Database - Setting up a Database

---

1. Create a .yaml file to apply to the cluster.
   (This is needed for the Postgresql service to run on Kubernetes)
2. Copy and paste this into a file named postgresql on your ubuntu home directory
3. Change the database name, user, and password accordingly

Yaml Code:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgresql
spec:
  selector:
    matchLabels:
      app: postgresql
  replicas: 1
  template:
     metadata:
     labels:
     app: postgresql
     spec:
     containers:
     - name: postgresql
     image: postgres:13
     env:
     - name: POSTGRES_DB
          value:
     - name: POSTGRES_USER
          value:
     - name: POSTGRES_PASSWORD
          value:
     ports:
     - containerPort: 5432
```

Apply the changes made:

```
kubectl apply -f postgresql-deployment.yaml
```

Check that the service is running with:

```
kubectl get service
```

```
ubuntu@ip-172-31-44-45:~$ kubectl get service
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
kubernetes         ClusterIP   10.100.0.1       <none>        443/TCP    11d
postgres-service   ClusterIP   10.100.241.200   <none>        5432/TCP   19h
```

Postgres-service should show up
Run: `kubectl get pod`

It should show that the pod is up and running

```
ubuntu@ip-172-31-44-45:~$ kubectl get pod
NAME                                   READY   STATUS    RESTARTS   AGE
postgres-deployment-5d7f678d5c-b74cz   1/1     Running   0          18h
```

This is an example script to connect and put entries in a database to test connections

Use the cluster-ip running the postgres-service and the database name, user, and password to connect to the created database

```
import pandas as pd
from sqlalchemy import create_engine, text
# Example preprocessed DataFrame
data = {
      'column1': ['value1', 'value2', 'value3'],
      'column2': ['value4', 'value5', 'value6']
}
df = pd.DataFrame(data)

# PostgreSQL connection details
db_details = {
      'dbname': ',
      'user': ,
      'password': ',
      'host': '',
      'port': 5432
}

# Connection string
conn_string =
f"postgresql+psycopg2://{db_details['user']}:{db_details['password']}
@{db_details['host']}:{db_details['port']}/{db_details['dbname']}"

# Create an engine
engine = create_engine(conn_string)

# Write the DataFrame to PostgreSQL
table_name = 'data'
```

```
# Create table if it does not exist and insert data
df.to_sql(table_name, engine, if_exists='replace', index=False)

print(f"DataFrame stored in table '{table_name}' successfully.")
try:
     with engine.connect() as connection:
        df_fetched = pd.read_sql_table(table_name, con=connection)
     print("Data fetched successfully.")
     print(df_fetched)
except Exception as e:
     print(f"An error occurred while fetching data: {e}")
```

To see a UI overview of the database you can install pgadmin on the cluster.

This is the official page for pgadmin install: [Installing pgAdmin 4 on Ubuntu 22.04 or 20.04 or 18.04 | ComputingForGeeks]

The install links used may vary based on ubuntu version

You need to launch it through RDP because it has a UI component, you will get an error from launching in the terminal