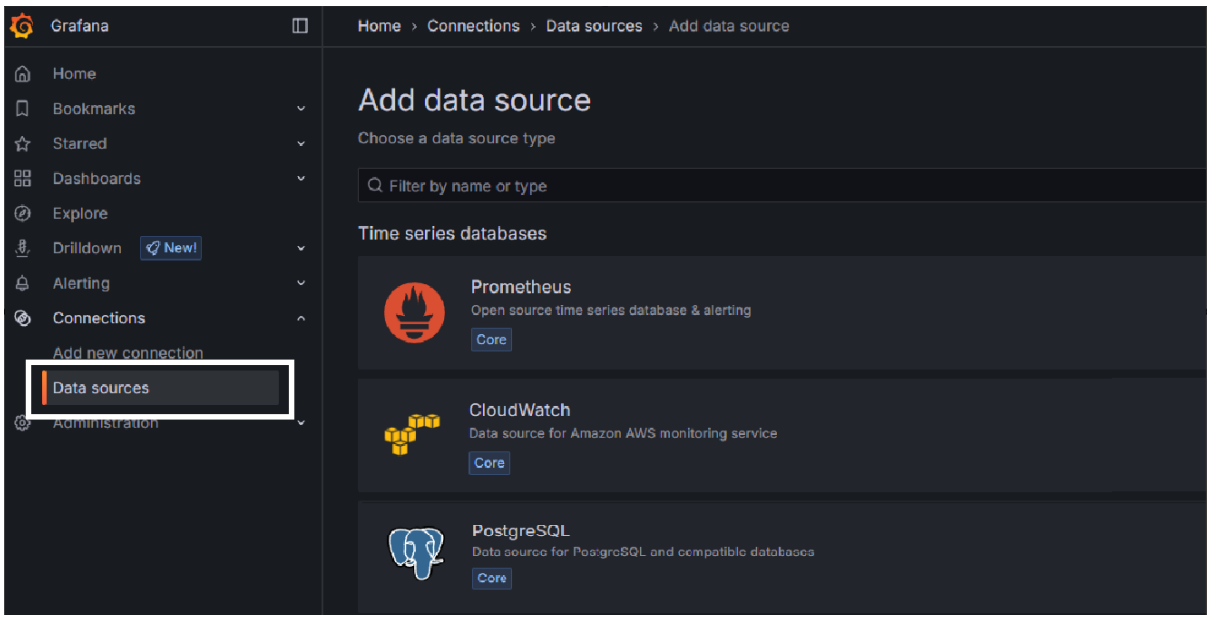
EC2 privada: servidor PostgreSQL con backups automáticos hacia S3.

IAM role: Permisos necesarios para que la EC2 pública acceda a los recursos de AWS (CloudWatch, S3).

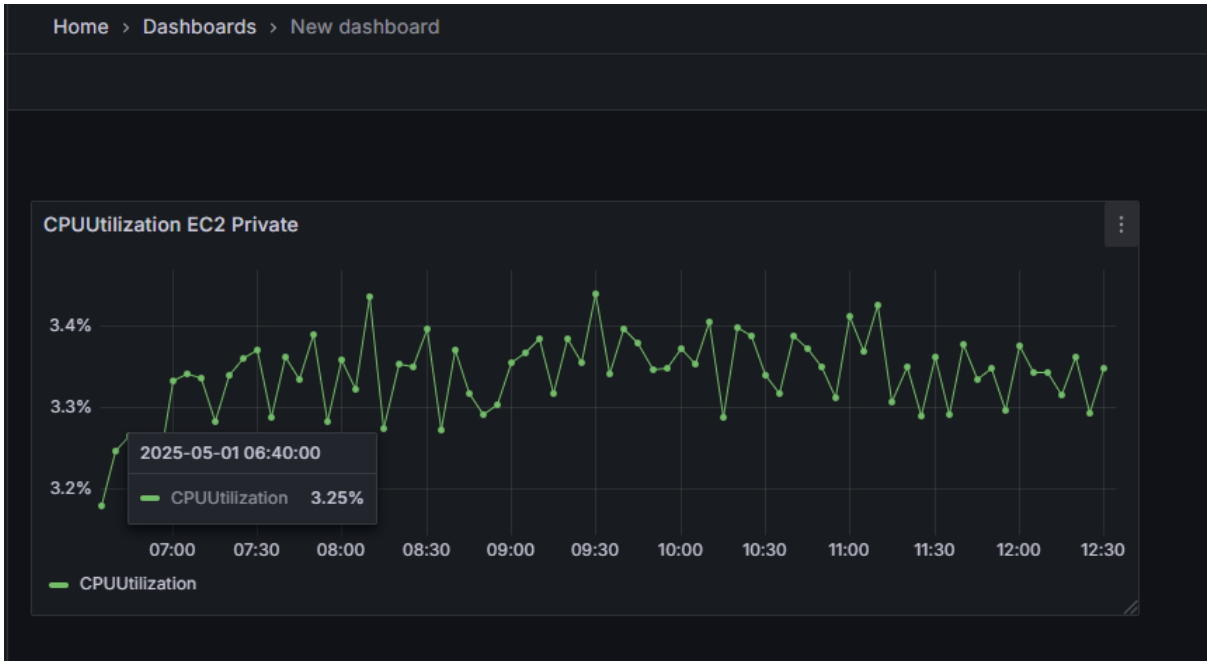
Configuración de monitoreo:

Instalación de Grafana vía EC2 user_data y Prometheus vía manifiesto K8s.

Añadir los Data Sources en Grafana:



Crear los Dashboards de grafana con los datos necesarios.



Utilizar alarmas de Grafana en los recursos críticos.

Automatización de tareas:

Scripts en Python y Bash para exportar métricas y backups diarios.

Cron jobs configurados para ejecución automática.

Requisitos:
Rellenar las variables de cada script con los correspondientes valores del despliegue.

Correr los scripts automatizados:
-setup_scripts.sh

-test_private_ec2.sh

Ejemplo de variables en los scripts:

```
region = "us-east-1"
private_instance_id = "InstancePrivateID"
bucket_name = "BucketName"
```

Donde `InstancePrivateID` corresponde a la Instancia creada en la subnet privada.

`BucketName` Es el nombre del bucket S3 creado el cual lo proporciona el Output de Terraform.

```
BASTION_IP="InstancePublicIP"
```

Donde `BASTION_IP` se refiere a la IP de EC2 pública.

3. Incidente de Conectividad

Descripción:

Durante la validación de la infraestructura, se detectó un fallo en la comunicación entre la instancia pública (bastión) y la privada (PostgreSQL). Esto impedía que las pruebas de conexión, así como los scripts de backup, funcionaran correctamente.

Diagnóstico:

Pruebas realizadas con nc (netcat) desde el bastión a la IP privada por puerto 5432.

Verificación de las reglas del Security Group.

Revisión de las rutas en las tablas de ruteo.

Solución:

El Security Group de la instancia privada permitiera tráfico desde el SG de la pública.

Las subredes privadas tuvieran una ruta `0.0.0.0/0` vía el NAT Gateway para salida a Internet.

La EC2 pública pudiera resolver DNS (configuración `enable_dns_support` y `enable_dns_hostnames`).

Correr el script de pruebas: `test_private_ec2.sh`.

4. Despliegue de Kubernetes en EC2

Para desplegar las aplicaciones de Kubernetes en EC2 es necesario tener el script de `setup_scripts.sh` correctamente configurado.

Este script hará una conexión ssh al bastion host, dentro de este generará una carpeta y copiará en ella el manifiesto YAML de Kubernetes.

Este manifiesto YAML tiene como funcionalidad:
Crear una aplicación de Nginx en el puerto 30080 con NodePort.
Levantar un servicio de Prometheus en el puerto 31080.

Se utiliza NodePort para asignarle un puerto a través de la IP pública de la Instancia EC2.

5. Documentación Técnica

Terraform Outputs: generados para obtener IPs públicas, privadas, bucketname, etc.

Scripts:

`ec2_logs_to_s3.py`: exporta métricas de CloudWatch a S3.

`test_private_ec2.sh`: Hace pruebas de conexión entre instancias.

`postgresql_daily_backup.sh`: backup de base de datos.

`setup_scripts.sh`: automatiza la subida de scripts y configuración de cron.

6. Conclusión

Se logró desplegar una infraestructura funcional con monitoreo activo, manejo de backups, y conectividad entre subredes correctamente configurada. La implementación de Prometheus y Nginx con 2 replicas sobre K3s demuestra escalabilidad futura. Las decisiones tomadas fueron documentadas y justificadas, y se encuentra todo listo para escalar o extender el entorno en siguientes etapas.

Links Útiles:
https://github.com/javieralmeida30/Reto-Tecnico-Night-Watch
https://www.linkedin.com/in/javieralmeida30/

Autor: Javier Almeida