Guía de Despliegue de Redis en Docker sobre EC2.

Esta guía detalla los pasos para desplegar una instancia de Redis utilizando contenedores Docker en una máquina virtual de Ubuntu (Instancia EC2) dentro de la Capa Gratuita de Amazon Web Services (AWS).

1. Prerrequisitos

Asegúrese de cumplir con los siguientes requisitos antes de comenzar:

- Una cuenta activa de AWS con acceso a la Capa Gratuita.
- Conocimientos básicos de la línea de comandos de Linux.
- Un par de claves SSH para conectarse a la instancia EC2.

2. Configuración en AWS

2.1 Creación de la Instancia EC2 (Ubuntu - Capa Gratuita)

1. **Lanzar Instancia:** Navegue al panel de EC2 y haga clic en "Lanzar instancias".
2. **Configuración:**
    - **AMI:** Seleccione una Imagen de Máquina de Amazon (AMI) de Ubuntu Server (la versión LTS elegible para la Capa Gratuita).
    - **Tipo de Instancia:** Seleccione **t3.small** (elegible para la Capa Gratuita).
    - **Par de Claves:** Seleccione o cree un par de claves (File Clave SSH) para la conexión.
3. **Configuración de Red (Grupo de Seguridad):**

- Cree un nuevo Grupo de Seguridad.
- Añada reglas de tráfico entrante:
  - **SSH (Puerto 22):** Permite el tráfico desde "Mi IP" o, temporalmente, 0.0.0.0/0.
  - **Redis (Puerto 6379):** Permite el tráfico desde 0.0.0.0/0 (para pruebas) o desde el rango de IP específico que necesite acceso.
  - **Aplicación Web (Puerto 8080):** Permite el tráfico desde 0.0.0.0/0 (para pruebas) o desde el rango de IP específico que necesite acceso.

3. Despliegue de Docker en la Instancia EC2

Una vez que la instancia EC2 esté en estado "running", conéctese a ella vía SSH.

3.1 Instalación Oficial de Docker y Docker Compose

Ejecute los siguientes comandos para instalar Docker y el plugin de Docker Compose de forma oficial en Ubuntu:# Actualizar los paquetes del sistema

sudo apt update

Instalar dependencias

sudo apt install ca-certificates curl gnupg lsb-release -y

Añadir la clave oficial de Docker

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

Añadir el repositorio oficial de Docker

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Instalar Docker y Docker Compose plugin

sudo apt update

sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

# Iniciar Docker y habilitar arranque automático

sudo systemctl start docker

sudo systemctl enable docker

# Configurar permisos para ejecutar Docker sin sudo

sudo usermod -aG docker $USER

newgrp docker

# Verificar instalación

docker --version

docker compose version

docker ps

# Probar Docker con un contenedor simple

docker run hello-world
```

3.2 Despliegue de la Aplicación y Redis con Docker Compose

Una vez que Docker esté instalado y funcionando (verificado con `docker run hello-world`), el siguiente paso es configurar y ejecutar Redis y su aplicación cliente usando Docker Compose.

A. **Preparación del Entorno**

a. **Crear el Directorio del Proyecto:**

```
mkdir redis-app
cd redis-app
```

B. **Definición de Docker Compose (`docker-compose.yml`)**

Cree el archivo `docker-compose.yml` en el directorio `redis-app`.

```
version: '3.9'
services:
  # Redis service
  redis:
    image: redis:7-alpine
    container_name: redis-server
    ports:
      - "6379:6379"
    volumes:
      # Redis data persistence (optional)
      - redis-data:/data
    command: redis-server --appendonly yes
    networks:
      - jedis-network
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 5s
      timeout: 3s
      retries: 5
```

```
  # Java application service
  app:
    image: traid128/aadd_jedis_test:8.0
    container_name: jedis-app
    depends_on:
      redis:
        condition: service_healthy
    environment:
      # These variables will be used to connect to Redis
      REDIS_HOST: redis
      REDIS_PORT: "6379"
    ports:
      - "8080:8080"
    volumes:
      # Mount custom properties file
      -
./src/main/resources/application.properties:/app/config/application.properties
    networks:
      - jedis-network
    # For interactive mode (terminal)
    stdin_open: true
    tty: true
# Persistent volumes
volumes:
  redis-data:
# Shared network for communication between containers
networks:
  jedis-network:
    driver: bridge
```

C. Ejecución del Despliegue

Finalmente, ejecute Docker Compose para levantar ambos servicios:docker compose up -d

Este comando descarga las imágenes, creará la red `jedis-network`, y levantará los contenedores en modo *detached* (segundo plano). La aplicación (`app`) solo se iniciará una vez que Redis (`redis`) pase su `healthcheck`.

D. **Verificación del Estado**

Puede verificar que los contenedores se hayan iniciado correctamente:docker compose ps

Debería ver tanto `redis-server` como `jedis-app` en estado `Up` y `healthy` (después de un momento).

Para verificar la ejecución de un solo contenedor o ejecutar comandos dentro de la aplicación (por ejemplo, para depuración): docker compose run --rm app