Notes

AWS Project Documentation

Connect A Web App with AWS Aurora

@sAnandha

Notes

AWS AURORA :

Amazon Aurora is a fully managed relational database engine designed for high performance, availability, and compatibility with popular database systems. It is part of Amazon's RDS (Relational Database Service) and supports both MySQL and PostgreSQL.

Key Features of Amazon Aurora

1. High Performance:
   - 5x faster than MySQL and 3x faster than PostgreSQL while maintaining compatibility.
   - Delivers throughput comparable to commercial databases at a fraction of the cost.
2. Scalability:
   - Automatically scales storage up to 128 TB per database instance.
   - Allows up to 15 low-latency read replicas to handle read-heavy workloads.
3. High Availability:
   - Provides 99.99% availability with failover support.
   - Automatically replicates data across multiple Availability Zones (AZs).

Notes

- Managed Service:
  - Handles database setup, patching, backups, maintenance
  - Offers automated backups with point-in-time recovery.
- Global Database:
  - Supports cross-region replication for low-latency access and disaster recovery.
- Security:
  - Integrated with AWS IAM, Amazon VPC, and encryption features.
  - Data is encrypted at rest and in transit.

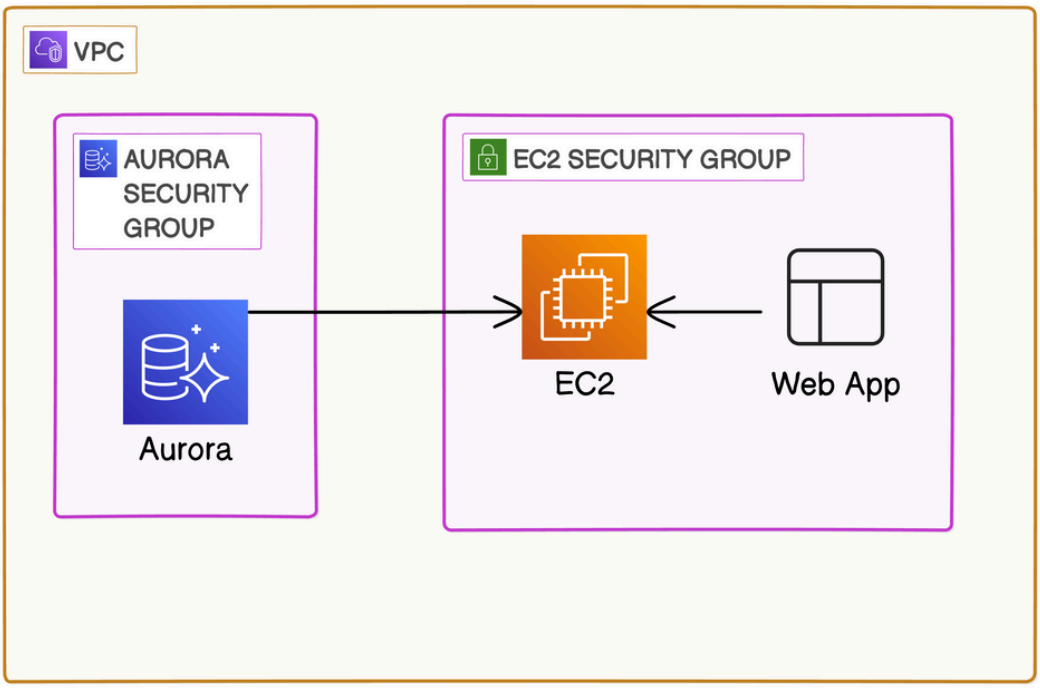
Benefits:

- Ease of Use: Fully managed by AWS, reducing the administrative burden.
- Flexibility: Compatible with popular open-source databases.
- Performance: Optimized for high throughput with minimal latency.

Common Use Cases :

- Web and Mobile Applications: High-performance databases for e-commerce, social media, and SaaS platforms.
- Analytics and Reporting: Handles large volumes of data for real-time analysis.

Architecture :



Steps to build :

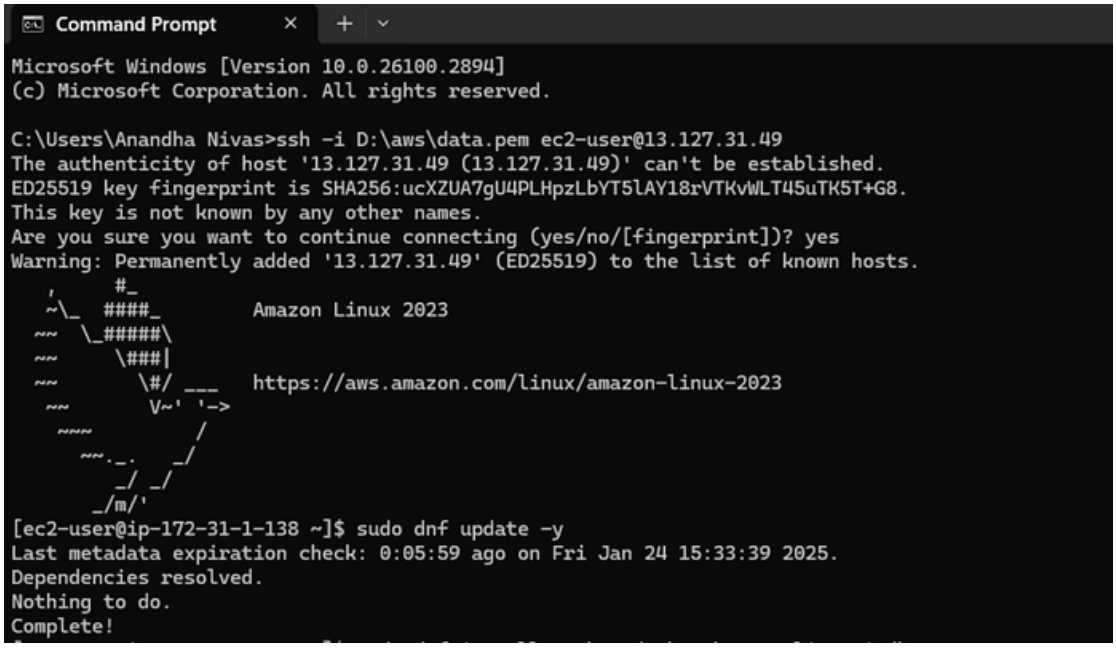
To connect a web app with an Amazon Aurora database using an EC2 instance and PHP, follow these steps:

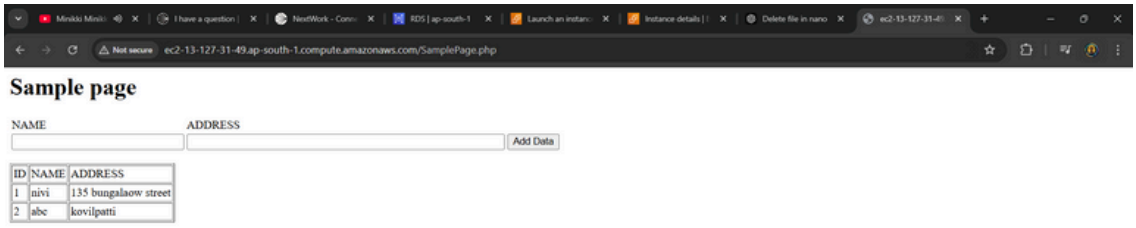
1. Set Up the EC2 Instance
   - Launch an EC2 instance in the same region as the Aurora cluster.
   - Assign the instance to a security group that allows outbound traffic to the Aurora database.
   - Install PHP and a web server (e.g., Apache or Nginx) on the EC2 instance.
2. Create an Aurora Cluster
   - Set up an Aurora database cluster in AWS, choosing the appropriate engine (MySQL or PostgreSQL).

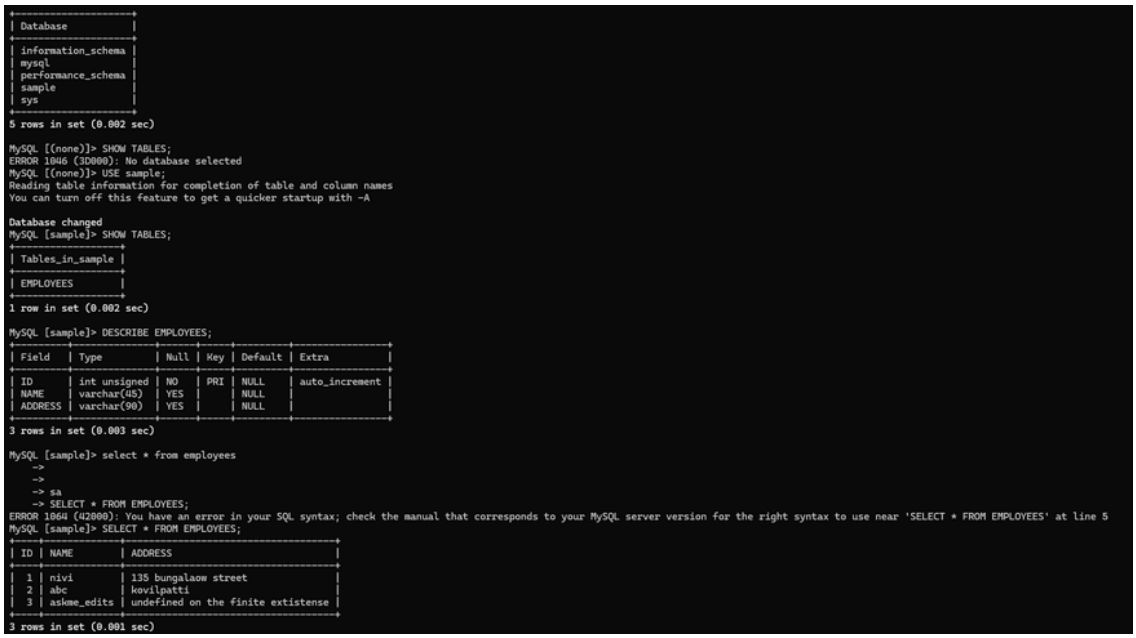
Notes

- 3. Configure Security Groups
- Ensure the Aurora cluster's security group allows inbound traffic from the EC2 instance's security group on the database port (e.g., 3306 for MySQL).
- Verify both resources are in the same VPC or establish connectivity between VPCs if necessary.
- 4. Configure the Web App
- On the EC2 instance, create or upload the PHP application files.
- Store database connection credentials (endpoint, username, password) securely, such as in environment variables or AWS Secrets Manager.
- 5. Establish the Connection
- Use PHP's database extensions (e.g., PDO or MySQLi) to connect to the Aurora database using the cluster endpoint and credentials.
- Test the connection by running queries against the database.
- 6. Enhance Security
- Use IAM roles and database authentication for better security, if supported by your Aurora engine.
- 7. Monitor and Optimize
- Monitor the EC2 instance and Aurora cluster performance using AWS CloudWatch.
- Scale resources as needed for optimal performance and cost-efficiency.
- This setup ensures a secure and efficient connection between your web app and Aurora database.

Sample Output:







For Any References :

- https://learn.nextwork.org/projects/aws-databases-webapp?track=high