Serverless means …

AWS Lambda

Serverless means …

No servers to provision or manage

Scales with usage

Pay for value

Availability and fault tolerance built in

Serverless means:

Serverless means:

Greater agility

Less overhead

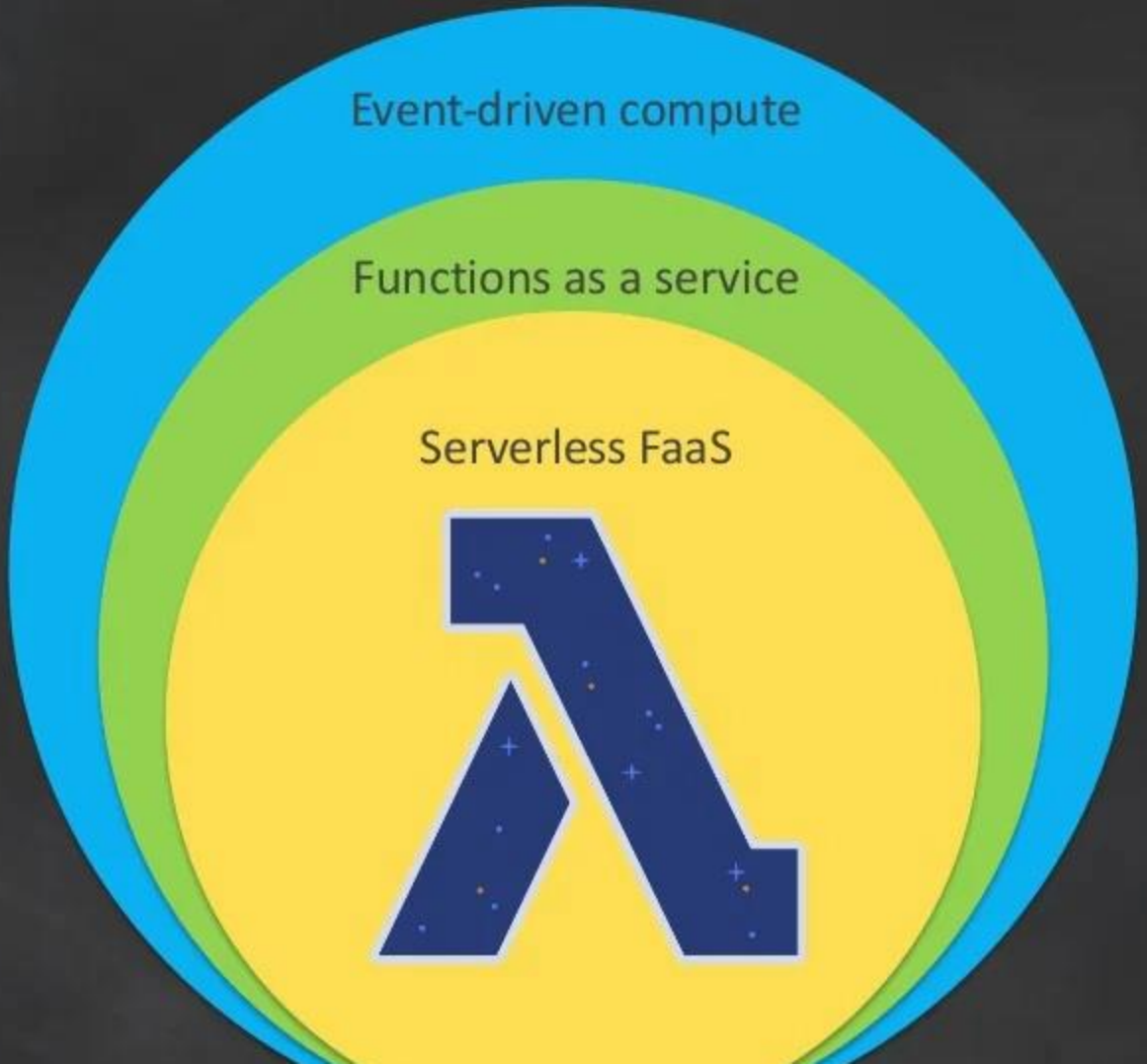
Better focus

Increased scale

More flexibility

Faster time to market

Today's focus:


Event-driven compute
Functions as a service
Serverless FaaS

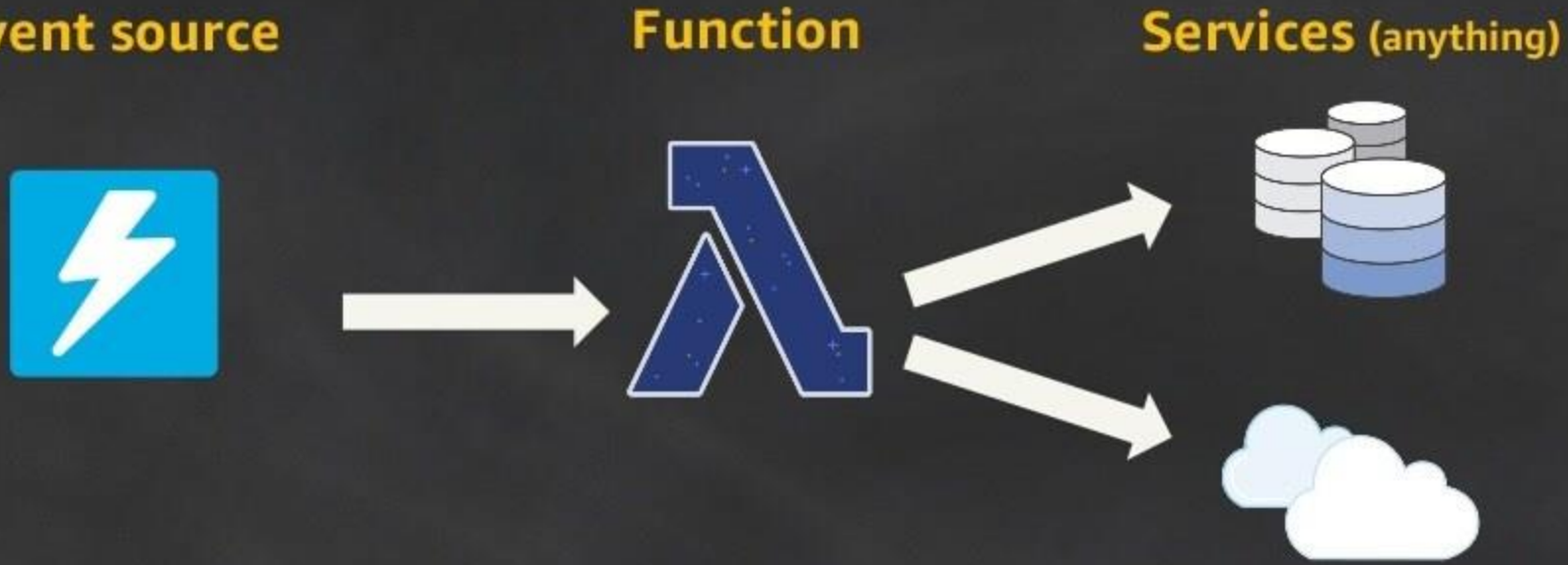
Lambda Handles

Load Balancing
Auto Scaling
Handling Failures
Security Isolation
OS Management
Managing Utilization
(and many other things) for you

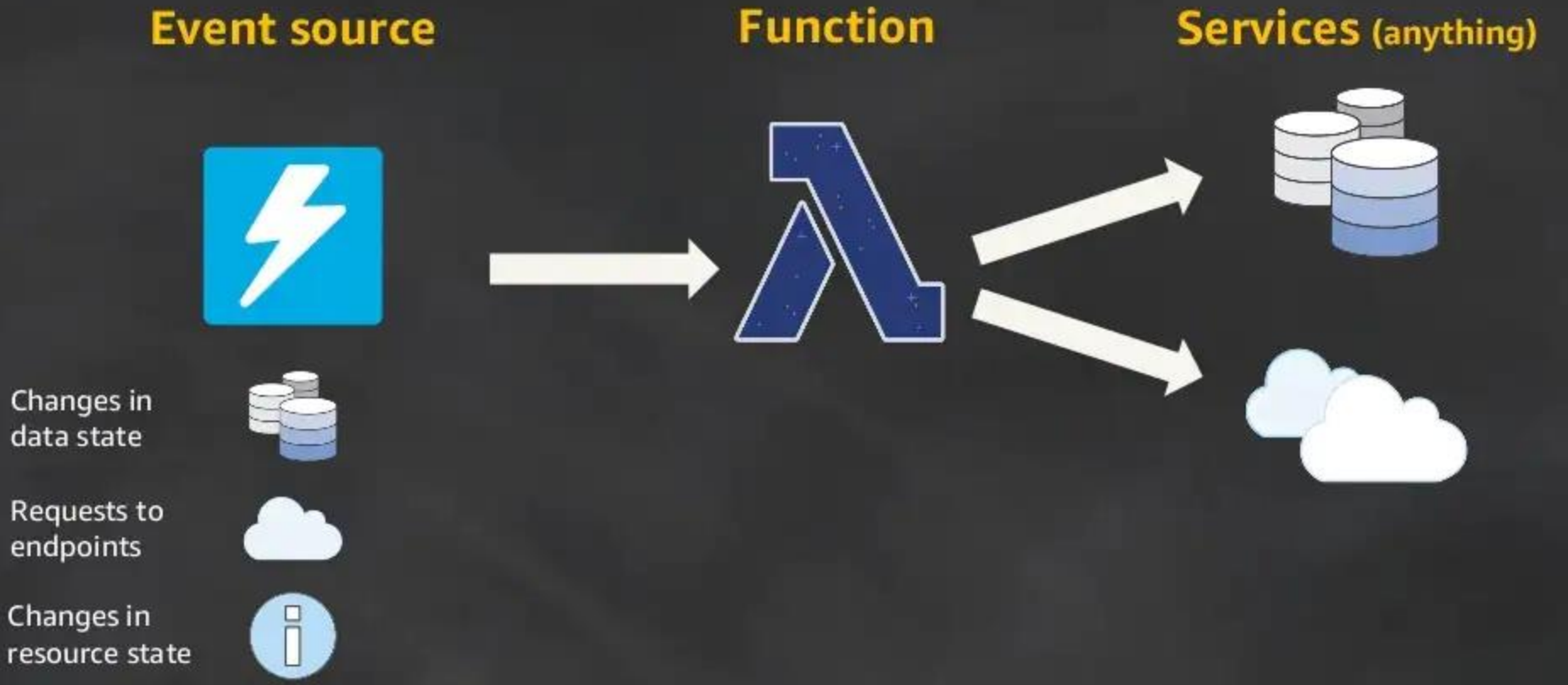
AWS Lambda release history

*As of October 2018, does not include region launches

Serverless applications



Serverless applications



Serverless applications

Event source

Changes in data state

Requests to endpoints

Changes in resource state

Function

NEW!

Node.js
Python
Java
C#
Go
Ruby
Runtime API

Services (anything)

Anatomy of a Lambda function

Handler() function

Function to be executed upon invocation

Event object

Data sent during Lambda function Invocation

Context object

Methods available to interact with runtime information (request I log group, more)

```
public String handleRequest(Book book, Context context) {
    saveBook(book);

    return book.getName() + " saved!";
}
```

Introducing: AWS Lambda runtime API and layers

NEW

Features that allow developers to share, discover, and deploy both libraries and languages as part of their serverless applications

Runtime API enables developers to use Lambda with any programming language.

Layers let functions easily share code. Upload layer once, reference within any function.

Layers promote separation of responsibilities, lets developers focus on writing business logic.

Combined, runtime API and layers allow developers to share any programming language or language version with others

Fine-grained pricing

Free Tier

1M requests and 400,000 GBs of compute.

Every month, every customer.

Buy compute time in 100ms increments

Low request charge

No hourly, daily, or monthly minimums

No per-device fees

Never pay for idle

Tweak your function's computer power

Lambda exposes only a memory control, with the **% of CPU core and network capacity** allocated to a function proportionally

Is your code CPU, Network or memory-bound? If so, it could be **cheape** to choose more memory.

Smart resource allocation

Match resource allocation (up to 3 GB!) to logic

Stats for Lambda function that calculates **1000 times** all prime numbers **<= 1000000**

128 MB	11.722965sec	$0.024628
256 MB	6.678945sec	$0.028035
512 MB	3.194954sec	$0.026830
1024 MB	1.465984sec	$0.024638

Green==Best **Red**==Worst

Smart resource allocation

Match resource allocation (up to 3 GB!) to logic

Stats for Lambda function that calculates **1000 times** all prime numbers **<= 1000000**

Lambda execution model

Lambda API

API provided by the Lambda service

Used by all other services that invoke Lambda across all models

Supports sync and async

Can pass any event payload structure you want

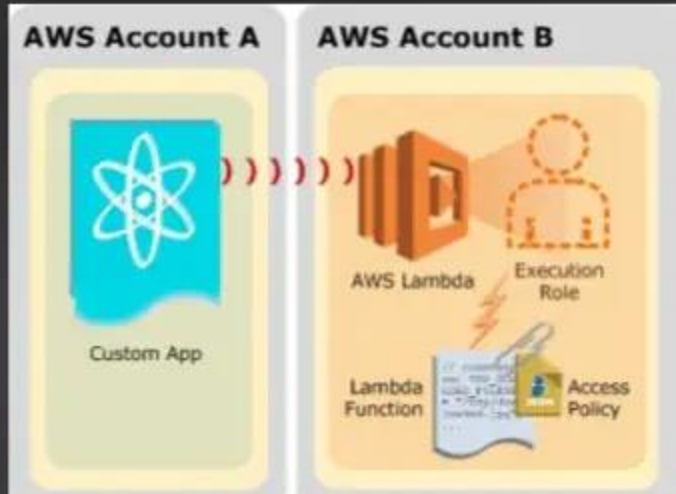
Client included in every SDK

Lambda permissions model

Fine grained security controls for both execution and invocation:

- **Execution policies:**
- Define what AWS resources/API calls can this function access via IAM
- Used in streaming invocations
- E.g. “Lambda function A can read from DynamoDB table users”
- **Function policies:**
- Used for sync and async invocations
- E.g. “Actions on bucket X can invoke Lambda function Z"
- Resource policies allow for cross account

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```



Common Lambda use cases

Web Applications

- Static websites
- Complex web apps
- Packages for Flask and Express

Backends

- Apps & services
- Mobile
- IoT

Data Processing

- Real time
- MapReduce
- Batch

Chatbots

- Powering chatbot logic

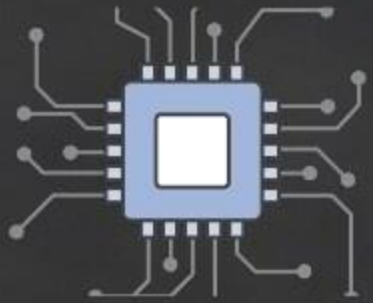
Amazon Alexa

- Powering voice-enabled apps
- Alexa Skills Kit

IT Automatio

- Policy engi
- Extending AWS servic
- Infrastructu manageme

Amazon API Gateway



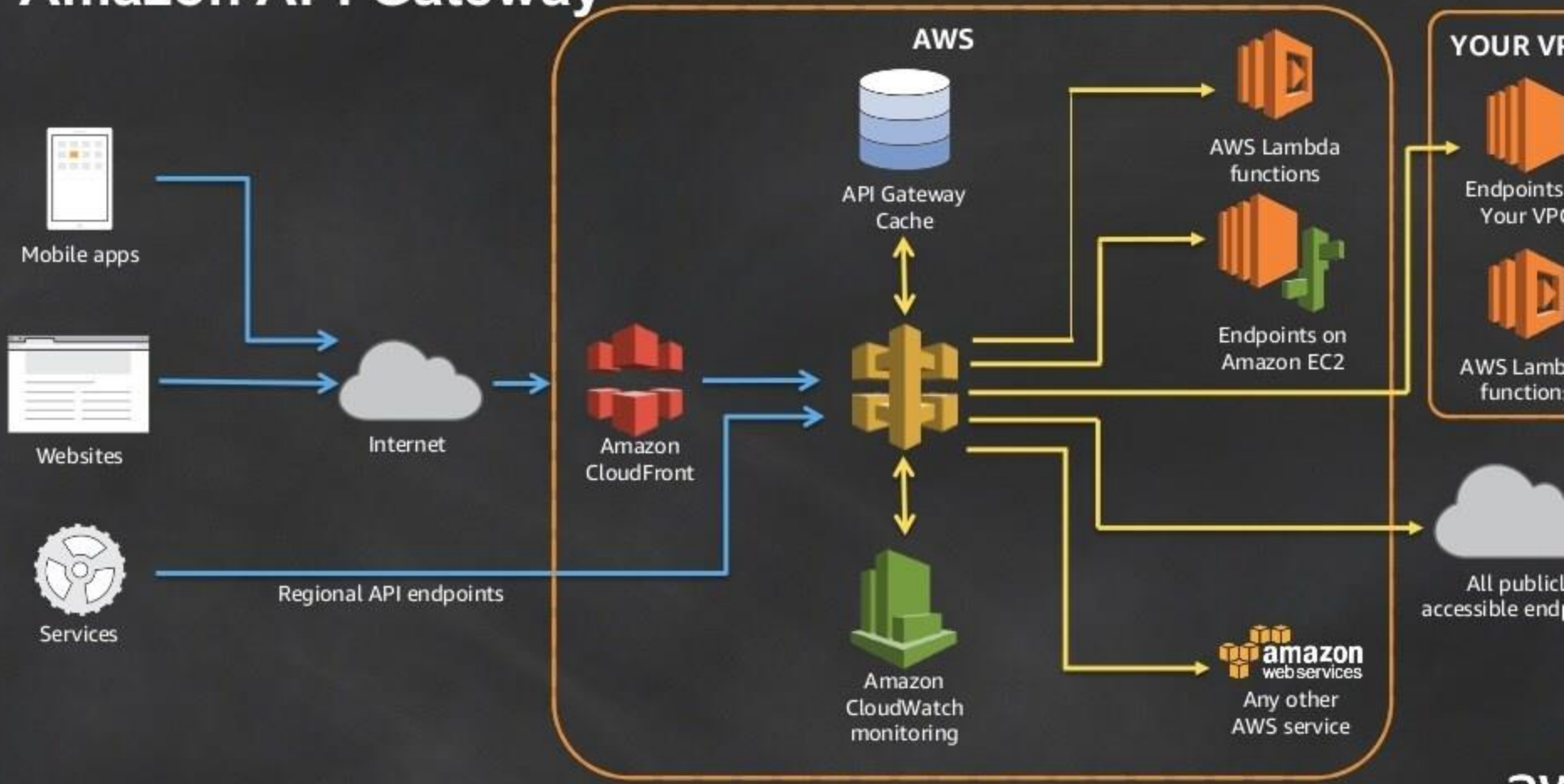
Create a unified API frontend for multiple micro-services

DDoS protection and throttling for your backend

Authenticate and authorize requests to a backend

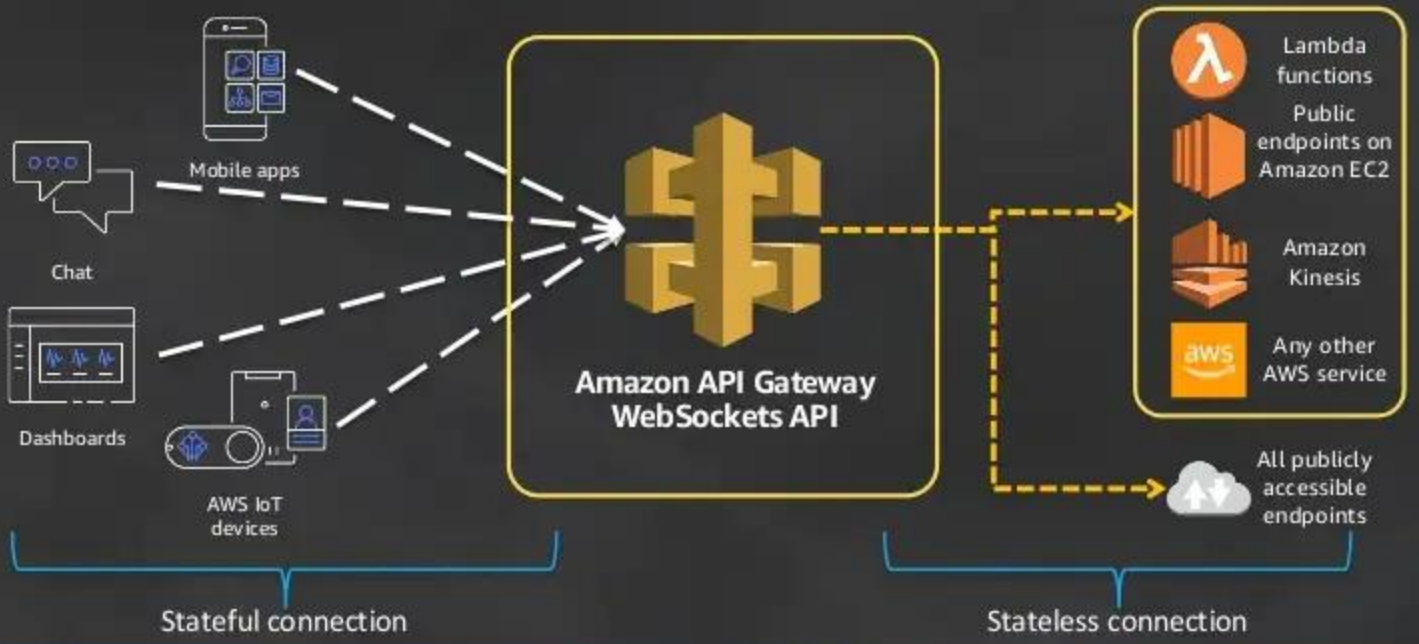
Throttle, meter, and monetize API usage by third-party developers

Amazon API Gateway



Introducing: API Gateway WebSockets

Coming soon!



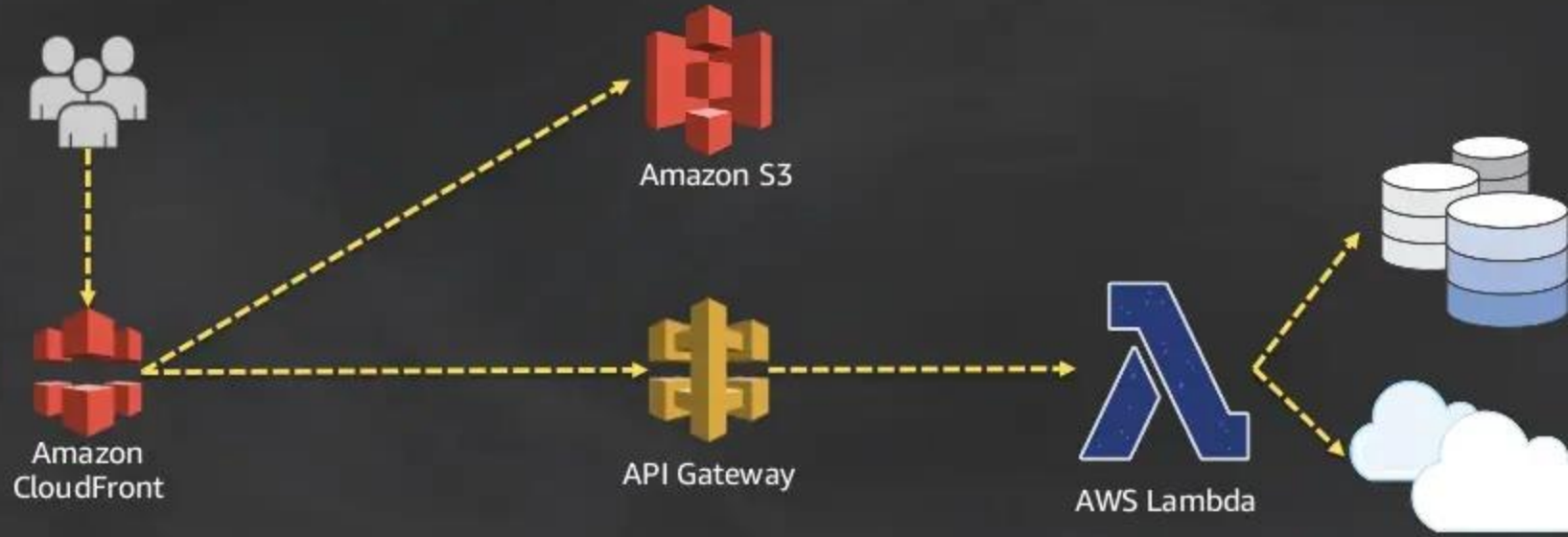
Build real-time two-way communication applications
chat, alerts and notifications, and streaming dashboards

Fully managed APIs
to handle connections and messages transfer between users and backend services

Invoke AWS services
like Lambda, Kinesis, or any HTTP endpoint based on message content

Pay for what you u
based on connection m
and messages transfe

Serverless web application with API Gateway

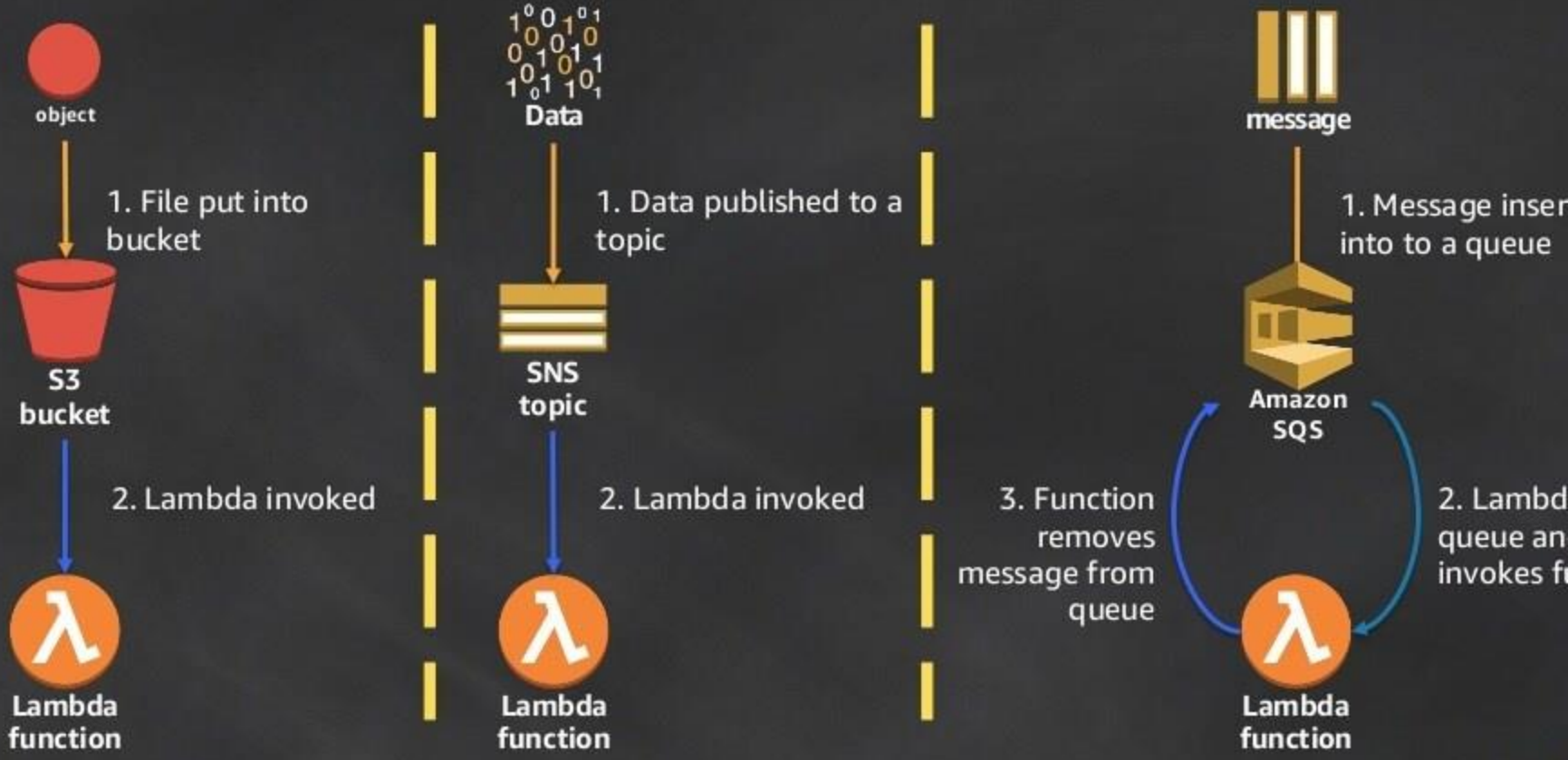


Amazon Simple Storage Service (Amazon S3) stores all of your static content: CSS, JS, images, and more. You would typically front this with a CDN such as CloudFront.

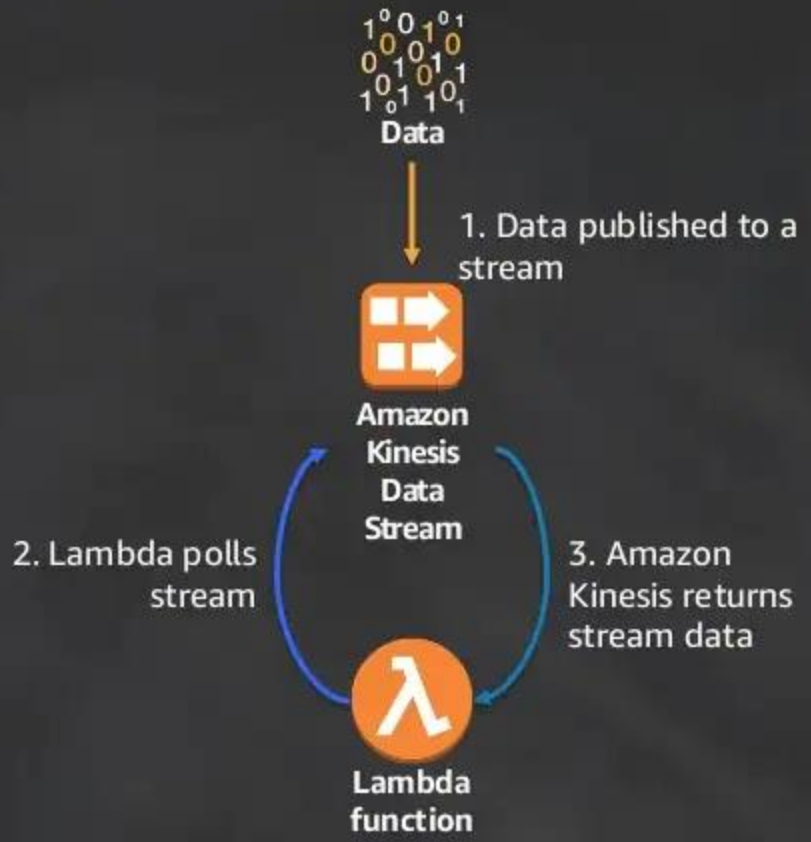
API Gateway handles all your application routing. It can handle authentication and authorization, throttling, DDOS protection, and more.

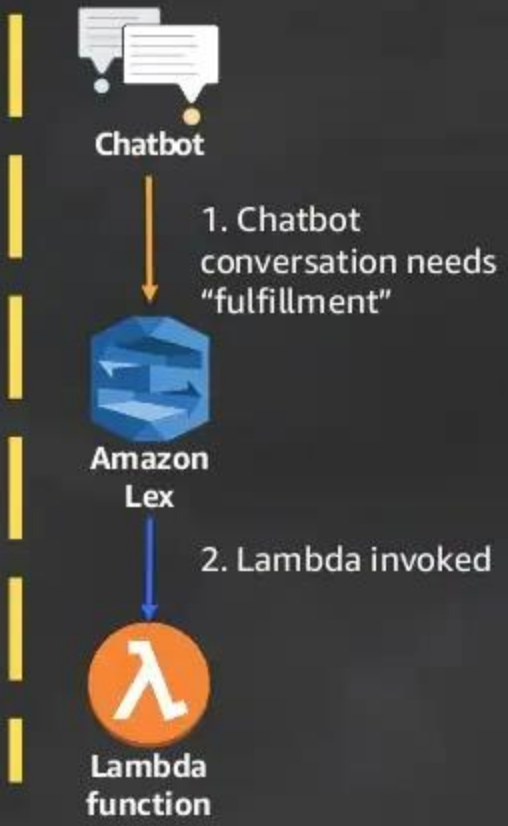
Lambda runs all the logic behind your website and interfaces with databases, other backend services, or anything else your site needs.

Serverless architectures

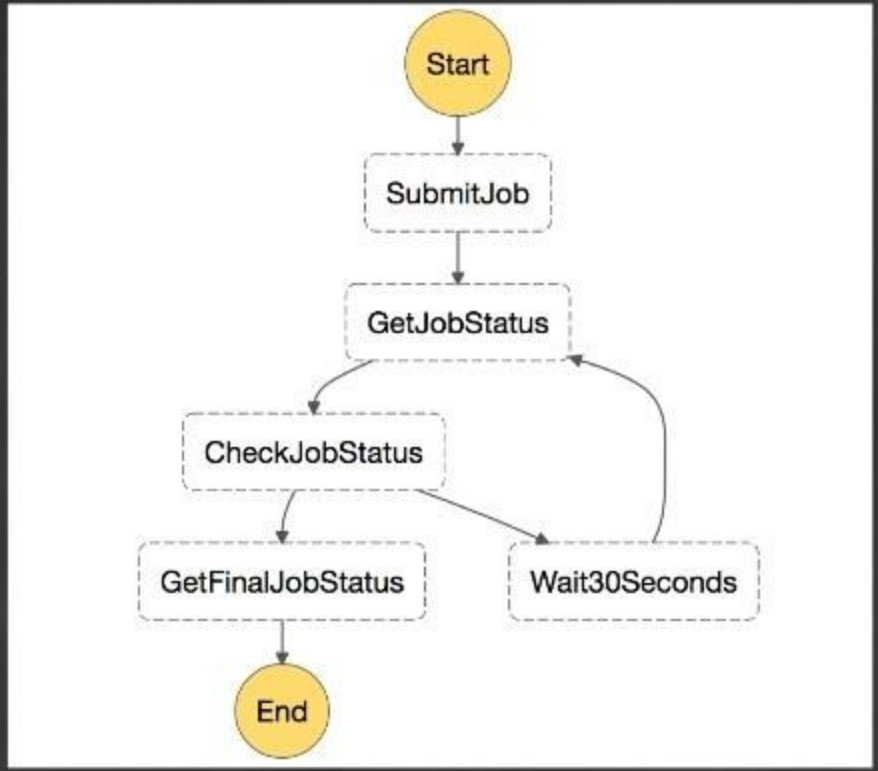


Serverless architectures





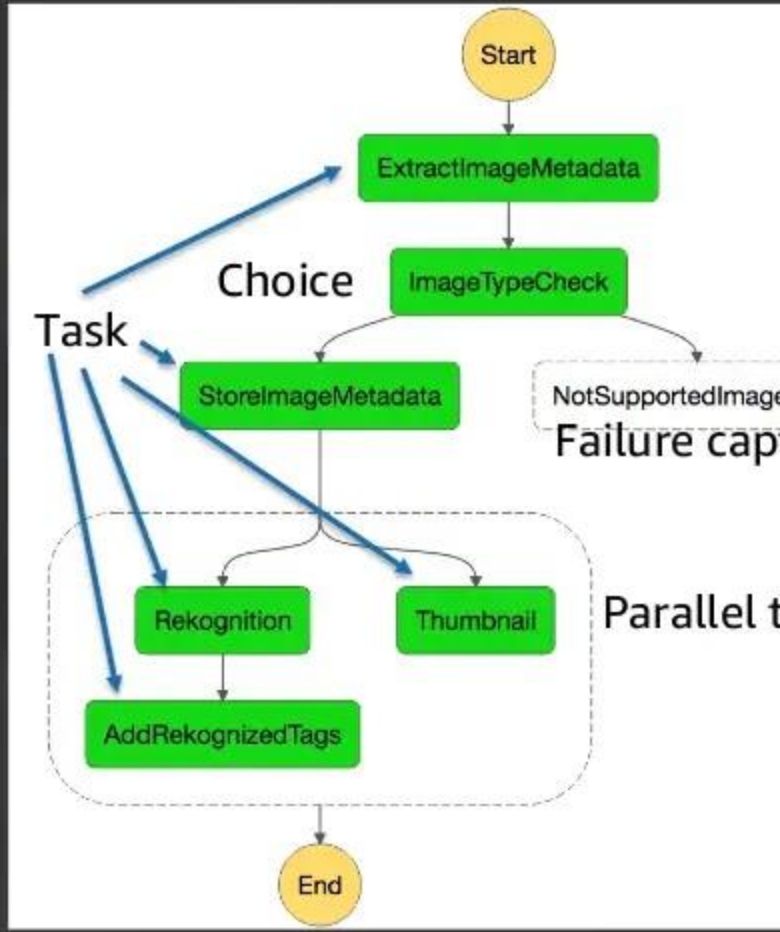
Keep orchestration out of code



AWS Step Functions

"Serverless" workflow management with zero administration

- Makes it easy to coordinate the components of distributed applications and microservices using visual workflows
- Automatically triggers and tracks each step and retries when there are errors, so your application executes in order and as expected
- Logs the state of each step, so when things do go wrong, you can diagnose and debug problems quickly



Build PCI and HIPAA compliant serverless applications!

Serverless platform services that can be used in both:


AWS Lambda


Amazon S3


Amazon CloudFront


Amazon DynamoDB


Amazon Kinesis Data Streams


Amazon Cognito


Amazon API Gateway


Amazon SNS

Metrics and logging are a universal right!

- **CloudWatch Metrics:**
- 7 Built in metrics for Lambda
  - Invocation Count, Invocation duration, Invocation errors, Throttled Invocation, Iterator Age, DLQ Errors, Concurrency
  - Can call "put-metric-data" from your function code for custom metrics
- 7 Built in metrics for API-Gateway
  - API Calls Count, Latency, 4XXs, 5XXs, Integration Latency, Cache Hit Count, Cache Miss Count
  - Error and Cache metrics support *averages and percentiles*

Metrics and logging are a universal right!

- **CloudWatch Logs:**
- API Gateway Logging
  - 2 Levels of logging, ERROR and INFO
  - Optionally log method request/body content
  - Set globally in stage, or override per method
- Lambda Logging
  - Logging directly from your code with your language's equivalent of console.log()
  - Basic request information included
- Log Pivots
  - Build metrics based on log filters
  - Jump to logs that generated metrics
- Export logs to AWS ElastiCache or S3

Messaging Systems in AWS

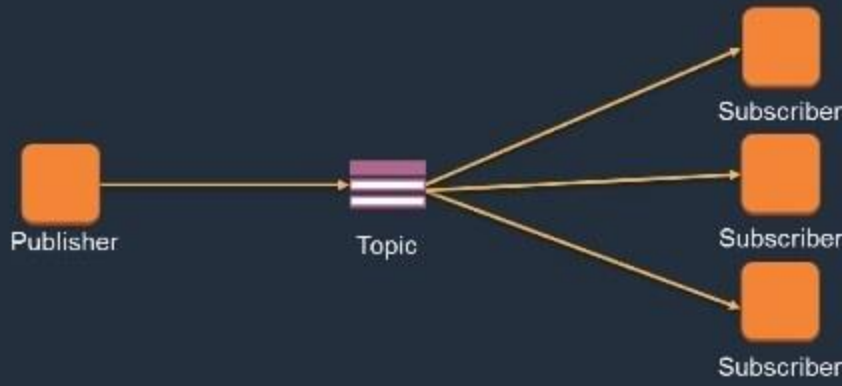
Messaging Enables Decoupling

Message Queueing

- Asynchronous
- Point-to-point

Publish-subscribe (pub-sub)

- Broadcast
- Point-to-multipoint



AWS Messaging Services

Amazon SQS

- Simple, fast, reliable, managed message queues
- Build highly scalable decoupled applications

Amazon SNS

- Simple, fast, reliable, managed pub/sub
- Highly scalable push messaging to users or applications

Amazon Kinesis

- Highly scalable streaming data analysis
- Ingest and analyze real-time data

AWS IoT

- Managed service to connect devices to the cloud
- Supports billions of subscribers

Amazon Pinpoint

- Mobile notification and engagement platform
- Targeted campaigns over email, SMS, and mobile

Amazon MQ

- Managed message broker service for Apache ActiveMQ

Amazon SQS: key features

- Persistent message queue with high durability and availability

- Messages are stored across multiple AZs

- Messages retained until deleted— up to 14 days

- Nearly unlimited throughput

Amazon SQS: key features

- Amazon CloudWatch metrics and alerts

- Message payloads up to 256 KB (5 TB using Amazon S3
- Message batching to increase throughput and reduce c

- Secure: uses AWS Identity & Access Management (IAM) and HTTPS/TLS

Two SQS Queue Types

Standard Queues

- At least once delivery
- Best-effort ordering
- Nearly unlimited transaction rate

FIFO Queues

- First-in First-out delivery preserving message
- Exactly once processing
- 300 transactions per second
- Messages groups for

SQS Extended Client Library

- For sending message payloads larger than 256KB
- Message payloads are stored in an S3 bucket
- SQS is used to transmit a reference to the S3 payload
- Abstracted from developer with Java client-side library

Amazon SNS: key features

- Proven reliability with messages are stored across multiple AZ

- Flexible message delivery over multiple transport protocols
- Nearly unlimited throughput

Amazon SNS: key features

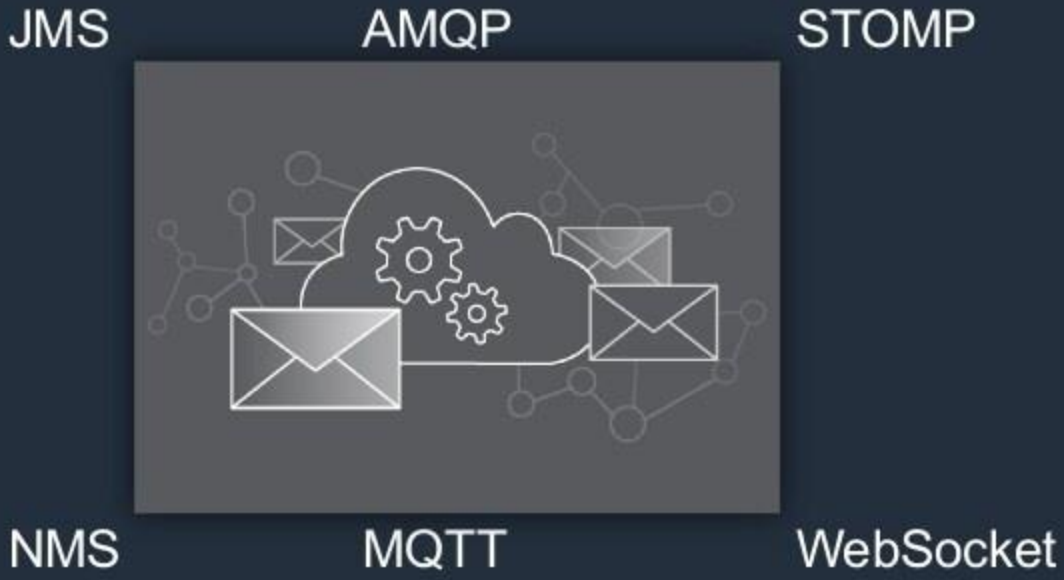
- Instantaneous or delayed, push-based delivery

- Simple APIs and easy integration

- Amazon CloudWatch metrics and alerts

- Message payloads up to 256 KB

Amazon MQ: Key Features



Compatible with industry-standard APIs and protocols.

Managed message broker service for Apache ActiveMQ.

Amazon MQ: Key Features

- Transient & persistent messaging
- Local & distributed transactions (XA)
- Queues & topics (with FIFO)
- Composite & virtual destinations
- Message filtering
- Request/reply
- Scheduled messages
- Unlimited message size
- Unlimited message retention

FAQ – when should I use Amazon MQ vs SQS/SNS?

SQS & SNS

- For cloud-native applications
- Simple
- Unlimited throughput
- Fully managed

Amazon MQ Service

- For application migration
- API-compatible
- Feature-rich
- Limited scale
- Managed infrastructure