## Successfully Connecting PostgreSQL on EC2 with EKS and Migrating Data from RDS

### Step 1 Setting Up PostgreSQL on EC2

I started with a fresh Ubuntu EC2 instance and installed PostgreSQL

sudo apt update

sudo apt install y postgresql postgresql contrib

Then I configured PostgreSQL to listen on all network interfaces

sudo nano /etc/postgresql/16/main/postgresql.conf

Set:

listen\_addresses = '\*'

Restart PostgreSQL:

sudo systemctl restart postgresql

This ensured that PostgreSQL could accept remote connections

### Step 2 Checking Security Group Rules

Since my database is on an EC2 instance I needed to allow incoming connections from EKS worker nodes

I updated the PostgreSQL EC2 security group

* **Port:** 5432
* **Protocol:** TCP
* **Source:** sg-00ak(EKS security group)

Additionally, I ensured the following security group rules were in place:

| **Security Group Rule ID** | **Type** | **Protocol** | **Port** | **Source** | **Description** |
| --- | --- | --- | --- | --- | --- |
| sgr-06953defdf | PostgreSQL | TCP | 5432 | 172.31.0.0/22­ | Only users connected to VPN |
| sgr-04fcd92ad2 | SSH | TCP | 22 | 172.31.4.201/32 | On VPN only |
| sgr-0735c78 | PostgreSQL | TCP | 5432 | (EKS security group) | EKS Cluster |

This allowed EKS worker nodes and VPN users to connect to PostgreSQL

### Step 3 Configuring VPC and Subnets for Security

To ensure smooth communication between EKS and PostgreSQL I verified my EKS cluster’s VPC and subnets.

My Ip range

192.168.0.0/16

This confirmed that EKS worker nodes were using IPs in the 192.168.0.0/16 range, which I needed to allow in PostgreSQL

### Step 4 Updating pg\_hba.conf for EKS Connectivity

To allow EKS pods, I updated pg\_hba.conf

sudo nano /etc/postgresql/16/main/pg\_hba.conf

Added:

# Allow VPN users

host all all 172.31.0.0/22 md5

# Allow EKS worker nodes

host all all 192.168.0.0/16 md5

Restarted PostgreSQL:

sudo systemctl restart postgresql

### Step 5 Optimizing PostgreSQL for Production

I optimized PostgreSQL for better performance by modifying /etc/postgresql/16/main/postgresql.conf

#### Memory Settings

shared\_buffers = 512MB # 25-40% of system memory

work\_mem = 16MB # Depending on complex query needs

maintenance\_work\_mem = 256MB # Allocate for maintenance operations like VACUUM

#### Connection Management

max\_connections = 100

#### WAL (Write-Ahead Logging) Settings

wal\_level = replica

checkpoint\_timeout = 5min

#### Enabling Logging for Performance Tuning

logging\_collector = on

log\_directory = '/var/log/postgresql'

log\_filename = 'postgresql.log'

Restarted PostgreSQL to apply changes:

sudo systemctl restart postgresql

### Step 6 Migrating Data from RDS to EC2 PostgreSQL

Since I had an existing RDS database I needed to migrate all data to the new EC2-hosted PostgreSQL instance

#### 1. Create a Dump from RDS

#### 2. Transfer Backup to EC2

#### 3. Restore Data to EC2 PostgreSQL

### Step 7 Testing Connection from EKS

Finally, I tested whether my EKS application pods could reach PostgreSQL

Success My EKS application could now connect to PostgreSQL without issues

## Key Takeaways

* **Security Group Configuration is Key** Ensure EKS worker nodes are allowed in PostgreSQL’s EC2 security group
* **Use the Right VPC CIDR Block** EKS pods use private IPs; allow the correct CIDR in pg\_hba.conf
* **Verify VPC and Subnets** Understanding how your EKS and PostgreSQL are networked helps prevent connectivity issues
* **Optimize PostgreSQL for Production** Adjust memory, connection, and logging settings to ensure stable performance
* **Seamless Data Migration** pg\_dump and pg\_restore make it easy to move data from RDS to EC2 PostgreSQL
* **Testing is Crucial** Use kubectl run to simulate an application pod and verify database connectivity

After these changes, my application is running smoothly with a production-ready PostgreSQL database on EC2 and an application on EKS

Have you faced similar networking or database migration challenges with AWS? Drop your experiences in the comments