Database Monitoring and Performance Optimization

Step-by-Step Guide for Database Monitoring and Performance Optimization

Phase 1: Setting Up MySQL with Docker

1. Install Docker:

- Ensure Docker is installed on your Linux machine. Follow the official Docker installation guide for your specific Linux distribution.

2. Pull MySQL Docker Image:

- Open a terminal and run the following command to pull the latest MySQL image:

```
docker pull mysql:latest
```

3. Run MySQL Container:

- Create and run a MySQL container with a specified root password:

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=my-secret-pw -d -p 3306:3306 mysql:latest
```

4. Verify MySQL Container:

- Check the status of the running container:

```
docker ps
```

Phase 2: Monitoring MySQL with Prometheus and Grafana

1. Set Up mysqld_exporter:

- Pull the mysqld_exporter Docker image:

```
docker pull prom/mysqld-exporter
```

- Run mysqld_exporter, linking it to your MySQL container:

```
docker run -d --name mysqld-exporter -e DATA_SOURCE_NAME="root:my-secret-pw@(mysql-container:3306)/" -p 9104:9104 prom/mysqld-exporter
```

2. Install Prometheus:

- Create a directory for Prometheus configuration:

  mkdir -p ~/prometheus

  cd ~/prometheus

- Create a configuration file prometheus.yml:

  global:

    scrape_interval: 15s

  scrape_configs:

    - job_name: 'mysql'

      static_configs:

        - targets: ['localhost:9104'] # MySQL Exporter

3. Run Prometheus:

- Pull the Prometheus Docker image:

  docker pull prom/prometheus

- Run Prometheus with the configuration file:

  docker run -d --name prometheus -p 9090:9090 -v ~/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus

4. Install Grafana:

- Pull the Grafana Docker image:

  docker pull grafana/grafana

- Run Grafana:

  docker run -d --name grafana -p 3000:3000 grafana/grafana

- Access Grafana by navigating to http://localhost:3000 in a web browser.

5. Configure Grafana Data Source:

- Log in to Grafana (default credentials are admin/admin).

- Go to Configuration > Data Sources > Add data source.

- Select Prometheus and set the URL to http://prometheus:9090.

6. Set Up Dashboards:

- Use existing Percona or Prometheus MySQL dashboards to visualize the metrics collected.

Phase 3: Automating Configuration with Ansible

1. Install Ansible:

- Follow the official Ansible installation guide for your Linux distribution.

2. Create Ansible Playbook:

- Create a directory for your Ansible project:

```
mkdir -p ~/ansible_projects/mysql_monitoring
cd ~/ansible_projects/mysql_monitoring
```

- Create a playbook file mysql_playbook.yml:

```
- hosts: localhost
  tasks:
    - name: Deploy MySQL Container
      docker_container:
        name: mysql-container
        image: mysql:latest
        state: started
        restart_policy: always
        env:
          MYSQL_ROOT_PASSWORD: my-secret-pw
        ports:
          - "3306:3306"
    - name: Deploy mysqld-exporter Container
      docker_container:
        name: mysqld-exporter
        image: prom/mysqld-exporter
        state: started
```

```
    restart_policy: always
    env:
      DATA_SOURCE_NAME: "root:my-secret-pw@(mysql-container:3306)/"
    ports:
      - "9104:9104"
  - name: Deploy Prometheus
    docker_container:
      name: prometheus
      image: prom/prometheus
      state: started
      restart_policy: always
      volumes:
        - ~/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml
      ports:
        - "9090:9090"
  - name: Deploy Grafana
    docker_container:
      name: grafana
      image: grafana/grafana
      state: started
      restart_policy: always
      ports:
        - "3000:3000"
```

3. Run Ansible Playbook:

   - Execute the playbook to set up your monitoring stack:

     ansible-playbook mysql_playbook.yml

Phase 4: Create Bash Scripts for Maintenance

1. Database Backup Script:

- Create a backup script backup_script.sh:

```
#!/bin/bash
BACKUP_DIR="/backup/mysql"
DB_USER="root"
DB_PASS="my-secret-pw"
TIMESTAMP=$(date +"%F")
mkdir -p $BACKUP_DIR
mysqldump -u $DB_USER -p$DB_PASS --all-databases | gzip > $BACKUP_DIR/db_backup_$TIMESTAMP.sql.gz
```

2. Monitoring Script:

- Create a monitoring script monitoring_script.sh:

```
#!/bin/bash
MYSQL_STATUS=$(mysqladmin -u root -p my-secret-pw status)
echo "MySQL Status: $MYSQL_STATUS" >> /var/log/mysql-monitor.log
```

3. Schedule Scripts with Cron:

- Open the crontab editor:

```
crontab -e
```

- Add entries to schedule the backup and monitoring scripts:

```
# Backup script every day at midnight
0 0 * * * /path/to/backup_script.sh
# Monitoring script every hour
0 * * * * /path/to/monitoring_script.sh
```

Phase 5: Store Monitoring Data in MySQL

1. Create Performance Data Table:

- Use the MySQL client to create a table for storing performance data:

```
CREATE TABLE performance_data (
  id INT AUTO_INCREMENT PRIMARY KEY,
  query VARCHAR(255),
  execution_time FLOAT,
  timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

2. Insert Performance Metrics:

- Write a script to periodically insert performance metrics:

```
#!/bin/bash
QUERY="SELECT COUNT(*) FROM my_table"
 EXECUTION_TIME=$( { time mysql -u root -p my-secret-pw -e "$QUERY"; } 2>&1 | grep real | awk '{print $2}' )
  mysql -u root -p my-secret-pw -e "INSERT INTO performance_data (query, execution_time) VALUES ('$QUERY', $EXECUTION_TIME)"
```

Conclusion

By following this detailed step-by-step guide, you will have implemented a comprehensive Database Monitoring and Performance Optimization solution utilizing MySQL, Docker, Ansible, Bash scripting, Prometheus, and Grafana. This solution allows you to monitor performance metrics in real-time, automate backups, and continuously optimize database operations. Feel free to adjust any steps to fit your specific environment and requirements!