

Lesson End Project

Monitoring MySQL Database Using Prometheus and Setting up Alerts

Project agenda: To configure Prometheus for monitoring a MySQL database, collect key performance metrics using MySQL Exporter, and set up alerting rules using Alertmanager to notify of any issues for proactive database management and performance optimization

Description: You are a DevOps engineer responsible for maintaining the performance and stability of MySQL databases within your organization. Due to performance bottlenecks during peak traffic periods, you have been tasked with implementing a monitoring solution. By using Prometheus and MySQL Exporter, you will gather essential MySQL metrics, analyze performance, and set up alerting rules through Alertmanager to notify the team in real time of any potential performance degradation or failures.

Tools required: Linux operating system and Docker

Prerequisites: Ensure that Docker is installed in the lab before proceeding.

Expected deliverables: A setup for MySQL monitoring with Prometheus, including MySQL Exporter configuration, PromQL queries for performance analysis, and alerting rules for key MySQL performance indicators

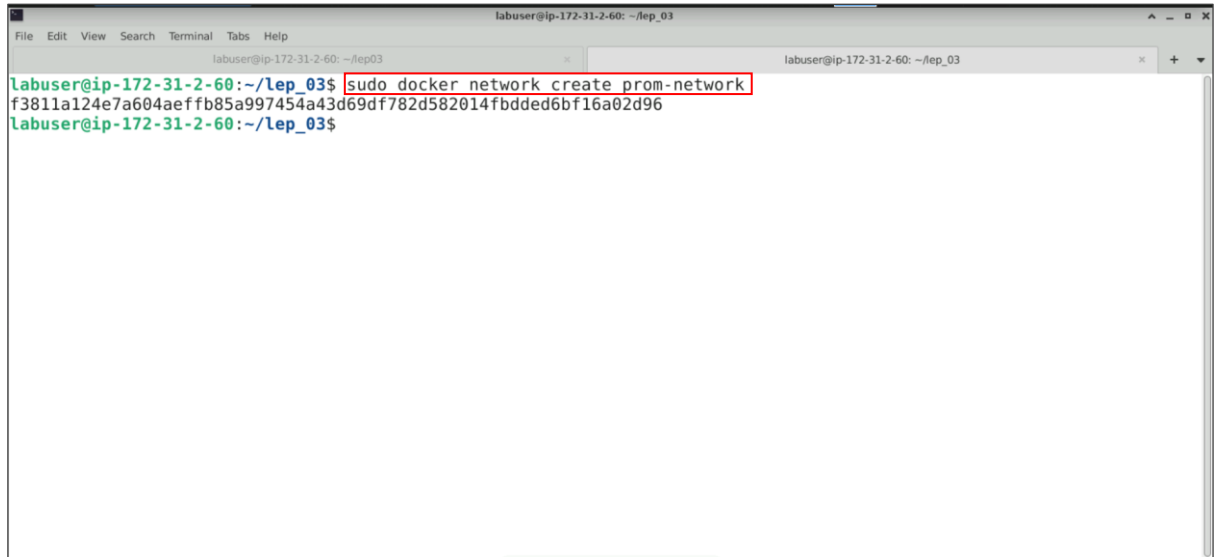
Steps to be followed:

1. Set up MySQL using Docker
2. Set up MySQL Server Exporter with Docker
3. Configure Alertmanager using Docker
4. Configure and start Prometheus in a Docker container
5. Explore MySQL Exporter metrics and alerting config on Prometheus
6. Simulate MySQL failure

Step 1: Set up MySQL using Docker

1.1 Open the terminal and run the following command to create a Docker network:

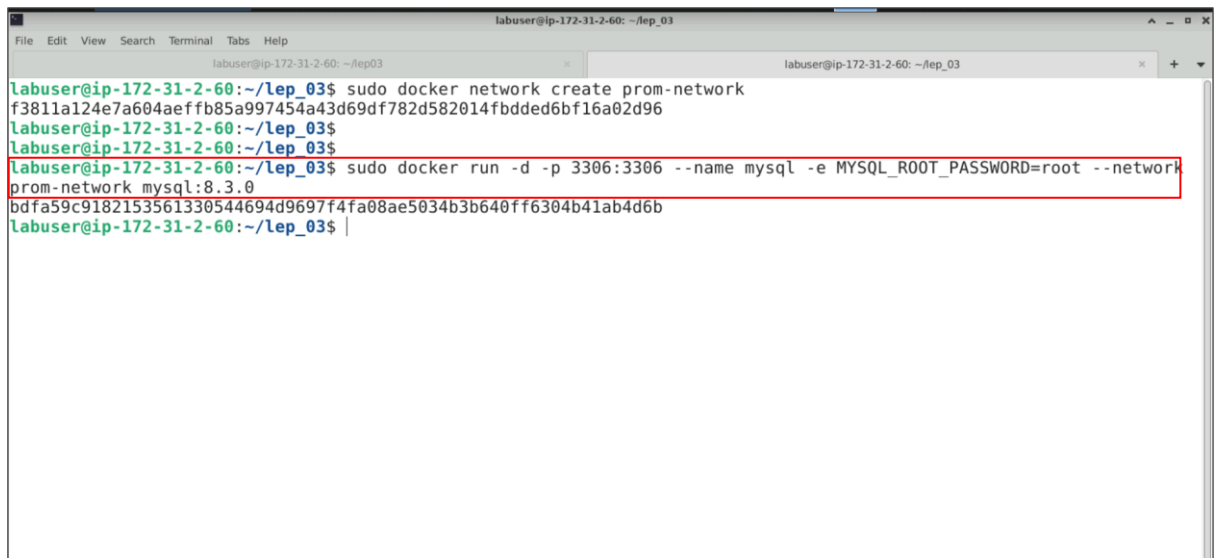
sudo docker network create prom-network

A terminal window titled 'labuser@ip-172-31-2-60: ~/lep_03' with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help). The terminal shows the command 'sudo docker network create prom-network' being executed. The output is a long alphanumeric string: 'f3811a124e7a604aefb85a997454a43d69df782d582014fbdded6bf16a02d96'. The prompt is 'labuser@ip-172-31-2-60:~/lep_03\$'.

```
labuser@ip-172-31-2-60:~/lep_03$ sudo docker network create prom-network
f3811a124e7a604aefb85a997454a43d69df782d582014fbdded6bf16a02d96
labuser@ip-172-31-2-60:~/lep_03$
```

1.2 Run the following command to set up MySQL in a Docker container:

**sudo docker run -d -p 3306:3306 --name mysql -e
MYSQL_ROOT_PASSWORD=root --network prom-network mysql:8.3.0**

A terminal window titled 'labuser@ip-172-31-2-60: ~/lep_03' with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help). The terminal shows the command 'sudo docker run -d -p 3306:3306 --name mysql -e MYSQL_ROOT_PASSWORD=root --network prom-network mysql:8.3.0' being executed. The output is a long alphanumeric string: 'bdfa59c9182153561330544694d9697f4fa08ae5034b3b640ff6304b41ab4d6b'. The prompt is 'labuser@ip-172-31-2-60:~/lep_03\$'.

```
labuser@ip-172-31-2-60:~/lep_03$ sudo docker network create prom-network
f3811a124e7a604aefb85a997454a43d69df782d582014fbdded6bf16a02d96
labuser@ip-172-31-2-60:~/lep_03$
labuser@ip-172-31-2-60:~/lep_03$ sudo docker run -d -p 3306:3306 --name mysql -e MYSQL_ROOT_PASSWORD=root --network
prom-network mysql:8.3.0
bdfa59c9182153561330544694d9697f4fa08ae5034b3b640ff6304b41ab4d6b
labuser@ip-172-31-2-60:~/lep_03$
```

Step 2: Set up MySQL Server Exporter with Docker

2.1 Run the following command to list all the running containers to identify the MySQL container id:

docker ps -a

```
labuser@ip-172-31-2-60: ~/lep_03
labuser@ip-172-31-2-60:~/lep_03$ sudo docker run -d -p 3306:3306 --name mysql -e MYSQL_ROOT_PASSWORD=root --network prom-network mysql:8.3.0
bdfa59c9182153561330544694d9697f4fa08ae5034b3b640ff6304b41ab4d6b
labuser@ip-172-31-2-60:~/lep_03$
labuser@ip-172-31-2-60:~/lep_03$ docker exec -it 679ad782cde7 /bin/bash
Error response from daemon: No such container: 679ad782cde7
labuser@ip-172-31-2-60:~/lep_03$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS
bdfa59c91821   mysql:8.3.0                        "docker-entrypoint.s..." About a minute ago Up About a minute
0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   mysql
37c9b2772110   prom/prometheus:latest            "/bin/prometheus --c..." 3 hours ago    Exited (0) About an hour
ago
48ac42966aa4   metrics-demo_flask-app            "python3 server.py"      3 hours ago    Exited (137) About an ho
ur ago
d1f548ce38be   bitnami/node-exporter:latest       "/opt/bitnami/node-e..." 3 hours ago    Exited (2) About an hour
ago
09740a73cc4d   grafana/grafana                   "/run.sh"                 3 hours ago    Exited (0) About an hour
ago
labuser@ip-172-31-2-60:~/lep_03$ docker exec -it bdfa59c91821 /bin/bash
bash-4.4#
```

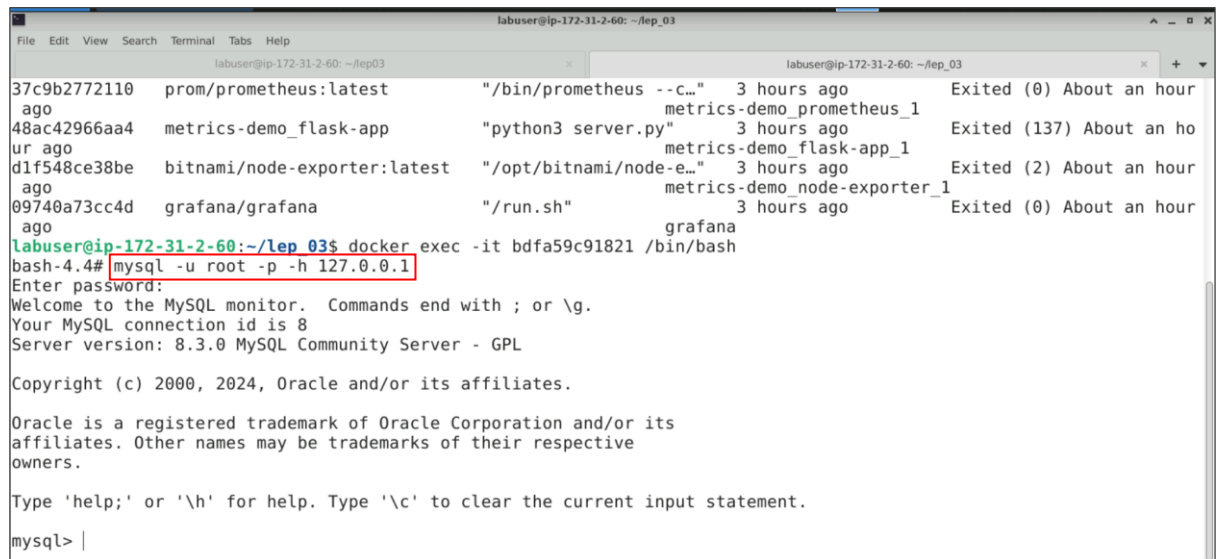
2.2 Run the given command to open an interactive Bash shell inside the MySQL Docker container:

docker exec -it bdfa59c91821 /bin/bash

```
labuser@ip-172-31-2-60: ~/lep_03
labuser@ip-172-31-2-60:~/lep_03$ sudo docker run -d -p 3306:3306 --name mysql -e MYSQL_ROOT_PASSWORD=root --network prom-network mysql:8.3.0
bdfa59c9182153561330544694d9697f4fa08ae5034b3b640ff6304b41ab4d6b
labuser@ip-172-31-2-60:~/lep_03$
labuser@ip-172-31-2-60:~/lep_03$ docker exec -it 679ad782cde7 /bin/bash
Error response from daemon: No such container: 679ad782cde7
labuser@ip-172-31-2-60:~/lep_03$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS
bdfa59c91821   mysql:8.3.0                        "docker-entrypoint.s..." About a minute ago Up About a minute
0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   mysql
37c9b2772110   prom/prometheus:latest            "/bin/prometheus --c..." 3 hours ago    Exited (0) About an hour
ago
48ac42966aa4   metrics-demo_flask-app            "python3 server.py"      3 hours ago    Exited (137) About an ho
ur ago
d1f548ce38be   bitnami/node-exporter:latest       "/opt/bitnami/node-e..." 3 hours ago    Exited (2) About an hour
ago
09740a73cc4d   grafana/grafana                   "/run.sh"                 3 hours ago    Exited (0) About an hour
ago
labuser@ip-172-31-2-60:~/lep_03$ docker exec -it bdfa59c91821 /bin/bash
bash-4.4#
```

Note: Make sure to replace the **CONTAINER ID** in the above command with the correct **CONTAINER ID** of your MySQL Docker container per the list.

- 2.3 Run the following command to connect to the MySQL server using the root credentials (**password=root**):
mysql -u root -p -h 127.0.0.1



```
labuser@ip-172-31-2-60: ~/lep_03
37c9b2772110 prom/prometheus:latest "/bin/prometheus --c..." 3 hours ago Exited (0) About an hour ago
48ac42966aa4 metrics-demo_flask-app "python3 server.py" 3 hours ago Exited (137) About an hour ago
d1f548ce38be bitnami/node-exporter:latest "/opt/bitnami/node-e..." 3 hours ago Exited (2) About an hour ago
09740a73cc4d grafana/grafana "/run.sh" 3 hours ago Exited (0) About an hour ago
labuser@ip-172-31-2-60:~/lep_03$ docker exec -it bdfa59c91821 /bin/bash
bash-4.4# mysql -u root -p -h 127.0.0.1
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.3.0 MySQL Community Server - GPL

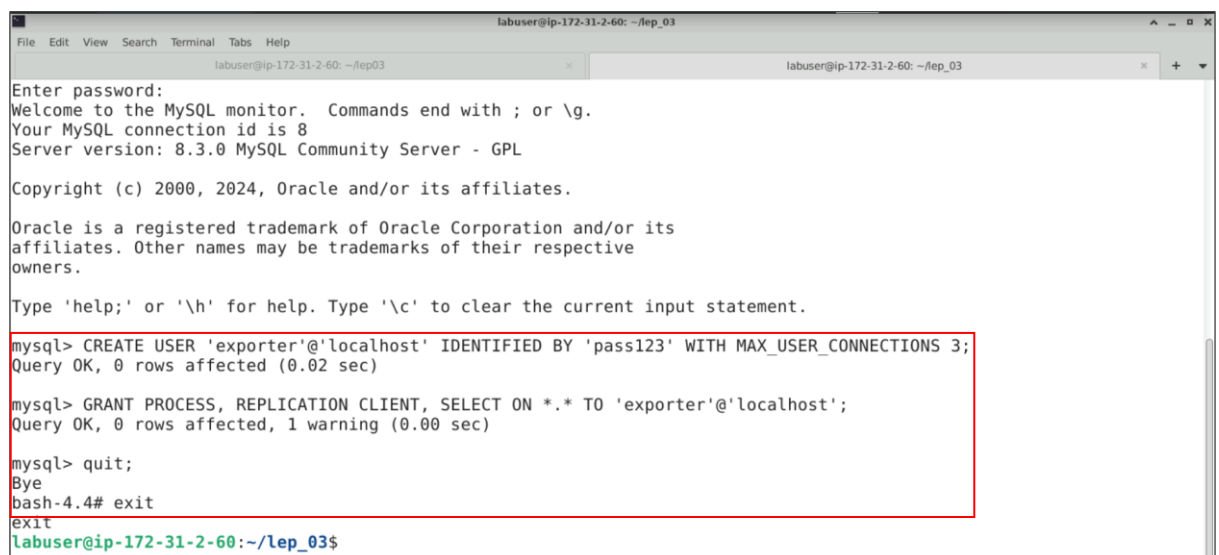
Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> |
```

- 2.4 Enter the following commands to create a new MySQL user **exporter** with a password **pass123** and grant privileges to the **exporter** user on all databases; exit MySQL and then exit the bash session
CREATE USER 'exporter'@'localhost' IDENTIFIED BY 'pass123' WITH MAX_USER_CONNECTIONS 3;
GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'exporter'@'localhost';
quit;
exit



```
labuser@ip-172-31-2-60: ~/lep_03
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE USER 'exporter'@'localhost' IDENTIFIED BY 'pass123' WITH MAX_USER_CONNECTIONS 3;
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'exporter'@'localhost';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> quit;
Bye
bash-4.4# exit
exit
labuser@ip-172-31-2-60:~/lep_03$
```

```
sudo vim config.my-cnf
```

```
labuser@ip-172-31-2-60: ~/lep_03
File Edit View Search Terminal Tabs Help
labuser@ip-172-31-2-60: ~/lep_03
labuser@ip-172-31-2-60: ~/lep_03

Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE USER 'exporter'@'localhost' IDENTIFIED BY 'pass123' WITH MAX_USER_CONNECTIONS 3;
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'exporter'@'localhost';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> quit;
Bye
bash-4.4# exit
exit
labuser@ip-172-31-2-60:~/lep_03$ sudo vim config.my.cnf
```

2.6 Copy and paste the following configuration, then save and exit the file:

```
[client]
user = exporter
password = pass123
host = mysql
```



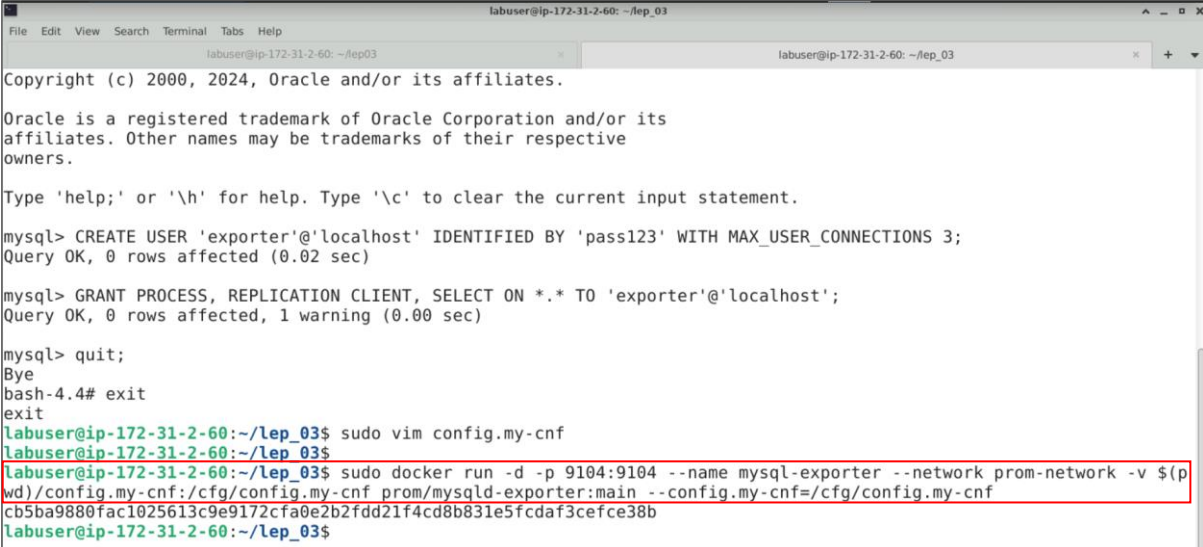
The screenshot shows a terminal window with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help) and a title bar (labuser@ip-172-31-2-60: ~/lep_03). The terminal has two tabs: 'labuser@ip-172-31-2-60: ~/lep_03' and 'labuser@ip-172-31-2-60: ~/lep_03'. The first tab is active and contains the following text:

```
[client]
user = exporter
password = pass123
host = mysql
```

The text is enclosed in a red rectangular box. Below the box, there are several blue tilde characters (~) indicating a list of files or directories. In the bottom right corner, the text '5,0-1' and 'All' are visible.

2.7 Run the following command to initialize the MySQL exporter container:

```
sudo docker run -d -p 9104:9104 --name mysql-exporter --network prom-network  
-v $(pwd)/config.my-cnf:/cfg/config.my-cnf prom/mysql-exporter:main --  
config.my-cnf=/cfg/config.my-cnf
```

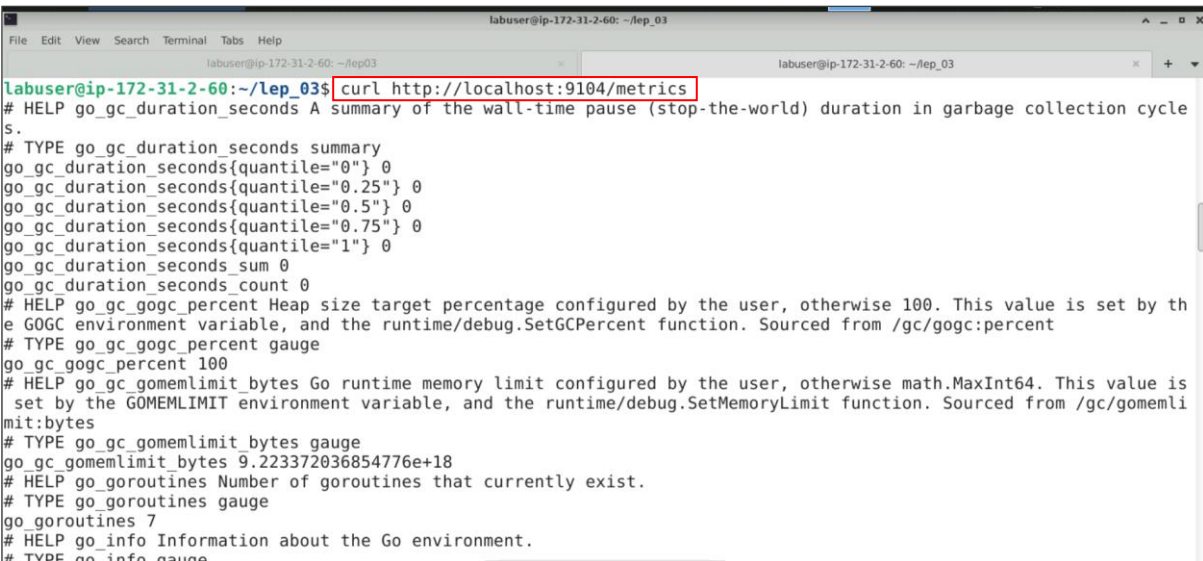


A terminal window titled 'labuser@ip-172-31-2-60: ~/lep_03' showing the following commands and output:

```
Copyright (c) 2000, 2024, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> CREATE USER 'exporter'@'localhost' IDENTIFIED BY 'pass123' WITH MAX_USER_CONNECTIONS 3;  
Query OK, 0 rows affected (0.02 sec)  
  
mysql> GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'exporter'@'localhost';  
Query OK, 0 rows affected, 1 warning (0.00 sec)  
  
mysql> quit;  
Bye  
bash-4.4# exit  
exit  
labuser@ip-172-31-2-60:~/lep_03$ sudo vim config.my-cnf  
labuser@ip-172-31-2-60:~/lep_03$  
labuser@ip-172-31-2-60:~/lep_03$ sudo docker run -d -p 9104:9104 --name mysql-exporter --network prom-network -v $(p  
wd)/config.my-cnf:/cfg/config.my-cnf prom/mysql-exporter:main --config.my-cnf=/cfg/config.my-cnf  
cb5ba9880fac1025613c9e9172cfa0e2b2fdd21f4cd8b831e5fcdaf3cefce38b  
labuser@ip-172-31-2-60:~/lep_03$
```

2.8 Use the following command to verify if the MySQL Exporter is running:

```
curl http://localhost:9104/metrics
```



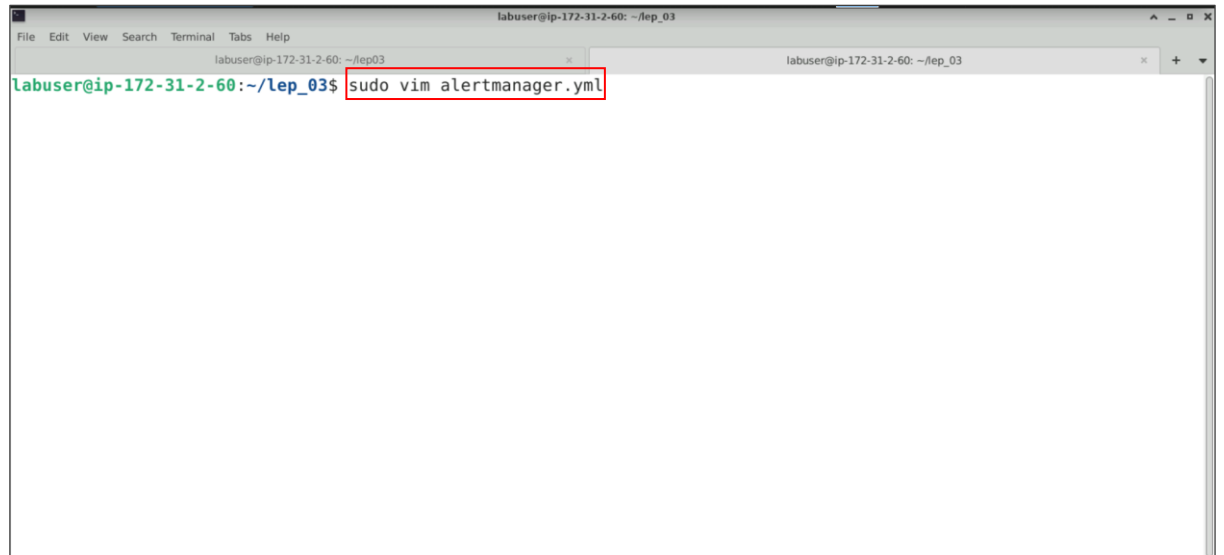
A terminal window titled 'labuser@ip-172-31-2-60: ~/lep_03' showing the output of the command `curl http://localhost:9104/metrics`:

```
labuser@ip-172-31-2-60:~/lep_03$ curl http://localhost:9104/metrics  
# HELP go_gc_duration_seconds A summary of the wall-time pause (stop-the-world) duration in garbage collection cycle  
s.  
# TYPE go_gc_duration_seconds summary  
go_gc_duration_seconds{quantile="0"} 0  
go_gc_duration_seconds{quantile="0.25"} 0  
go_gc_duration_seconds{quantile="0.5"} 0  
go_gc_duration_seconds{quantile="0.75"} 0  
go_gc_duration_seconds{quantile="1"} 0  
go_gc_duration_seconds_sum 0  
go_gc_duration_seconds_count 0  
# HELP go_gc_gogc_percent Heap size target percentage configured by the user, otherwise 100. This value is set by th  
e GOGC environment variable, and the runtime/debug.SetGCPercent function. Sourced from /gc/gogc:percent  
# TYPE go_gc_gogc_percent gauge  
go_gc_gogc_percent 100  
# HELP go_gc_gomemlimit_bytes Go runtime memory limit configured by the user, otherwise math.MaxInt64. This value is  
set by the GOMEMLIMIT environment variable, and the runtime/debug.SetMemoryLimit function. Sourced from /gc/gomemli  
mit:bytes  
# TYPE go_gc_gomemlimit_bytes gauge  
go_gc_gomemlimit_bytes 9.223372036854776e+18  
# HELP go_goroutines Number of goroutines that currently exist.  
# TYPE go_goroutines gauge  
go_goroutines 7  
# HELP go_info Information about the Go environment.  
# TYPE go_info gauge
```

Step 3: Configure Alertmanager using Docker

3.1 Run the following command to create the configuration file **alertmanager.yml** using **vim** editor:

sudo vim alertmanager.yml



3.2 Copy and paste the following configuration, then save and exit the file:

route:

receiver: 'mail'

repeat_interval: 4h

group_by: [alertname]

receivers:

- **name:** 'mail'

email_configs:


- **smarthost:** 'smtp.gmail.com:587'

auth_username: '<your_username>'

auth_password: "<your-password>"

from: '<your-email>'

to: '<some-email>'



The screenshot shows a terminal window with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help) and a title bar (labuser@ip-172-31-2-60: ~/lep_03). The terminal content is as follows:

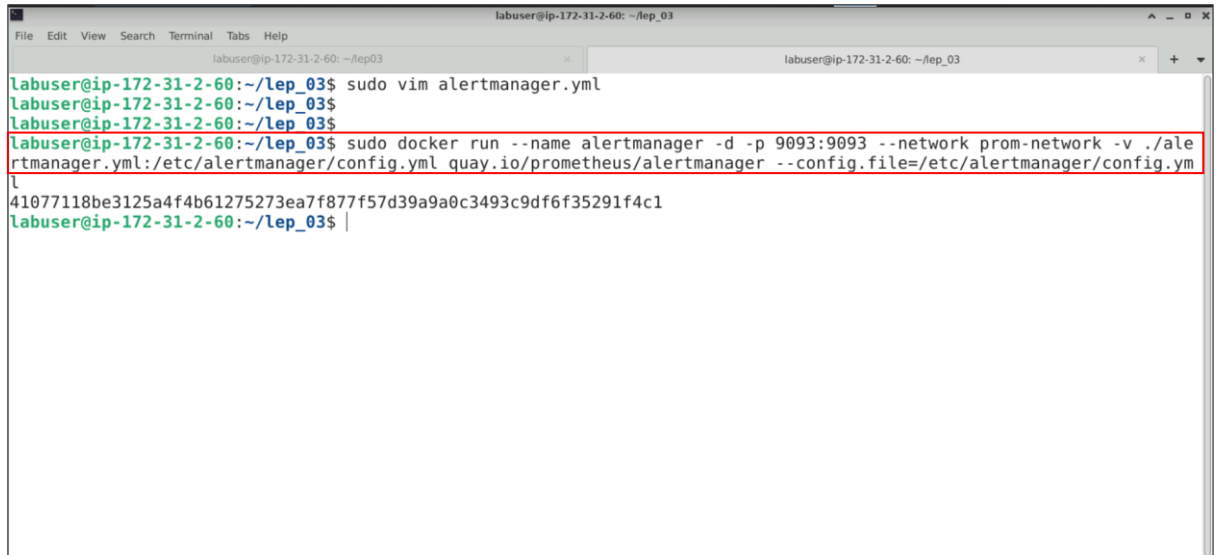
```
route:
  receiver: 'mail'
  repeat_interval: 4h
  group_by: [ alertname ]

receivers:
  - name: 'mail'
    email_configs:
      - smarthost: 'smtp.gmail.com:587'
        auth_username: [REDACTED]
        auth_password: "<your-password>"
        from: [REDACTED]
        to: [REDACTED]
```

At the bottom of the terminal, there is a status bar with the text "-- INSERT --" on the left, "15,1" in the center, and "All" on the right.

3.3 Run the following command to start the Alertmanager container:

```
sudo docker run --name alertmanager -d -p 9093:9093 --network prom-network -v ./alertmanager.yml:/etc/alertmanager/config.yml quay.io/prometheus/alertmanager --config.file=/etc/alertmanager/config.yml
```



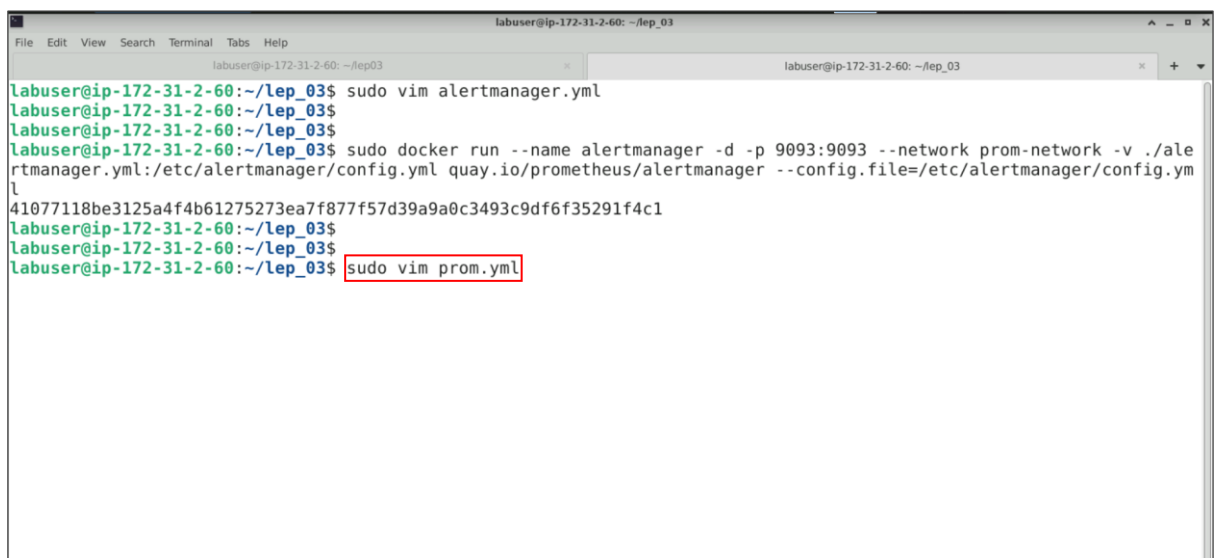
A terminal window titled 'labuser@ip-172-31-2-60: ~/lep_03' showing a series of commands. The first three commands are 'sudo vim alertmanager.yml', 'labuser@ip-172-31-2-60:~/lep_03\$', and 'labuser@ip-172-31-2-60:~/lep_03\$'. The fourth command, 'labuser@ip-172-31-2-60:~/lep_03\$ sudo docker run --name alertmanager -d -p 9093:9093 --network prom-network -v ./alertmanager.yml:/etc/alertmanager/config.yml quay.io/prometheus/alertmanager --config.file=/etc/alertmanager/config.yml', is highlighted with a red box. The output of this command is a long alphanumeric string: '41077118be3125a4f4b61275273ea7f877f57d39a9a0c3493c9df6f35291f4c1'. The prompt then returns to 'labuser@ip-172-31-2-60:~/lep_03\$'.

Step 4: Configure and start Prometheus in a Docker container

4.1 Run the following command to use vim editor to create the configuration file

prom.yml for Prometheus:

```
sudo vim prom.yml
```



A terminal window titled 'labuser@ip-172-31-2-60: ~/lep_03' showing a series of commands. The first four commands are 'sudo vim alertmanager.yml', 'labuser@ip-172-31-2-60:~/lep_03\$', 'labuser@ip-172-31-2-60:~/lep_03\$', and 'labuser@ip-172-31-2-60:~/lep_03\$ sudo docker run --name alertmanager -d -p 9093:9093 --network prom-network -v ./alertmanager.yml:/etc/alertmanager/config.yml quay.io/prometheus/alertmanager --config.file=/etc/alertmanager/config.yml'. The output of the last command is the same long alphanumeric string as in the previous screenshot. The prompt then returns to 'labuser@ip-172-31-2-60:~/lep_03\$'. The fifth command, 'labuser@ip-172-31-2-60:~/lep_03\$ sudo vim prom.yml', is highlighted with a red box.

4.2 Copy and paste the following configuration, then save and exit the file:

```
global:
  scrape_interval: 15s # By default, scrape targets every 15 seconds.

  # Attach these labels to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    monitor: 'codelab-monitor'

rule_files:
  - "rules.yml"
alerting:
  alertmanagers:
    - static_configs:
        - targets: ["alertmanager:9093"]

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped
  # from this config.
  - job_name: 'prometheus'

    # Override the global default and scrape targets from this job every 5 seconds.
    scrape_interval: 5s

    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'mysql'

    params:
      auth_module: [client]

    scrape_interval: 5s

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

    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        # The mysqld_exporter host:port
        replacement: mysql-exporter:9104
```

```
labuser@ip-172-31-2-60: ~/lep_03
File Edit View Search Terminal Tabs Help
labuser@ip-172-31-2-60: ~/lep_03
labuser@ip-172-31-2-60: ~/lep_03

global:
  scrape_interval: 15s # By default, scrape targets every 15 seconds.

  # Attach these labels to any time series or alerts when communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
    monitor: 'codeLab-monitor'

rule_files:
  - "rules.yml"
alerting:
  alertmanagers:
    - static_configs:
        - targets: ["alertmanager:9093"]

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label 'job=<job_name>' to any timeseries scraped from this config.
  - job_name: 'prometheus'

    # Override the global default and scrape targets from this job every 5 seconds.
    scrape_interval: 5s

    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'mysql'

    params:
      auth_module: [client]

    scrape_interval: 5s

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

    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target

-- INSERT --
1,1 Top
```

4.3 Run the following command to create the **rules.yml** configuration file using **vim** editor:

sudo vim rules.yml

```
labuser@ip-172-31-2-60: ~/lep_03
File Edit View Search Terminal Tabs Help
labuser@ip-172-31-2-60: ~/lep_03
labuser@ip-172-31-2-60: ~/lep_03
labuser@ip-172-31-2-60: ~/lep_03$ sudo docker run --name alertmanager -d -p 9093:9093 --network prom-network -v ./alertmanager.yml:/etc/alertmanager/config.yml quay.io/prometheus/alertmanager --config.file=/etc/alertmanager/config.yml
41077118be3125a4f4b61275273ea7f877f57d39a9a0c3493c9df6f35291f4c1
labuser@ip-172-31-2-60: ~/lep_03$
labuser@ip-172-31-2-60: ~/lep_03$
labuser@ip-172-31-2-60: ~/lep_03$ sudo vim prom.yml
labuser@ip-172-31-2-60: ~/lep_03$ sudo vim rules.yml
```

4.4 Copy and paste the following configuration, then save and exit the file:

groups:

- name: MysqldExporter

rules:

- alert: MysqlDown

```
expr: mysql_up == 0
```

for: 0m

labels:

severity: critical

annotations:

summary: MySQL down (instance {{ \$labels.instance }})

```
description: "MySQL instance is down on {{ $labels.instance }}\n VALUE = {{  
ue }}\n LABELS = {{ $labels }}"
```

```
labuser@ip-172-31-2-60: ~/lep_03
```

```
File Edit View Search Terminal Tabs Help
```

```
labuser@ip-172-31-2-60: ~/lep_03 labuser@ip-172-31-2-60: ~/lep_03
```

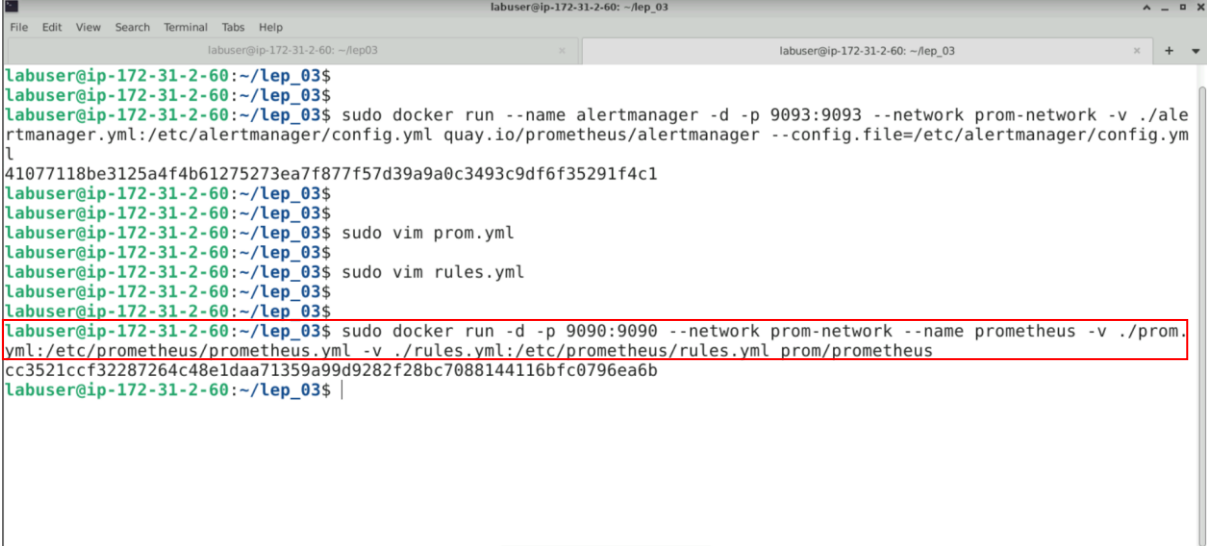
```
groups:  
- name: MysqlExporter  
  rules:  
    - alert: MySQLDown  
      expr: mysql_up == 0  
      for: 0m  
      labels:  
        severity: critical  
      annotations:  
        summary: MySQL down (instance {{ $labels.instance }})  
        description: "MySQL instance is down on {{ $labels.instance }}\n VALUE = {{ $value }}\n LABELS = {{ $lab  
els }}"
```

```
~  
~  
~  
~  
~  
~  
~
```

```
1,1 All
```

4.5 Execute the following command to run the container after both configuration files are created:

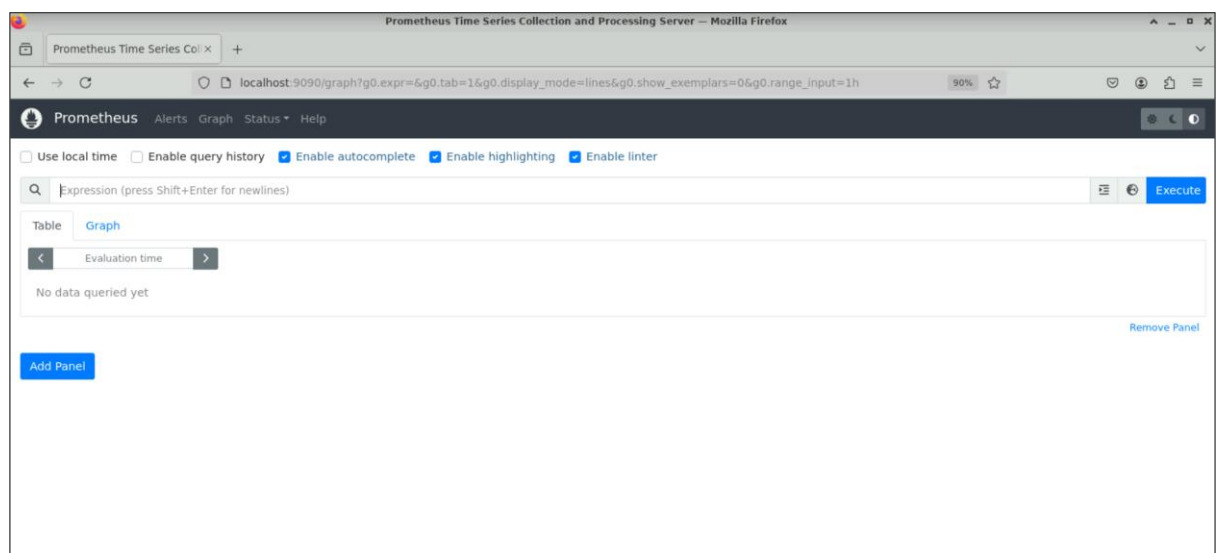
```
sudo docker run -d -p 9090:9090 --network prom-network --name prometheus -v ./prom.yml:/etc/prometheus/prometheus.yml -v ./rules.yml:/etc/prometheus/rules.yml prom/prometheus
```



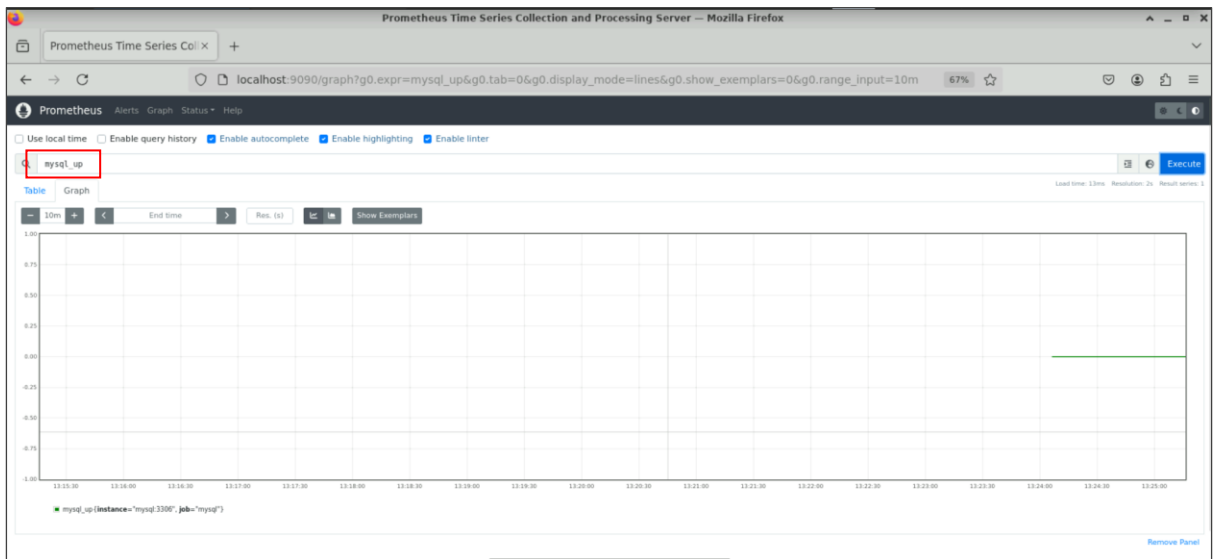
```
labuser@ip-172-31-2-60: ~/lep_03
labuser@ip-172-31-2-60:~/lep_03$
labuser@ip-172-31-2-60:~/lep_03$ sudo docker run --name alertmanager -d -p 9093:9093 --network prom-network -v ./alertmanager.yml:/etc/alertmanager/config.yml quay.io/prometheus/alertmanager --config.file=/etc/alertmanager/config.yml
41077118be3125a4f4b61275273ea7f877f57d39a9a0c3493c9df6f35291f4c1
labuser@ip-172-31-2-60:~/lep_03$
labuser@ip-172-31-2-60:~/lep_03$ sudo vim prom.yml
labuser@ip-172-31-2-60:~/lep_03$ sudo vim rules.yml
labuser@ip-172-31-2-60:~/lep_03$
labuser@ip-172-31-2-60:~/lep_03$ sudo docker run -d -p 9090:9090 --network prom-network --name prometheus -v ./prom.yml:/etc/prometheus/prometheus.yml -v ./rules.yml:/etc/prometheus/rules.yml prom/prometheus
cc3521ccf32287264c48e1daa71359a99d9282f28bc7088144116bfc0796ea6b
labuser@ip-172-31-2-60:~/lep_03$
```

Step 5: Explore MySQL Exporter metrics and alerting config on Prometheus

5.1 Open the browser and access the Prometheus UI using the URL <http://localhost:9090/>

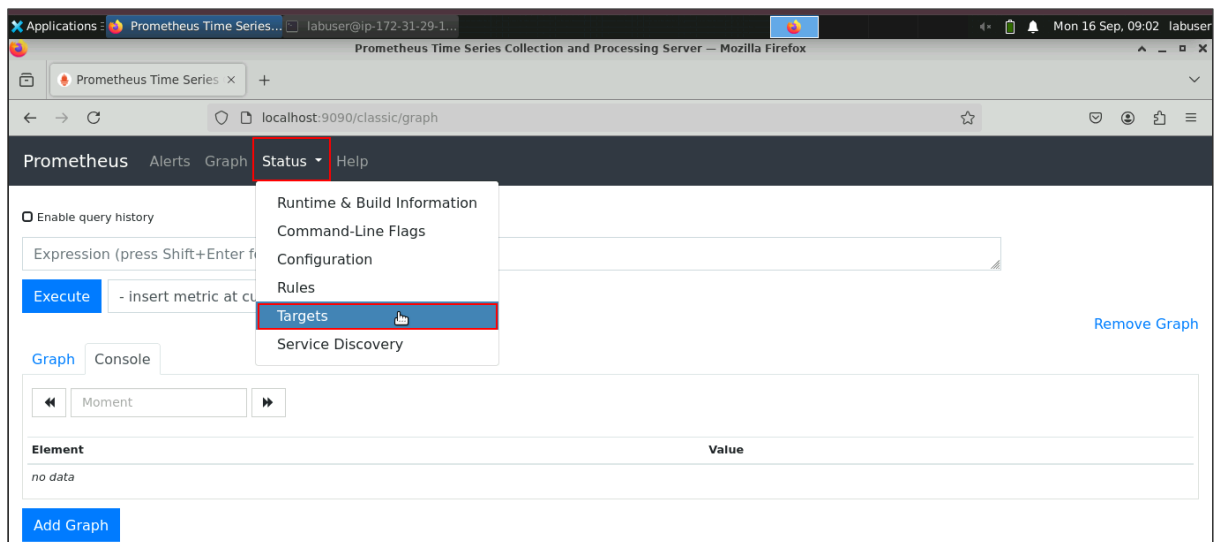


5.2 Execute the following query on the expression bar to visualize MySQL exporter metrics:
mysql_up

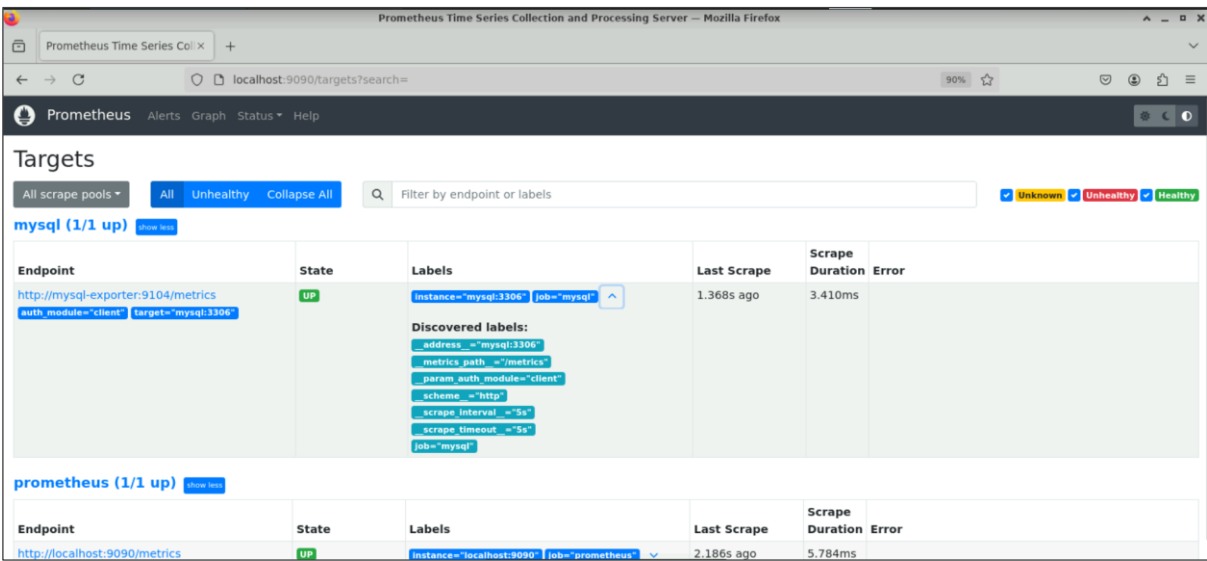


Step 6: Simulate MySQL failure

6.1 Navigate to the **Targets** section in the Prometheus UI and check the status of MySQL exporter



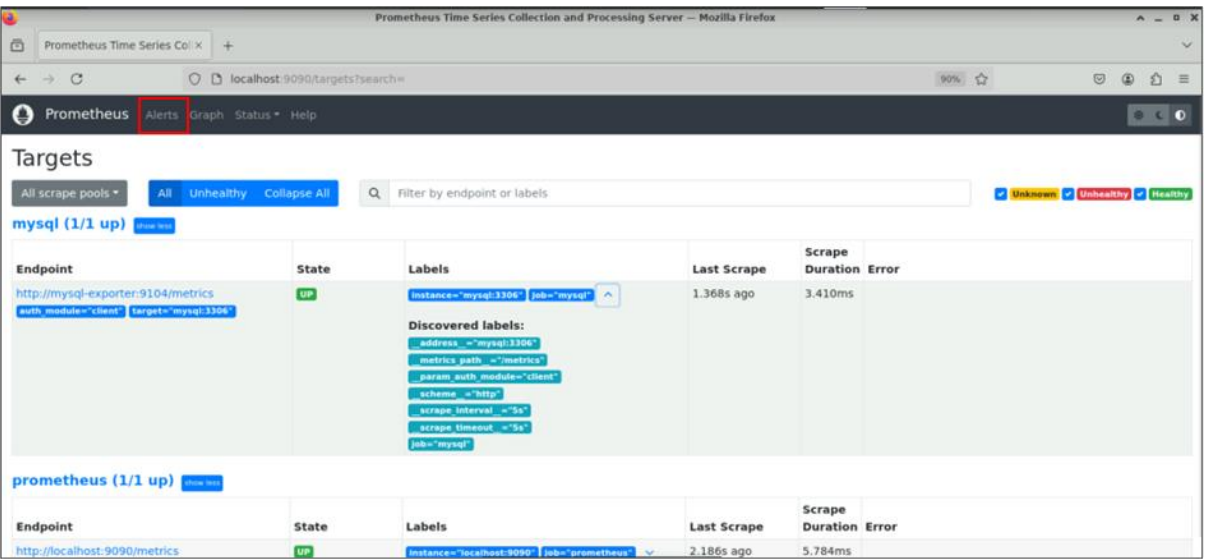
It appears as shown below:



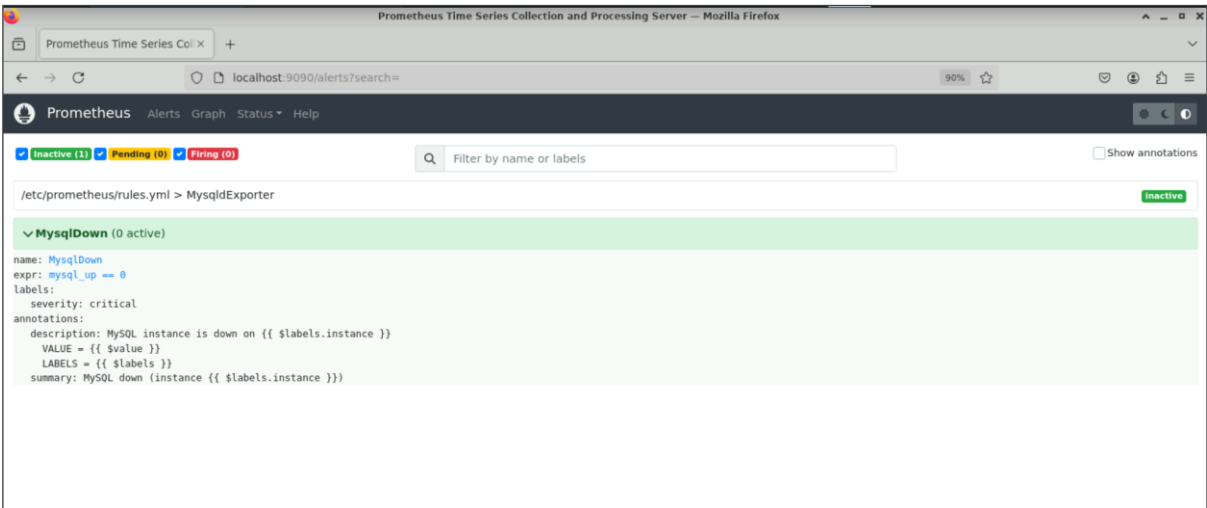
The screenshot shows the Prometheus web interface at localhost:9090. The 'Targets' tab is active, displaying a list of scrape targets. Two targets are listed: 'mysql' and 'prometheus', both with a state of 'UP'. The 'mysql' target has a last scrape time of 1.368s ago and a duration of 3.410ms. The 'prometheus' target has a last scrape time of 2.186s ago and a duration of 5.784ms. The interface includes a search bar, filter buttons, and a table with columns for Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://mysql-exporter:9104/metrics	UP	instance="mysql:3306" job="mysql"	1.368s ago	3.410ms	
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	2.186s ago	5.784ms	

6.2 Navigate to the **Alerts** section and check the alert rule details



This screenshot shows the same Prometheus interface as the previous one, but with the 'Alerts' tab selected in the top navigation bar. The 'Targets' section is still visible in the background, showing the same two targets in an 'UP' state.



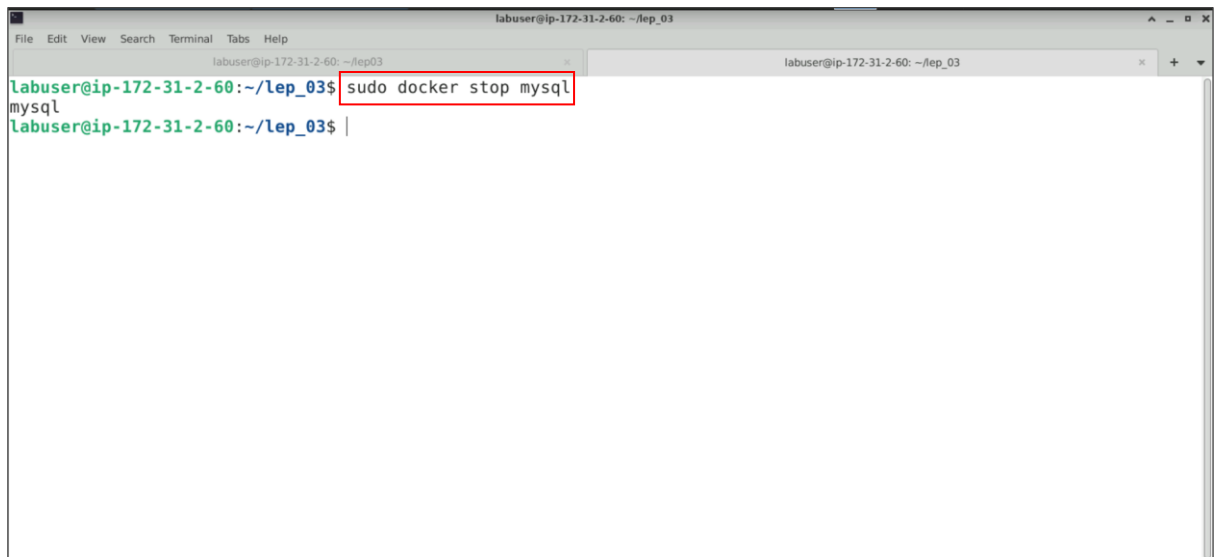
The screenshot shows the 'Alerts' tab in the Prometheus web interface. At the top, there are filters for alert status: 'Inactive (1)', 'Pending (0)', and 'Firing (0)'. Below this, the details for the 'MysqlDown' alert rule are displayed. The rule is currently 'Inactive'. The configuration for the rule is shown in a code block, including its name, expression, labels, severity, and annotations.

```
/etc/prometheus/rules.yml > MysqlDownExporter [inactive]

MysqlDown (0 active)
name: MysqlDown
expr: mysql_up == 0
labels:
severity: critical
annotations:
  description: MySQL instance is down on {{ $labels.instance }}
  VALUE = {{ $value }}
  LABELS = {{ $labels }}
  summary: MySQL down (instance {{ $labels.instance }})
```

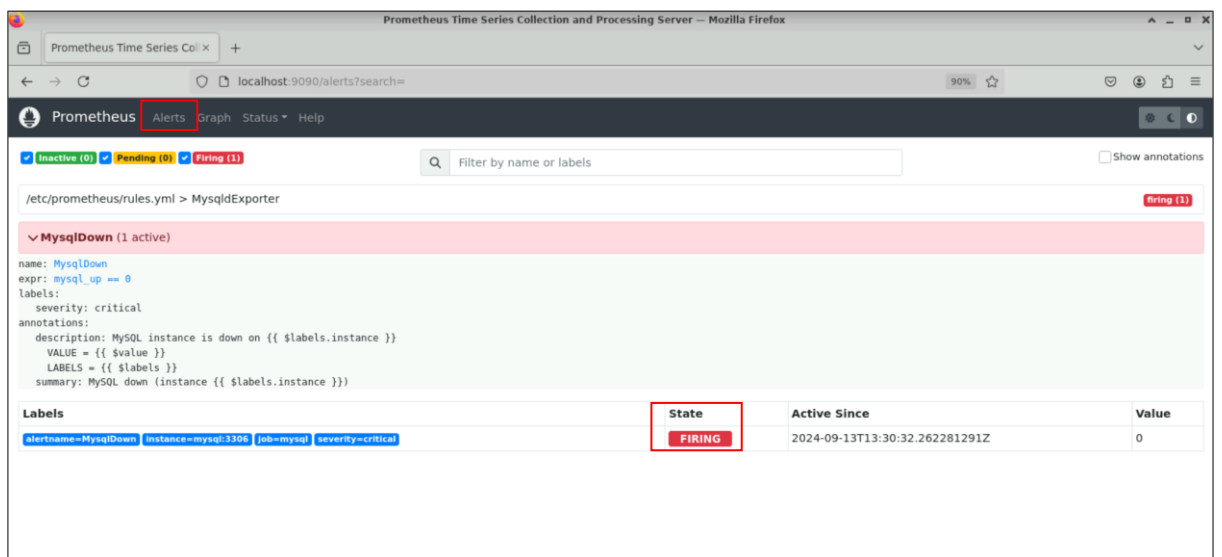
6.3 Navigate back to the terminal and run the following command to stop the MySQL container:

sudo docker stop mysql



```
labuser@ip-172-31-2-60: ~/lep_03
labuser@ip-172-31-2-60:~/lep_03$ sudo docker stop mysql
mysql
labuser@ip-172-31-2-60:~/lep_03$
```

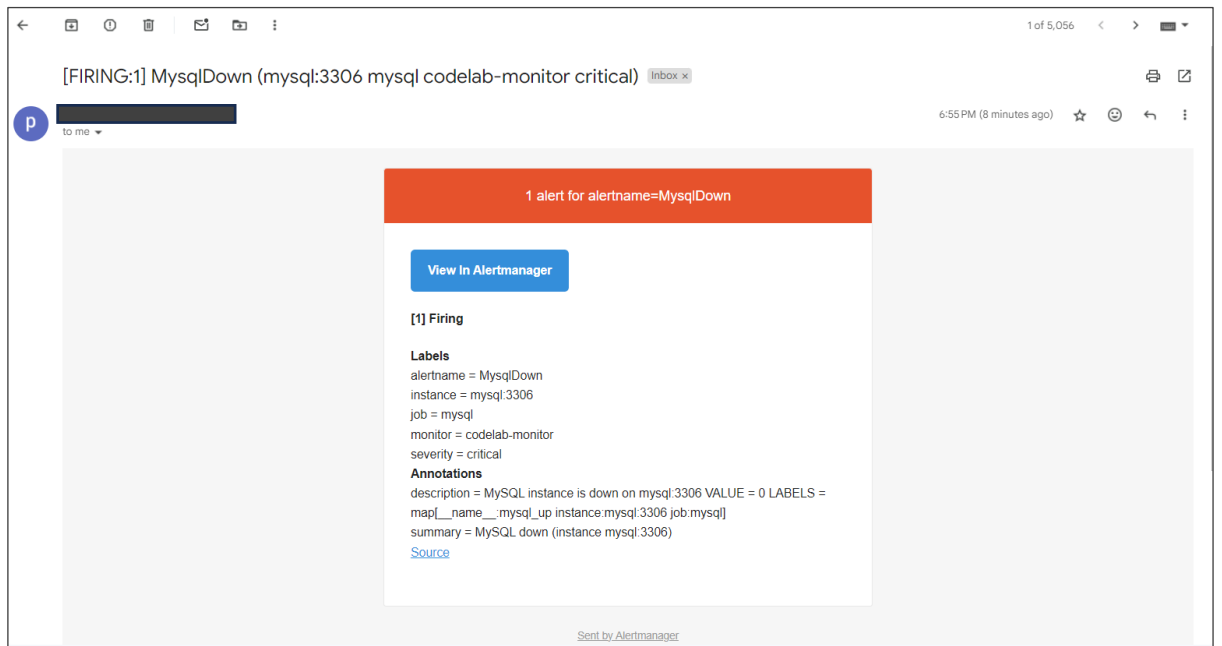
6.4 Check the **Alerts** section in Prometheus for a MySQL alert trigger



The screenshot shows the Prometheus Alerts page in a web browser. The 'Alerts' tab is selected. At the top, there are status indicators: Inactive (0), Pending (0), and Firing (1). A search bar is present with the text 'Filter by name or labels'. Below this, the alert rule is shown: `/etc/prometheus/rules.yml > MysqldExporter`. The alert is titled 'MySQLDown (1 active)'. The details section shows: `name: MySQLDown`, `expr: mysql_up == 0`, `labels: severity: critical`, and `annotations: description: MySQL instance is down on {{ $labels.instance }} VALUE = {{ $value }} LABELS = {{ $labels }} summary: MySQL down (instance {{ $labels.instance }})`. At the bottom, a table displays the alert details:

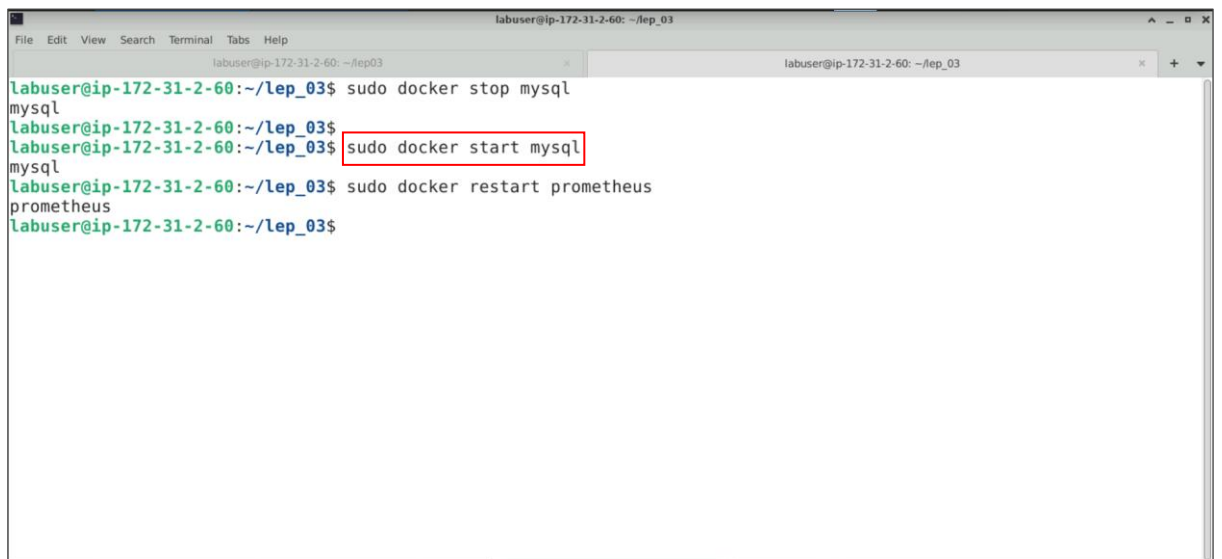
Labels	State	Active Since	Value
Alertname=MySQLDown Instance=mysql:3306 Job=mysql severity=critical	FIRING	2024-09-13T13:30:32.262281291Z	0

6.5 Check your email to verify that the alert notification has been received

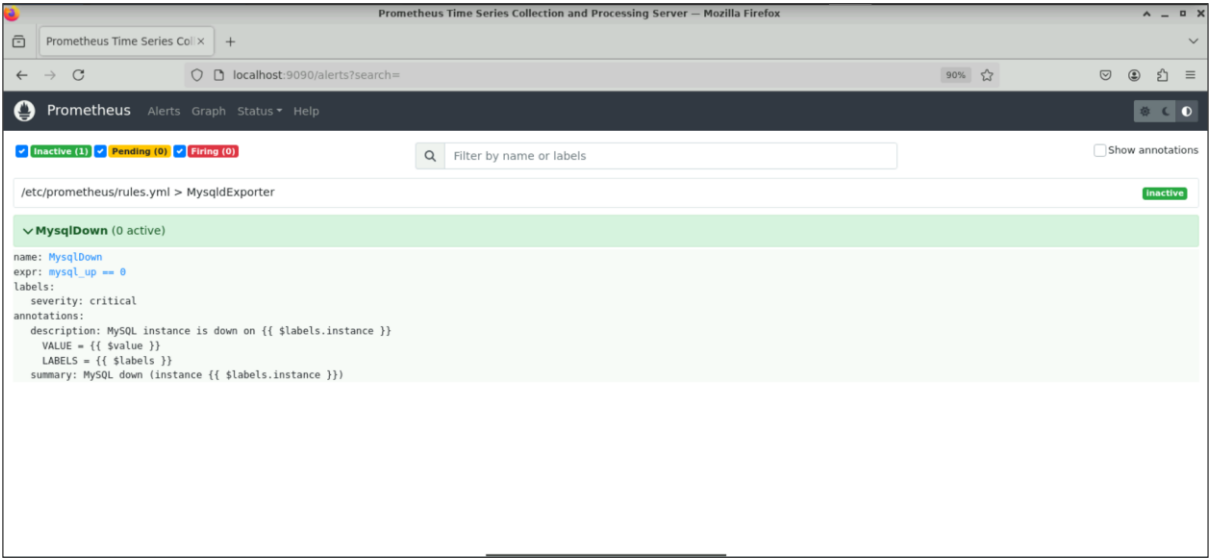


6.6 Run the following command to start the MySQL container to observe the alert status change:

sudo docker start mysql



Observe that the alert status has been updated in the Prometheus UI as shown below:



By following these steps, you have successfully configured Prometheus to monitor a MySQL database, collected key performance metrics using MySQL Exporter, and set up alerting rules with Alertmanager to notify of any issues, ensuring proactive database management and performance optimization.