Implementing Message Brokering with Amazon MQ

MESSAGING WITH AMAZON MQ

Michael Heydt
FREELANCE CLOUD DEV, ARCHITECT AND TRAINER

@mikeheydt www.smac.io

Module Overview

Introduction of Amazon MQ

Benefits of Amazon MQ

Use cases and user roles for Amazon MQ

Amazon MQ vs Amazon SQS

Essential elements of Amazon MQ

Creating a message broker with the AWS console

Using the ActiveMQ UI

Creating a message producer and consumer with Java

Overview of Amazon MQ

Amazon MQ

A fully managed implementation of Apache ActiveMQ offered by AWS

Allows decoupling and scaling of applications and microservices

Allows you to send, store and receive messages between your software components

Support for multiple messaging protocols and patterns

Migration path to the cloud for users of ActiveMQ

Benefits of Amazon MQ

Security

Durability

Availability

Compatibility

Simplified Operation

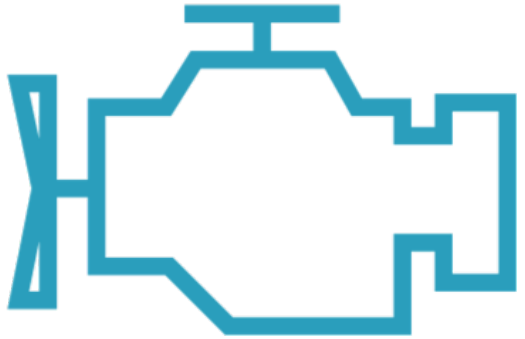
Low latency / high throughput

Getting Started with Amazon MQ

Use Cases for Amazon MQ

Migrating existing applications using ActiveMQ into AWS

Existing applications in AWS that will benefit from a managed messaging solution



Applications that need higher performance and enhanced messaging patterns beyond those in SQS

Roles of Users with Amazon MQ

Developer

Needs to know how to communicate with brokers using application code

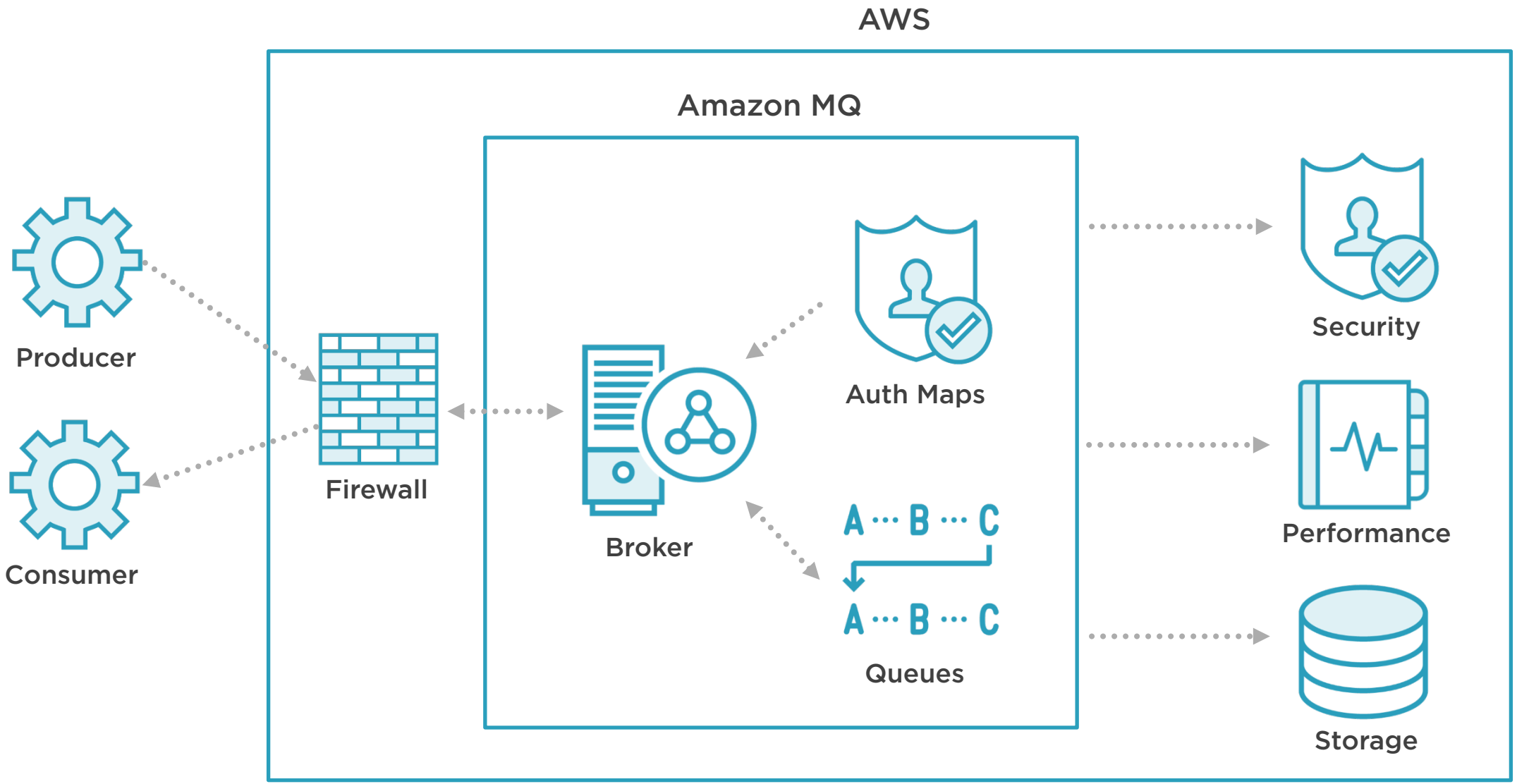
Administrator

Needs to know deploy, configure, secure and operate brokers in AWS

Critical Variances of Amazon MQ vs ActiveMQ

- Creating and configuring brokers
- Sending and receiving messages
- Securing access to the broker with authorization maps
- Managing availability of the broker
- Performance monitoring

Course Reference Model



Amazon MQ vs Amazon SQS

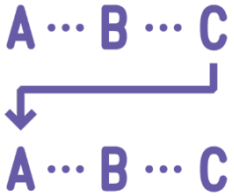
Amazon MQ vs Amazon SQS

Amazon MQ	Amazon SQS
Broker-based (star / hub and spoke)	Broker-less
Multiple broker patterns	Single architecture pattern
Compatible with many message brokers	Stand-alone service
API and Protocol support	REST only
High usage complexity	Low usage complexity
Support for complex messaging patterns	Queue
Manual vertical and horizontal scaling	Automatic / elastic global scaling

Essential Elements of Amazon MQ

Essential Elements of Amazon MQ

Brokers



Queues

Protocols

Storage

Monitoring

Security

Broker

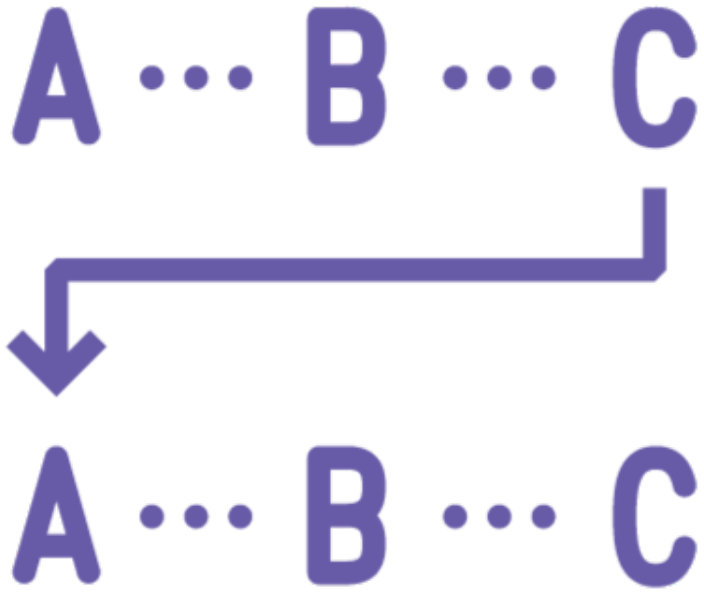
A broker coordinates indirect communication from publishers and consumers

Receives messages from clients, queues them, and then delivers them to consumers

Implemented as EC2 instances

The server(s) implementing the Amazon MQ broker are fully managed by AWS

Queue



A queue is a structure where messages sent from a publisher are stored until they are processed by a consumer.

Queues help us decouple application into smaller independent building blocks.

Storage

Brokers store messages in-memory (non-persistent) or in EFS or EBS (persistent)

EFS is the default persistent storage and supports high durability and replication across availability zones

EBS offers low latency and high throughput and is useful for optimizing clusters in a single availability zone

Protocols

Protocols define the rule and conventions of communicating with the Amazon MQ broker

Amazon MQ is a multi-protocol broker

All communications, regardless of protocol, are TLS encrypted

Producers and consumers can use different protocols

Amazon MQ Protocols

Protocol	Usage
OpenWire	Active MQ native protocol
AMQP	Advanced Message Queuing Protocol
MQTT	Message Queue Telemetry Transport
MQTT over WebSocket	MQTT tunneled through WebSocket
STOMP	Simple Text Oriented Message Protocol
STOMP over WebSocket	STOMP tunneled through WebSocket

Monitoring

Amazon MQ is fully integrated with CloudWatch for performance monitoring

API level monitoring can be enabled with CloudTrail

Security for Amazon MQ

- IAM / API access
- Encryption (in-flight and at-rest)
- Network / security group
- Connection
- Users and authorization maps
- Inter-broker

Demo

Creating a message broker

Demo

Using the ActiveMQ web UI

Demo

Creating a message producer and consumer with Java

Summary

- Reviewed benefits of Amazon MQ
- Identified several common use cases
- User roles and needed knowledge
- Course solution framework
- Differentiated Amazon MQ and SQS
- Amazon MQ components overview
- Created a single instance message broker
- Examined using the ActiveMQ UI
- Used Java to send and receive messages