**aws**

Modern Cloud-Native Payment Systems

# Payment System Interface Modernization on AWS

# Payment System Interface Modernization on AWS: Modern Cloud-Native Payment Systems

# Table of Contents

# Payment System Interface Modernization on AWS: Modern Cloud-Native Payment Systems

Publication date: **April 6, 2022 ([Diagram history](#))**

This architecture shows you how to build a microservices-based payment system to handle scale and optimized performance with improved container-based deployment, and using API and event-based models for handling different channels in payment.

## Payment System Interface Modernization on AWS Diagram
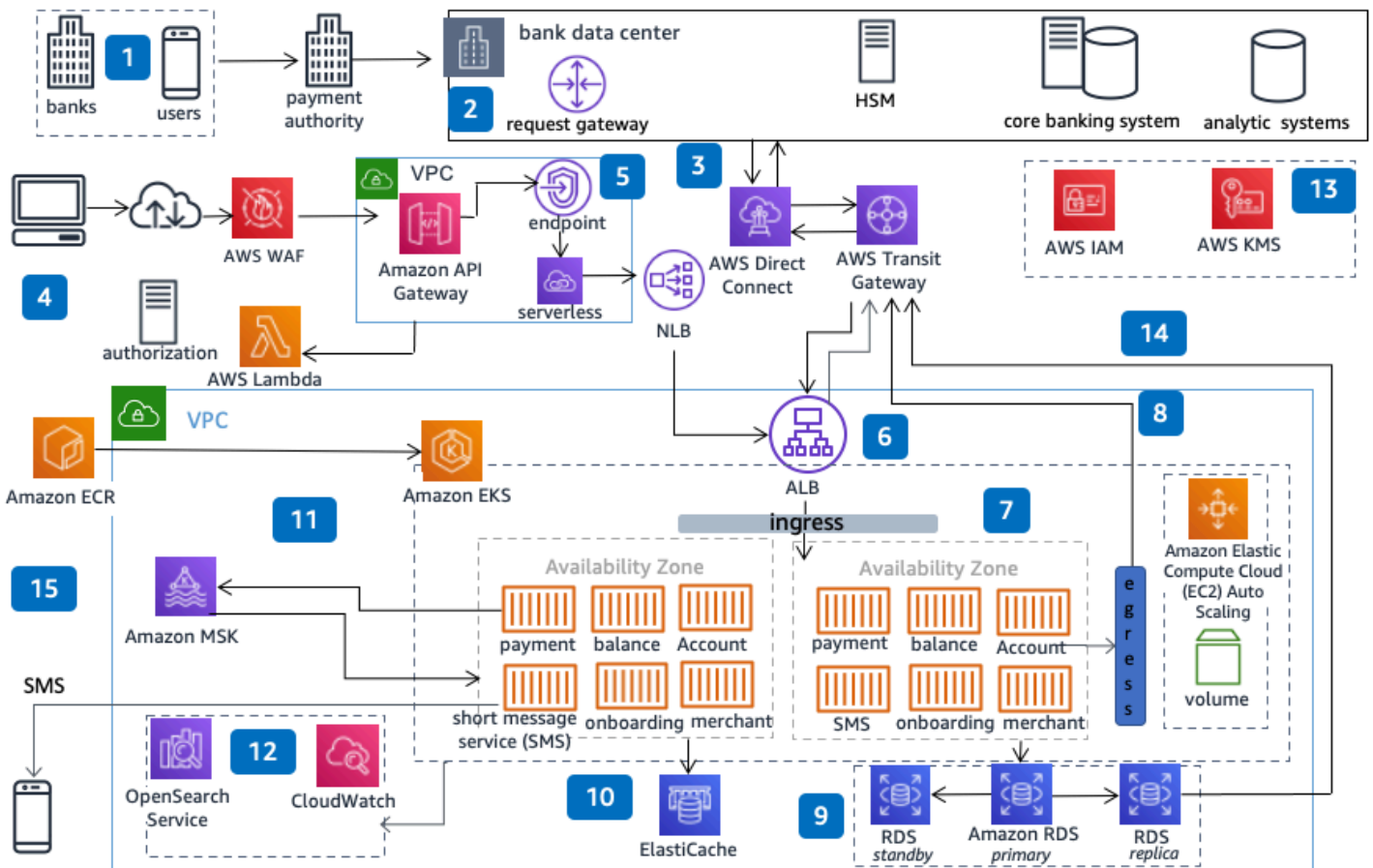


1. Users initiate transactions to payment interface system using mobile apps integrated with the payment system, or through bank-provided apps.

2. The request receiver accepts the request at the bank data center.

3. The request is routed to payment interface services in the AWS network from the bank using **AWS Direct Connect**.

4. A request from a merchant or other client arrives directly from the internet.

5. The request passes through **AWS WAF** and is received through the **Amazon API Gateway**, which routes requests to services using a VPC endpoint and **Network Load Balancer** (NLB).

6. In AWS, the request for payment interface services is received by the **Amazon Elastic Kubernetes Service** (Amazon EKS) cluster's **Application Load Balancer** (ALB) ingress controller.

7. Ingress rules redirect the request to a targeted microservice for purposes such as payment, balance, virtual payment address (VPA), merchant, account, and so on.

8. Payment interface services reach the bank hardware security model (HSM), core banking systems hosted inside the bank using **AWS Direct Connect**.

9. Payment interface services perform create, read, update, and delete (CRUD) operations in the **Amazon Relational Database Service** (Amazon RDS) unified payment interface (UPI) database.

10. Payment interface services maintain sessions in **Amazon ElastiCache for Redis**.

11. Payment interface services communicate with each other with an event-driven model using topics in **Amazon Managed Streaming for Apache Kafka** (Amazon MSK).

12. Payment interfaces microservices observability using **Amazon OpenSearch Service** and **Amazon CloudWatch**.

13. Use **AWS Identity and Access Management** (IAM), **AWS Key Management Service** (AWS KMS) , and **AWS Secrets Manager** to ensure the roles-based access, securing credentials, and data.

14. Payment interface near real-time transaction data is consumed by other systems in the bank DC using an **Amazon Relational Database Service** (Amazon RDS) replica over **AWS Direct Connect**.

15. Payment interface microservices container images are maintained inside **Amazon Elastic Container Registry** (Amazon ECR).

# Download editable diagram

To customize this reference architecture diagram based on your business needs, download the ZIP file which contains an editable PowerPoint.

# Create a free AWS account

Sign up now

Sign up for an AWS account. New accounts include 12 months of [AWS Free Tier](#) access, including the use of Amazon EC2, Amazon S3, and Amazon DynamoDB.

# Further reading

For additional information, refer to

- [AWS Architecture Icons](#)
- [AWS Architecture Center](#)
- [AWS Well-Architected](#)

# Diagram history

To be notified about updates to this reference architecture diagram, subscribe to the RSS feed.

| Change | Description | Date |
|---|---|---|
| [Initial publication](#) | Reference architecture diagram first published. | April 6, 2022 |

> ⓘ **Note**
>
> To subscribe to RSS updates, you must have an RSS plugin enabled for the browser you are using.