Crunchy-Postgres-Exporter

Task requirement

To set up Postgres exporter which captures the slow query also.

The **crunchy-postgres-exporter** container provides real time metrics about the PostgreSQL database via an API. These metrics are scraped and stored by a Prometheus time-series database and are then graphed and visualized through the open source data visualizer Grafana.

System Configuration

- RAM: 8 GB
- CPU: 4 cores

Environment details

- Podman - Version 3

List of tools and technologies

- Podman
- Postgres Version 12

Definition of tools

- **Podman** (the POD manager) is an open source tool for developing, managing, and running containers on your Linux systems. Originally developed by Red Hat® engineers along with the open source community, Podman manages the entire container ecosystem using the libpod library.
- **PostgreSQL**, also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance. It was originally named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California, Berkeley.
- /code

Command for the setup or configuration

1. Create Pod:

```
podman pod create --name crunchy-postgres --publish 9090:9090 --publish 9187:9187 --
```

- **podman pod create**: This command is used to create a new pod, which is a group of containers that share the same network namespace. Containers within a pod can communicate with each other using the loopback interface .
- **--name crunchy-postgres**: the pod will be named "crunchy-postgres".
- **--publish 9090:9090**: This flag maps port 9090 from the host to port 9090 within the pod. Port 9090 is commonly used for services like Prometheus.
- **--publish 9187:9187**: This flag maps port 9187 from the host to port 9187 within the pod. Port 9187 is the default port for the PostgreSQL Exporter, which exposes PostgreSQL performance metrics.
- **--publish 5432:5432**: This flag maps port 5432 from the host to port 5432 within the pod. Port 5432 is the default port for PostgreSQL database connections.
- **--publish 3000:3000**: This flag maps port 3000 from the host to port 3000 within the pod. Port 3000 is commonly used for Grafana.

2. Create Postgres container:

```
podman run -d --pod crunchy-postgres --name postgres_crunchy -e "POSTGRES_DB=postgre
```

- **-d**: This flag indicates that the container should run in detached mode (in the background).
- **-pod crunchy-postgres**: This flag specifies that the container should be part of the existing "crunchy-postgres" pod.
- **--name postgres_crunchy**: This flag assigns the name "postgres_crunchy" to the container.
- **-e "POSTGRES_DB=postgres"**: This flag sets the environment variable **POSTGRES_DB** within the container to "postgres",
- **-e "POSTGRES_USER=postgres"**: This flag sets the environment variable **POSTGRES_USER** within the container to "postgres",It indicate the username.
- **-e "POSTGRES_PASSWORD=redhat"**: This flag sets the environment variable **POSTGRES_PASSWORD** within the container to "redhat", which is the password for the PostgreSQL user.
- **-v /home/manoj/shiksha_portal/crunchy/postgres/data:/var/lib/postgresql/data**: This allows you to persist the PostgreSQL data outside the container, ensuring that the data is retained even if the container is removed.
- **docker.io/postgres:12**: This specifies the Docker image to use for the container. the container will be based on the "postgres:12" image from Docker Hub.

3. Do Changes in configuration file:

WE will need to modify our postgresql.conf configuration file to tell PostgreSQL to load

shared libraries.

```
echo "shared_preload_libraries = 'pg_stat_statements,auto_explain'" >> postgresql.co
```

4. Pull crunchy-postgres-exporter image:

Error faced when pulling this image without redhat credentials

```
manoj@keen:~$ podman pull registry.connect.redhat.com/crunchydata/crunchy-postgres-e
Trying to pull registry.connect.redhat.com/crunchydata/crunchy-postgres-exporter:lat
Error: initializing source docker://registry.connect.redhat.com/crunchydata/crunchy-
```

NOTE:- Redhat Credentials are required for pulling this image.

Steps to resolved

1st login in redhat registry then pull the image

```
podman login registry.connect.redhat.com -u manoj@fosteringlinux.com  -p 123455
Login Succeeded!
```

```
podman pull registry.connect.redhat.com/crunchydata/crunchy-postgres-exporter:latest
Trying to pull registry.connect.redhat.com/crunchydata/crunchy-postgres-exporter:lat
Getting image source signatures
Copying blob 07e9ce81867b done
Copying blob 3a0c43549655 done
Writing manifest to image destination
Storing signatures
```

5. Create a demo container of crunchy for setup.sql.

```
podman run -itd --pod crunchy-postgres --name crunchy -e EXPORTER_PG_PASSWORD=redhat
```

6. Get the setup.sql from /opt/cpm/conf/pgxx/setup.sql from the crunchy container according to your postgres version.

```
podman cp crunchy:/opt/cpm/conf/pg12/setup.sql .
```

NOTE:- **setup.sql** Creates **ccp_monitoring** role with all necessary grants. Creates all necessary database objects (functions, tables, etc) required for monitoring.

This will copy the **setup.sql** file from the container to the directory where you executed the command.

NOTE:- Dont miss the (.) which is taking place after setup.sql because it represent copy in your current directory.

7. Remove the test container of crunchy:

```
podman rm -f crunchy
```

8. Push setup.sql in postgres database:

Get the **setup.sql** from **/opt/cpm/conf/pg12/setup.sql** from the exporter container.

```
 psql -h 127.0.0.1 -U postgres -d template1 < setup.sql
```

This command is used to execute SQL commands from the setup.sql file within a PostgreSQL database, making changes to the database schema, data, or settings as specified in the SQL file.

9. Create Extension:

```
psql -h 127.0.0.1 -U postgres -d template1 -c "CREATE EXTENSION pg_stat_statements;"
```

This command is used to enable the **pg_stat_statements** extension in the PostgreSQL database, allowing you to collect statistics about executed SQL statements for performance analysis.

10. Create password for user ccp_monitoring:

```
psql -h 127.0.0.1 -U postgres -d postgres
```

Here We will login in postgres database and then Create password for user ccp_monitoring and also will create database name yogendra

```
postgres=# \password ccp_monitoring
Enter new password for user "ccp_monitoring":
Enter it again:
postgres=# create database yogendra;
CREATE DATABASE
```

11. Now create crunchy-postgres-exporter container:

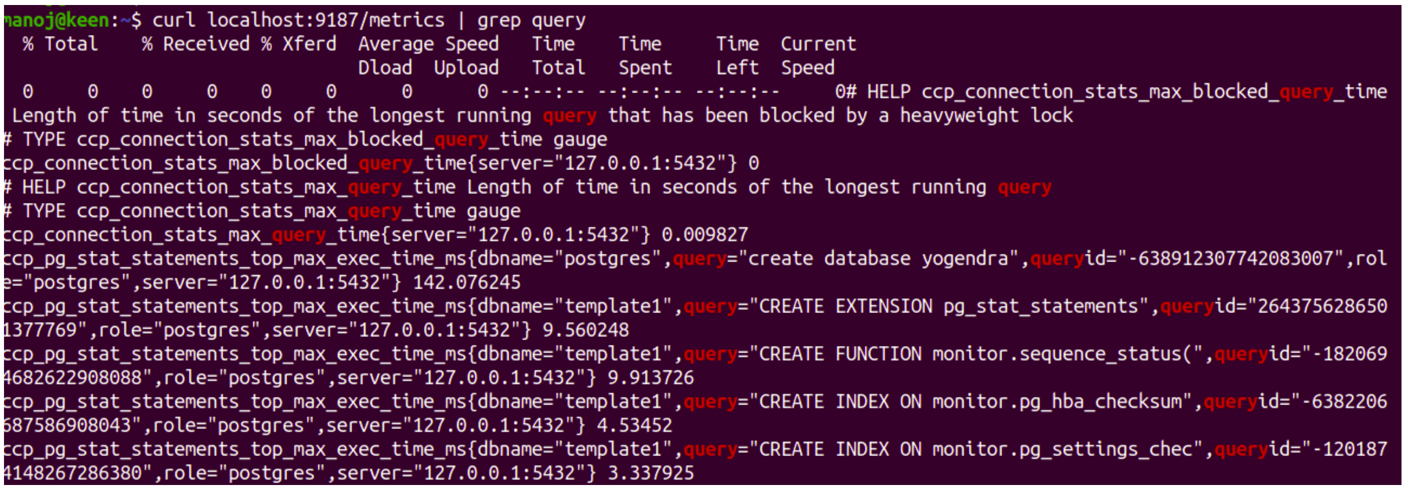
```
podman run -itd --pod crunchy-postgres --name crunchy -e EXPORTER_PG_PASSWORD=redhat
```

- **podman run**: This command is used to run a new container.
- **-itd**: These flags are used together for interactive (console input/output enabled), detached (background) mode.
- **--pod crunchy-postgres**: This flag specifies that the container should be part of the existing "crunchy-postgres" pod.

- **--name crunchy**: This flag assigns the name "crunchy" to the container.
- **-e EXPORTER_PG_PASSWORD=redhat**: This flag sets the environment variable **EXPORTER_PG_PASSWORD** within the container to "redhat". This likely represents the password required for PostgreSQL Exporter to connect to the PostgreSQL instance.
- **-e EXPORTER_PG_HOST=127.0.0.1**: This flag sets the environment variable **EXPORTER_PG_HOST** within the container to "127.0.0.1", indicating the host where the PostgreSQL database is located.
- **-e EXPORTER_PG_USER=ccp_monitoring**: This flag sets the environment variable **EXPORTER_PG_USER** within the container to "ccp_monitoring", which is likely the username used by the PostgreSQL Exporter to connect to the PostgreSQL instance.
- **-e DATA_SOURCE_NAME=...**: This flag sets the **DATA_SOURCE_NAME** environment variable. It specifies the connection details for the PostgreSQL Exporter to use when connecting to the PostgreSQL database. The provided URL includes the username, password, host, port, database name, and SSL mode settings.
- **83a59722eb87**: This represents the ID of the container image that you want to run as a container.

12. Check metrics:

```
curl localhost:9187/metrics | grep query
```



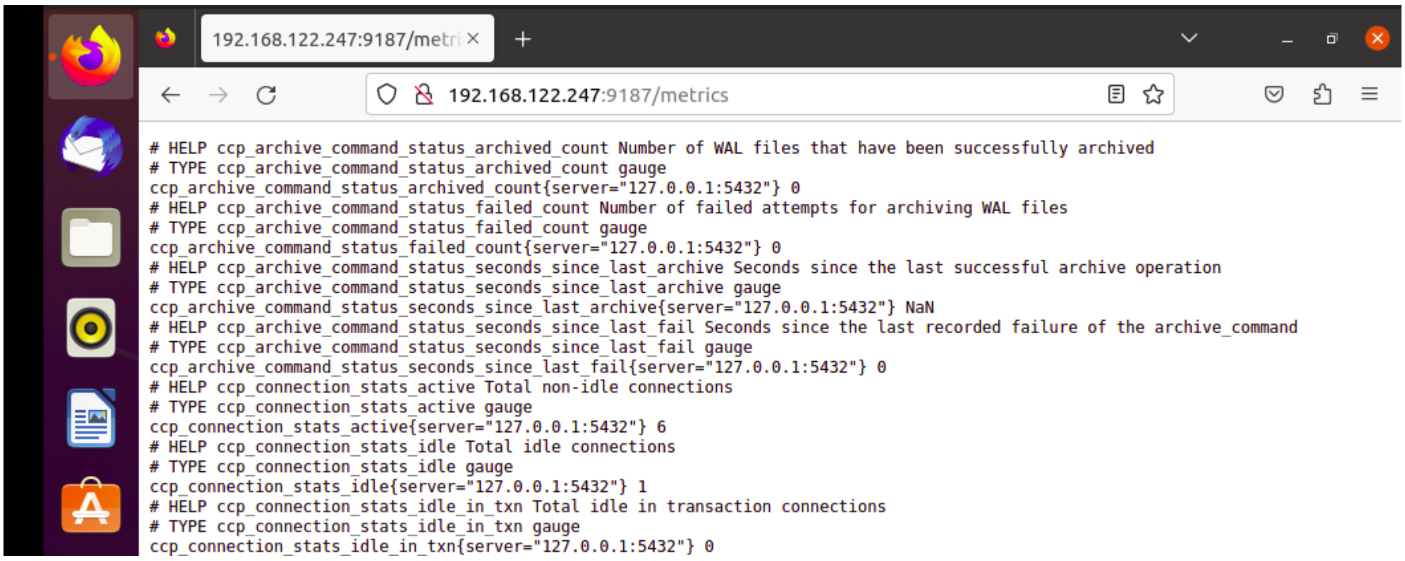
```
nanoj@keen:~$ curl localhost:9187/metrics | grep query
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0# HELP ccp_connection_stats_max_blocked_query_time
 Length of time in seconds of the longest running query that has been blocked by a heavyweight lock
# TYPE ccp_connection_stats_max_blocked_query_time gauge
ccp_connection_stats_max_blocked_query_time{server="127.0.0.1:5432"} 0
# HELP ccp_connection_stats_max_query_time Length of time in seconds of the longest running query
# TYPE ccp_connection_stats_max_query_time gauge
ccp_connection_stats_max_query_time{server="127.0.0.1:5432"} 0.009827
ccp_pg_stat_statements_top_max_exec_time_ms{dbname="postgres",query="create database yogendra",queryid="-638912307742083007",rol
e="postgres",server="127.0.0.1:5432"} 142.076245
ccp_pg_stat_statements_top_max_exec_time_ms{dbname="template1",query="CREATE EXTENSION pg_stat_statements",queryid="264375628650
1377769",role="postgres",server="127.0.0.1:5432"} 9.560248
ccp_pg_stat_statements_top_max_exec_time_ms{dbname="template1",query="CREATE FUNCTION monitor.sequence_status(",queryid="-182069
4682622908088",role="postgres",server="127.0.0.1:5432"} 9.913726
ccp_pg_stat_statements_top_max_exec_time_ms{dbname="template1",query="CREATE INDEX ON monitor.pg_hba_checksum",queryid="-6382206
687586908043",role="postgres",server="127.0.0.1:5432"} 4.53452
ccp_pg_stat_statements_top_max_exec_time_ms{dbname="template1",query="CREATE INDEX ON monitor.pg_settings_chec",queryid="-120187
4148267286380",role="postgres",server="127.0.0.1:5432"} 3.337925
```

- **curl** is a command-line tool to transfer data to or from a server
- **grep query**: This part of the command uses the **grep** command to search for lines in the input that contain the word "query". This is used to filter the metrics output to only show lines related to queries.

13. Create prometheus container:

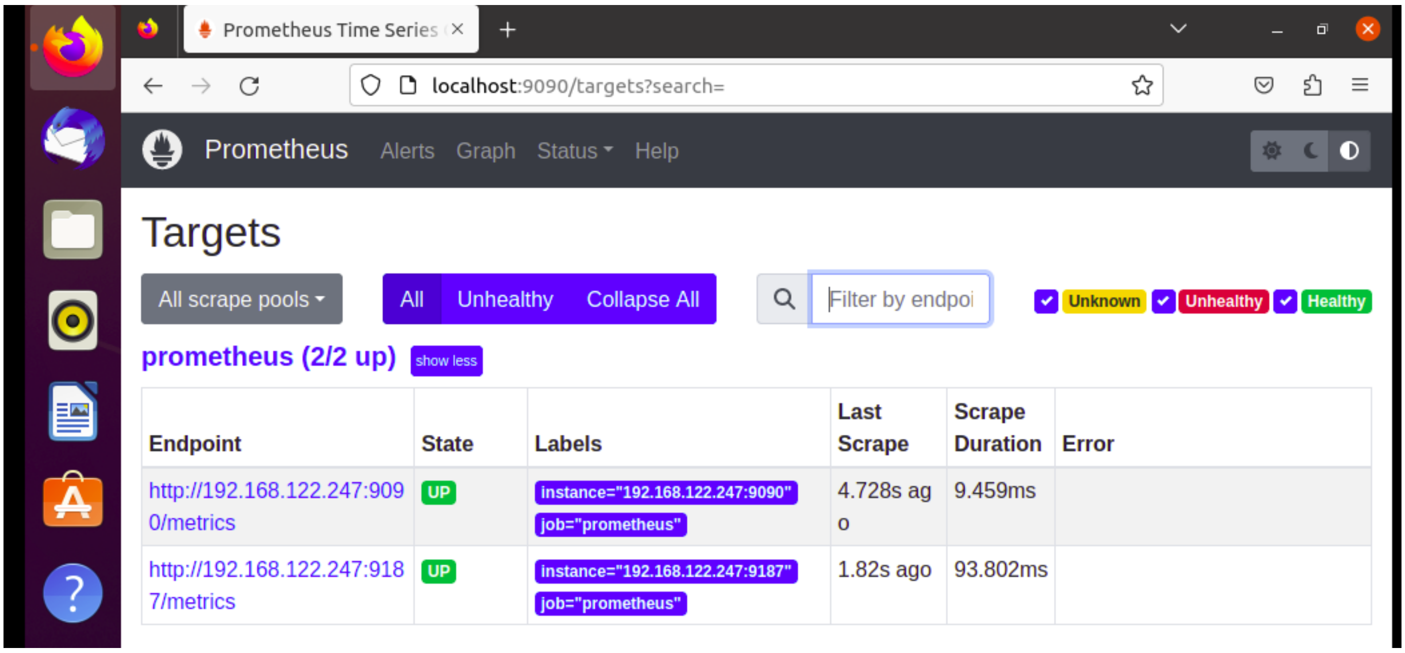
```
podman run -itd --pod crunchy-postgres --name prometheus_crunchy -v /home/yogendra/s
```

Set the target in prometheus.yml file to get metrics in prometheus



```
# HELP ccp_archive_command_status_archived_count Number of WAL files that have been successfully archived
# TYPE ccp_archive_command_status_archived_count gauge
ccp_archive_command_status_archived_count{server="127.0.0.1:5432"} 0
# HELP ccp_archive_command_status_failed_count Number of failed attempts for archiving WAL files
# TYPE ccp_archive_command_status_failed_count gauge
ccp_archive_command_status_failed_count{server="127.0.0.1:5432"} 0
# HELP ccp_archive_command_status_seconds_since_last_archive Seconds since the last successful archive operation
# TYPE ccp_archive_command_status_seconds_since_last_archive gauge
ccp_archive_command_status_seconds_since_last_archive{server="127.0.0.1:5432"} NaN
# HELP ccp_archive_command_status_seconds_since_last_fail Seconds since the last recorded failure of the archive_command
# TYPE ccp_archive_command_status_seconds_since_last_fail gauge
ccp_archive_command_status_seconds_since_last_fail{server="127.0.0.1:5432"} 0
# HELP ccp_connection_stats_active Total non-idle connections
# TYPE ccp_connection_stats_active gauge
ccp_connection_stats_active{server="127.0.0.1:5432"} 6
# HELP ccp_connection_stats_idle Total idle connections
# TYPE ccp_connection_stats_idle gauge
ccp_connection_stats_idle{server="127.0.0.1:5432"} 1
# HELP ccp_connection_stats_idle_in_txn Total idle in transaction connections
# TYPE ccp_connection_stats_idle_in_txn gauge
ccp_connection_stats_idle_in_txn{server="127.0.0.1:5432"} 0
```

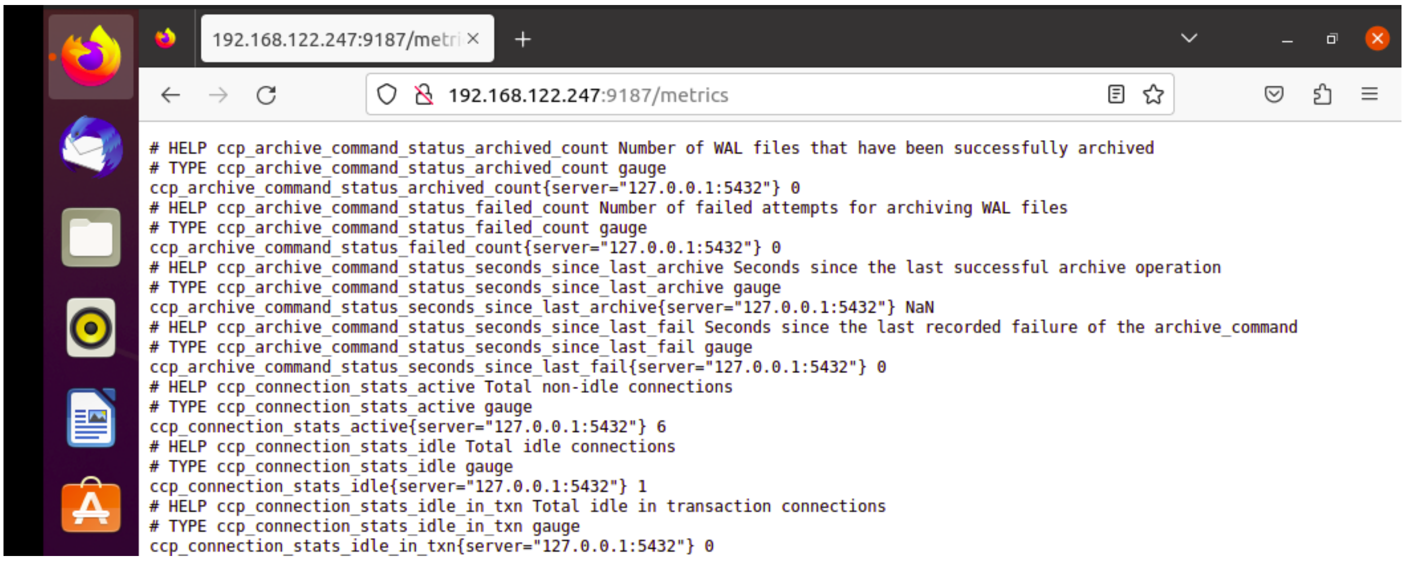
hit on browser: http://localhost:9090/



Targets

prometheus (2/2 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.122.247:9090/metrics	UP	instance="192.168.122.247:9090" job="prometheus"	4.728s ago	9.459ms	
http://192.168.122.247:9187/metrics	UP	instance="192.168.122.247:9187" job="prometheus"	1.82s ago	93.802ms	



```
# HELP ccp_archive_command_status_archived_count Number of WAL files that have been successfully archived
# TYPE ccp_archive_command_status_archived_count gauge
ccp_archive_command_status_archived_count{server="127.0.0.1:5432"} 0
# HELP ccp_archive_command_status_failed_count Number of failed attempts for archiving WAL files
# TYPE ccp_archive_command_status_failed_count gauge
ccp_archive_command_status_failed_count{server="127.0.0.1:5432"} 0
# HELP ccp_archive_command_status_seconds_since_last_archive Seconds since the last successful archive operation
# TYPE ccp_archive_command_status_seconds_since_last_archive gauge
ccp_archive_command_status_seconds_since_last_archive{server="127.0.0.1:5432"} NaN
# HELP ccp_archive_command_status_seconds_since_last_fail Seconds since the last recorded failure of the archive_command
# TYPE ccp_archive_command_status_seconds_since_last_fail gauge
ccp_archive_command_status_seconds_since_last_fail{server="127.0.0.1:5432"} 0
# HELP ccp_connection_stats_active Total non-idle connections
# TYPE ccp_connection_stats_active gauge
ccp_connection_stats_active{server="127.0.0.1:5432"} 6
# HELP ccp_connection_stats_idle Total idle connections
# TYPE ccp_connection_stats_idle gauge
ccp_connection_stats_idle{server="127.0.0.1:5432"} 1
# HELP ccp_connection_stats_idle_in_txn Total idle in transaction connections
# TYPE ccp_connection_stats_idle_in_txn gauge
ccp_connection_stats_idle_in_txn{server="127.0.0.1:5432"} 0
```

14. Create Grafana Container:

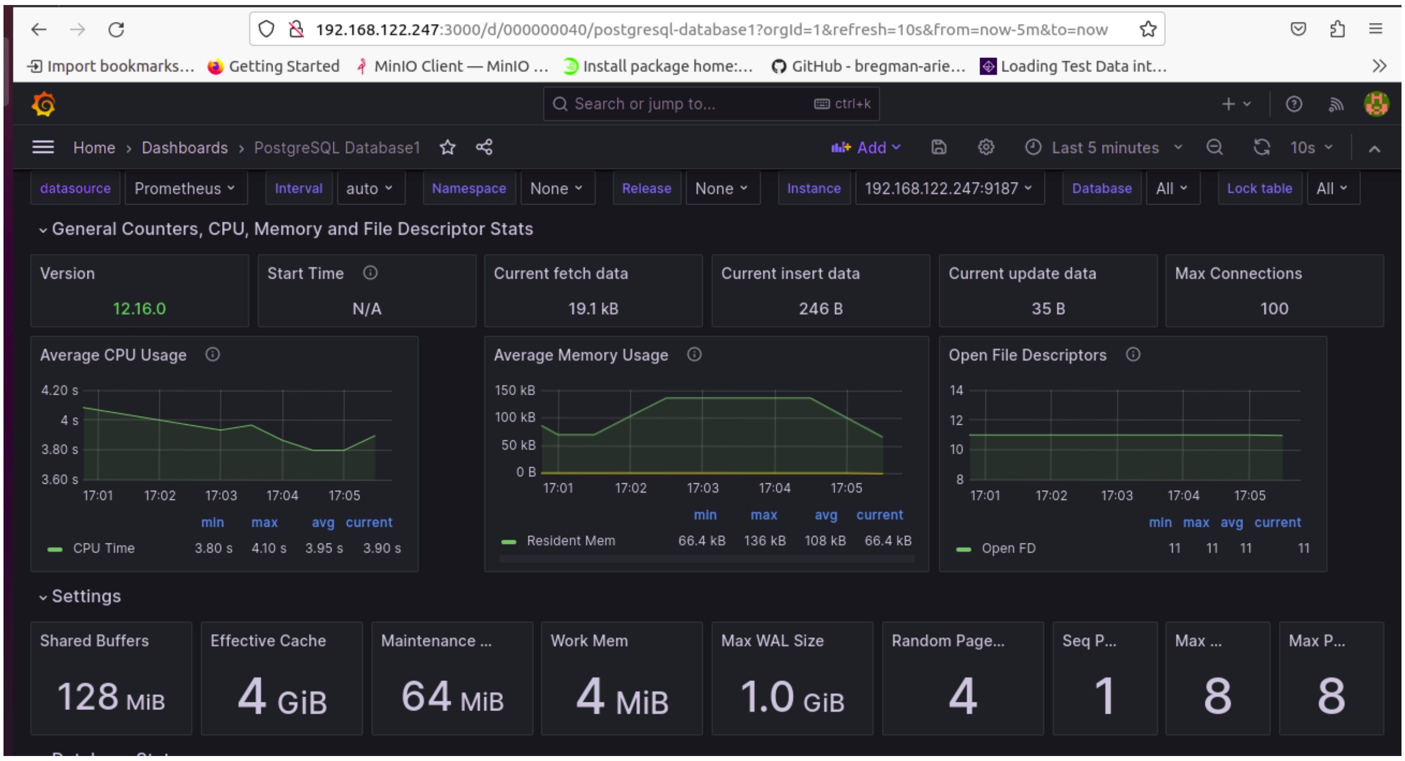
```
podman run -itd --pod crunchy-postgres --name grafana_crunchy docker.io/grafana/grafa
```

Hit on browser:

http://localhost:3000/

Select the prometheus as a datasource and import the dashboard.

9628



192.168.122.247:3000/d/000000040/postgresql-database1?orgId=1&refresh=10s&from=now-5m&to=now

Home › Dashboards › PostgreSQL Database1

datasource: Prometheus | Interval: auto | Namespace: None | Release: None | Instance: 192.168.122.247:9187 | Database: All | Lock table: All

General Counters, CPU, Memory and File Descriptor Stats

Version	Start Time	Current fetch data	Current insert data	Current update data	Max Connections
12.16.0	N/A	19.1 kB	246 B	35 B	100

Average CPU Usage

	min	max	avg	current
CPU Time	3.80 s	4.10 s	3.95 s	3.90 s

Average Memory Usage

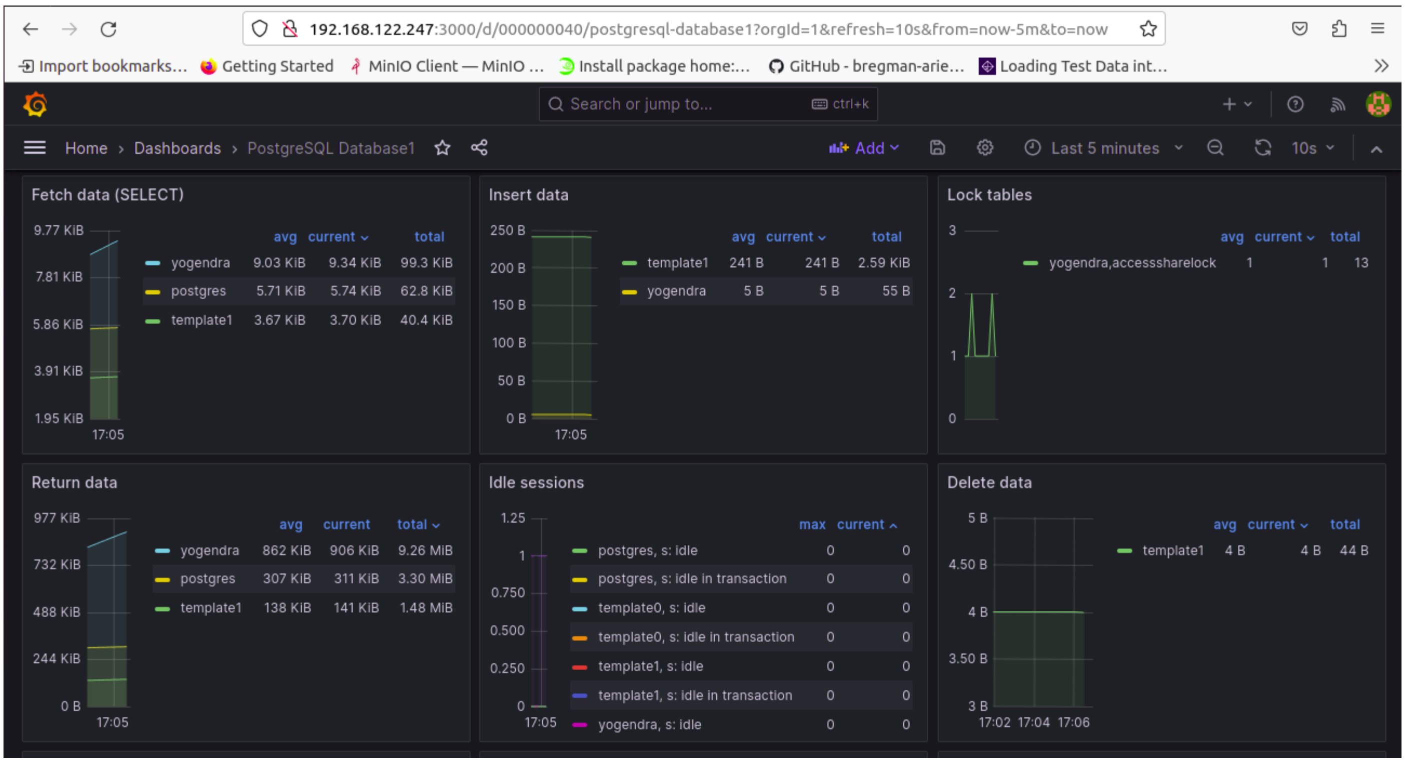
	min	max	avg	current
Resident Mem	66.4 kB	136 kB	108 kB	66.4 kB

Open File Descriptors

	min	max	avg	current
Open FD	11	11	11	11

Settings

Shared Buffers	Effective Cache	Maintenance ...	Work Mem	Max WAL Size	Random Page...	Seq P...	Max ...	Max P...
128 MiB	4 GiB	64 MiB	4 MiB	1.0 GiB	4	1	8	8



192.168.122.247:3000/d/000000040/postgresql-database1?orgId=1&refresh=10s&from=now-5m&to=now

Home › Dashboards › PostgreSQL Database1

Fetch data (SELECT)

	avg	current	total
yogendra	9.03 KiB	9.34 KiB	99.3 KiB
postgres	5.71 KiB	5.74 KiB	62.8 KiB
template1	3.67 KiB	3.70 KiB	40.4 KiB

Insert data

	avg	current	total
template1	241 B	241 B	2.59 KiB
yogendra	5 B	5 B	55 B

Lock tables

	avg	current	total
yogendra,accesssharelock	1	1	13

Return data

	avg	current	total
yogendra	862 KiB	906 KiB	9.26 MiB
postgres	307 KiB	311 KiB	3.30 MiB
template1	138 KiB	141 KiB	1.48 MiB

Idle sessions

	max	current
postgres, s: idle	0	0
postgres, s: idle in transaction	0	0
template0, s: idle	0	0
template0, s: idle in transaction	0	0
template1, s: idle	0	0
template1, s: idle in transaction	0	0
yogendra, s: idle	0	0

Delete data

	avg	current	total
template1	4 B	4 B	44 B

Test cases list

Note : NA

Reference link

https://access.crunchydata.com/documentation/pgmonitor/2.2/exporter/index.html