Step by step to install **PostgreSQL 18** on **Ubuntu 24.04** running on an **AWS EC2 instance**.
For cloud setup, including security considerations.

Step 1: Launch and Connect to Your EC2 Instance

1. Launch an EC2 instance with **Ubuntu 24.04**.
2. Make sure **port 22** (SSH) is open in your Security Group.
3. Connect via SSH:

ssh -i /path/to/your-key.pem ubuntu@your-ec2-public-ip

Step 2: Update the System

Always update packages first:

sudo apt update && sudo apt upgrade -y

Step 3: Import PostgreSQL Repository

PostgreSQL 18 is not included in Ubuntu default repos. Use PostgreSQL's official repository:

1. Install dependencies:

sudo apt install -y wget gnupg2 lsb-release

2. Import the PostgreSQL signing key:

wget -qO - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo gpg --dearmor -o /usr/share/keyrings/pgdg.gpg

3. Add PostgreSQL 18 repository:

echo "deb [signed-by=/usr/share/keyrings/pgdg.gpg] http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main" | sudo tee /etc/apt/sources.list.d/pgdg.list

Step 4: Install PostgreSQL 18

1. Update package lists:

sudo apt update

2. Install PostgreSQL 18:

sudo apt install -y postgresql-18

3. Verify installation:

psql --version

You should see something like psql (PostgreSQL) 18.x.

Step 5: Start and Enable PostgreSQL Service

sudo systemctl start postgresql

sudo systemctl enable postgresql

sudo systemctl status postgresql

PostgreSQL should now be running.

Step 6: Switch to PostgreSQL User

PostgreSQL creates a postgres system user by default:

sudo -i -u postgres

Test access:

psql

You should enter the PostgreSQL shell (postgres=#). Exit with:

\q

Step 7: Create a Database and User (Optional)

sudo -i -u postgres

psql

Inside psql shell:

CREATE DATABASE mydb;

```
CREATE USER myuser WITH ENCRYPTED PASSWORD 'mypassword';
GRANT ALL PRIVILEGES ON DATABASE mydb TO myuser;
\q
```

Step 8: Configure Remote Access (Optional for AWS)

By default, PostgreSQL only listens on localhost. To allow external access:

1. Edit PostgreSQL config:

```
sudo nano /etc/postgresql/18/main/postgresql.conf
```

Find:

```
#listen_addresses = 'localhost'
```

Change to:

```
listen_addresses = '*'
```

2. Edit host-based authentication:

```
sudo nano /etc/postgresql/18/main/pg_hba.conf
```

Add at the end:

```
host    all             all             0.0.0.0/0               md5
```

3. Restart PostgreSQL:

```
sudo systemctl restart postgresql
```

4. Open **port 5432** in your EC2 Security Group for remote access.

Step 9: Test Remote Connection

From your local machine:

```
psql -h <EC2_PUBLIC_IP> -U myuser -d mydb
```

✅ **PostgreSQL 18 is now installed and ready on Ubuntu 24 EC2.**

3

Fully automated Bash script for installing **PostgreSQL 18** on **Ubuntu 24** EC2, including optional remote access setup.
You can copy-paste this into your EC2 terminal.

```
#!/bin/bash

# ============================
# PostgreSQL 18 Automated Setup
# Ubuntu 24 on AWS EC2
# ============================

# Exit immediately if a command exits with a non-zero status
set -e

# Variables - change these if needed
PG_VERSION=18
DB_NAME="mydb"
DB_USER="myuser"
DB_PASSWORD="mypassword"
ALLOW_REMOTE_ACCESS=true  # set to false to allow localhost only

# Update system
echo "Updating system packages..."
sudo apt update && sudo apt upgrade -y

# Install dependencies
echo "Installing dependencies..."
sudo apt install -y wget gnupg2 lsb-release

# Import PostgreSQL GPG key
echo "Adding PostgreSQL repository key..."
wget -qO - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo gpg --dearmor -o /usr/share/keyrings/pgdg.gpg

# Add PostgreSQL repository
echo "Adding PostgreSQL repository..."
echo "deb [signed-by=/usr/share/keyrings/pgdg.gpg] http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main" | sudo
tee /etc/apt/sources.list.d/pgdg.list

# Update package lists
sudo apt update

# Install PostgreSQL
echo "Installing PostgreSQL $PG_VERSION..."
sudo apt install -y postgresql-$PG_VERSION postgresql-client-$PG_VERSION

# Start and enable PostgreSQL service
sudo systemctl enable --now postgresql

# Create database and user
echo "Creating PostgreSQL user and database..."
sudo -i -u postgres psql <<EOF
CREATE DATABASE $DB_NAME;
```

```
CREATE USER $DB_USER WITH ENCRYPTED PASSWORD '$DB_PASSWORD';
GRANT ALL PRIVILEGES ON DATABASE $DB_NAME TO $DB_USER;
EOF

# Optional: Configure remote access
if [ "$ALLOW_REMOTE_ACCESS" = true ]; then
   echo "Configuring PostgreSQL for remote access..."

   # Update listen_addresses
   sudo sed -i "s/#listen_addresses = 'localhost'/listen_addresses = '*'/" /etc/postgresql/$PG_VERSION/main/postgresql.conf

   # Update pg_hba.conf to allow all IPs
   echo "host    all             all             0.0.0.0/0               md5" | sudo tee -a /etc/postgresql/$PG_VERSION/main/pg_hba.conf

   # Restart PostgreSQL
   sudo systemctl restart postgresql
   echo "Remote access enabled. Remember to open port 5432 in your EC2 Security Group."
fi

# Final status
echo "PostgreSQL $PG_VERSION installation complete!"
psql --version
echo "Database: $DB_NAME"
echo "User: $DB_USER"
```


✅ **How to use:**

1. Save it as install_pg18.sh on your EC2:

```
nano install_pg18.sh
```

2. Make it executable:

```
chmod +x install_pg18.sh
```

3. Run it:

```
./install_pg18.sh
```

This will:

- Install PostgreSQL 18
- Create a database and user
- Enable remote access (optional)
- Start PostgreSQL service automatically

Production-ready, secure PostgreSQL 18 setup for **Ubuntu 24 on AWS EC2**. This version will:

- Only allow your specific IP to connect (instead of 0.0.0.0/0)
- Use strong passwords
- Keep default PostgreSQL security settings intact
- Include all database and user setup

Secure PostgreSQL 18 Installation Script

```
#!/bin/bash

# ============================
# Secure PostgreSQL 18 Setup
# Ubuntu 24 on AWS EC2
# ============================

set -e

# ====== User Config ======
PG_VERSION=18
DB_NAME="mydb"
DB_USER="myuser"
DB_PASSWORD="StrongPassword123!"   # Change to a strong password
ALLOW_REMOTE_ACCESS=true          # Set to false for localhost-only
YOUR_IP="YOUR_PUBLIC_IP/32"       # Replace YOUR_PUBLIC_IP with your current IP
# ==========================

# Update system
echo "Updating system packages..."
sudo apt update && sudo apt upgrade -y

# Install dependencies
echo "Installing dependencies..."
sudo apt install -y wget gnupg2 lsb-release

# Import PostgreSQL GPG key
echo "Adding PostgreSQL repository key..."
wget -qO - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo gpg --dearmor -o /usr/share/keyrings/pgdg.gpg

# Add PostgreSQL repository
echo "Adding PostgreSQL repository..."
echo "deb [signed-by=/usr/share/keyrings/pgdg.gpg] http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main" | sudo
tee /etc/apt/sources.list.d/pgdg.list

# Update package lists
sudo apt update

# Install PostgreSQL
echo "Installing PostgreSQL $PG_VERSION..."
sudo apt install -y postgresql-$PG_VERSION postgresql-client-$PG_VERSION

# Start and enable PostgreSQL service
```

```
sudo systemctl enable --now postgresql

# Create database and user
echo "Creating PostgreSQL user and database..."
sudo -i -u postgres psql <<EOF
CREATE DATABASE $DB_NAME;
CREATE USER $DB_USER WITH ENCRYPTED PASSWORD '$DB_PASSWORD';
GRANT ALL PRIVILEGES ON DATABASE $DB_NAME TO $DB_USER;
EOF

# Configure secure remote access
if [ "$ALLOW_REMOTE_ACCESS" = true ]; then
  echo "Configuring PostgreSQL for remote access from $YOUR_IP..."

  # Update listen_addresses
  sudo sed -i "s/#listen_addresses = 'localhost'/listen_addresses = '*'/" /etc/postgresql/$PG_VERSION/main/postgresql.conf

  # Update pg_hba.conf to allow only your IP
  echo "host    all             all             $YOUR_IP            md5" | sudo tee -a /etc/postgresql/$PG_VERSION/main/pg_hba.conf

  # Restart PostgreSQL
  sudo systemctl restart postgresql

  echo "Remote access configured. Make sure port 5432 is open in your EC2 Security Group for IP: $YOUR_IP"
fi

# Final status
echo "----------------------------------------"
echo "PostgreSQL $PG_VERSION installation complete!"
echo "Database: $DB_NAME"
echo "User: $DB_USER"
echo "Password: $DB_PASSWORD"
echo "Secure remote access: $ALLOW_REMOTE_ACCESS"
psql --version
echo "----------------------------------------"
```


How to Use:

1. Replace YOUR_PUBLIC_IP with your local/public IP (you can find it via curl https://checkip.amazonaws.com).
2. Save the file on your EC2 instance, e.g., secure_pg18.sh
3. Make it executable:

```
chmod +x secure_pg18.sh
```

4. Run it:

```
./secure_pg18.sh
```

✅ **What this does differently:**

- Only allows your IP to access PostgreSQL (no wide open access)
- Keeps all other PostgreSQL security defaults
- Strong password enforced
- Prepares database and user for immediate use

Let's upgrade setup to **PostgreSQL 18 with SSL/TLS** on **Ubuntu 24 EC2**, so all remote connections are encrypted. This is highly recommended for production environments.
We'll generate **self-signed SSL certificates** for PostgreSQL, configure it, and still keep the **secure IP-only access**.

Secure PostgreSQL 18 with SSL Script

```
#!/bin/bash

# ================================
# PostgreSQL 18 Secure Setup with SSL
# Ubuntu 24 on AWS EC2
# ================================

set -e

# ====== User Config ======
PG_VERSION=18
DB_NAME="mydb"
DB_USER="myuser"
DB_PASSWORD="StrongPassword123!"      # Change to a strong password
ALLOW_REMOTE_ACCESS=true              # Set to false for localhost-only
YOUR_IP="YOUR_PUBLIC_IP/32"           # Replace with your public IP
SSL_DIR="/etc/postgresql/$PG_VERSION/main/ssl"
# ============================

# Update system
echo "Updating system packages..."
sudo apt update && sudo apt upgrade -y

# Install dependencies
echo "Installing dependencies..."
sudo apt install -y wget gnupg2 lsb-release openssl

# Import PostgreSQL GPG key
echo "Adding PostgreSQL repository key..."
wget -qO - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo gpg --dearmor -o /usr/share/keyrings/pgdg.gpg

# Add PostgreSQL repository
echo "Adding PostgreSQL repository..."
echo "deb [signed-by=/usr/share/keyrings/pgdg.gpg] http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main" | sudo tee /etc/apt/sources.list.d/pgdg.list

# Update package lists
sudo apt update

# Install PostgreSQL
echo "Installing PostgreSQL $PG_VERSION..."
sudo apt install -y postgresql-$PG_VERSION postgresql-client-$PG_VERSION

# Start and enable PostgreSQL service
sudo systemctl enable --now postgresql
```

```
# Create database and user
echo "Creating PostgreSQL user and database..."
sudo -i -u postgres psql <<EOF
CREATE DATABASE $DB_NAME;
CREATE USER $DB_USER WITH ENCRYPTED PASSWORD '$DB_PASSWORD';
GRANT ALL PRIVILEGES ON DATABASE $DB_NAME TO $DB_USER;
EOF

# Create SSL certificates
echo "Creating SSL certificates for PostgreSQL..."
sudo mkdir -p $SSL_DIR
sudo chmod 700 $SSL_DIR
sudo openssl req -new -x509 -days 365 -nodes -text -out $SSL_DIR/server.crt -keyout $SSL_DIR/server.key -subj "/CN=$DB_USER"
sudo chmod 600 $SSL_DIR/server.key
sudo chown postgres:postgres $SSL_DIR/server.*

# Configure PostgreSQL for SSL
sudo sed -i "s/#ssl = off/ssl = on/" /etc/postgresql/$PG_VERSION/main/postgresql.conf
sudo sed -i "s|#ssl_cert_file = 'server.crt'|ssl_cert_file = '$SSL_DIR/server.crt'|" /etc/postgresql/$PG_VERSION/main/postgresql.conf
sudo sed -i "s|#ssl_key_file = 'server.key'|ssl_key_file = '$SSL_DIR/server.key'|"
/etc/postgresql/$PG_VERSION/main/postgresql.conf

# Configure secure remote access
if [ "$ALLOW_REMOTE_ACCESS" = true ]; then
    echo "Configuring PostgreSQL for remote access from $YOUR_IP..."

    # Update listen_addresses
    sudo sed -i "s/#listen_addresses = 'localhost'/listen_addresses = '*'/" /etc/postgresql/$PG_VERSION/main/postgresql.conf

    # Update pg_hba.conf to allow SSL connections only from your IP
    echo "hostssl    all          all          $YOUR_IP            md5" | sudo tee -a /etc/postgresql/$PG_VERSION/main/pg_hba.conf

    # Restart PostgreSQL
    sudo systemctl restart postgresql

    echo "Remote SSL access configured. Ensure port 5432 is open in your EC2 Security Group for IP: $YOUR_IP"
fi

# Final status
echo "----------------------------------------"
echo "PostgreSQL $PG_VERSION installation complete with SSL!"
echo "Database: $DB_NAME"
echo "User: $DB_USER"
echo "Password: $DB_PASSWORD"
echo "SSL certificates located at: $SSL_DIR"
echo "Secure remote access: $ALLOW_REMOTE_ACCESS"
psql --version
echo "----------------------------------------"
```

How to Use

1. Replace YOUR_PUBLIC_IP with your current public IP:

curl https://checkip.amazonaws.com

2. Save the file as secure_pg18_ssl.sh on your EC2 instance.
3. Make it executable:

chmod +x secure_pg18_ssl.sh

4. Run the script:

./secure_pg18_ssl.sh

Testing SSL Connection

From your local machine:

psql "host=<EC2_PUBLIC_IP> port=5432 dbname=mydb user=myuser password=StrongPassword123! sslmode=require"

The sslmode=require ensures your connection is encrypted.

PostgreSQL 18 on Ubuntu 24 EC2 with **Let's Encrypt SSL certificates**.
This way, your PostgreSQL server uses **publicly trusted certificates**, so clients don't have to bypass SSL warnings.
We'll use **Certbot** to obtain certificates and configure PostgreSQL to use them, with **secure IP-only access**.

Prerequisites

1. You need a **public domain name** pointing to your EC2 instance. PostgreSQL SSL with Let's Encrypt cannot use just an IP. Example: db.example.com.
2. **Port 80** must be temporarily open in your Security Group for domain validation.
3. Your domain must point to the EC2 public IP.

Secure PostgreSQL 18 with Let's Encrypt SSL Script

```
#!/bin/bash

# ================================
# PostgreSQL 18 Secure Setup with Let's Encrypt SSL
# Ubuntu 24 on AWS EC2
# ================================

set -e

# ====== User Config ======
PG_VERSION=18
DB_NAME="mydb"
DB_USER="myuser"
DB_PASSWORD="StrongPassword123!"     # Change to a strong password
ALLOW_REMOTE_ACCESS=true             # Set to false for localhost-only
YOUR_IP="YOUR_PUBLIC_IP/32"          # Replace with your public IP
DOMAIN_NAME="db.example.com"         # Replace with your domain
SSL_DIR="/etc/letsencrypt/live/$DOMAIN_NAME"
# ============================

# Update system
echo "Updating system packages..."
sudo apt update && sudo apt upgrade -y

# Install dependencies
echo "Installing dependencies..."
sudo apt install -y wget gnupg2 lsb-release software-properties-common certbot

# Import PostgreSQL GPG key
echo "Adding PostgreSQL repository key..."
wget -qO - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo gpg --dearmor -o /usr/share/keyrings/pgdg.gpg

# Add PostgreSQL repository
echo "Adding PostgreSQL repository..."
echo "deb [signed-by=/usr/share/keyrings/pgdg.gpg] http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -cs)-pgdg main" | sudo
tee /etc/apt/sources.list.d/pgdg.list

# Update package lists
sudo apt update
```

```
# Install PostgreSQL
echo "Installing PostgreSQL $PG_VERSION..."
sudo apt install -y postgresql-$PG_VERSION postgresql-client-$PG_VERSION

# Start and enable PostgreSQL service
sudo systemctl enable --now postgresql

# Create database and user
echo "Creating PostgreSQL user and database..."
sudo -i -u postgres psql <<EOF
CREATE DATABASE $DB_NAME;
CREATE USER $DB_USER WITH ENCRYPTED PASSWORD '$DB_PASSWORD';
GRANT ALL PRIVILEGES ON DATABASE $DB_NAME TO $DB_USER;
EOF

# Obtain SSL certificate using Certbot (standalone)
echo "Obtaining Let's Encrypt SSL certificate for $DOMAIN_NAME..."
sudo systemctl stop postgresql
sudo certbot certonly --standalone -d $DOMAIN_NAME --non-interactive --agree-tos -m admin@$DOMAIN_NAME
sudo systemctl start postgresql

# Configure PostgreSQL for SSL
echo "Configuring PostgreSQL to use Let's Encrypt SSL..."
sudo sed -i "s/#ssl = off/ssl = on/" /etc/postgresql/$PG_VERSION/main/postgresql.conf
sudo sed -i "s|#ssl_cert_file = 'server.crt'|ssl_cert_file = '$SSL_DIR/fullchain.pem'|"
/etc/postgresql/$PG_VERSION/main/postgresql.conf
sudo sed -i "s|#ssl_key_file = 'server.key'|ssl_key_file = '$SSL_DIR/privkey.pem'|"
/etc/postgresql/$PG_VERSION/main/postgresql.conf
sudo chown postgres:postgres $SSL_DIR/privkey.pem $SSL_DIR/fullchain.pem
sudo chmod 600 $SSL_DIR/privkey.pem

# Configure secure remote access
if [ "$ALLOW_REMOTE_ACCESS" = true ]; then
    echo "Configuring PostgreSQL for remote access from $YOUR_IP..."

    # Update listen_addresses
    sudo sed -i "s/#listen_addresses = 'localhost'/listen_addresses = '*'/" /etc/postgresql/$PG_VERSION/main/postgresql.conf

    # Update pg_hba.conf to allow SSL connections only from your IP
    echo "hostssl    all             all             $YOUR_IP                md5" | sudo tee -a /etc/postgresql/$PG_VERSION/main/pg_hba.conf

    # Restart PostgreSQL
    sudo systemctl restart postgresql

    echo "Remote SSL access configured. Ensure port 5432 is open in your EC2 Security Group for IP: $YOUR_IP"
fi

# Setup auto-renewal for PostgreSQL certificates
echo "Configuring auto-reload of PostgreSQL after SSL renewal..."
```

```
sudo bash -c 'echo "postrenew_hook='systemctl restart postgresql'" >> /etc/letsencrypt/renewal-hooks/deploy/postgresql_restart.sh'
sudo chmod +x /etc/letsencrypt/renewal-hooks/deploy/postgresql_restart.sh

# Final status
echo "----------------------------------------"
echo "PostgreSQL $PG_VERSION installation complete with Let's Encrypt SSL!"
echo "Database: $DB_NAME"
echo "User: $DB_USER"
echo "Password: $DB_PASSWORD"
echo "Domain: $DOMAIN_NAME"
echo "Secure remote access: $ALLOW_REMOTE_ACCESS"
psql --version
echo "----------------------------------------"
```

How to Use

1. Replace:
    - YOUR_PUBLIC_IP → your local/public IP
    - DOMAIN_NAME → your domain pointing to the EC2 instance
2. Save as secure_pg18_letsencrypt.sh on EC2.
3. Make it executable:

```
chmod +x secure_pg18_letsencrypt.sh
```

4. Run the script:

```
./secure_pg18_letsencrypt.sh
```


Testing SSL Connection

```
psql "host=db.example.com port=5432 dbname=mydb user=myuser password=StrongPassword123! sslmode=require"
```

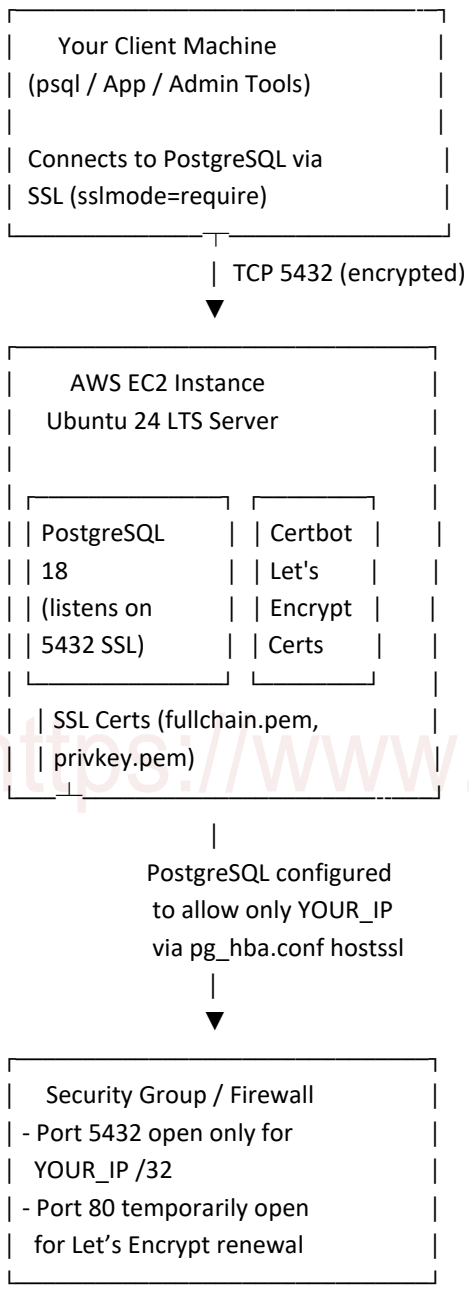
✅ sslmode=require ensures a trusted encrypted connection.

This setup is **production-ready**, secure, and automatically renews SSL certificates without downtime.

Diagram of secure PostgreSQL 18 setup on AWS EC2 with Let's Encrypt SSL.

This will describe the architecture so you can visualize the flow of connections, security, and components.

Architecture Diagram (Text/ASCII Version)



Flow Explanation

1. **Client → EC2**
   - Your client connects using psql or an app.
   - Connection is **encrypted using SSL** (fullchain.pem and privkey.pem).
   - Only your IP is allowed in the firewall and PostgreSQL (pg_hba.conf) for remote access.
2. **EC2 Components**
   - **PostgreSQL 18** handles all database operations.
   - **Certbot** manages Let's Encrypt SSL certificates.
   - PostgreSQL points to the **Let's Encrypt certs** for SSL encryption.
   - Automatic reload of PostgreSQL after certificate renewal ensures zero downtime.

3. **Security Layer**
    - AWS Security Group only allows **port 5432** for your IP.
    - **Port 80** is temporarily used by Certbot for SSL validation and can be closed afterward.
    - All traffic to PostgreSQL is encrypted; no plaintext passwords over the network.