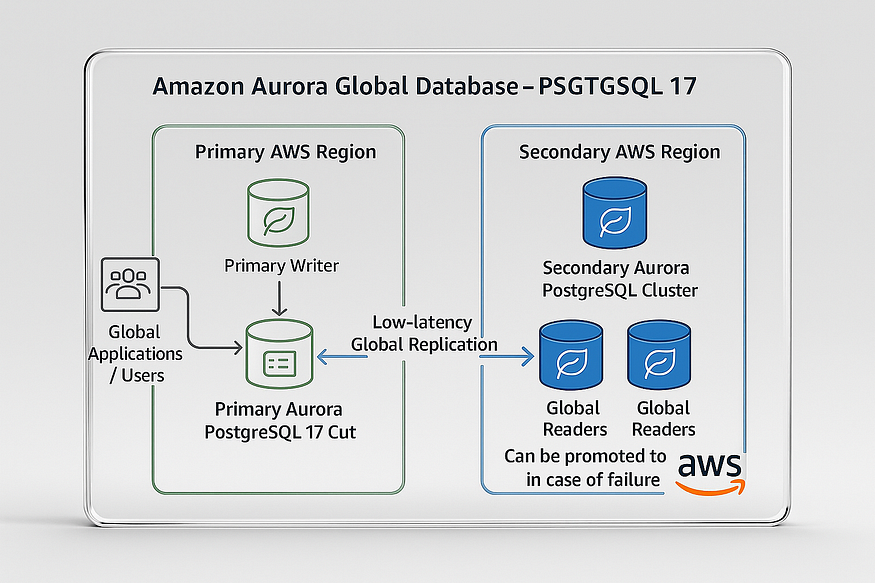
# **01 - Aurora Global Database: Creating an Amazon Aurora Global Database with PostgreSQL 17**



Amazon Aurora Global Database enables low-latency global reads and fast disaster recovery across AWS Regions. With Aurora PostgreSQL-Compatible edition (now supporting PostgreSQL 17), you can create a global database directly from the AWS Management Console, the AWS CLI, or the RDS API.

**Note**: If you already have an Aurora DB cluster running a global-compatible Aurora PostgreSQL engine, you don’t need to start from scratch. You can simply add another AWS Region to the existing DB cluster to extend it into a global database.

## **Using the AWS Management Console**

The process begins in an AWS Region that supports Aurora Global Database. You’ll need to ensure that you select a supported Region and engine version. For reference, AWS provides an up-to-date list of Regions and engines that support Aurora Global Database.

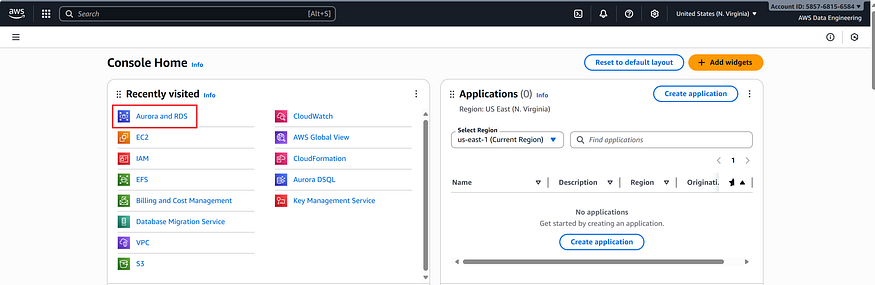
Before you begin, one of the key setup tasks is selecting a Virtual Private Cloud (VPC). Aurora DB clusters run inside a VPC, so you can either use the default VPC or create your own for greater control. If you choose to create a custom VPC, you should also prepare any related subnets, subnet groups, and security groups in advance. AWS provides a step-by-step tutorial: *Create a VPC for use with a DB cluster (IPv4 only)*.

For general Aurora DB cluster creation concepts, see *Creating an Amazon Aurora DB cluster*. Once your networking environment is ready, you can proceed with the following steps.

## **Step-by-Step: Creating a Global Database**

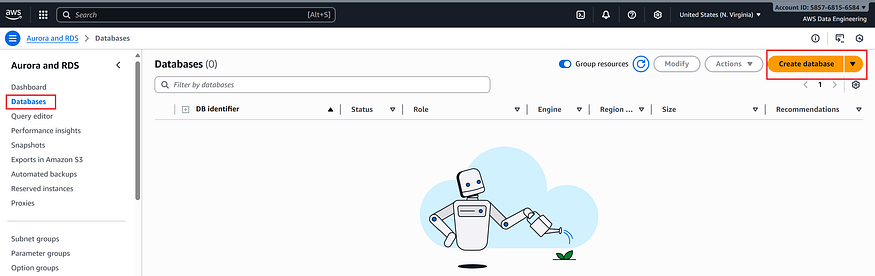
1. ****Sign in**** to the AWS Management Console and open the [Amazon RDS Console](https://console.aws.amazon.com/rds/" \t "https://medium.com/@jramcloud1/_blank).

Press enter or click to view image in full size



2. Choose ****Create database****.

Press enter or click to view image in full size



3. On the *Create database* page, select:

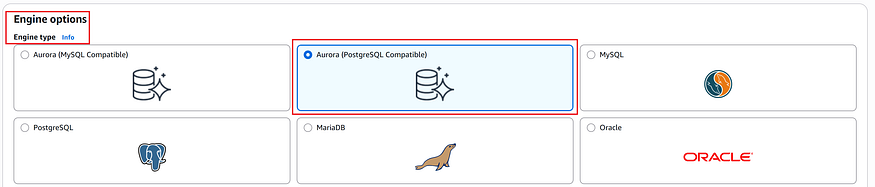
* ****Standard create**** (not Easy create).

Press enter or click to view image in full size



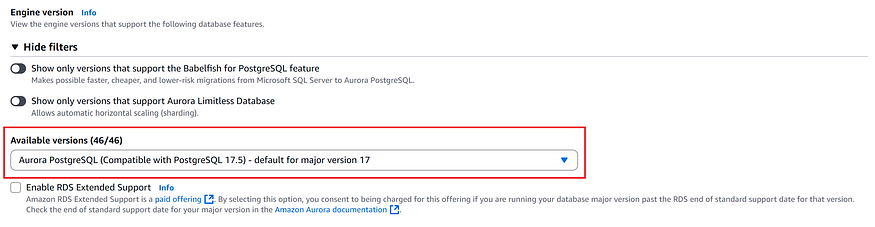
* ****Engine type****: Aurora (PostgreSQL-Compatible).

Press enter or click to view image in full size



* ****Engine version****: PostgreSQL 17 (latest global-compatible release).

Press enter or click to view image in full size



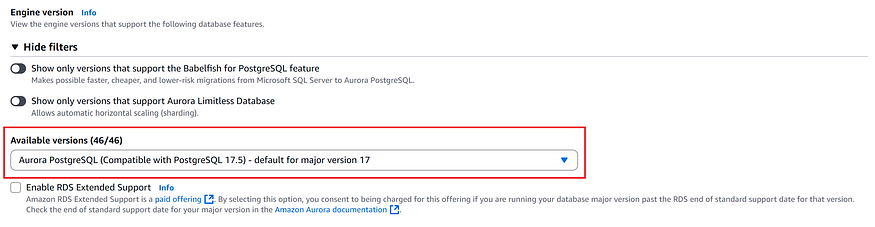
4. Continue with the following configuration details.

## **Creating a Global Database Using Aurora PostgreSQL**

1. ****Engine Options****

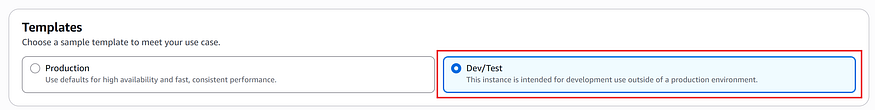
* ****Engine version****: Select PostgreSQL 17.

Press enter or click to view image in full size



* ****Templates****: Choose ****Production**** for high availability. Use ****Dev/Test**** only for non-production workloads.

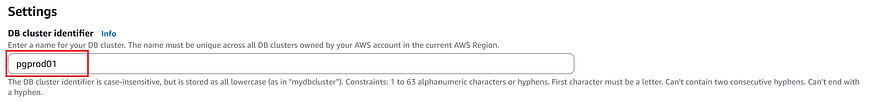
Press enter or click to view image in full size



2. ****Settings****

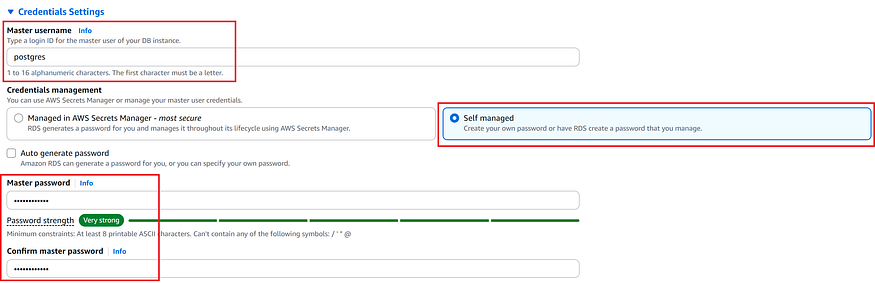
* Provide a meaningful ****DB cluster identifier**** (this identifies the primary DB cluster of your global database).

Press enter or click to view image in full size



* Set the ****admin password**** manually, or allow Aurora to auto-generate it and copy it securely.

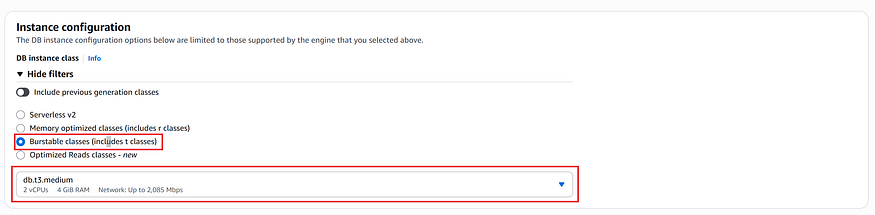
Press enter or click to view image in full size



3. ****DB Instance Class****

* Choose db.r5.large or higher. We recommend a memory-optimized class such as db.r5 or larger for production-grade performance.

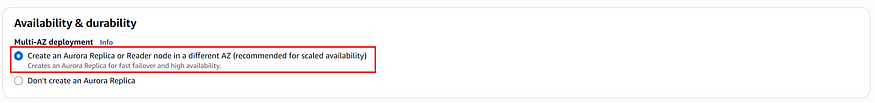
Press enter or click to view image in full size



4. ****Availability & Durability****

* Enable creation of an ****Aurora Replica**** in a different Availability Zone (AZ) for high availability. If skipped, you must add one later.

Press enter or click to view image in full size



5. ****Connectivity****

✅ 1. ****Compute Resource****

* ****Selected Option****: *Don’t connect to an EC2 compute resource*
* This means you’re ****not linking the database to an EC2 instance**** during creation.
* You can still connect to it later from an EC2 instance or any external compute resource.

💡 Use this option if you’re managing compute separately or using serverless access patterns.

✅ 2. ****Network Type****

* ****Selected Option****: *IPv4*
* Ensures your database is reachable over standard IPv4 networking.
* Make sure your ****RDS DB subnet group**** is associated with a subnet in the selected VPC.

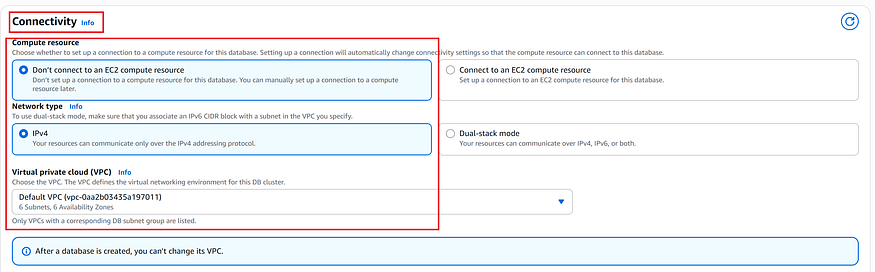
🧠 Tip: IPv4 is widely supported and typically sufficient unless you need dual-stack (IPv4/IPv6) compatibility.

✅ 3. ****Virtual Private Cloud (VPC)****

* ****Selected Option****: *Default VPC (vpc-0e4a3b55a51437011)*
* Your database will be deployed inside this VPC, which defines its private network boundaries.

⚠️ Note: Once the database is created, ****you cannot change its VPC**** — choose carefully.

Press enter or click to view image in full size



6. ****Configuring Network Settings for Your Aurora Global Database****

When creating an Aurora global database, one of the critical steps is configuring the network settings. These settings determine how your DB cluster will communicate within your AWS environment and whether it can be accessed publicly.

****1. DB Subnet Group****

Aurora DB clusters run inside a ****Virtual Private Cloud (VPC)****, and subnets define which Availability Zones (AZs) and IP ranges your cluster can use.

* Choose the ****DB subnet group**** from the dropdown list.
* In most cases, you’ll see a default VPC-based subnet group (for example, default-vpc-0aa2b03435a197011) that spans multiple subnets and Availability Zones.
* If you want tighter control, you can pre-create a custom subnet group with the exact subnets your workload requires.

This ensures your cluster is deployed in multiple Availability Zones, improving fault tolerance and availability.

****2. Public Access****

Next, you decide whether your Aurora cluster will be accessible from outside the VPC:

* ****Yes**** → RDS assigns a ****public IP address**** to the cluster. This allows Amazon EC2 instances and external resources outside of the VPC to connect.
* ****No**** → No public IP is assigned. Only Amazon EC2 instances and other resources ****inside the VPC**** can connect.

For production databases, ****“No” is recommended**** for security. If you choose “Yes,” ensure that your security group rules are tightly controlled to prevent unauthorized access.

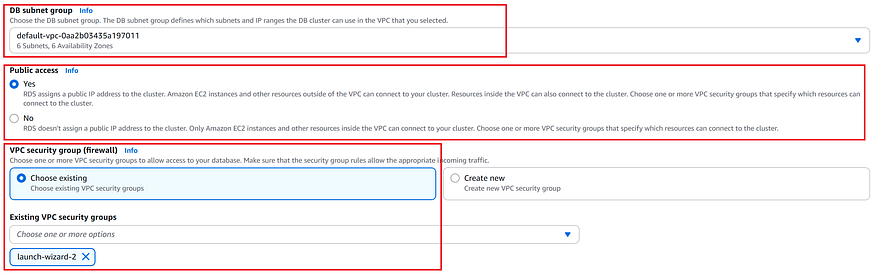
****3. VPC Security Group (Firewall)****

Finally, you configure the firewall rules by selecting a ****VPC security group****:

* Choose ****“Existing”**** to reuse a previously created security group (for example, launch-wizard-2).
* Or select ****“Create new”**** if you want a dedicated security group for this cluster.

The security group defines which inbound and outbound traffic is allowed to your Aurora DB cluster. For example, you can allow connections only from specific EC2 instances, application servers, or IP ranges.

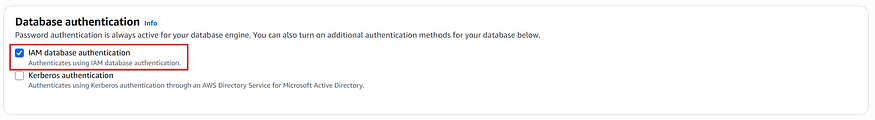
Press enter or click to view image in full size



7. ****Database Authentication****

* By default, password authentication is enabled. You can optionally configure IAM authentication or Kerberos later.

Press enter or click to view image in full size



8. ****Additional Configuration****

* Provide an ****Initial database name**** to create the primary writer node.
* Leave defaults for ****DB cluster parameter groups**** unless you have custom groups.
* Accept defaults for encryption, logging, and monitoring unless your organization requires custom settings.

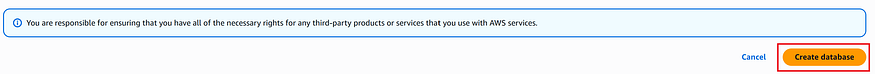
Press enter or click to view image in full size



9. ****Create Database****

* Choose ****Create database****. The process may take several minutes while Aurora provisions the cluster, writer node, and replica(s).
* Once ready, the cluster status changes to ****Available****. This cluster becomes the ****primary DB cluster**** of your Aurora Global Database.

Press enter or click to view image in full size



At this point, you can extend your deployment to additional AWS Regions by adding ****secondary clusters****, which provide global read scaling and cross-Region disaster recovery.

## **Verifying the Aurora DB Cluster**

Once you finish creating your Aurora DB cluster, you can verify that it is ready in the ****Amazon RDS console****.

1. ****Navigate to Databases****

* In the left-hand navigation pane, choose ****Databases****.
* This opens the list of all your Aurora and RDS databases.

2. ****Check the Cluster Status****

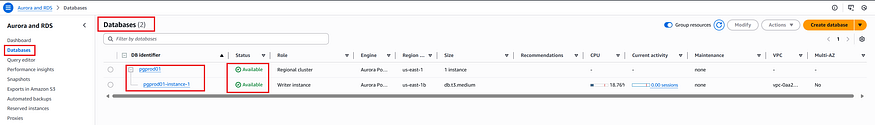
* Locate your newly created cluster (for example, ****pgprod01****).
* Expand the cluster to view the DB instance (for example, ****pgprod01-instance-1****).
* Ensure that both the ****Regional cluster**** and the ****Writer instance**** show the status ****Available**** (indicated by a green check mark).

3. ****Confirm the Details****

* The console will also display important details such as the DB engine (Aurora PostgreSQL 17), Region (e.g., us-east-1), instance size (e.g., db.t3.medium), and VPC configuration.
* At this point, your cluster is up and running, and you can proceed to add additional Regions to create a global database.

The screenshot below shows an example where the cluster ****pgprod01**** and its writer instance are both in the ****Available**** state:

Press enter or click to view image in full size



✅ With these steps, you’ve successfully created the primary cluster of an Aurora Global Database running PostgreSQL 17.