PostgreSQL Kubernetes

Create namespace for PostgreSQL
`kubectl create namespace postgresql`

```
[ronslim@ansiblec postgresql]$ kubectl create namespace postgresql
namespace/postgresql created
```

Create secret yaml code for Postgre SQL Database Username and Password (e.g. `postgresecrets.yaml`)

```
postgresecrets.yaml:
apiVersion: v1
kind: Secret
metadata:
  name: postgres-secrets
  namespace: postgresql
data:
  POSTGRES_PASSWORD: <Base64 password>
  POSTGRES_USER: <Base64 username>

Example:
Desired POSTGRES_PASSWORD = Bake
Use the command to convert the desired password to base64
echo -n "Bake" | base64
QmFrZQ==

Then put to the yaml
apiVersion: v1
kind: Secret
metadata:
  name: postgres-secrets
  namespace: postgresql
data:
  POSTGRES_PASSWORD: QmFrZQ==
  POSTGRES_USER: <Base64 username>

Desired POSTGRES_USER: postgres
echo -n "postgres" | base64
cG9zdGdyZXM=

Then put to the yaml
apiVersion: v1
kind: Secret
metadata:
  name: postgres-secrets
  namespace: postgresql
data:
  POSTGRES_PASSWORD: QmFrZQ==
  POSTGRES_USER: cG9zdGdyZXM=
```

Then execute kubectl apply to apply the yaml file

```
[ronslim@ansiblec postgresql]$ kubectl apply -f postgresecrets.yaml
secret/postgres-secrets created
```

Create ConfigMap yaml code for Postgre SQL Database DB Name (e.g. `postgreconfigmap.yaml`)

```
postgreconfigmap.yaml:
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-configmap
  namespace: postgresql
```

```
data:
  POSTGRES_DB: postgres
```

Note: POSTGRES_DB = Desired DB Name of PostgreSQL

Then execute kubectl apply to apply the yaml file

```
[ronslim@ansiblec postgresql]$ kubectl apply -f postgreconfigmap.yaml
configmap/postgres-configmap created
```

Then create Persistent Volume Claim yaml code to store postgre SQL data (e.g. `postgrevolclaim.yaml`)

```
postgrevolclaim.yaml:
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-pv-claim
  namespace: postgresql
  labels:
    app: postgres
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Then execute kubectl apply to apply the yaml file

```
[ronslim@ansiblec postgresql]$ kubectl apply -f postgrevolclaim.yaml
persistentvolumeclaim/postgres-pv-claim created
```

Then create Statefulset yaml code for deployment (e.g. `postgrestatefulset.yaml`)

```
postgrestatefulset.yaml:
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: postgres-statefulset
  namespace: postgresql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
      - name: postgres
        image: postgres:17-alpine3.21
        ports:
          - containerPort: 5432
        env:
        - name: POSTGRES_USER
          valueFrom:
            secretKeyRef:
              name: postgres-secrets
              key: POSTGRES_USER
        - name: POSTGRES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: postgres-secrets
              key: POSTGRES_PASSWORD
        - name: POSTGRES_DB
```

```
          valueFrom:
            configMapKeyRef:
              name: postgres-configmap
              key: POSTGRES_DB
        volumeMounts:
        - name: postgres-database-storage
          mountPath: /var/lib/pgsql/data
      volumes:
      - name: postgres-database-storage
        persistentVolumeClaim:
          claimName: postgres-pv-claim
```

Notice the ff:
secretKeyRef is from created secrets postgres-secrets
ConfigMapKeyRef is from created configmap postgres-configmap
Volumes claimName is from created Persistent Volume Claim postgres-pv-claim

Then execute kubectl apply to apply the yaml file

```
[ronslim@ansiblec postgresql]$ kubectl apply -f postgrestatefulset.yaml
statefulset.apps/postgres-statefulset created
```

Check if the Statefulset is already 1/1 ready

kubectl get statefulset -n postgresql

```
[ronslim@ansiblec postgresql]$ kubectl get statefulset -n postgresql
NAME                   READY   AGE
postgres-statefulset   1/1     2m4s
```

Then create Service yaml code to expose postgreSQL via ClusterIP (e.g. `postgresvc.yaml`)

```
postgresvc.yaml:
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
  namespace: postgresql
spec:
  type: ClusterIP
  ports:
    - port: 5432
      targetPort: 5432
  selector:
    app: postgres
```

Then execute kubectl apply to apply the yaml file

```
[ronslim@ansiblec postgresql]$ kubectl apply -f postgresvc.yaml
service/postgres-service created
```

Then get pods to get pod name

```
[ronslim@ansiblec postgresql]$ kubectl get pods -n postgresql
NAME                     READY   STATUS    RESTARTS   AGE
postgres-statefulset-0   1/1     Running   0          14m
```

Try to connect to postgresql via Pod. Then supply password

kubectl exec -it pods/<pod name> -n postgresql -- psql -U <Desired and not Base64 POSTGRES_USER> -d <POSTGRES_DB from ConfigMap> -h <Postgre Service Name>
e.g kubectl exec -it pods/postgres-statefulset-0 -n postgresql -- psql -U postgres -d postgres -h postgres-service

```
[ronslim@ansiblec postgresql]$ kubectl exec -it pods/postgres-statefulset-0 -n postgresql -- psql -U postgres -d postgres -h postgres-service
Password for user postgres:
psql (17.5)
Type "help" for help.

postgres=# |
```

Then try to get version and exit after
`select version();`

```
postgres=# select version();
                                    version
-----------------------------------------------------------------------------------------
 PostgreSQL 17.5 on x86_64-pc-linux-musl, compiled by gcc (Alpine 14.2.0) 14.2.0, 64-bit
(1 row)

postgres=# |
```

Then port-forward to expose the pod outside for pgadmin DB access later

```
kubectl port-forward --address localhost,<VM IP> pods/<Pod Name> -n <postgre sql namespace> 5432:5432
kubectl port-forward --address localhost,192.168.16.133 pods/postgres-statefulset-0 -n postgresql 5432:5432
```

```
[ronslim@ansiblec ~]$ kubectl port-forward --address localhost,192.168.16.133 pods/postgres-statefulset-0 -n postgresql 5432:5432
Forwarding from 127.0.0.1:5432 -> 5432
Forwarding from 192.168.16.133:5432 -> 5432
Forwarding from [::1]:5432 -> 5432
```

pgadmin Kubernetes

Using same namespace postgresql
Create secret yaml code for pgadmin Password (e.g. `pgadminsecret.yaml`)

```
pgadminsecret.yaml:
apiVersion: v1
kind: Secret
metadata:
  name: pgadmin-secrets
  namespace: postgresql
data:
  pgadmin-password: <Base64 Password>
```

Then execute kubectl apply to apply the yaml file

```
[ronslim@ansiblec pgadmin]$ kubectl apply -f pgadminsecret.yaml
secret/pgadmin-secrets created
```

Create ConfigMap yaml code for pgadmin server configuration (e.g. `pgadminconfigmap.yaml`)

```
pgadminconfigmap.yaml:
apiVersion: v1
kind: ConfigMap
metadata:
  name: pgadmin-configmap
  namespace: postgresql
data:
  servers.json: |
    {
       "Servers": {
         "1": {
           "Name": "PostgreSQL DB",
           "Group": "Servers",
           "Port": 5432,
           "Username": "postgres",
           "Host": "<VM IP>",
           "SSLMode": "prefer",
           "MaintenanceDB": "postgres"
         }
        }
```

```
    }
```

Then execute kubectl apply to apply the yaml file

```
[ronslim@ansiblec pgadmin]$ kubectl apply -f pgadminconfigmap.yaml
configmap/pgadmin-configmap created
```

Create Statefulset yaml code for pgadmin deployment (e.g. `pgadminstateful.yaml`)

```
pgadminstateful.yaml:
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: pgadmin-statefulset
  namespace: postgresql
spec:
  serviceName: pgadmin-service
  podManagementPolicy: Parallel
  replicas: 1
  updateStrategy:
    type: RollingUpdate
  selector:
    matchLabels:
      app: pgadmin
  template:
    metadata:
      labels:
        app: pgadmin
    spec:
      containers:
        - name: pgadmin
          image: dpage/pgadmin4:9.3.0
          imagePullPolicy: Always
          env:
          - name: PGADMIN_DEFAULT_EMAIL
            value: <email user@domain.com e.g. ron_lim_az500_ms@outlook.com>
          - name: PGADMIN_DEFAULT_PASSWORD
            valueFrom:
              secretKeyRef:
                name: pgadmin-secrets
                key: pgadmin-password
          ports:
          - name: http
            containerPort: 80
            protocol: TCP
          volumeMounts:
          - name: pgadmin-configvol
            mountPath: /pgadmin4/servers.json
            subPath: servers.json
            readOnly: true
          - name: pgadmin-data
            mountPath: /var/lib/pgadmin
      volumes:
      - name: pgadmin-configvol
        configMap:
          name: pgadmin-configmap
  volumeClaimTemplates:
  - metadata:
      name: pgadmin-data
      namespace: postgresql
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 3Gi

Notice the ff:
SecretKeyRef pgadmin-password from pgadmin-secrets
Volume configmap = pgadmin-configmap
```

```
MountPath = /pgadmin4/servers.json (servers.json from pgadmin-configmap)
```

Then execute kubectl apply to apply the yaml file

```
[ronslim@ansiblec pgadmin]$ kubectl apply -f pgadminstateful.yaml
statefulset.apps/pgadmin-statefulset created
```

Then check the Statefulset pgadmin-statefulset if 1/1 ready

```
[ronslim@ansiblec pgadmin]$ kubectl get statefulset -n postgresql
NAME                    READY   AGE
pgadmin-statefulset     1/1     46s
postgres-statefulset    1/1     54m
[ronslim@ansiblec pgadmin]$
```

If ready, then create Service yaml code to expose pgadmin via NodePort (e.g. `pgadminsvc.yaml`)

```
pgadminsvc.yaml:
apiVersion: v1
kind: Service
metadata:
  name: pgadmin-service
  namespace: postgresql
spec:
  ports:
  - protocol: TCP
    port: 80
    targetPort: http
  selector:
    app: pgadmin
  type: NodePort
```

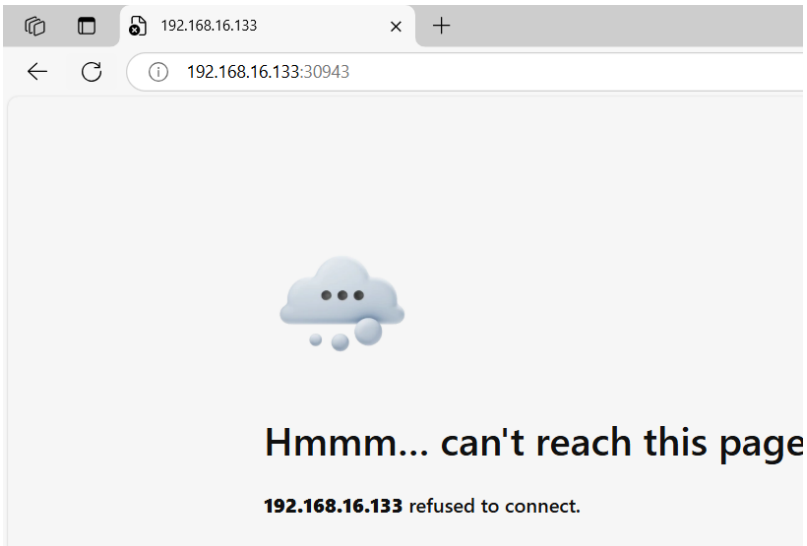
Then execute kubectl apply to apply the yaml file

```
[ronslim@ansiblec pgadmin]$ kubectl apply -f pgadminsvc.yaml
service/pgadmin-service created
```

Then get the pgadmin-service to retrieve the Node Port for port-forward

```
[ronslim@ansiblec pgadmin]$ kubectl get svc pgadmin-service -n postgresql
NAME              TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
pgadmin-service   NodePort   10.100.188.109   <none>        80:30943/TCP   50s
```

Try accessing the pgadmin web via http://<VM IP>/:<NodePort> (e.g. http://192.168.16.133:30943). Notice that the Prometheus is not yet accessible outside VM browser.



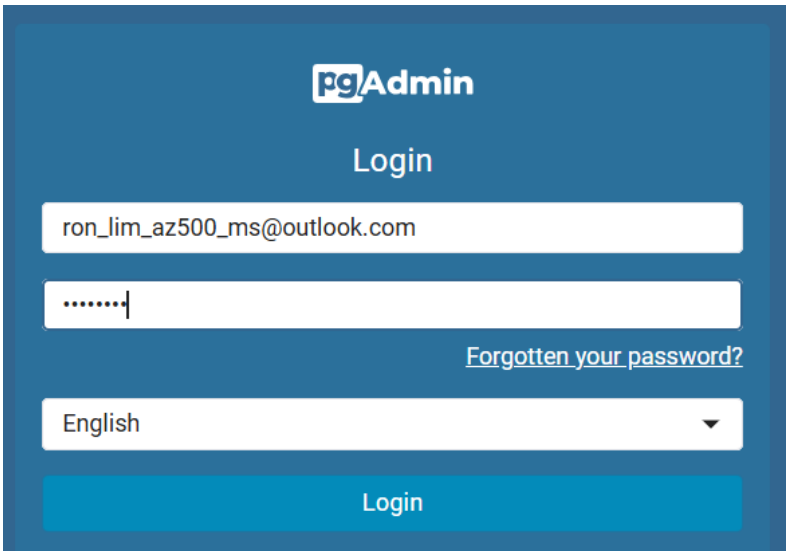
Then port-forward in a separate window to expose the pgadmin service outside

```
kubectl port-forward --address localhost,<VM IP> service/<pgadmin service name> -n
<postgre sql namespace> <NodePort>:80
e.g. kubectl port-forward --address localhost,192.168.16.133 service/pgadmin-service
-n postgresql 30943:80
```

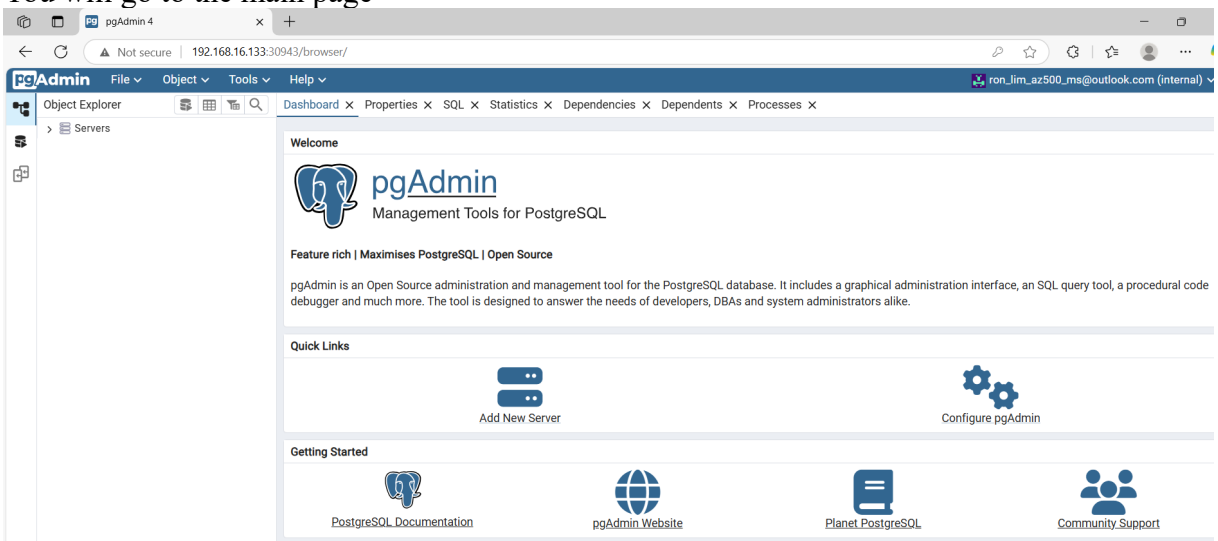
```
[ronslim@ansiblec ~]$ kubectl port-forward --address localhost,192.168.16.133 service/pgadmin-service -n postgresql 30943:80
Forwarding from 127.0.0.1:30943 -> 80
Forwarding from 192.168.16.133:30943 -> 80
Forwarding from [::1]:30943 -> 80
```

Try accessing the pgadmin again via http://<VM IP>/:<NodePort> (e.g. http://192.168.16.133:30943). Notice that the pgadmin is now accessible outside VM Browser.

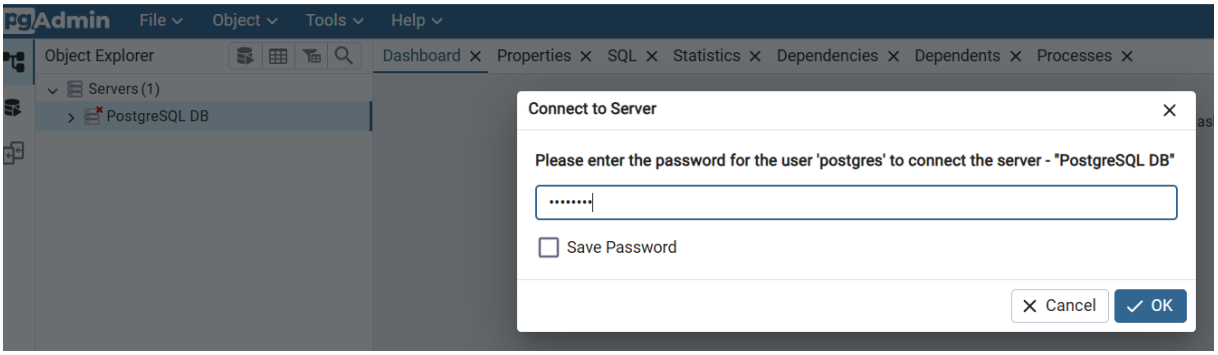
Then use PGADMIN_DEFAULT_EMAIL for username and desired password and not Base64 pgadmin-password



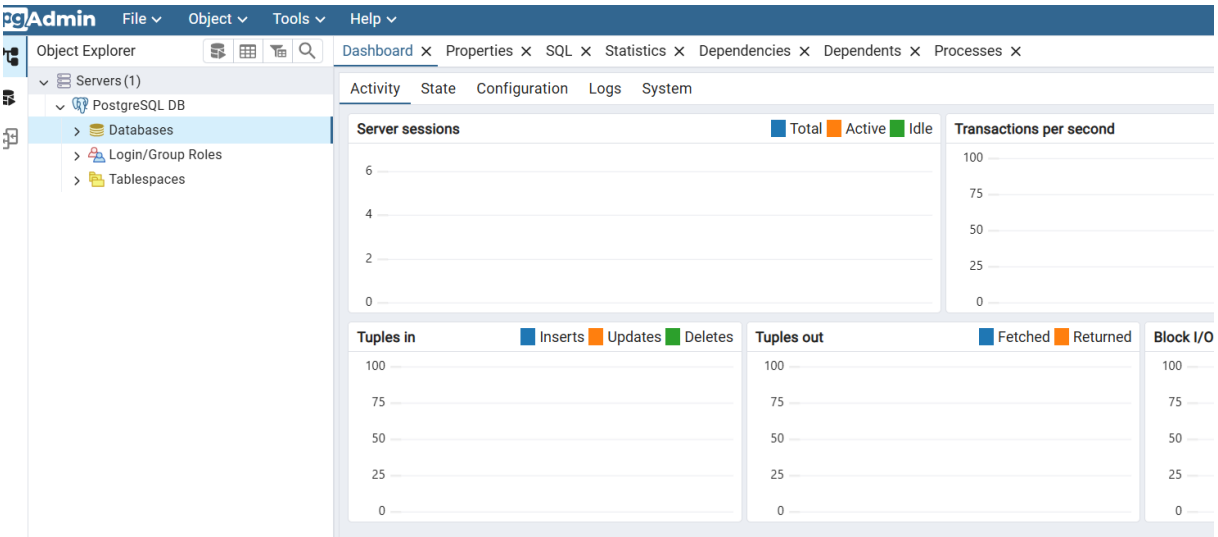
You will go to the main page



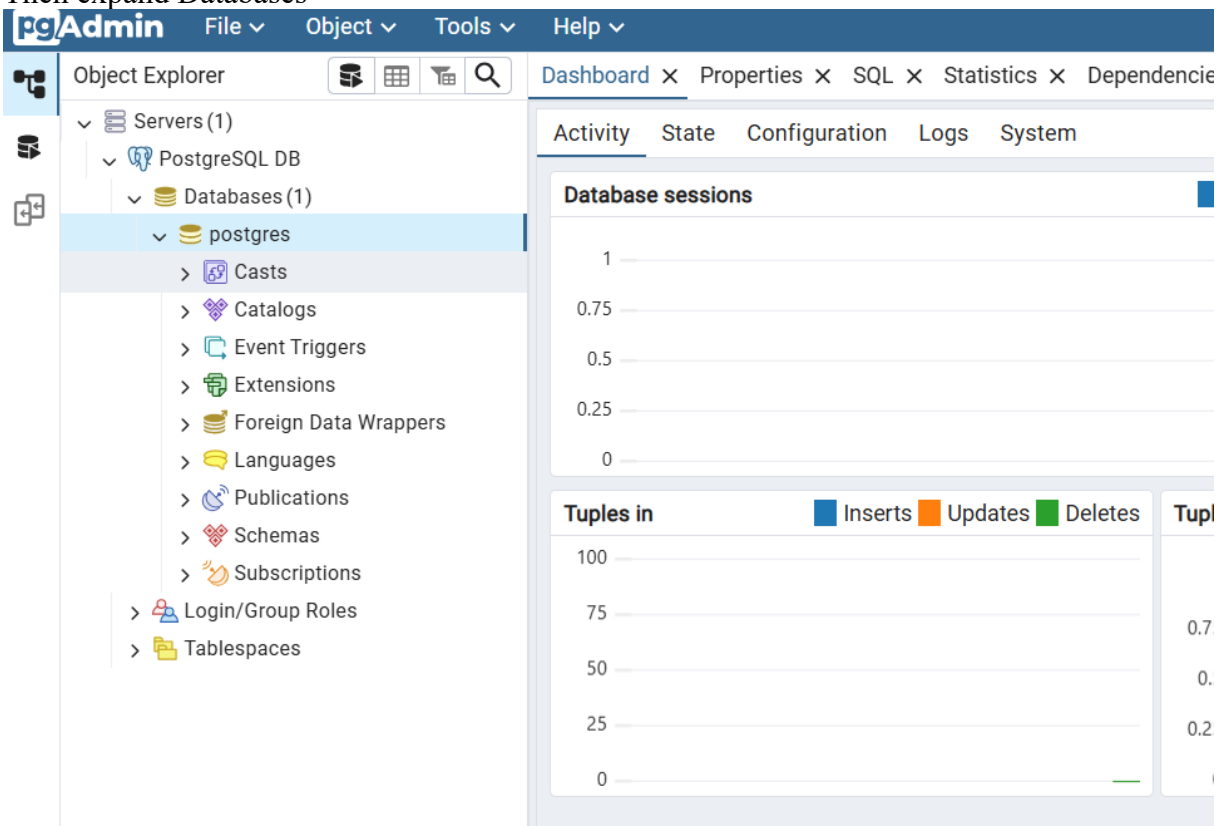
Then expand Servers and input the desired password and not Base64 pgadmin-password and click OK.



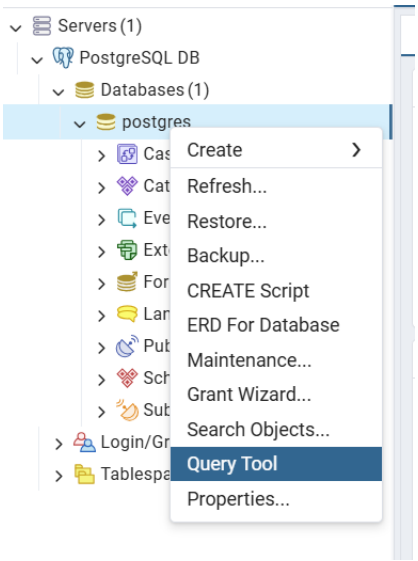
You are now logged in to the Postgre Database

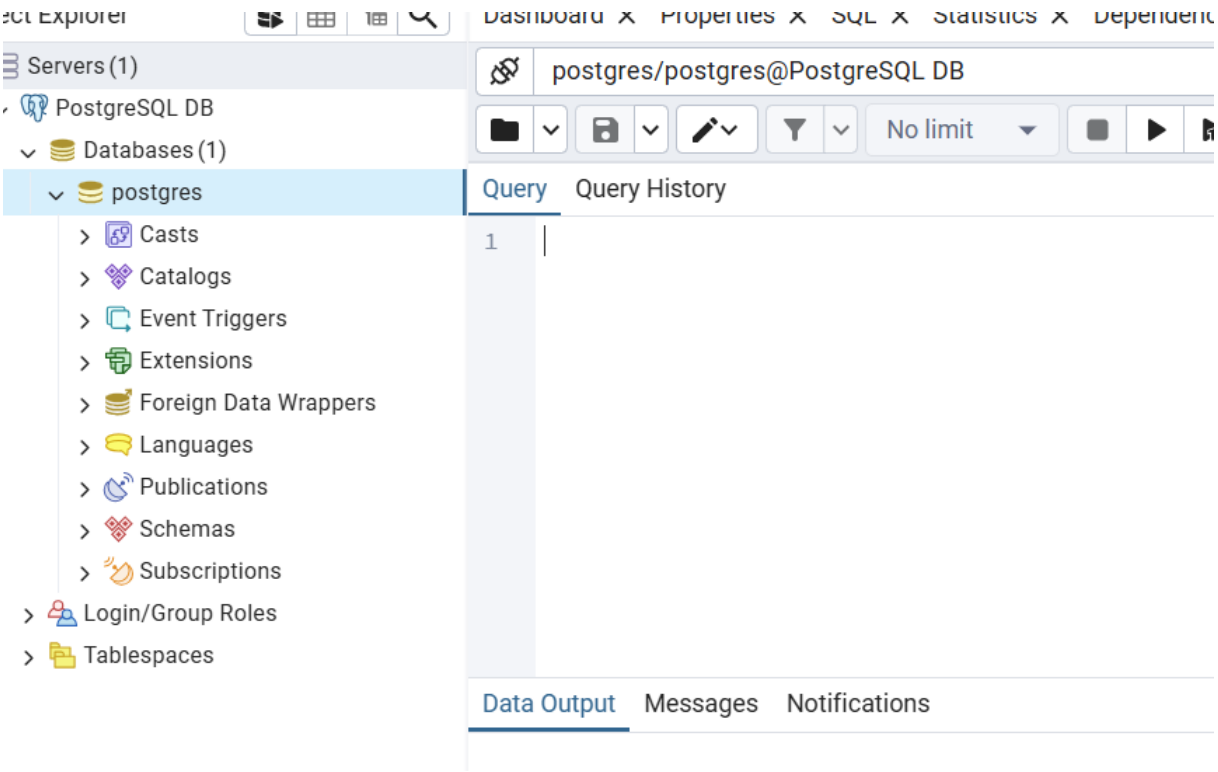


Then expand Databases



Then either right-click then click Query Tools to go to SQL Window and perform queries





Try to select the version
select version();

Query Query History

```
1  select version();
```

Data Output Messages Notifications

	version text
1	PostgreSQL 17.5 on x86_64-pc-linux-musl, compiled by gcc (Alpine 14.2.0) 14.2.0, 64-bit