**Step 1: Launch an EC2 Instance on AWS**

1. **Log in to AWS Console**.
2. Navigate to the **EC2 Dashboard** and click on **Launch Instance**.
3. **Configure instance details**:
   * Choose **Amazon Linux 2** or **Ubuntu** as the operating system.
   * Select an instance type (e.g., t2.micro for testing).
   * Ensure the instance has a **key pair** and **security group** configured.
4. **Security Group Configuration**:
   * Allow SSH (port 22) for your IP.
   * Allow MySQL access (port 3306) from your IP or custom IPs as required.
   * Allow HTTP/HTTPS (optional, if you want to expose additional services).
5. **Launch the EC2 instance** and connect via SSH:

ssh -i <your-key>.pem ec2-user@<your-instance-public-ip>

**Step 2: Install Docker on EC2**

Once connected to the EC2 instance:

1. **Update the package index**:

sudo yum update -y # for Amazon Linux

sudo apt update -y # for Ubuntu

1. **Install Docker**: For Amazon Linux 2:

sudo amazon-linux-extras install docker

For Ubuntu:

sudo apt install docker.io -y

1. **Start and enable Docker**:

sudo systemctl start docker

sudo systemctl enable docker

1. **Add your user to the Docker group** (optional, for running Docker commands without sudo):

sudo usermod -aG docker $(whoami)

1. **Verify Docker installation**:

docker --version

**Step 3: Run MySQL in Docker with Host Disk Mount**

1. **Create a directory on the host for MySQL data**: This will allow you to store MySQL data on the host disk, making it persistent.

mkdir -p ~/mysql-data

1. **Run the MySQL container**: Use the following command to run MySQL with host disk mounted for data persistence and configure a root password (my-secret-pw in this example).

docker run -d \

--name mysql-container \

-e MYSQL\_ROOT\_PASSWORD=my-secret-pw \

-v ~/mysql-data:/var/lib/mysql \

-p 3306:3306 \

--health-cmd='mysqladmin ping --silent' \

--health-interval=30s \

--health-timeout=5s \

--health-retries=3 \

mysql:latest

Explanation of flags:

* + -d: Runs the container in detached mode (in the background).
  + -e MYSQL\_ROOT\_PASSWORD: Sets the MySQL root password.
  + -v ~/mysql-data:/var/lib/mysql: Mounts a directory from the host (~/mysql-data) to the MySQL data directory in the container.
  + -p 3306:3306: Maps port 3306 on the host to port 3306 in the container.
  + --health-cmd='mysqladmin ping --silent': Health check command that pings MySQL to ensure it’s running.
  + --health-interval=30s: Checks the health every 30 seconds.
  + --health-timeout=5s: If a health check takes longer than 5 seconds, it’s considered failed.
  + --health-retries=3: Number of times a health check can fail before being marked as unhealthy.

**Step 4: Monitor Container Health**

To monitor the health of the container, use the following commands:

1. **Check container health status**:

docker ps

This will show a column called **STATUS** with the health status (e.g., healthy, unhealthy).

1. **Inspect detailed health status**:

docker inspect --format='{{json .State.Health}}' mysql-container

This gives you detailed information about the health checks, such as the number of retries, last check result, and more.

1. **Check logs for MySQL**:

docker logs mysql-container

1. **Monitor the container continuously**: You can use Docker's built-in events to monitor the health of the container in real-time:

docker events --filter container=mysql-container

**Step 5: Access MySQL from the Host or Remote Machine**

1. **Connect to MySQL from the host** (inside the EC2 instance):

docker exec -it mysql-container mysql -u root -p

1. **Connect to MySQL from a remote machine**: Make sure port 3306 is open in the EC2 security group and use the public IP of the EC2 instance:

mysql -h <EC2-public-IP> -P 3306 -u root -p

**Step 6: Optional - Set Up Monitoring with Docker Compose and Prometheus (Advanced)**

For advanced monitoring, you can use **Docker Compose** with tools like **Prometheus** and **Grafana** to monitor container health and resource usage.

1. Install Docker Compose:

sudo curl -L "https://github.com/docker/compose/releases/download/v2.5.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

1. Create a docker-compose.yml file to configure Prometheus and Grafana.

**Docker Compose File (docker-compose.yml)**

version: '3.8'

services:

mysql:

image: mysql:latest

container\_name: mysql-container

restart: unless-stopped

environment:

MYSQL\_ROOT\_PASSWORD: my-secret-pw

volumes:

- ./mysql-data:/var/lib/mysql

ports:

- "3306:3306"

healthcheck:

test: ["CMD", "mysqladmin", "ping", "--silent"]

interval: 30s

timeout: 5s

retries: 3

prometheus:

image: prom/prometheus:latest

container\_name: prometheus

volumes:

- ./prometheus.yml:/etc/prometheus/prometheus.yml

ports:

- "9090:9090"

grafana:

image: grafana/grafana:latest

container\_name: grafana

ports:

- "3000:3000"

volumes:

- grafana-storage:/var/lib/grafana

depends\_on:

- prometheus

volumes:

grafana-storage:

driver: local

**Explanation of Services**

1. **MySQL Service**:
   * Uses the official mysql:latest image.
   * Exposes port 3306 for MySQL access.
   * Mounts ./mysql-data on the host to /var/lib/mysql inside the container for persistent data storage.
   * Defines a healthcheck that runs mysqladmin ping --silent every 30 seconds to monitor container health.
2. **Prometheus Service**:
   * Uses the prom/prometheus image.
   * Exposes port 9090 for accessing Prometheus metrics and dashboard.
   * Requires a **Prometheus configuration file** (prometheus.yml) to scrape metrics from Docker.
3. **Grafana Service**:
   * Uses the grafana/grafana image.
   * Exposes port 3000 for Grafana's dashboard.
   * Mounts a persistent volume grafana-storage for storing Grafana data.

**Prometheus Configuration (prometheus.yml)**

Create a file named prometheus.yml in the same directory as your docker-compose.yml file. This file configures Prometheus to scrape metrics from Docker and your MySQL container. This would be a yaml script.

global:

scrape\_interval: 15s

scrape\_configs:

- job\_name: 'mysql'

static\_configs:

- targets: ['mysql-container:3306']

- job\_name: 'docker'

static\_configs:

- targets: ['localhost:9323']

**Steps to Use This Setup**

1. **Create the Required Files**:
   * Create a directory for your Docker Compose project and navigate into it:

mkdir mysql-docker-compose

cd mysql-docker-compose

* + Create the docker-compose.yml file in the directory.
  + Create the prometheus.yml file in the same directory.

1. **Start the Containers**: Run the following command to start all the services (MySQL, Prometheus, and Grafana):

docker-compose up -d

1. **Access Services**:
   * **MySQL**: Connect via localhost:3306 (or your EC2 public IP, depending on how you configured the security group).
   * **Prometheus**: Open http://<your-ec2-ip>:9090 to access Prometheus.
   * **Grafana**: Open http://<your-ec2-ip>:3000 to access Grafana (default credentials: admin/admin).
2. **Configure Grafana**:
   * Add **Prometheus** as a data source in Grafana (http://prometheus:9090 as the URL).
   * Import a MySQL dashboard from Grafana’s built-in library to monitor the health and performance of MySQL.

**Monitoring Container Health in Prometheus**

Prometheus will automatically scrape health metrics from the MySQL container, thanks to the health check defined in the docker-compose.yml.

**Stopping the Containers**

To stop all services, run:

docker-compose down

This setup allows you to run MySQL with data persistence, monitor its health, and visualize metrics using Grafana and Prometheus. Let me know if you need more details or help with setting up dashboards in Grafana!