Crunchy-Postgres-Exporter

# Overview

This document provides a step-by-step overview of set-up Crunchy-Postgres-Exporter using Podman, Crunchy Exporter, PostgreSQL, Prometheus and Grafana on an Ubuntu 20.04 system.

# Environment details

* Ubuntu version 20.04
* RAM 4 GB

# List of tools and technologies

* Podman
* PostgreSQL
* Grafana
* Prometheus

# Definition of tools

* **Podman :** Podman is an open-source containerization tool that works on Linux operating systems. It allows you to create, manage, and run containers, and it offers many features that are compatible with docker commands.
* **PostgreSQL :** PostgreSQL is an open-source relational database management system used for data storage and retrieval. It offers advanced features and scalability, making it a popular open-source RDBMS.
* **Grafana :** Grafana is an open-source tool used for data visualization and monitoring. It allows data to be visually represented in the form of graphs, charts, making data analysis and performance monitoring easier.
* **Prometheus :** Prometheus is an open-source monitoring and alerting system used for system and application performance monitoring. Its primary purpose is data collection, storage, querying, and alerting, allowing the detection of real-time insights and performance issues.

# Command for the setup or configuration

## Step 1. Install podman.

| sudo sh -c "echo 'deb https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/xUbuntu\_$(lsb\_release -rs)/ /' > /etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list" |
| --- |



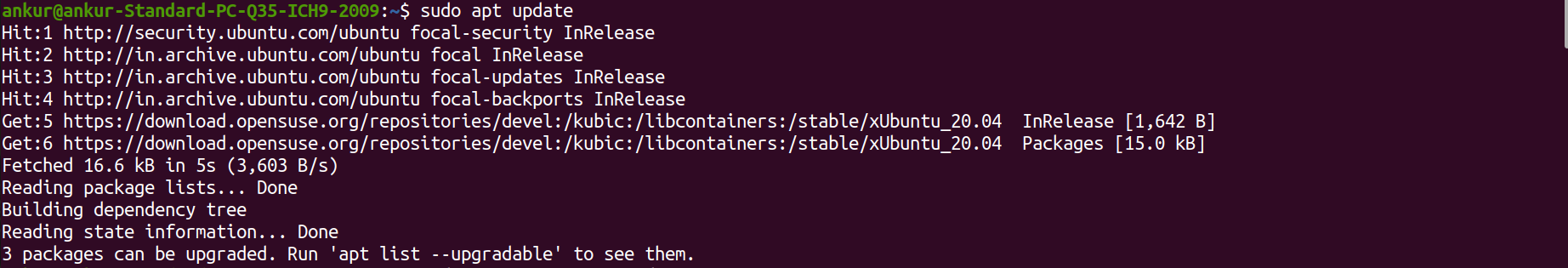
| wget https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/xUbuntu\_$(lsb\_release -rs)/Release.key |
| --- |



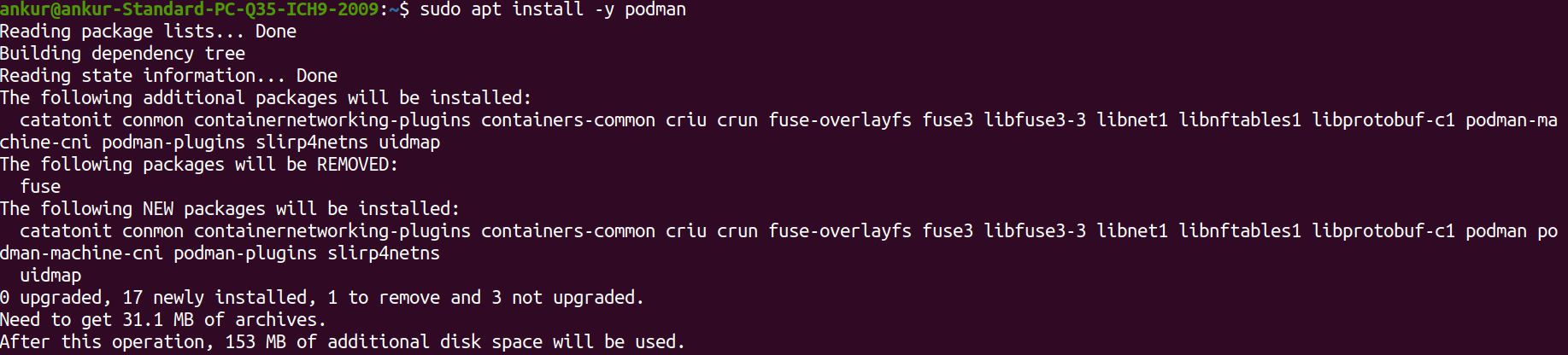
| sudo apt-key add - < Release.key |
| --- |



| sudo apt update |
| --- |



| sudo apt install -y podman |
| --- |

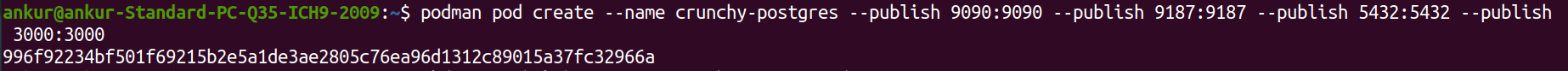


| podman --version |
| --- |



## Step 2. Create Pod name crunchy-postgres for all the 4 containers.

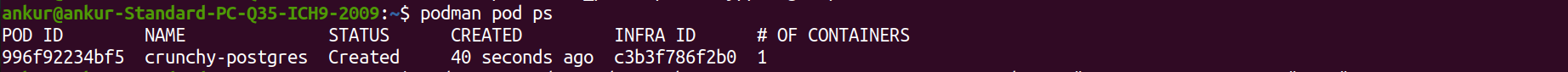
| podman pod create --name crunchy-postgres --publish 9090:9090 --publish 9187:9187 --publish 5432:5432 --publish 3000:3000 |
| --- |



* **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.

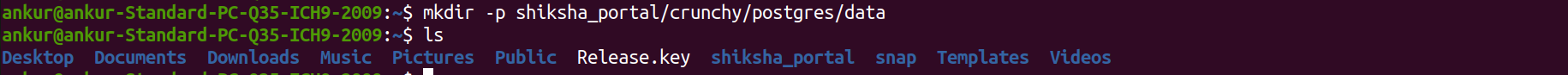
## Step 3. Check the created pod status.

| podman pod ps |
| --- |

****

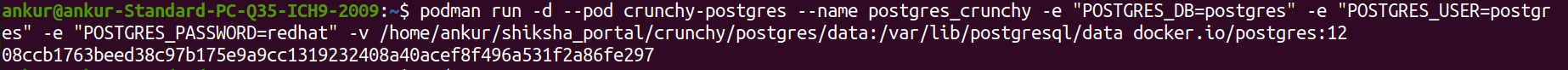
## Step 4. Create a directory for mounting the persistent storage for the postgres data.

| mkdir -p shiksha\_portal/crunchy/postgres/data |
| --- |



## Step 5. Create a postgres container in this pod which was created with the name crunchy-postgres.

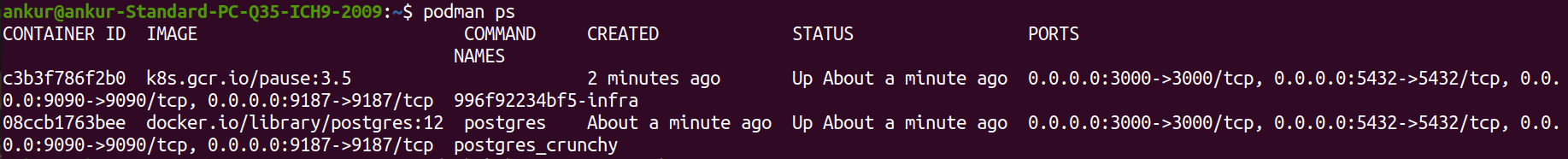
| podman run -d --pod crunchy-postgres --name postgres\_crunchy -e "POSTGRES\_DB=postgres" -e "POSTGRES\_USER=postgres" -e "POSTGRES\_PASSWORD=redhat" -v /home/ankur/shiksha\_portal/crunchy/postgres/data:/var/lib/postgresql/data docker.io/postgres:12 |
| --- |

****

* **podman run**: This command is used to run a new container.
* **-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 indicates 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/ankur/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.

## Step 6. Check the container status.

| podman ps |
| --- |



## Step 7. Do changes in postgresql.conf file.

We will need to modify your postgresql.conf configuration file to tell PostgreSQL to load shared libraries.

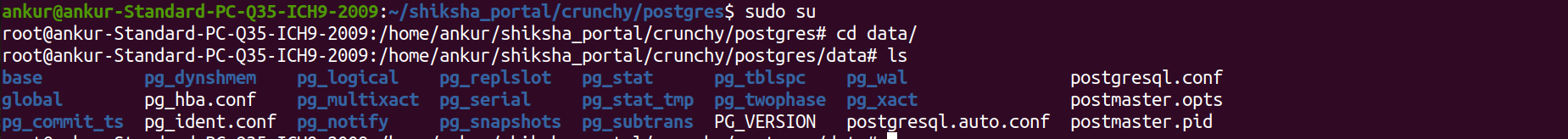
**Change directory to postgres then run sudo su for root privilege and change directory to data.**

| cd shiksha\_portal/cruchy/postgres/ |
| --- |

| sudo su |
| --- |

| cd data/ |
| --- |

****

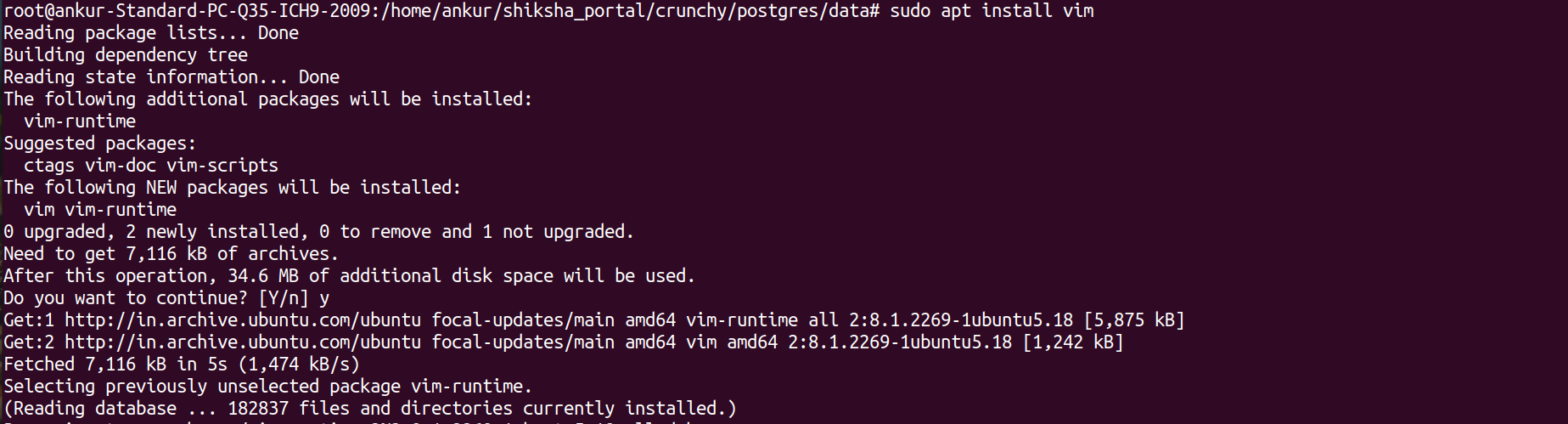
****

| **home/ankur/shiksha\_portal/crunchy/postgres/data# echo "shared\_preload\_libraries = 'pg\_stat\_statements,auto\_explain'" >> postgresql.conf** |
| --- |



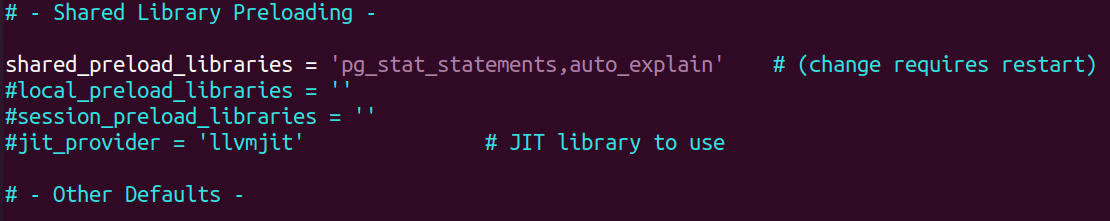
## Step 8. Install vim.

| sudo apt install vim |
| --- |



"We can also make these configuration changes in the postgresql.conf file. Search for '**shared\_preload\_libraries**' and remove the '#' symbol to uncomment it and add following line “**pg\_stat\_statements,auto\_explain**”

| vim postgresql.conf |
| --- |

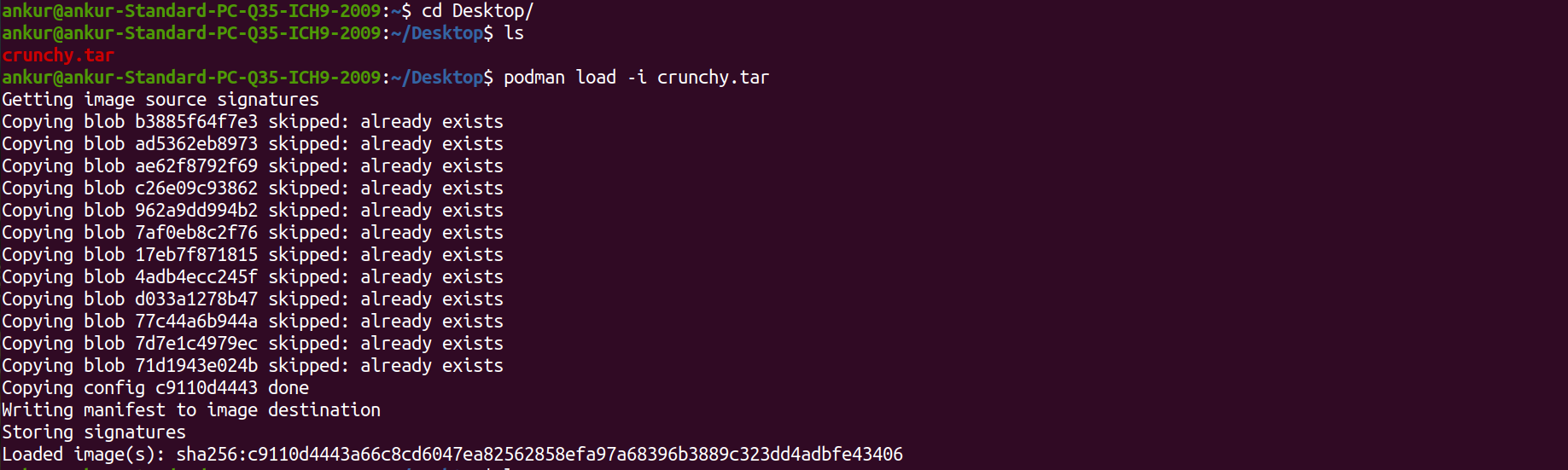
****

## Step 9. Crunchy-postgres-exporter image.

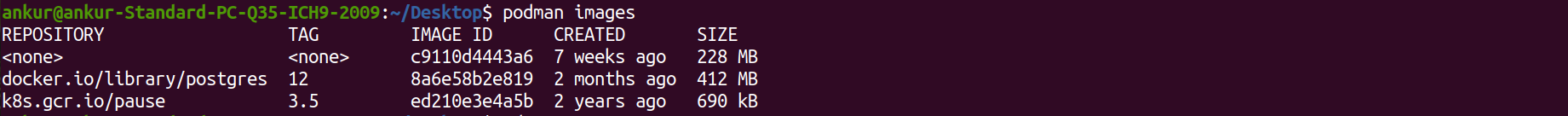
Download tar file from the given link. and run the given command on the same path.

Link:- <https://drive.google.com/file/d/1pTXscjFxxsv1g0NFMCDofeRD-vf_I06M/view>

| podman load -i crunchy.tar |
| --- |



| podman images |
| --- |



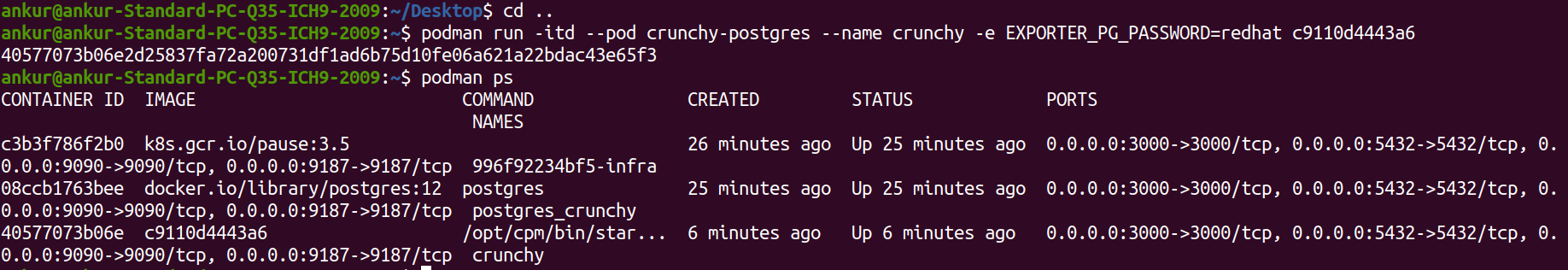
Then run the container with the image ID and use your container's image ID.

| podman run -itd --pod crunchy-postgres --name crunchy -e EXPORTER\_PG\_PASSWORD=redhat c9110d4443a6 |
| --- |

* **podman run:** This is the basic command for running containers with Podman.
* **-itd:** These are flags used with the podman run command:
* **-i:** This option makes the container interactive, allowing you to interact with it through the terminal.
* **-t:** This option allocates a pseudo-TTY, which is often used in conjunction with -i to provide a terminal interface.
* **-d:** This option runs the container in detached mode, meaning it runs in the background, and you can continue using your terminal for other tasks.
* **--pod crunchy-postgres:** This specifies the name of the pod to which the container should be added.
* **--name crunchy:** This assigns a name to the container, in this case, "crunchy."
* **-e EXPORTER\_PG\_PASSWORD=redhat:** This option sets an environment variable within the container. In this case, it sets the environment variable EXPORTER\_PG\_PASSWORD with the value redhat.
* **c9110d4443a6:** This is the container image identifier, typically an image ID.

Again check container status.

| podman ps |
| --- |



## Step 10. Copy setup.sql from Container to Host:

Copy the **setup.sql** file from the **/opt/cpm/conf/pg12/** directory within the Crunchy container to your current working directory on the host machine..

| **podman cp crunchy:/opt/cpm/conf/pg12/setup.sql .** |
| --- |

* **podman cp:** This is a command used to copy files between your host system and a container using Podman.
* **crunchy:/opt/cpm/conf/pg12/setup.sql:** This part specifies the source path of the file inside the container you want to copy.
* **(.):** This indicates the destination path on your host system where the file will be copied.

**NOTE:- Don't miss the ( . ) which is taking place after setup.sql because it represents a copy in your current directory.**

## Step 11. Remove the Test Container.

Remove the **crunchy** container.

| **podman rm -f crunchy** |
| --- |



Here, we will copy the **setup.sql** file from the host to inside the PostgreSQL container in the **/var/lib/postgresql** directory.

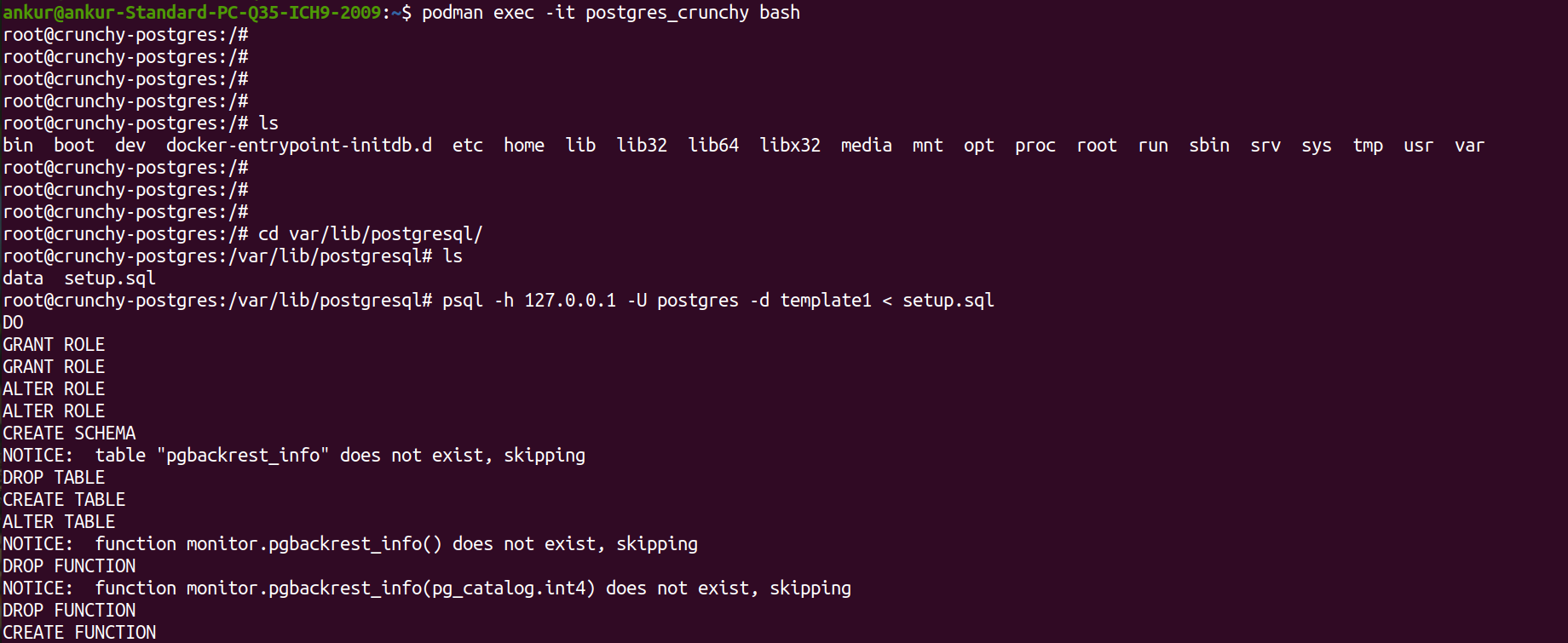
| **podman cp /home/ankur/setup.sql postgres\_crunchy:/var/lib/postgresql** |
| --- |

* **podman cp:** This is a command used to copy files between your host system and a container using Podman.
* **/home/ankur/setup.sql:** This is the source path on your local system, specifying the file you want to copy.
* **postgres\_crunchy:** This is the name of the destination container where you want to copy the file. Containers are often given names for easy reference.
* **/var/lib/postgresql:** This part specifies the destination path within the container. It means you want to copy the file to the /var/lib/postgresql directory inside the "postgres\_crunchy" container.



## Step 12. Go inside the postgres container & Push setup.sql in postgres database.

| podman exec -it postgres\_crunchy bash  root@crunchy-postgres:/# ls  root@crunchy-postgres:/# cd var/lib/postgresql/  root@crunchy-postgres:/var/lib/postgresql# ls  root@crunchy-postgres:/var/lib/postgresql# 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. It facilitates making changes to the database schema, data, or settings as specified in the SQL file.

## Step 13. Create an extension.

| root@crunchy-postgres:/var/lib/postgresql# psql -h 127.0.0.1 -U postgres -d template1 -c "CREATE EXTENSION pg\_stat\_statements;" |
| --- |

* **/var/lib/postgresql#:** This part of the command appears to indicate the directory or path from which the command is being executed. It's not part of the actual PostgreSQL command.
* **psql:** This is the PostgreSQL command-line client, used to interact with PostgreSQL databases.
* **-h 127.0.0.1:** This specifies the hostname or IP address of the PostgreSQL server you want to connect to. In this case, it's connecting to the local machine (127.0.0.1).
* **-U postgres:** This option specifies the username ("postgres") that you want to use to connect to the database.
* **-d template1:** This specifies the name of the database ("template1") to which you want to connect.
* **-c "CREATE EXTENSION pg\_stat\_statements;":** This part is the SQL command that you want to execute within the specified database. In this case, it's creating a PostgreSQL extension called "pg\_stat\_statements."



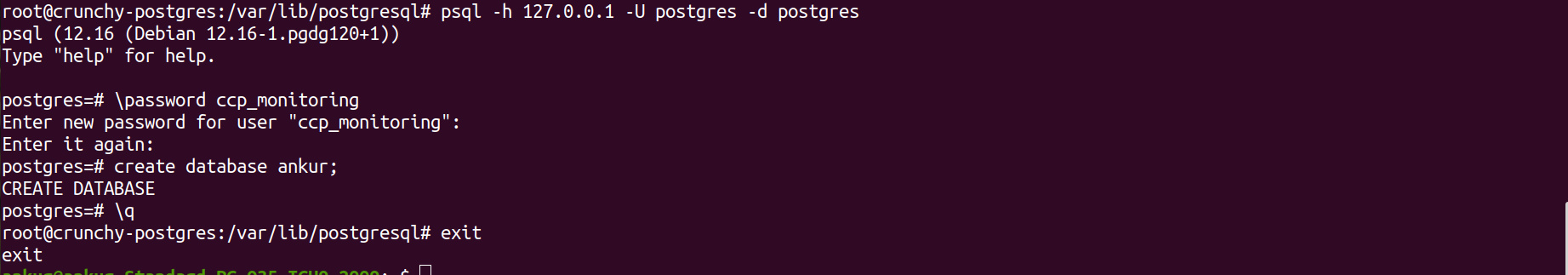
This command is used to enable the '**pg\_stat\_statements**' extension in the PostgreSQL database. It enables the collection of statistics about executed SQL statements for performance analysis.

## Step 14. Here, we will log in to the PostgreSQL database, create a password for the user 'ccp\_monitoring,' and also create a database named 'ankur.'

| root@crunchy-postgres:/var/lib/postgresql# psql -h 127.0.0.1 -U postgres -d postgres  postgres=# \password ccp\_monitoring Enter new password for user "ccp\_monitoring": (I am giving here redhat) Enter it again: postgres=# create database ankur; |
| --- |

In the PostgreSQL database, to exit, use the command '**\q**' and then press '**Enter**.'

* **psql:** This is the command-line utility for interacting with PostgreSQL databases. It allows you to run SQL queries, manage databases, and perform various database-related tasks.
* **-h 127.0.0.1**: This flag specifies the host where the PostgreSQL database is running.
* **-U postgres**: This flag specifies the username to use when connecting to the database. In this case, you're connecting as the PostgreSQL superuser "postgres."
* **-d postgres**: This flag specifies the name of the database to connect to. In this case, you're connecting to a database named "postgres."

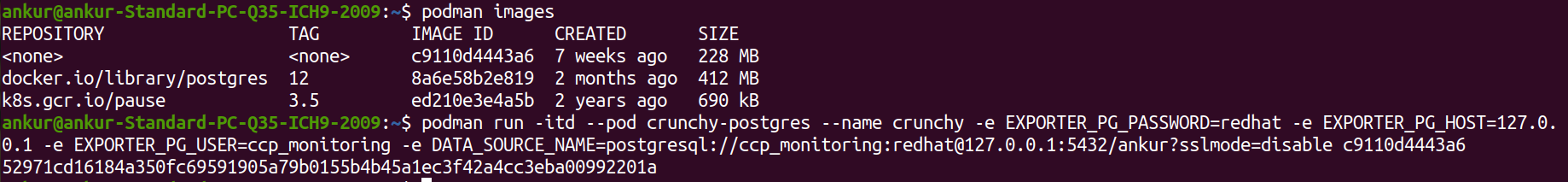


## Step 15. Now create a crunchy-postgres-exporter container.

| podman run -itd --pod crunchy-postgres --name crunchy -e EXPORTER\_PG\_PASSWORD=redhat -e EXPORTER\_PG\_HOST=127.0.0.1 -e EXPORTER\_PG\_USER=ccp\_monitoring -e DATA\_SOURCE\_NAME=postgresql://ccp\_monitoring:redhat@127.0.0.1:5432/ankur?sslmode=disable c9110d4443a6 |
| --- |

* **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.
* **c9110d4443a6**: This represents the ID of the container image that you want to run as a container.

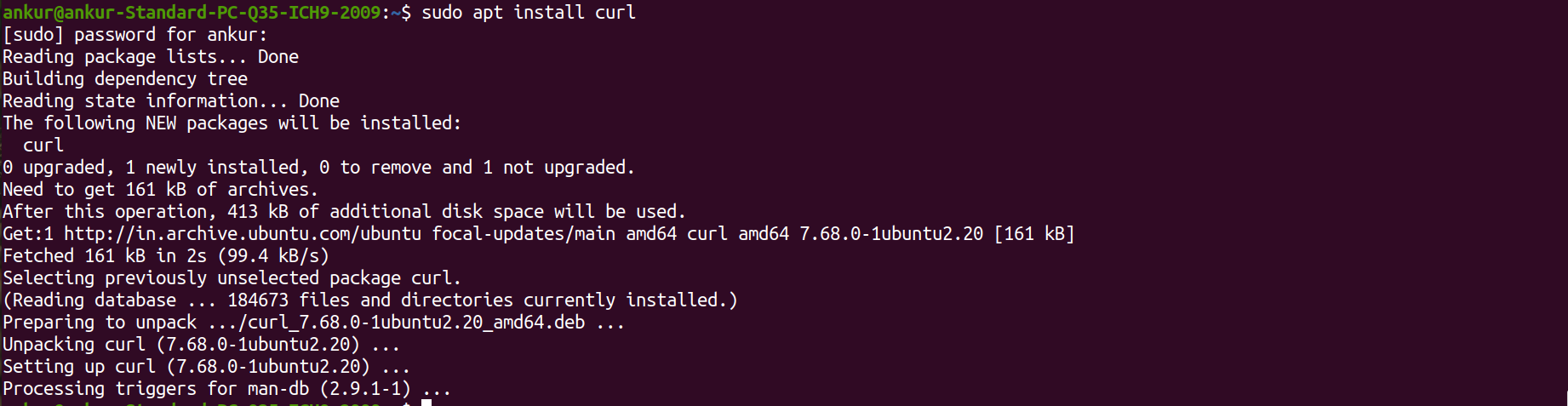
**Podman images.**



## Step 16. Install curl.

| sudo apt install curl |
| --- |

* **curl:** is a command-line tool to transfer data to or from a server.

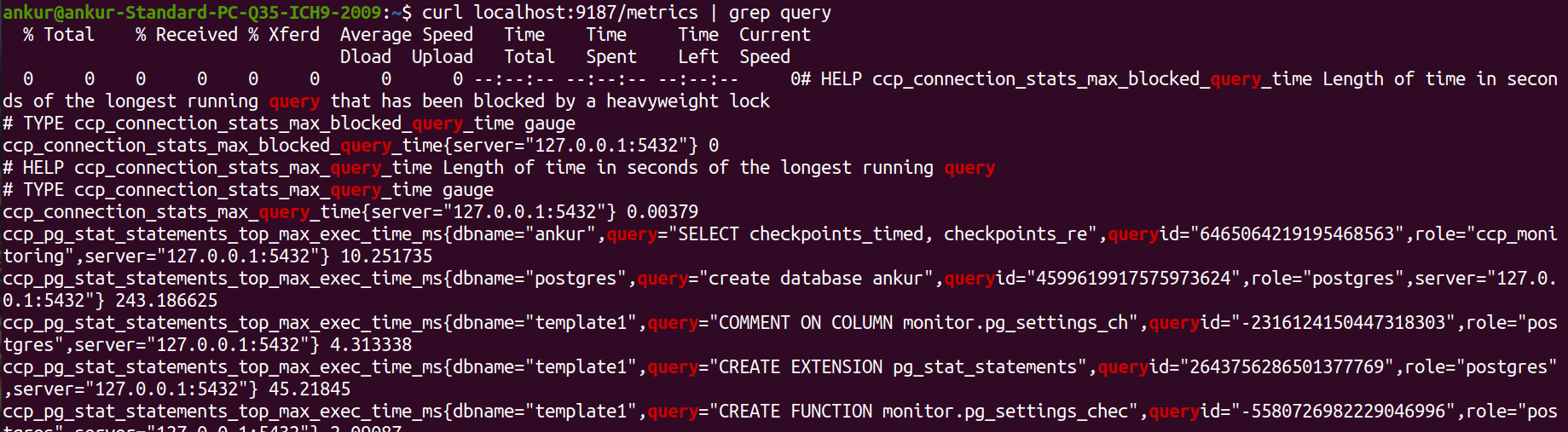
****

**Check metrics.**

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

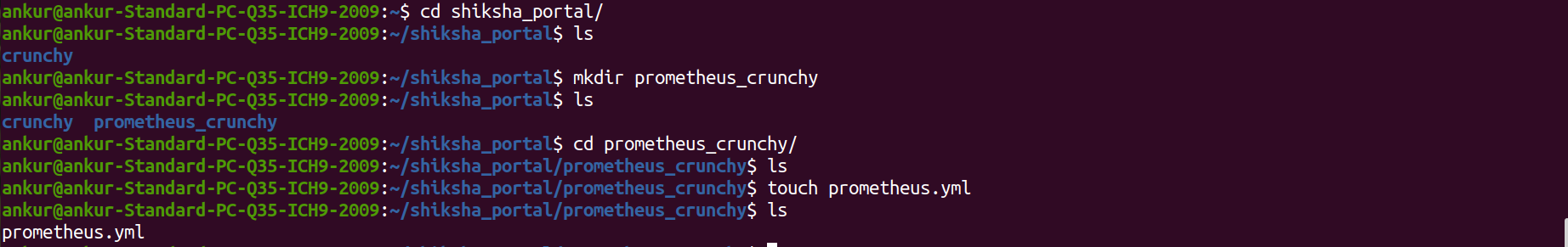
.

* **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



Now, for the Prometheus container, we need to create the 'prometheus\_crunchy' directory and the 'prometheus.yml' file inside the directory. Change directory to shiksha portal and create file.

| :~$ **cd shiksha\_portal/**  :~/shiksha\_portal$ mkdir prometheus\_crunchy :~/shiksha\_portal$ ls :~/shiksha\_portal$ cd prometheus\_crunchy/ :~/shiksha\_portal/prometheus\_crunchy$ touch prometheus.yml |
| --- |

****

## Step 17. Create a prometheus container.

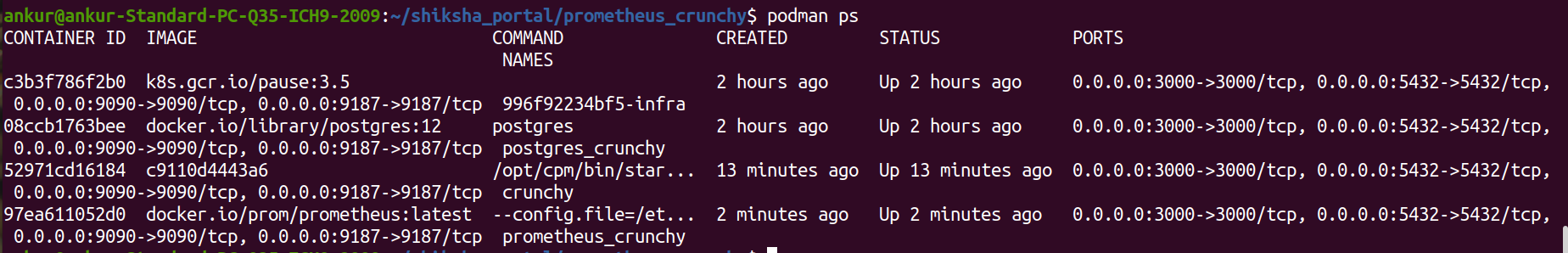
Please check the path name **/home/ankur/shiksha\_portal/prometheus\_crunchy.**

| podman run -itd --pod crunchy-postgres --name prometheus\_crunchy -v /home/ankur/shiksha\_portal/prometheus\_crunchy/prometheus.yml:/etc/prometheus/prometheus.yml docker.io/prom/prometheus |
| --- |



**Check all container status**

| podman ps |
| --- |



## Step 18. Configure the prometheus.yml file.

Check IP

| hostman -I |
| --- |

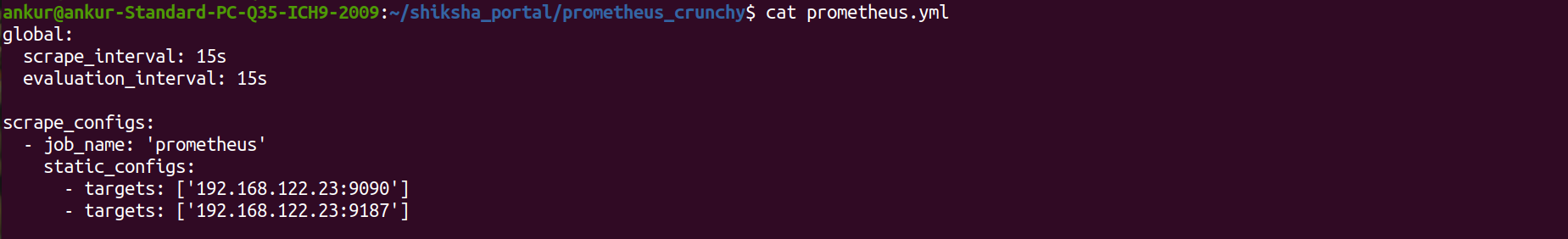
****

| vim Prometheus.yml |
| --- |

To insert the given content into the file, use 'i', add the content, then press 'Esc' followed by ':wq!' to save and exit.

| /shiksha\_portal/prometheus\_crunchy$ cat prometheus.yml |
| --- |

| **global:**  **scrape\_interval: 15s**  **evaluation\_interval: 15s**  **scrape\_configs:**  **- job\_name: 'prometheus'**  **static\_configs:**  **- targets: ['192.168.122.23:9090']**  **- targets: ['192.168.122.23:9187']** |
| --- |

****

Set the target in the prometheus.yml file to get metrics in prometheus and hit the browser (after restarting the prometheus container).

| **:~/shiksha\_portal/prometheus\_crunchy$ podman restart prometheus\_crunchy** |
| --- |

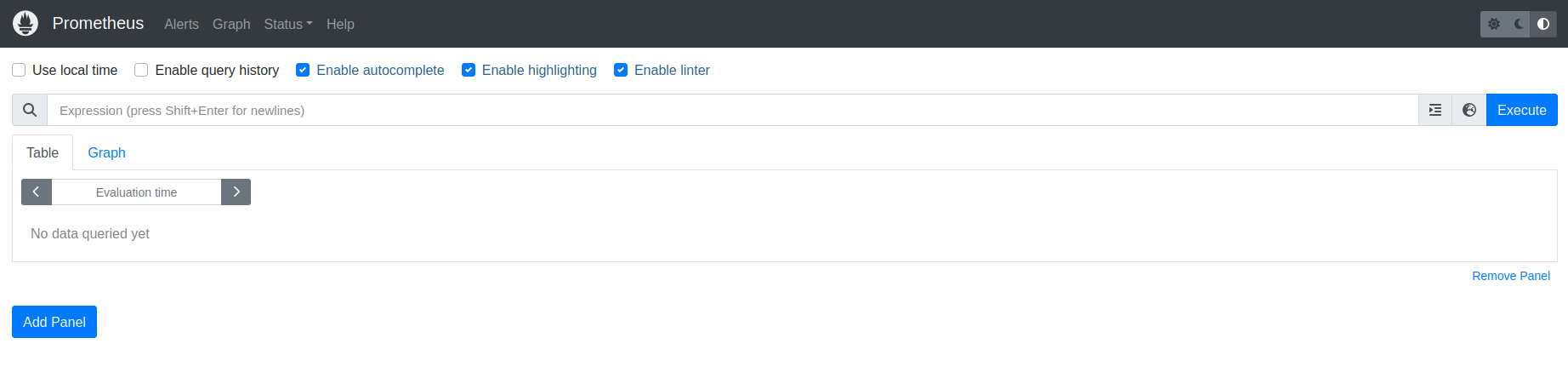


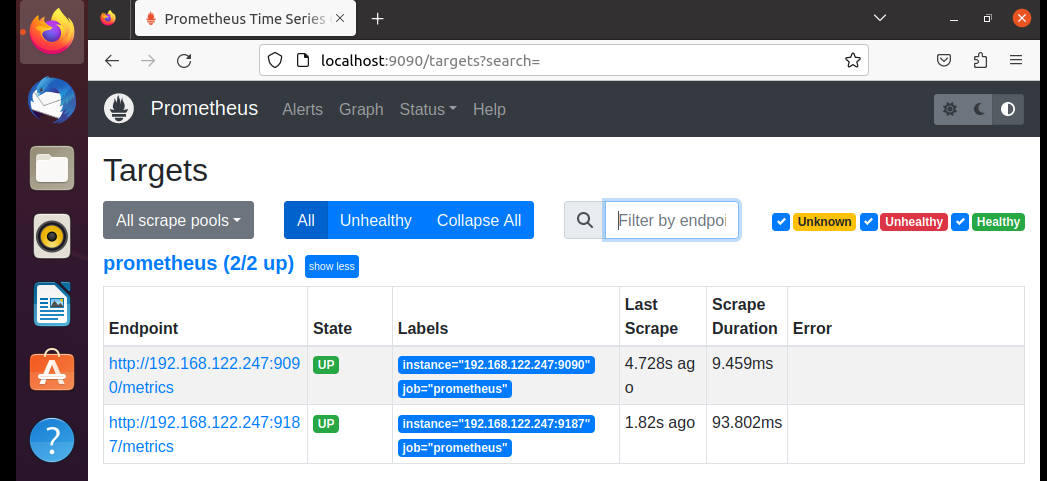
[**http://**](http://localhost:9090/)**192.168.122.23**[**:9090/**](http://localhost:9090/)

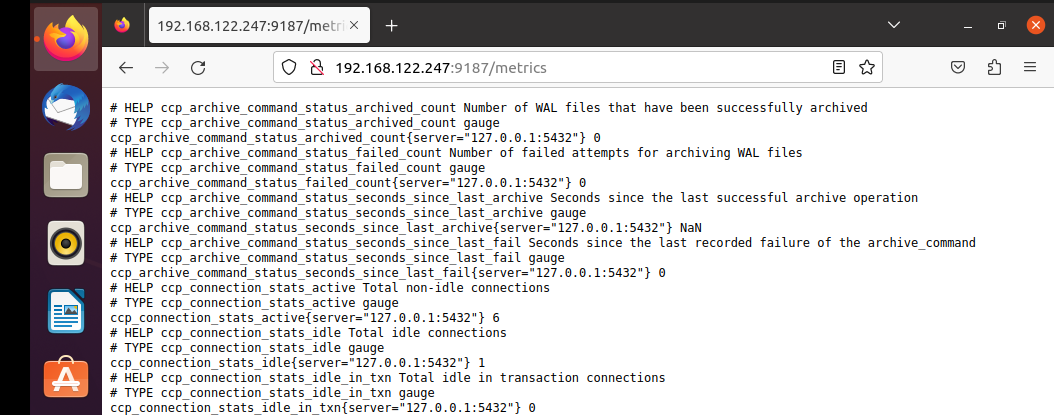
Or

[**http://localhost:9090/**](http://localhost:9090/)

Open Prometheus in the browser by using the IP or 'localhost' and then click 'Status' > 'Targets' > 'Status' to confirm that all statuses are up.

****





## Step 19. Create a grafana container for Visualisation of the metrics data.

| podman run -itd --pod crunchy-postgres --name grafana\_crunchy docker.io/grafana/grafana |
| --- |

* **podman run:** This is the command to run a container using Podman.
* **-itd:** These are options:

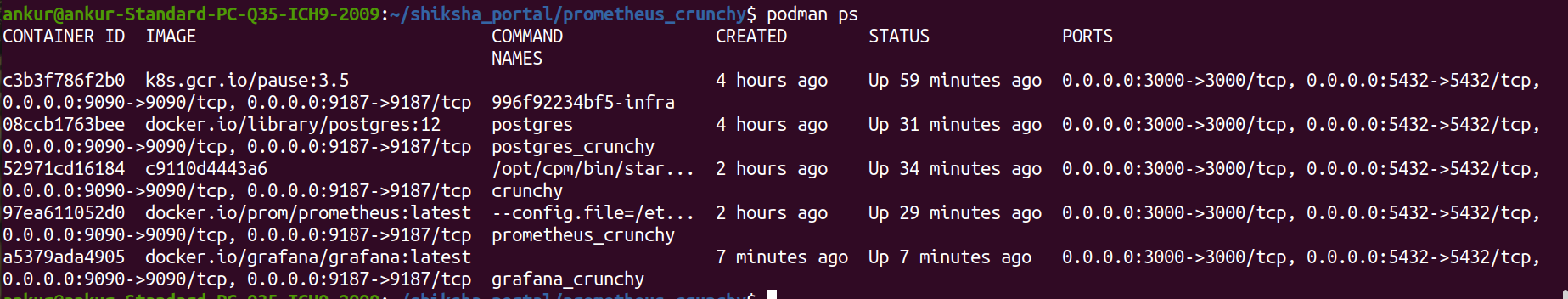
**-i:** Keep STDIN open even if not attached (i.e., interactive mode).

**-t:** Allocate a pseudo-TTY (i.e., terminal).

**-d:** Run the container in the background (detached mode).

* **--pod crunchy-postgres:** This specifies that the container should be part of the "crunchy-postgres" pod.
* **--name grafana\_crunchy:** This assigns the name "grafana\_crunchy" to the running container.
* **docker.io/grafana/grafana:** This is the image you are using to create the container. It specifies the image's location, which is "docker.io" in this case, and the image name "grafana/grafana."

****

**Now check all container status.  
**

Visit[**http://localhost:3000/**](http://localhost:3000/)

Login to Grafana using the default ID and password (admin), and then change the password.

Click on **"Data source" > Select "Prometheus" > Paste the Prometheus URL > Save and Test.**

Next, click on **"Create dashboard" > "Import dashboard" > Enter ID 9628 > Choose "DS Prometheus" as the data source we created.**

Select **"Prometheus"** as the data source and import the dashboard with **ID 9628.**

