Sivahari
1239619

Distributed Load Testing Using JMeter Integration of Influx DB and Grafana with Docker on AWS EC2

1. Introduction

Distributed load testing is an approach to simulate large user traffic by distributing the load across multiple machines. This document outlines the process of setting up distributed load testing with JMeter in a master-slave configuration, integrated with InfluxDB and Grafana for real-time performance monitoring. Docker is used to containerize all components, ensuring portability and ease of deployment on AWS EC2 instances.

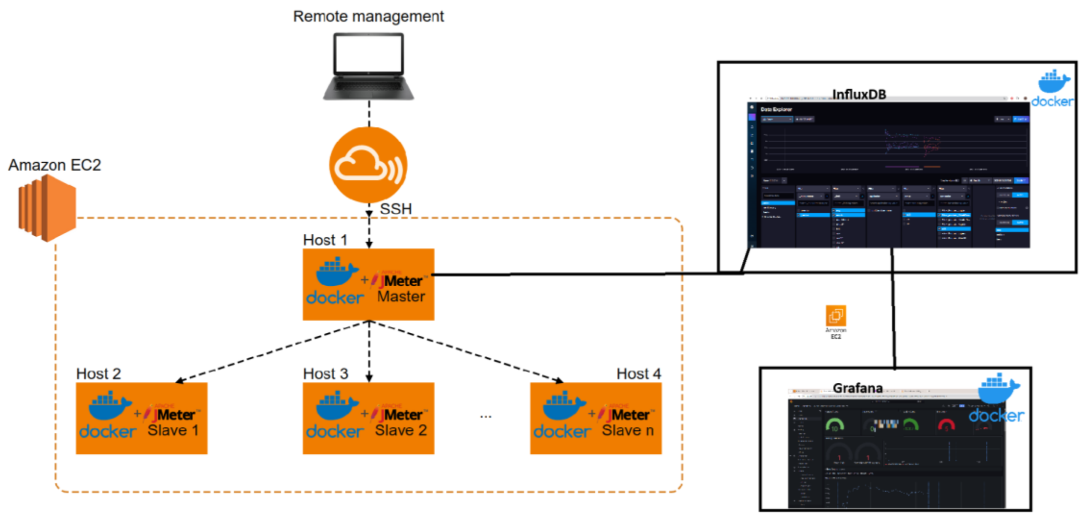
2. Components Overview

- **JMeter:** Open-source tool for performance and load testing.
- Master-Slave Architecture: Master sends test scripts to Slave instances which execute the tests concurrently.
- **InfluxDB:** Time-series database for storing performance metrics.
- Grafana: A visualization tool for monitoring and creating dashboards from InfluxDB data.
- Docker: A containerization platform that simplifies managing services and their dependencies.
- AWS EC2: Virtual machines in AWS for hosting JMeter, InfluxDB, and Grafana.
- Git: Update the test scripts and clone the repo in machine.

3. Infrastructure Overview

- AWS EC2: t2.medium instances (2vCPUs, 4GB RAM)
- Operating System: Ubuntu Linux
- Tools:
  - Docker
  - Docker Compose
  - AWS CLI for uploading results to S3

After testing, results will be transferred from the EC2 instances to an S3 bucket for persistent storage and analysis.



4. JMeter Master-Slave Setup

1. **Launch EC2 Instances:**
   - Set up one EC2 instance for the JMeter Master and at least two more for the Slaves.
   - Ensure the security group allows necessary ports for communication 8086 and 3000.

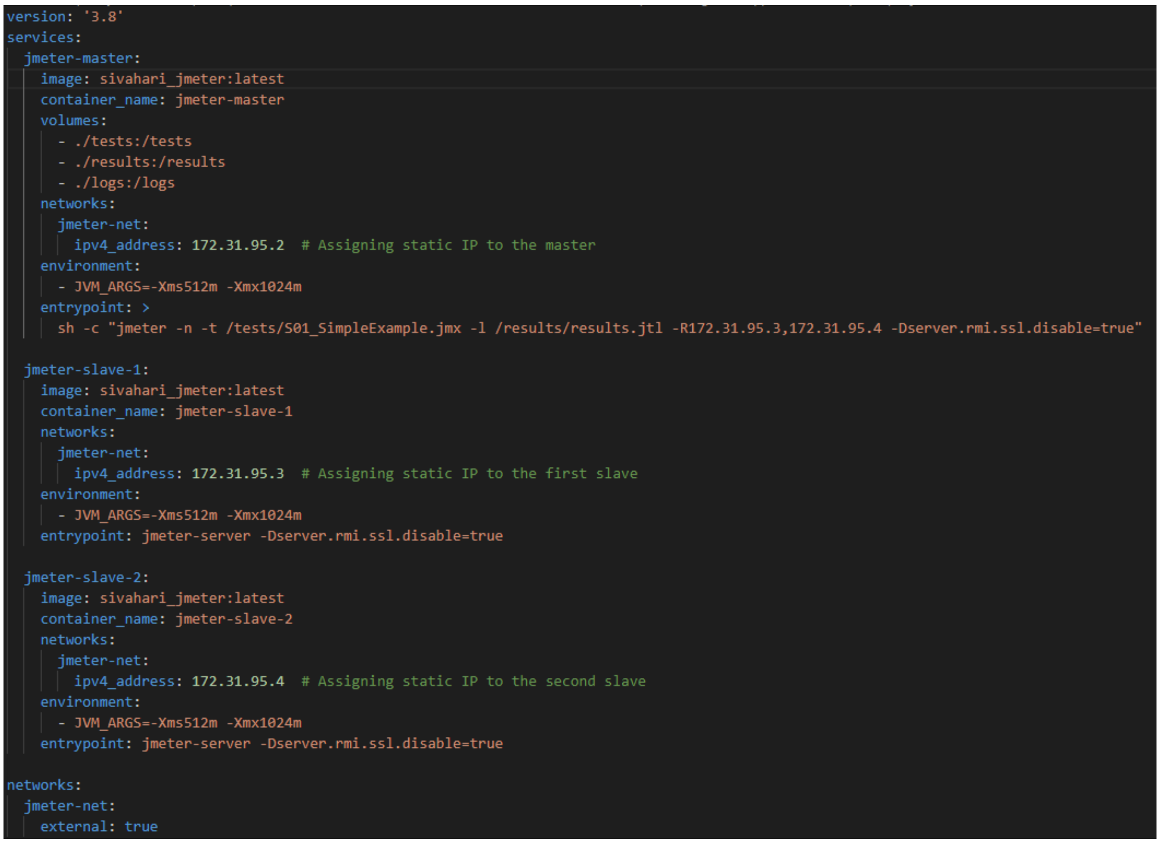
2. **Install Docker and Docker Compose:**
   Install Docker and Docker Compose on each EC2 instance (Master and Slaves).
   sudo apt update
   sudo apt install docker.io docker-compose -y

3. **Set Up JMeter Master-Slave Containers:**
   Run the JMeter Master and Slaves in separate Docker containers using the Docker
   Run the Distributed Load Test:
   On the Master instance, use the JMeter command line to run the test plan:

Dockerfile

FROM justb4/jmeter:latest
COPY /plugins/.jar /usr/share/jmeter/lib/*

Compose file:



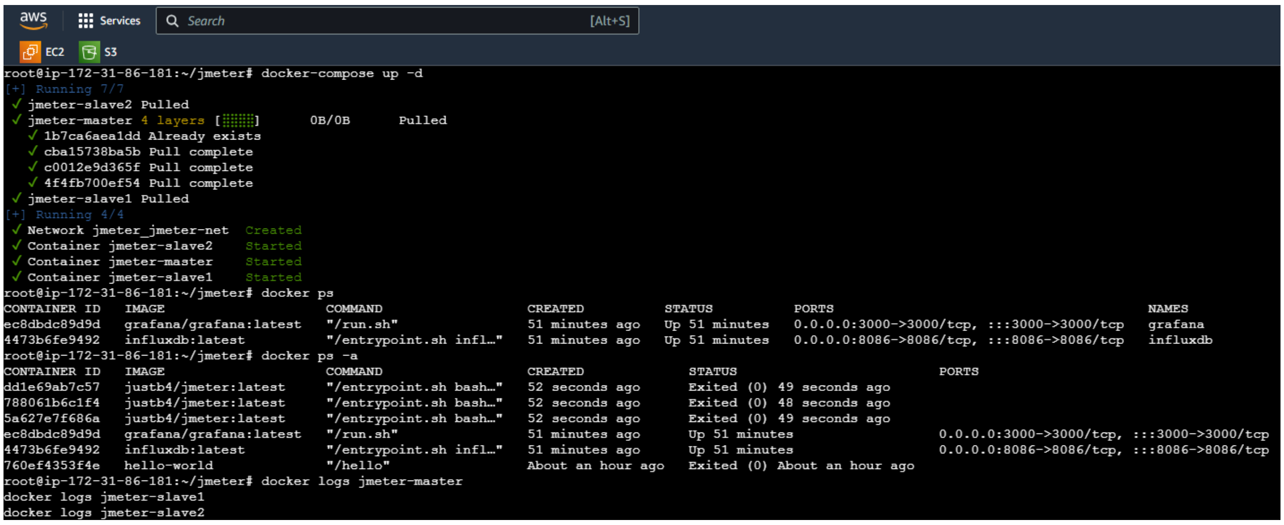
```yaml
version: '3.8'
services:
  jmeter-master:
    image: sivahari_jmeter:latest
    container_name: jmeter-master
    volumes:
      - ./tests:/tests
      - ./results:/results
      - ./logs:/logs
    networks:
      jmeter-net:
        ipv4_address: 172.31.95.2  # Assigning static IP to the master
    environment:
      - JVM_ARGS=-Xms512m -Xmx1024m
    entrypoint: >
      sh -c "jmeter -n -t /tests/S01_SimpleExample.jmx -l /results/results.jtl -R172.31.95.3,172.31.95.4 -Dserver.rmi.ssl.disable=true"

  jmeter-slave-1:
    image: sivahari_jmeter:latest
    container_name: jmeter-slave-1
    networks:
      jmeter-net:
        ipv4_address: 172.31.95.3  # Assigning static IP to the first slave
    environment:
      - JVM_ARGS=-Xms512m -Xmx1024m
    entrypoint: jmeter-server -Dserver.rmi.ssl.disable=true

  jmeter-slave-2:
    image: sivahari_jmeter:latest
    container_name: jmeter-slave-2
    networks:
      jmeter-net:
        ipv4_address: 172.31.95.4  # Assigning static IP to the second slave
    environment:
      - JVM_ARGS=-Xms512m -Xmx1024m
    entrypoint: jmeter-server -Dserver.rmi.ssl.disable=true

networks:
  jmeter-net:
    external: true
```

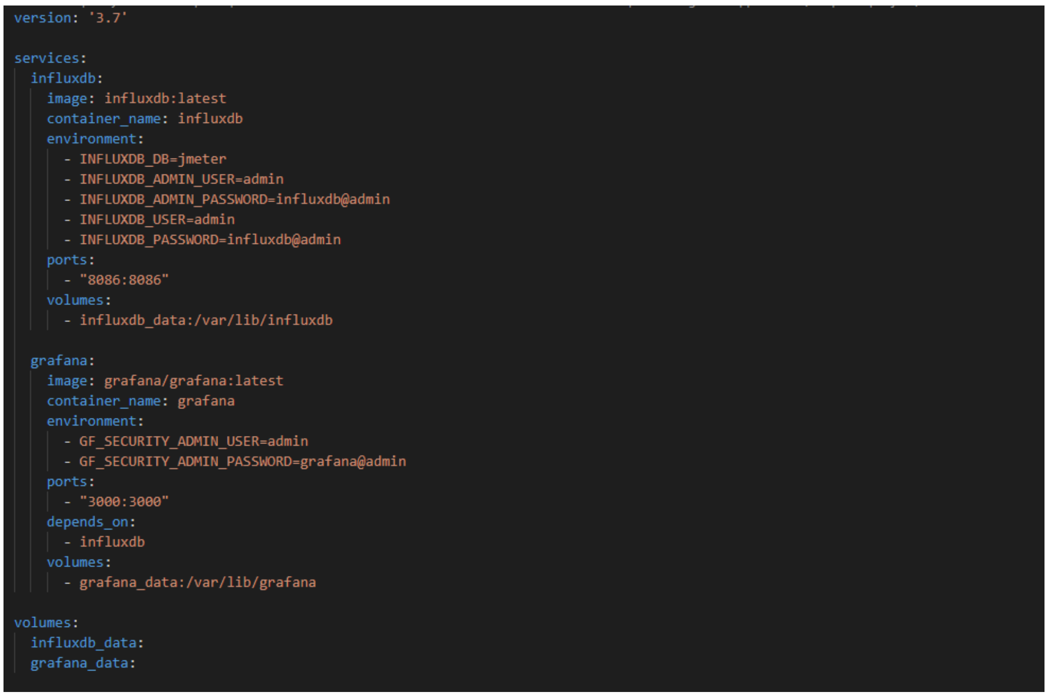
4. **InfluxDB and Grafana Setup**



```
root@ip-172-31-86-181:~/jmeter# docker-compose up -d
[+] Running 7/7
 ✓ jmeter-slave2 Pulled
 ✓ jmeter-master 4 layers [⣿⣿⣿⣿]      0B/0B      Pulled
   ✓ 1b7ca6aea1dd Already exists
   ✓ cba15738ba5b Pull complete
   ✓ c0012e9d365f Pull complete
   ✓ 4f4fb700ef54 Pull complete
 ✓ jmeter-slave1 Pulled
[+] Running 4/4
 ✓ Network jmeter_jmeter-net  Created
 ✓ Container jmeter-slave2    Started
 ✓ Container jmeter-master    Started
 ✓ Container jmeter-slave1    Started
root@ip-172-31-86-181:~/jmeter# docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS                                       NAMES
ec8dbdc89d9d   grafana/grafana:latest   "/run.sh"                51 minutes ago   Up 51 minutes   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   grafana
4473b6fe9492   influxdb:latest          "/entrypoint.sh infl…"   51 minutes ago   Up 51 minutes   0.0.0.0:8086->8086/tcp, :::8086->8086/tcp   influxdb
root@ip-172-31-86-181:~/jmeter# docker ps -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED             STATUS                         PORTS
dd1e69ab7c57   justb4/jmeter:latest     "/entrypoint.sh bash…"   52 seconds ago      Exited (0) 49 seconds ago
788061b6c1f4   justb4/jmeter:latest     "/entrypoint.sh bash…"   52 seconds ago      Exited (0) 48 seconds ago
5a627e7f686a   justb4/jmeter:latest     "/entrypoint.sh bash…"   52 seconds ago      Exited (0) 49 seconds ago
ec8dbdc89d9d   grafana/grafana:latest   "/run.sh"                51 minutes ago      Up 51 minutes                  0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
4473b6fe9492   influxdb:latest          "/entrypoint.sh infl…"   51 minutes ago      Up 51 minutes                  0.0.0.0:8086->8086/tcp, :::8086->8086/tcp
760ef4353f4e   hello-world              "/hello"                 About an hour ago   Exited (0) About an hour ago
root@ip-172-31-86-181:~/jmeter# docker logs jmeter-master
docker logs jmeter-slave1
docker logs jmeter-slave2
```

Once we start the container all setup will up and running.
Hit the Grafana and InfluxDB entpoint url to launch.
Verify using **Docker ps -a** cmd

1. Docker Compose for InfluxDB and Grafana (`docker-compose.yml`):
  monitoring-network:



```
version: '3.7'

services:
  influxdb:
    image: influxdb:latest
    container_name: influxdb
    environment:
      - INFLUXDB_DB=jmeter
      - INFLUXDB_ADMIN_USER=admin
      - INFLUXDB_ADMIN_PASSWORD=influxdb@admin
      - INFLUXDB_USER=admin
      - INFLUXDB_PASSWORD=influxdb@admin
    ports:
      - "8086:8086"
    volumes:
      - influxdb_data:/var/lib/influxdb

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=grafana@admin
    ports:
      - "3000:3000"
    depends_on:
      - influxdb
    volumes:
      - grafana_data:/var/lib/grafana

volumes:
  influxdb_data:
  grafana_data:
```

2. Deploy InfluxDB and Grafana containers on a EC2 instance:
bash
docker-compose.yml up -d

3. **Configure JMeter for InfluxDB:**
   In the JMeter test plan, add a Backend Listener and configure it to send data to InfluxDB using the following URL:

   http://172.65.23.1:8086/write?db=jmeter

  Access InfluxDB from browser using http:// 172.65.23.1:8086.

4. **Configure Grafana:**
   - Access Grafana from the browser using `http:// 172.65.23.1:3000.
   - Add InfluxDB as a data source in Grafana.
   - Create dashboards to visualize real-time data from JMeter.

5. Copying Results to S3

Install AWS CLI and configure :
Sudo apt install awscli -y
aws configure3.

Copy Results to S3: aws s3 cp /root/jmeter/results s3://jmeterbucket/results --recursive

6. Conclusion

This document provides a step-by-step guide to setting up a distributed load testing environment with JMeter in master-slave configuration, and real-time monitoring using InfluxDB and Grafana. Using Docker and Docker Compose ensures easy deployment and scalability on AWS EC2 instances. Following these steps will help in simulating high traffic loads and monitoring performance metrics efficiently.