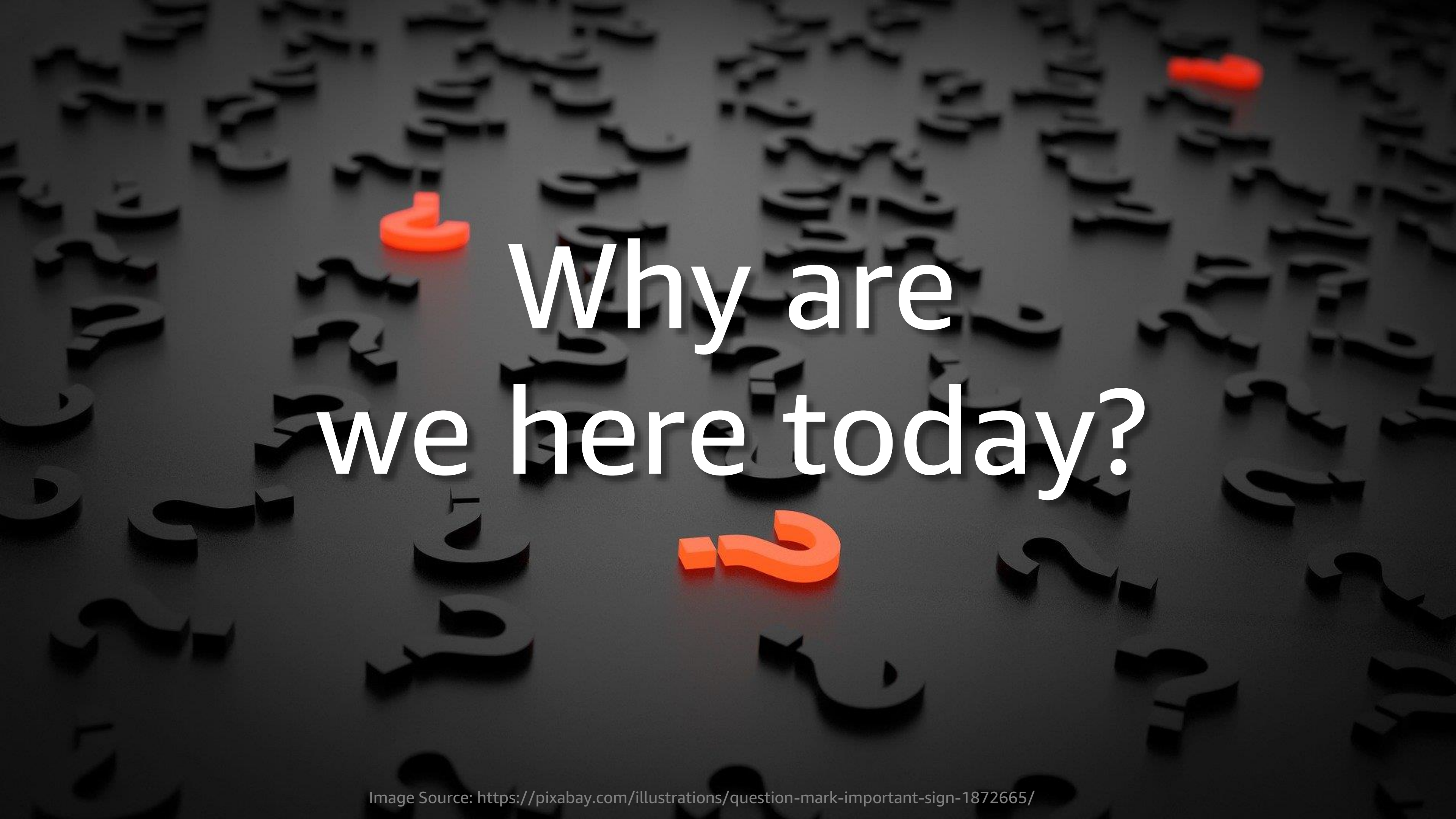
Advanced serverless messaging patterns for your applications

Talia Nassi
Senior Developer Advocate, AWS Serverless
@talia_nassi

Hi 👋

I build serverless things.
Then I talk and write about them.
Test Engineer > Developer Advocate

@talia_nassi



Why are we here today?

Image Source: https://pixabay.com/illustrations/question-mark-important-sign-1872665/

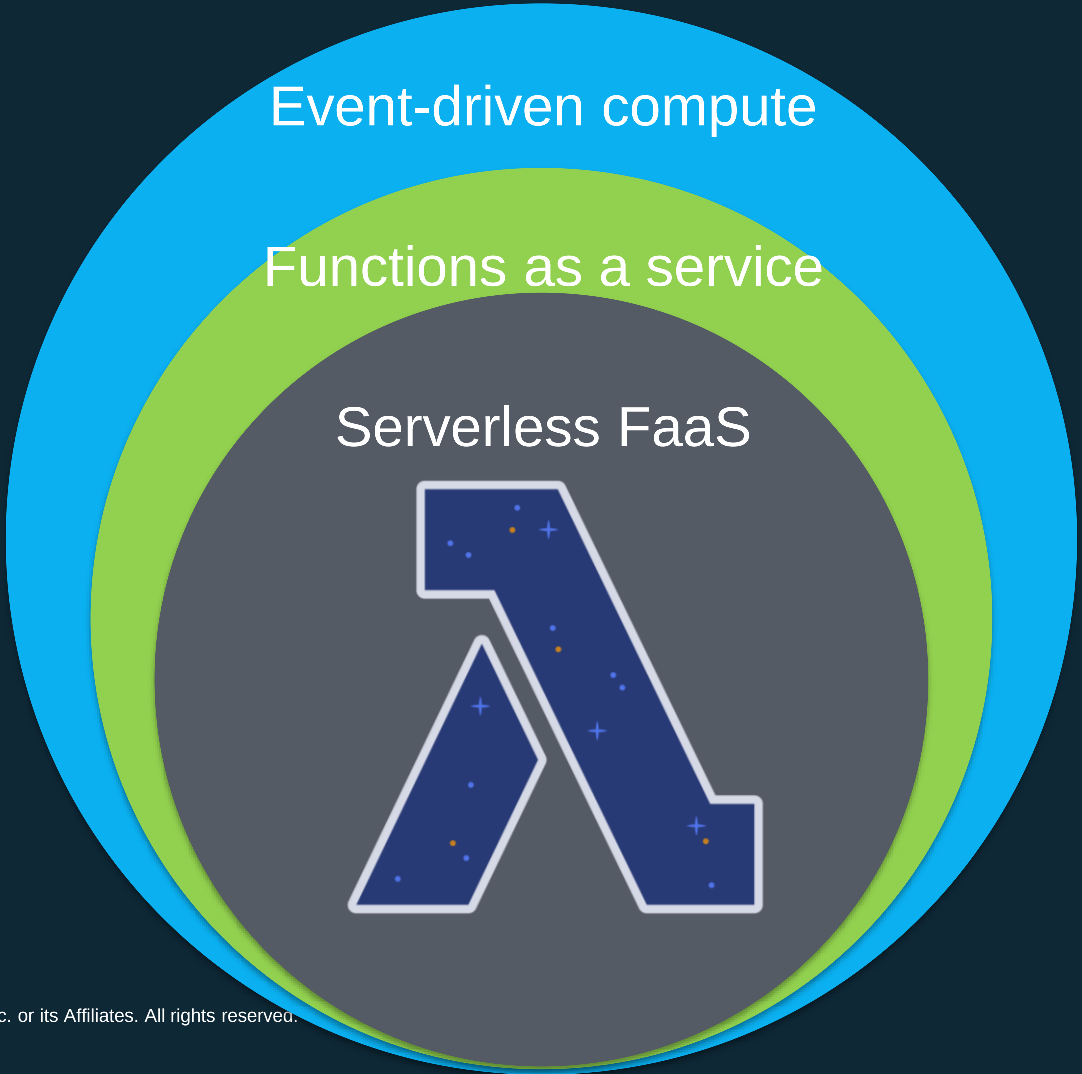
What is serverless?

No infrastructure provisioning, no management

Automatic scaling

Pay for value

Highly available and secure

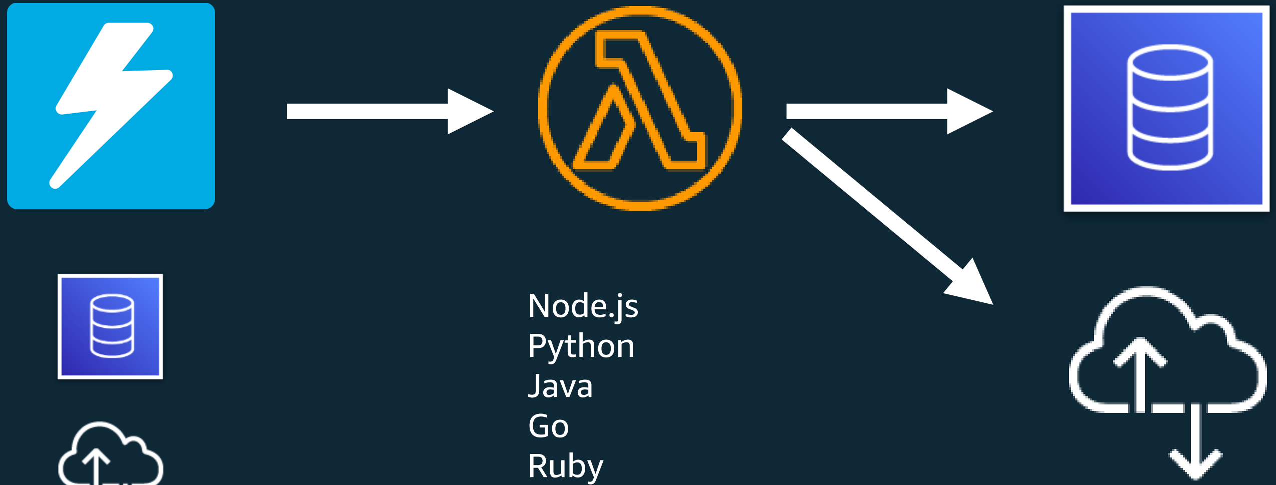

Event-driven compute
Functions as a service
Serverless FaaS

Serverless Applications

Event source

Function

Services (anything)



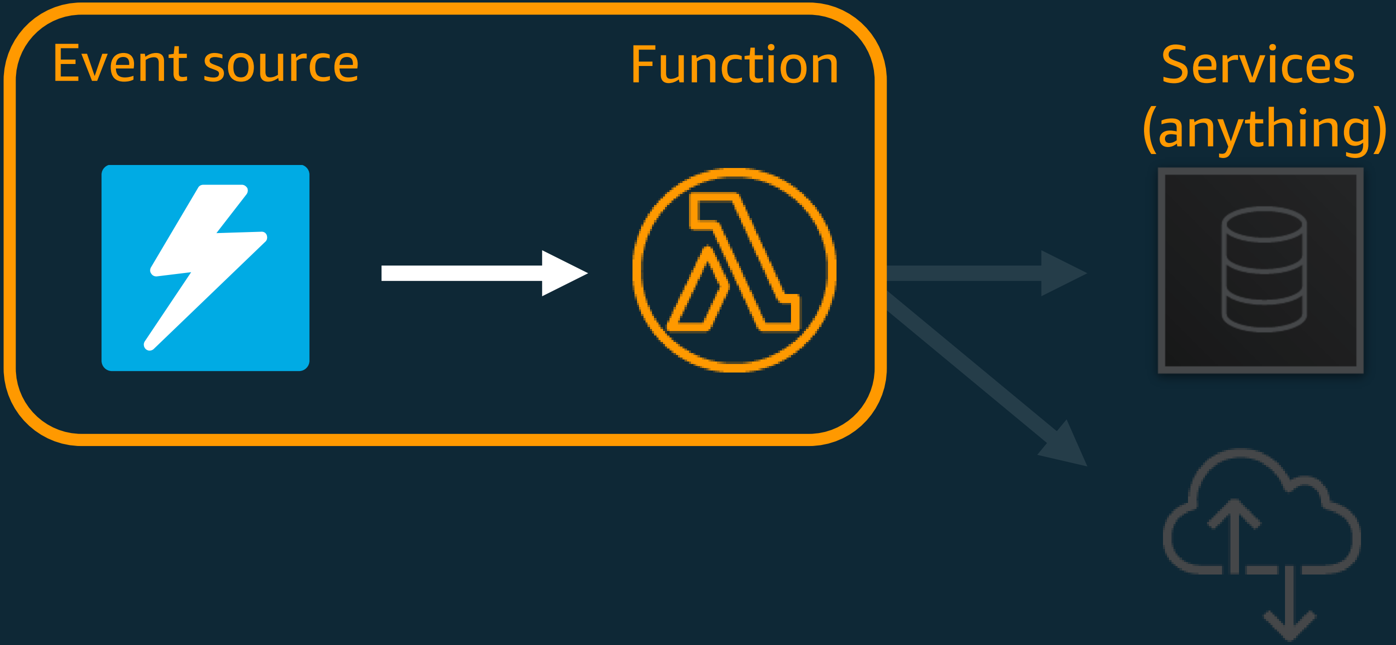
Changes in data state

Requests to endpoints

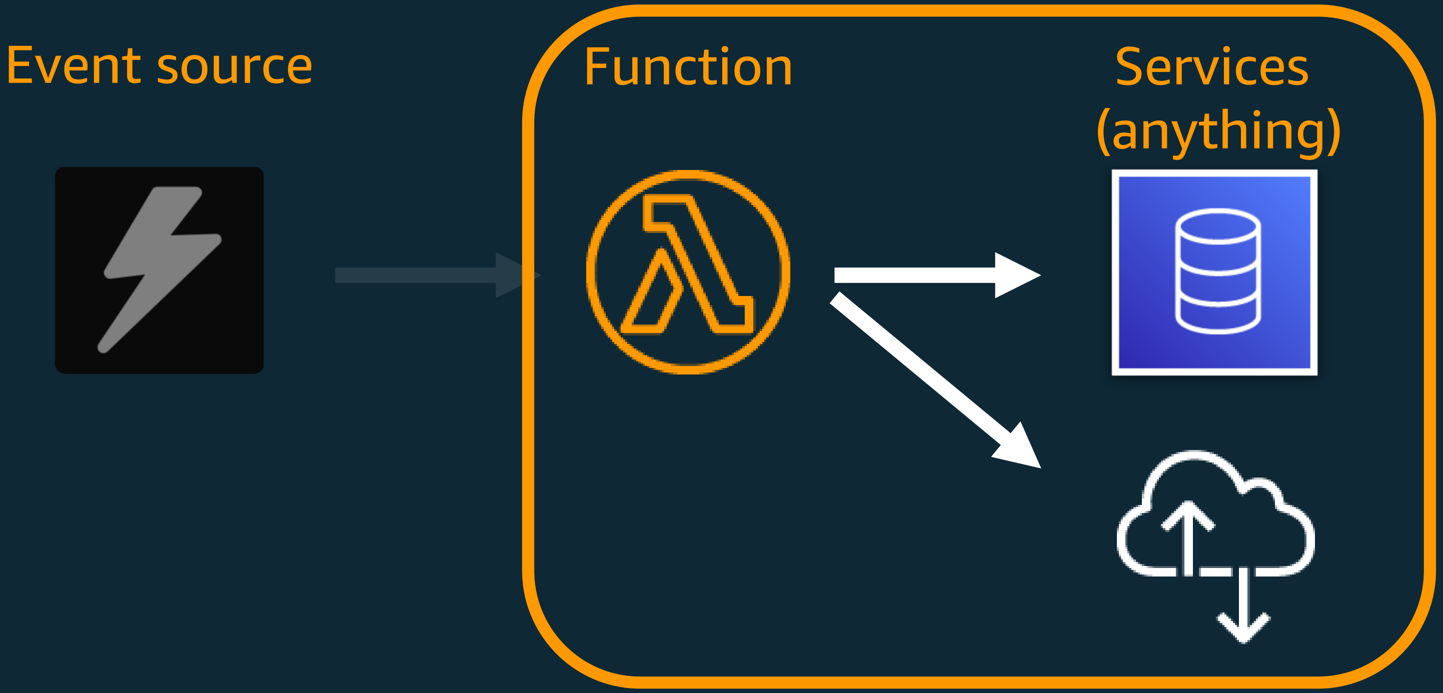
Changes in Resource state

Node.js
Python
Java
Go
Ruby
.Net (C#/Powershell)
Custom Runtime API

Serverless Applications

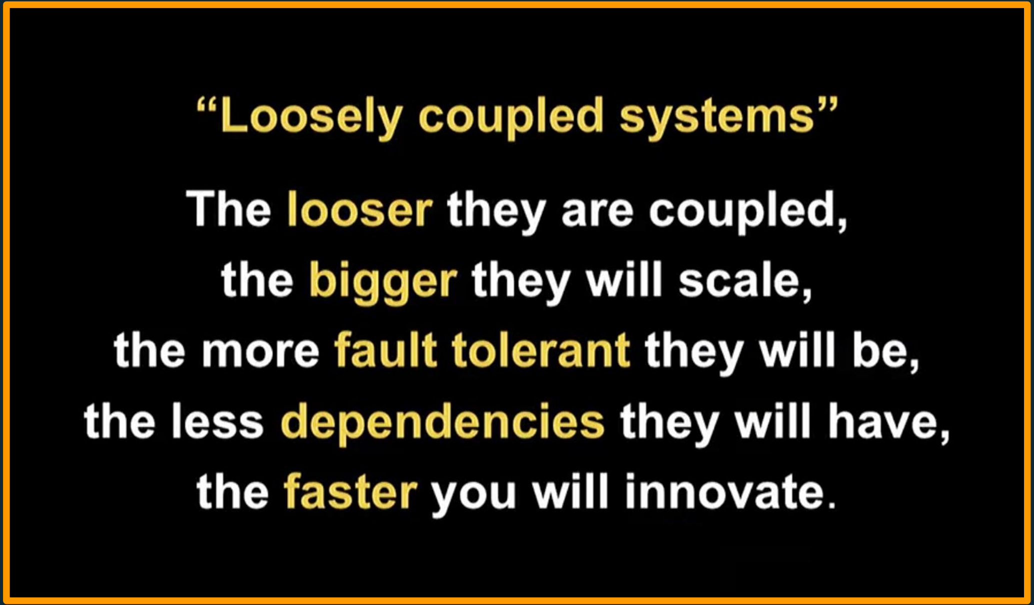


Serverless Applications



Messaging services with AWS Lambda

What is messaging?


"Loosely coupled systems"

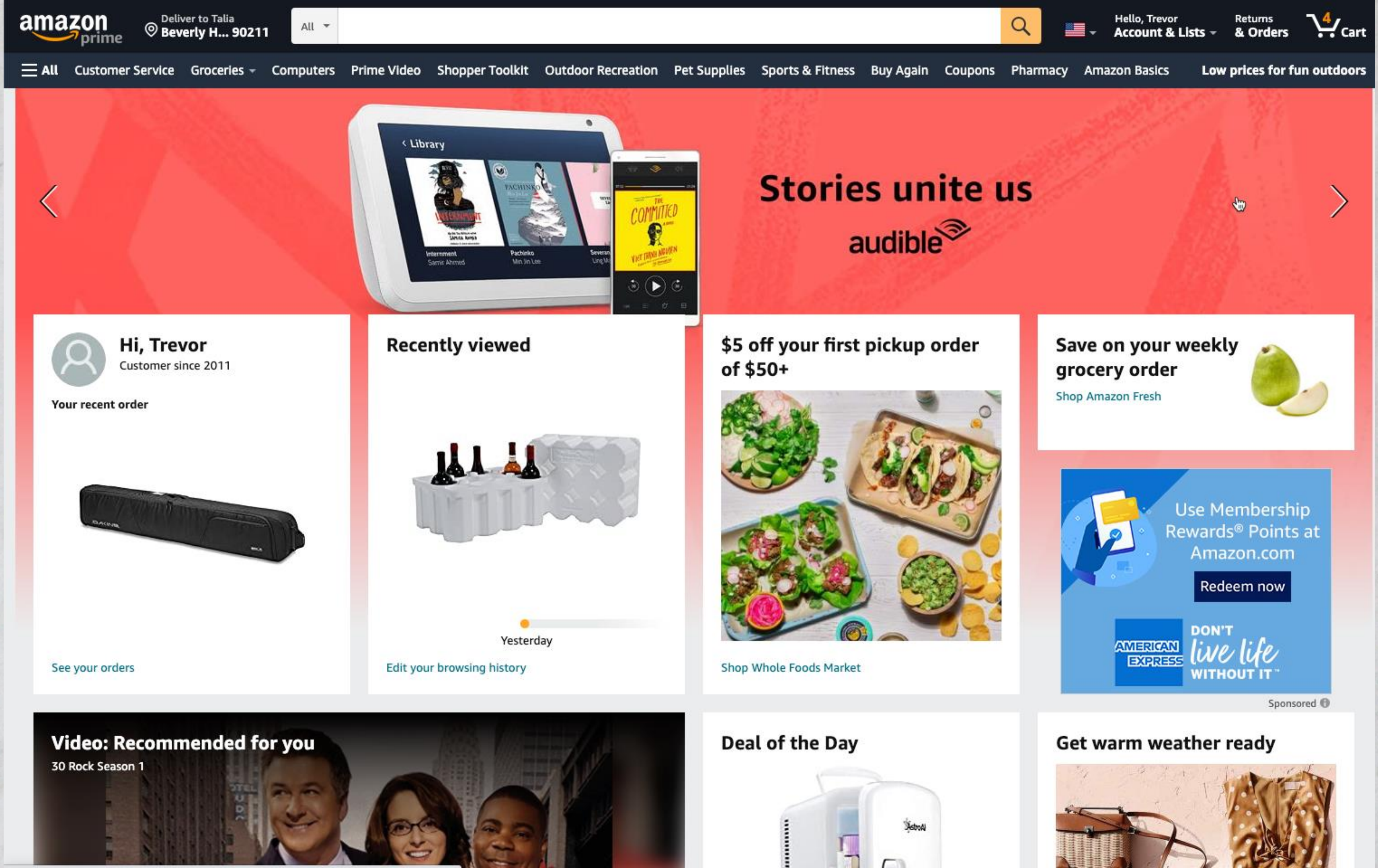
The looser they are coupled,
the bigger they will scale,
the more fault tolerant they will be,
the less dependencies they will have,
the faster you will innovate.

What does messaging provide?

Resilience

Availability

Scalability



Hi, Trevor
Customer since 2011
Your recent order
See your orders

Recently viewed
Yesterday
Edit your browsing history

$5 off your first pickup order of $50+
Shop Whole Foods Market

Save on your weekly grocery order
Shop Amazon Fresh

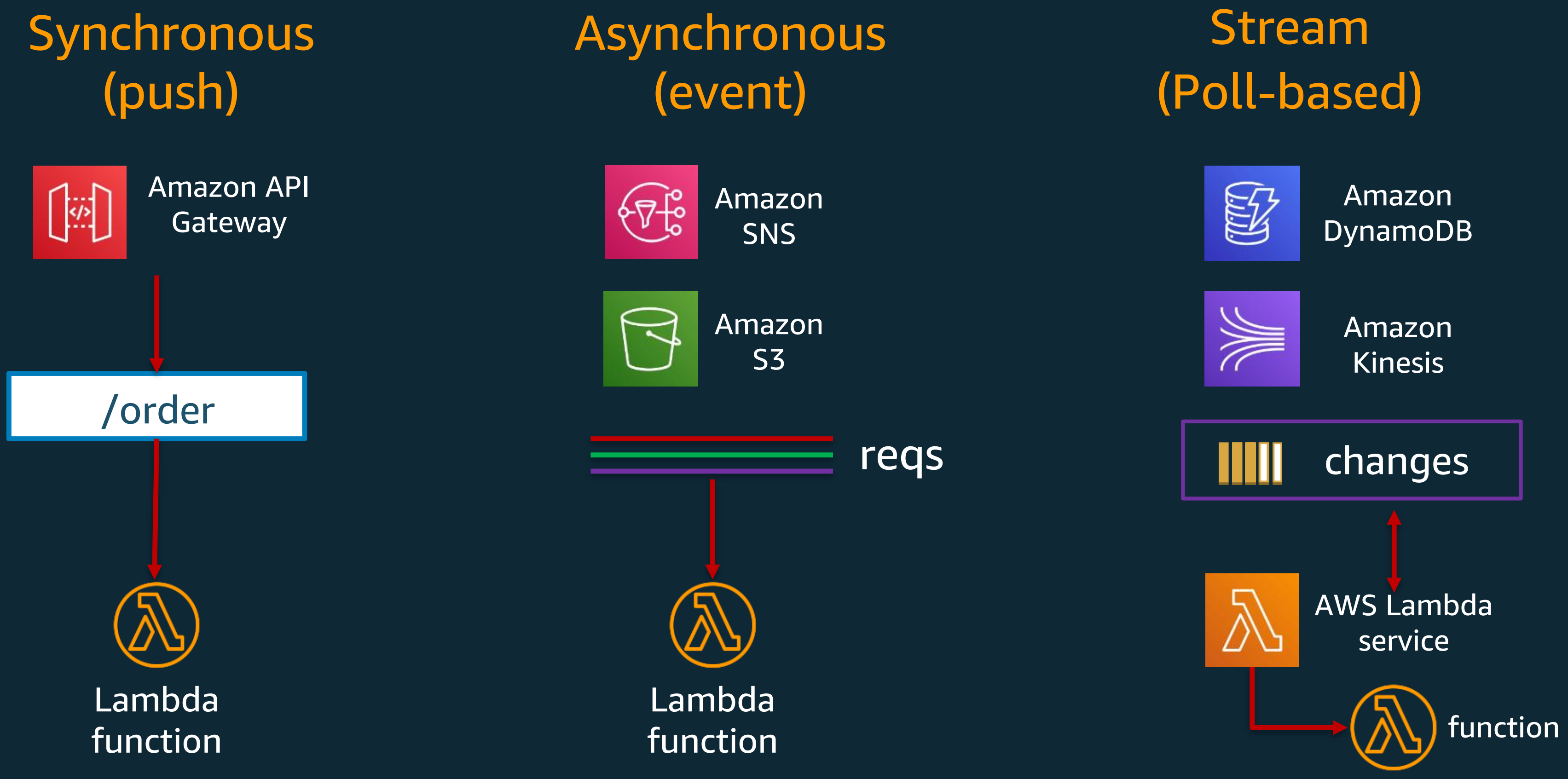
Use Membership Rewards® Points at Amazon.com
Redeem now
AMERICAN EXPRESS
DON'T live life WITHOUT IT™
Sponsored

Video: Recommended for you
30 Rock Season 1

Deal of the Day

Get warm weather ready

Lambda execution model



Messaging Services

Amazon SQS

Queues
Durable and scalable
Fully managed
Comprehensive security

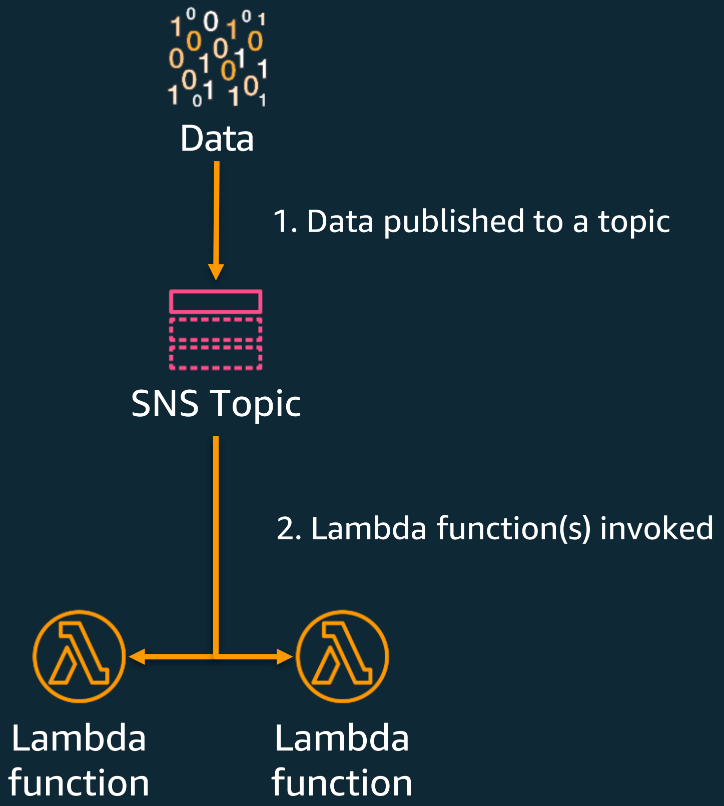
Amazon SNS

Pub/Sub
Performance at scale
Fully managed
Enterprise-ready

Amazon EventBridge

Event Bus
Serverless event bus for AWS services, your own applications, and SaaS providers

Amazon SNS + Lambda



Simple, flexible, secure, fully managed publish/subscribe messaging and mobile push notification service for high throughput, highly reliable many-to-many messaging.

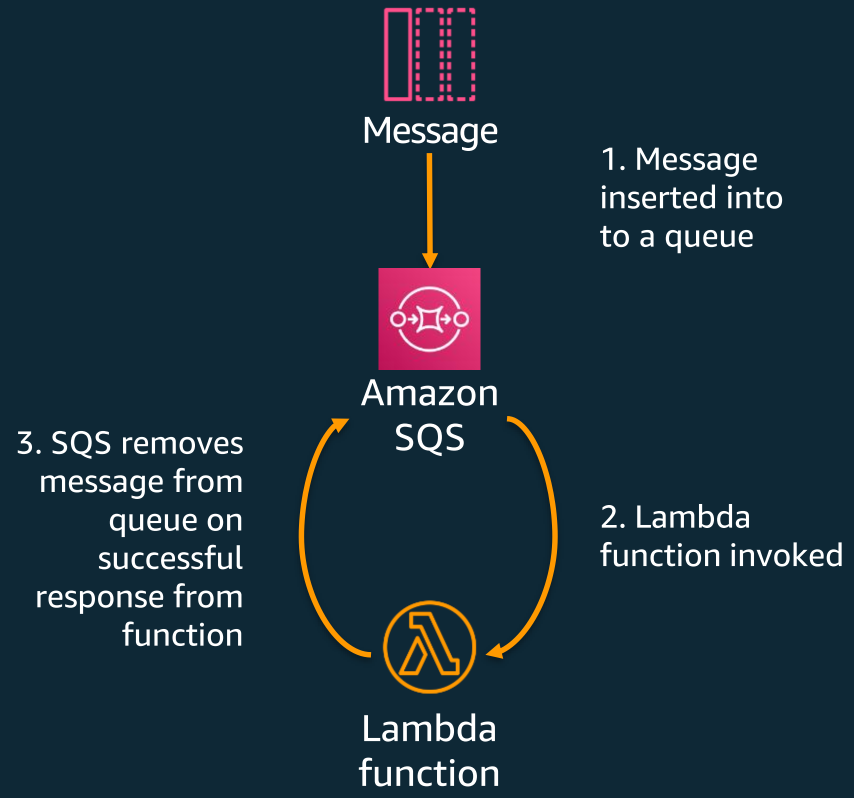
Messages are published to a Topic

Topics can have multiple subscribers (fanout)

Messages can be filtered and only sent to certain subscribers

Asynchronous

Amazon SQS + Lambda


Message
1. Message inserted into to a queue
Amazon SQS
3. SQS removes message from queue on successful response from function
2. Lambda function invoked
Lambda function

Simple, flexible, fully managed message queuing service to send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available

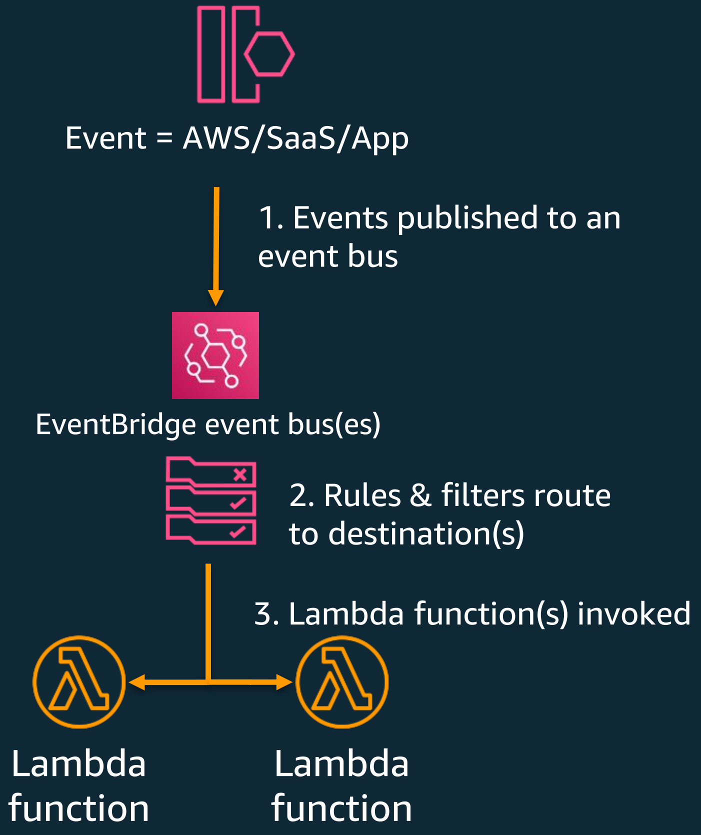
Processed in batches

Standard queue = at least once delivery
FIFO queue = ordered and exactly once

Visibility timeout allows for handling of failures during processing

Asynchronous

Amazon EventBridge + Lambda


Event = AWS/SaaS/App
1. Events published to an event bus
EventBridge event bus(es)
2. Rules & filters route to destination(s)
3. Lambda function(s) invoked
Lambda function
Lambda function

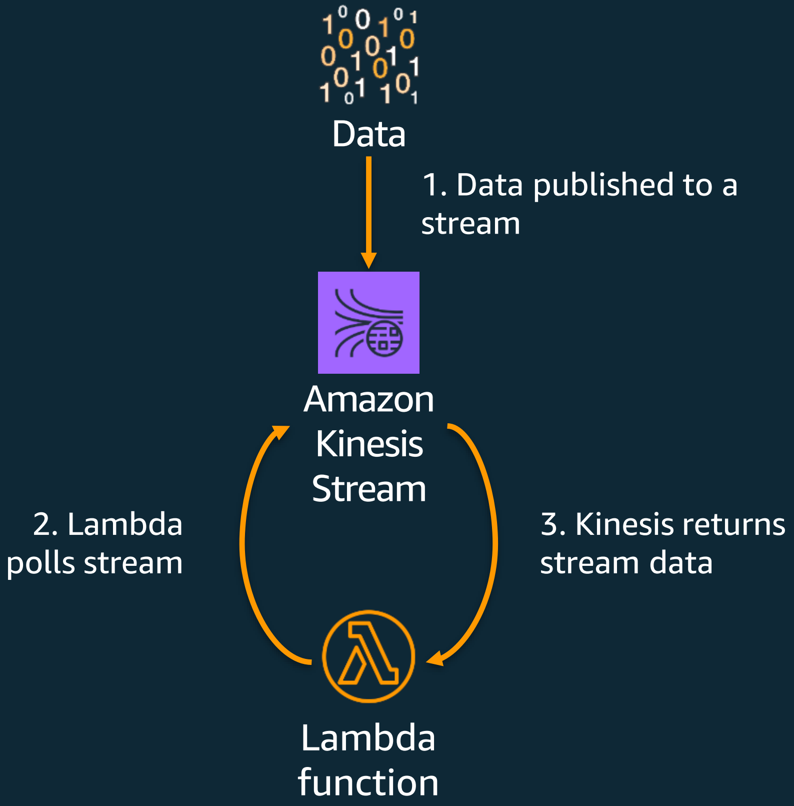
Simple, flexible, fully managed event bus router to connect applications together by ingesting and processing data across your own applications, AWS services and SaaS applications.

Events are published to an event bus

Set up rules to filter metadata and payload, and route events to targets

Asynchronous

Amazon Kinesis Streams + Lambda



Fully managed, highly scalable service for collecting and processing real-time data streams for analytics and machine learning

Stream consists of shards with a fixed amount of capacity and throughput

Lambda receives batches and potentially batches of batches

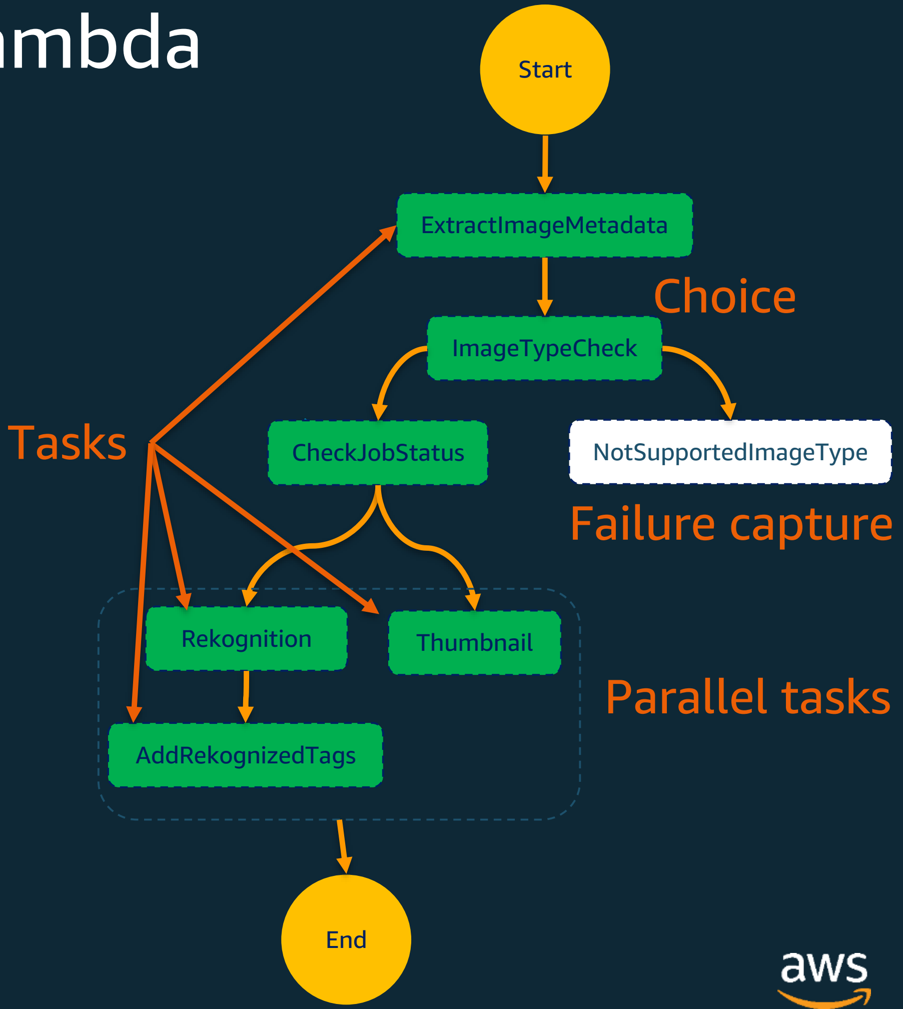
Can have different applications consuming the same stream

Stream

AWS Step Functions + Lambda

"Serverless" workflow management with zero administration:

- Coordinate microservices using visual workflows
- Automatically triggers and tracks each step
- Can handle custom failure messages from Lambda code



Awareness of messaging-payload size limits

Lambda	Sync: 6 MB Async: 256 KB
Amazon SQS	256 KB
Amazon SNS	256 KB (SMS) 1,600 b
Step Functions	32 KB
API Gateway	HTTP: 10 MB WebSockets: 128 KB (32-MB frames)

Comparing messaging services

Scale/Concurrency controls

Durability

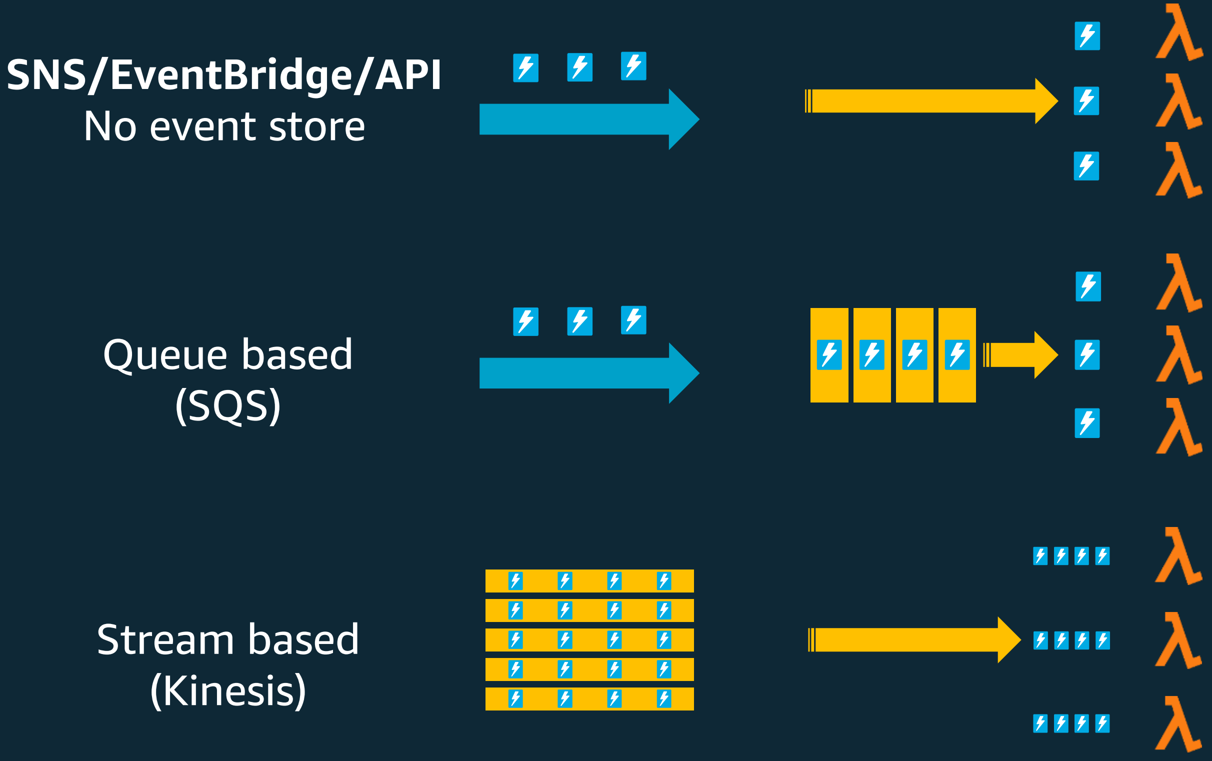
Persistence

Consumption models

Retries

Pricing

Concurrency across models



Lambda Dead Letter Queues

"You can configure your function with a dead-letter queue to save discarded events for further processing."

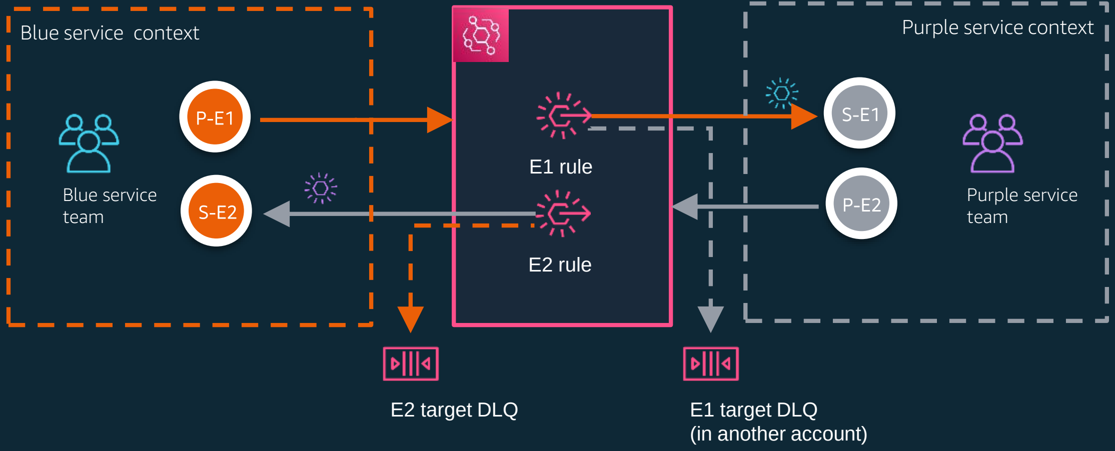
- Amazon SQS queue
  - Monitor via an SQS Queue length metric/alarm
- Amazon SNS topic
  - Send messages to something durable and/or a trusted endpoint for processing
  - Can send to Lambda functions in other regions

- **If and when things go "boom" DLQs can save your invocation event information**

Amazon EventBridge dead letter queues

Don't lose events and understand root cause

EventBridge now supports DLQ and custom retry policy (maximum # of retries or the maximum event age of the event) via customer-managed Amazon SQS queue



Possible root causes?

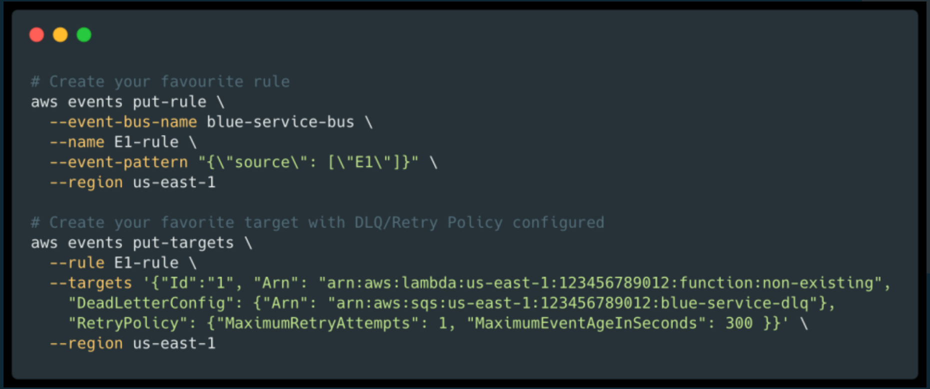
- Permissions not correct
- Service availability
- Deleted resource
- Throttling
- Cross account loop
- Invalid parameters

Amazon EventBridge dead letter queues

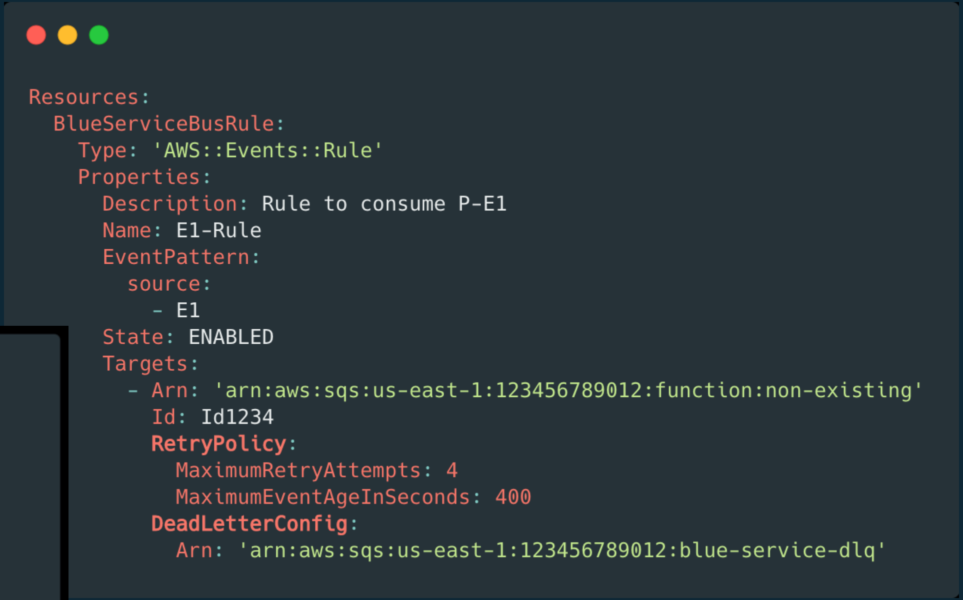
Don’t lose events and understand root cause

The DLQ and Customer Retry Policy are configured per EventBridge target and via the `PutTargets` API

DLQs for Amazon EventBridge come with AWS Console, AWS CLI and AWS CloudFormation support



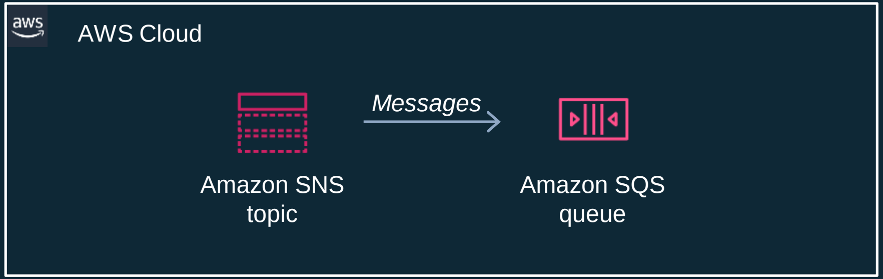
```
# Create your favourite rule
aws events put-rule \
  --event-bus-name blue-service-bus \
  --name E1-rule \
  --event-pattern "{\"source\": [\"E1\"]}" \
  --region us-east-1

# Create your favorite target with DLQ/Retry Policy configured
aws events put-targets \
  --rule E1-rule \
  --targets '{"Id":"1", "Arn": "arn:aws:lambda:us-east-1:123456789012:function:non-existing",
    "DeadLetterConfig": {"Arn": "arn:aws:sqs:us-east-1:123456789012:blue-service-dlq"},
    "RetryPolicy": {"MaximumRetryAttempts": 1, "MaximumEventAgeInSeconds": 300 }}' \
  --region us-east-1
```



```
Resources:
  BlueServiceBusRule:
    Type: 'AWS::Events::Rule'
    Properties:
      Description: Rule to consume P-E1
      Name: E1-Rule
      EventPattern:
        source:
          - E1
      State: ENABLED
      Targets:
        - Arn: 'arn:aws:sqs:us-east-1:123456789012:function:non-existing'
          Id: Id1234
          RetryPolicy:
            MaximumRetryAttempts: 4
            MaximumEventAgeInSeconds: 400
          DeadLetterConfig:
            Arn: 'arn:aws:sqs:us-east-1:123456789012:blue-service-dlq'
```

Combining Messaging Patterns

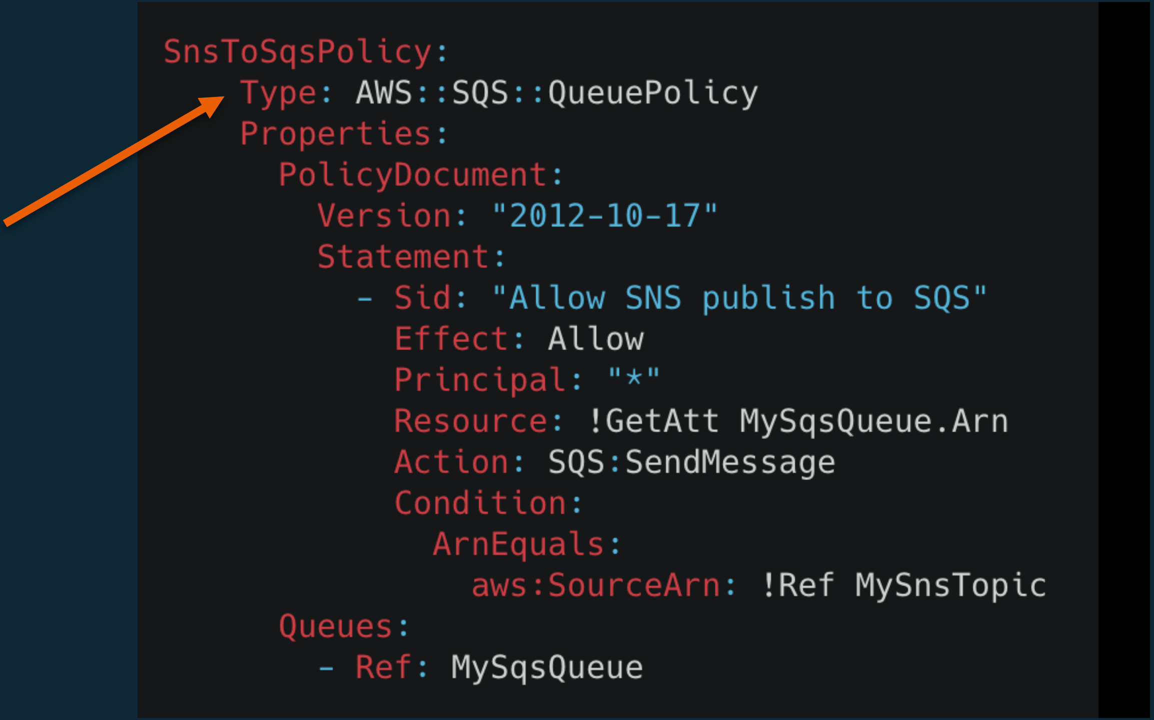
SNS → SQS



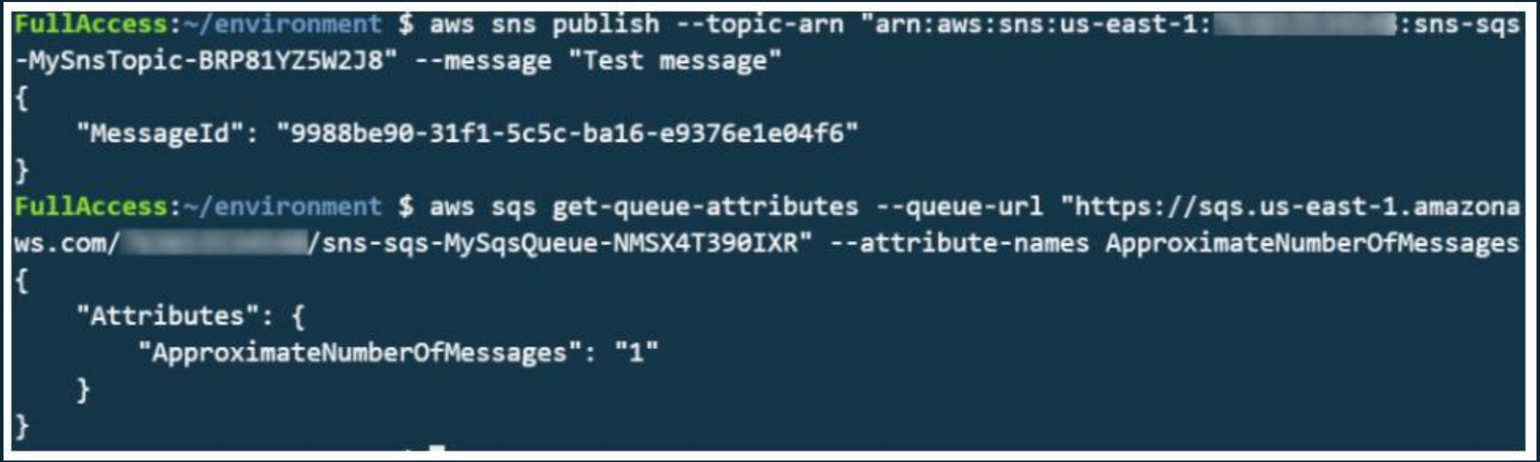
SNS → SQS

```yaml
MySqsQueue:
  Type: AWS::SQS::Queue

MySnsTopic:
  Type: AWS::SNS::Topic
  Properties:
    Subscription:
      - Protocol: sqs
        Endpoint: !GetAtt MySqsQueue.Arn
```

SNS → SQS



```
SnsToSqsPolicy:
    Type: AWS::SQS::QueuePolicy
    Properties:
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Sid: "Allow SNS publish to SQS"
            Effect: Allow
            Principal: "*"
            Resource: !GetAtt MySqsQueue.Arn
            Action: SQS:SendMessage
            Condition:
              ArnEquals:
                aws:SourceArn: !Ref MySnsTopic
      Queues:
        - Ref: MySqsQueue
```

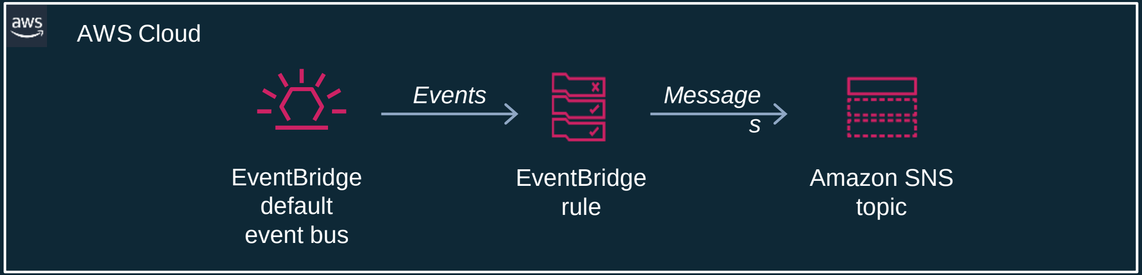


```
FullAccess:~/environment $ aws sns publish --topic-arn "arn:aws:sns:us-east-1:[redacted]:sns-sqs
-MySnsTopic-BRP81YZ5W2J8" --message "Test message"
{
    "MessageId": "9988be90-31f1-5c5c-ba16-e9376e1e04f6"
}
FullAccess:~/environment $ aws sqs get-queue-attributes --queue-url "https://sqs.us-east-1.amazona
ws.com/[redacted]/sns-sqs-MySqsQueue-NMSX4T390IXR" --attribute-names ApproximateNumberOfMessages
{
    "Attributes": {
        "ApproximateNumberOfMessages": "1"
    }
}
```

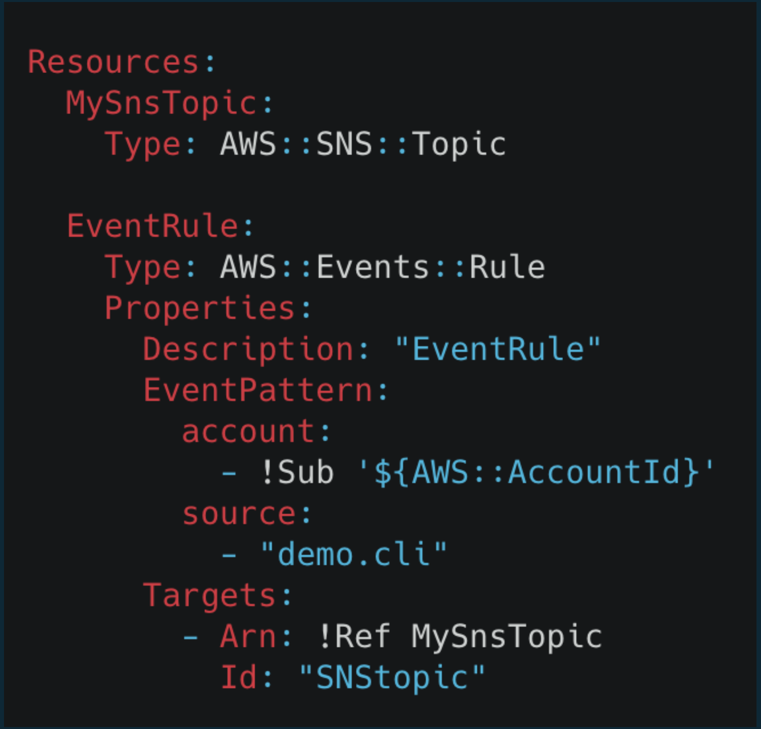
SNS → SQS

```
QueueSubcription:
   Type: 'AWS::SNS::Subscription'
   Properties:
     TopicArn: !Ref MySnsTopic
     Endpoint: !GetAtt MySqsQueue.Arn
     Protocol: sqs
     FilterPolicy:
       type:
       - orders
       - payments
     RawMessageDelivery: 'true'
```

EventBridge → SNS



EventBridge → SNS

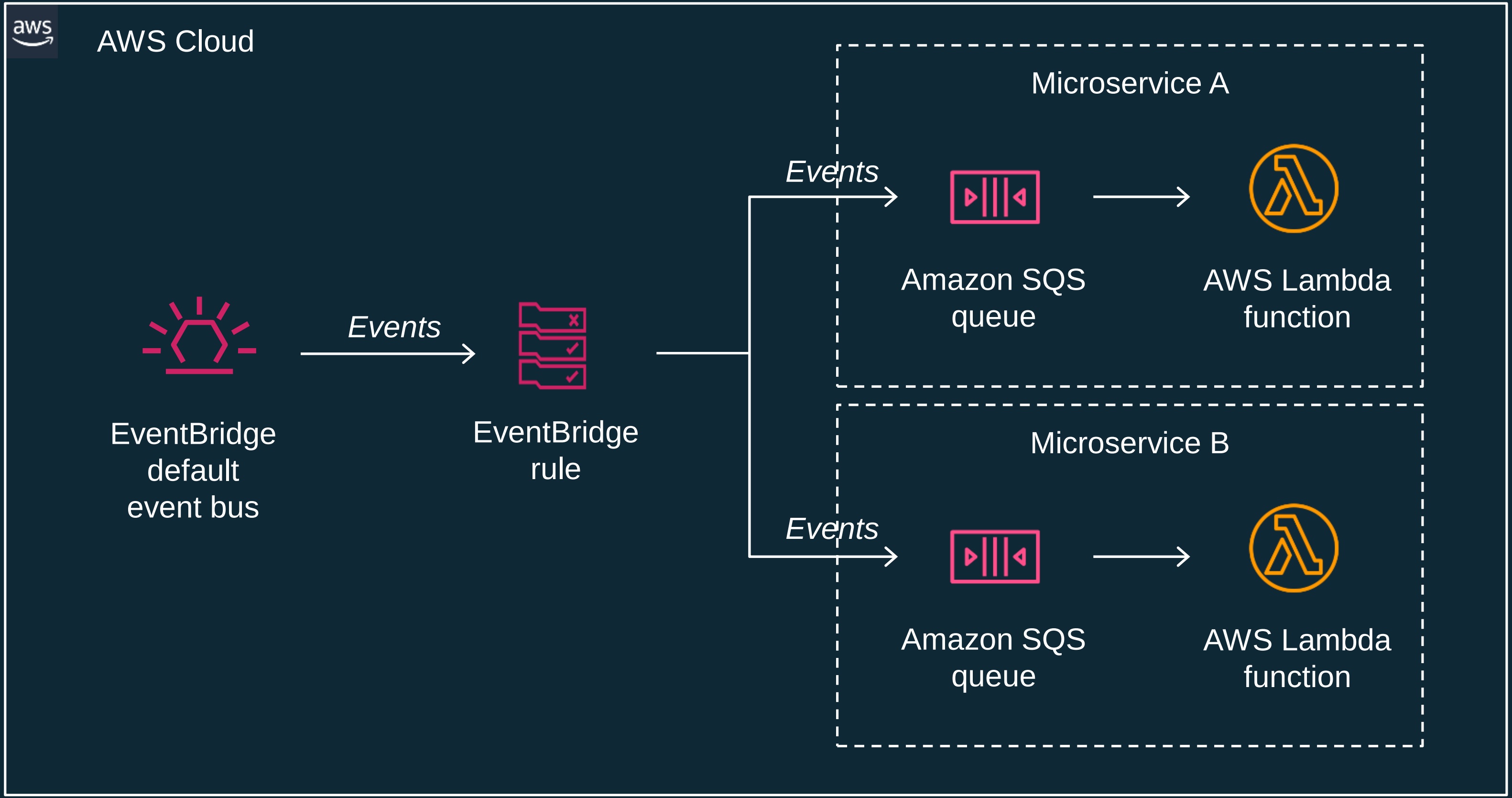


```yaml
Resources:
  MySnsTopic:
    Type: AWS::SNS::Topic

  EventRule:
    Type: AWS::Events::Rule
    Properties:
      Description: "EventRule"
      EventPattern:
        account:
          - !Sub '${AWS::AccountId}'
        source:
          - "demo.cli"
      Targets:
        - Arn: !Ref MySnsTopic
          Id: "SNStopic"
```

EventBridge → SNS

```
EventBridgeToToSnsPolicy:
  Type: AWS::SNS::TopicPolicy
  Properties:
    PolicyDocument:
      Statement:
      - Effect: Allow
        Principal:
          Service: events.amazonaws.com
        Action: sns:Publish
        Resource: !Ref MySnsTopic
    Topics:
      - !Ref MySnsTopic
```

EventBridge → SQS



EventBridge → SQS

```yaml
EventBridgeToToSqsPolicy:
  Type: AWS::SQS::QueuePolicy
  Properties:
   PolicyDocument:
    Statement:
    - Effect: Allow
      Principal:
       Service: events.amazonaws.com
      Action: SQS:SendMessage
      Resource:  !GetAtt MySqsQueue.Arn
   Queues:
    - Ref: MySqsQueue
```

OK, what do I do already!?

	Amazon EventBridge	Amazon SNS
Sources	More than 90 AWS services 21 SaaS integrations Custom applications	30 AWS services Custom applications
Targets	18 AWS services	2 AWS services + 4 web & mobile endpoints
Fan Out	5 targets per rule 400-2400 events/sec (soft, can be up to 100Ks) 750-4500 invocations / sec (soft)	Supports millions of subscribers per topic
Filtering	Rules apply to entire event body Advanced filtering rules, has input transformation, schema registry/discovery	Filters apply only to message attributes (10 per message) Content-based filtering done in code
Latency	Median of 560ms	Median of 25ms
Price	AWS event sources are free $1.00/million custom or SaaS events Free to deliver events to any AWS target	$0.50/million messages to a topic Deliveries to AWS services (SQS, Lambda) are free. $0.50/million for mobile push, $0.60/million for HTTP/S, $20/million for email, SMS deliveries vary by region

When to use X or EventBridge

CloudWatch Events

= replace with EventBridge

✗ only AWS services as sources, only uses default event bus. no SaaS integrations

SNS

✓ for high throughput (millions TPS), millions of subscribers, very low latency

✗ only limited targets, no ordering, filtering only on attributes, may need multiple topics

Kinesis

✓ for real-time processing at large scale, routing and storing, guarantees order

✗ limited consumers per stream, not serverless (does not scale automatically, not usage based pricing)

SQS

✓ need resiliency, ordering guarantees (FIFO queues), buffer downstream services

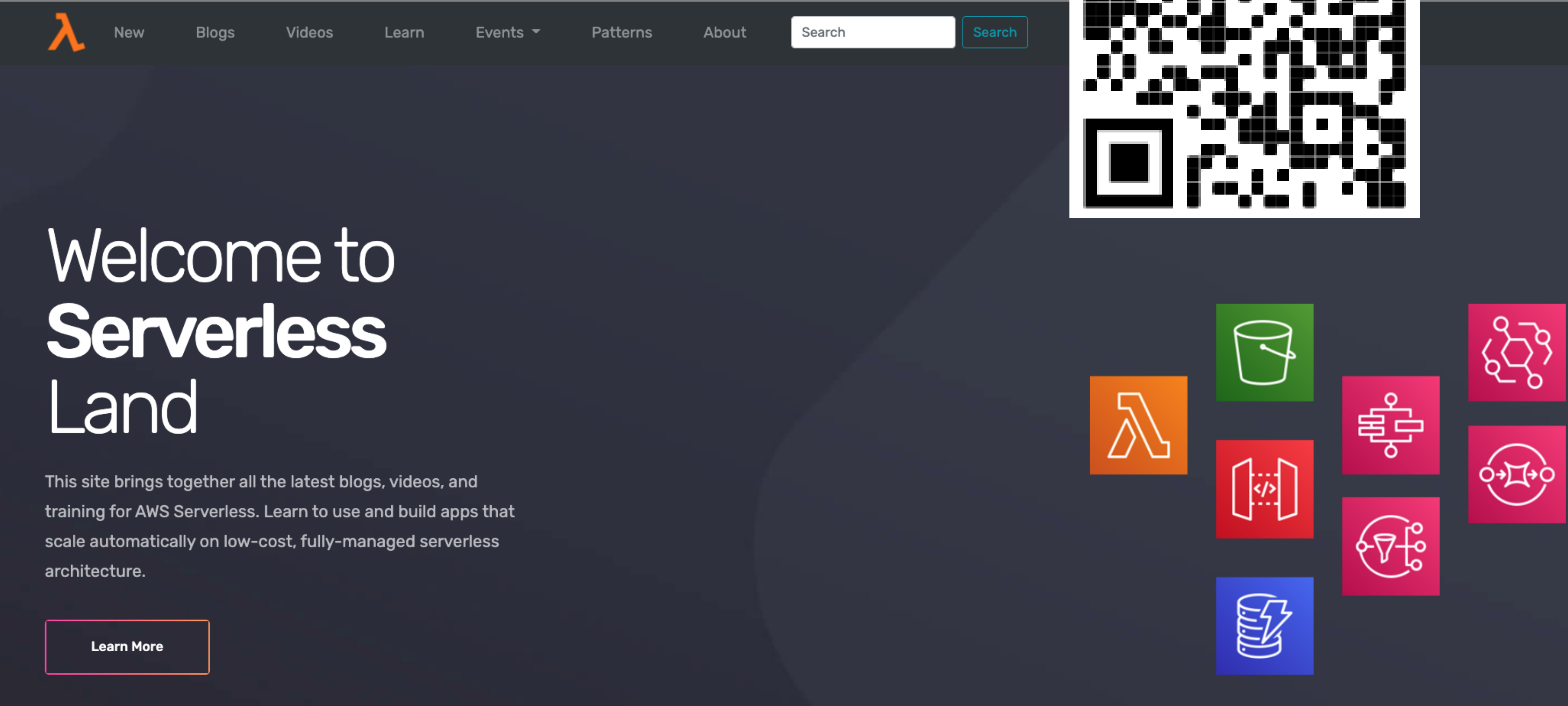
✗ no filtering, no ordering (standard queues)

Summary

There are many ways to get data between microservices!

- Kinesis, SNS, SQS, EventBridge, and the Lambda API are just a few of the ways.
- You **might** need an API that you create yourself.
- Compare scale, durability, persistence, consumption models, retries, and pricing.
- May need more than one in some part of your infrastructure.
- Evaluate and test using SAM CLI.
- Serverless pricing models make testing new ideas low cost and easy to get started with!

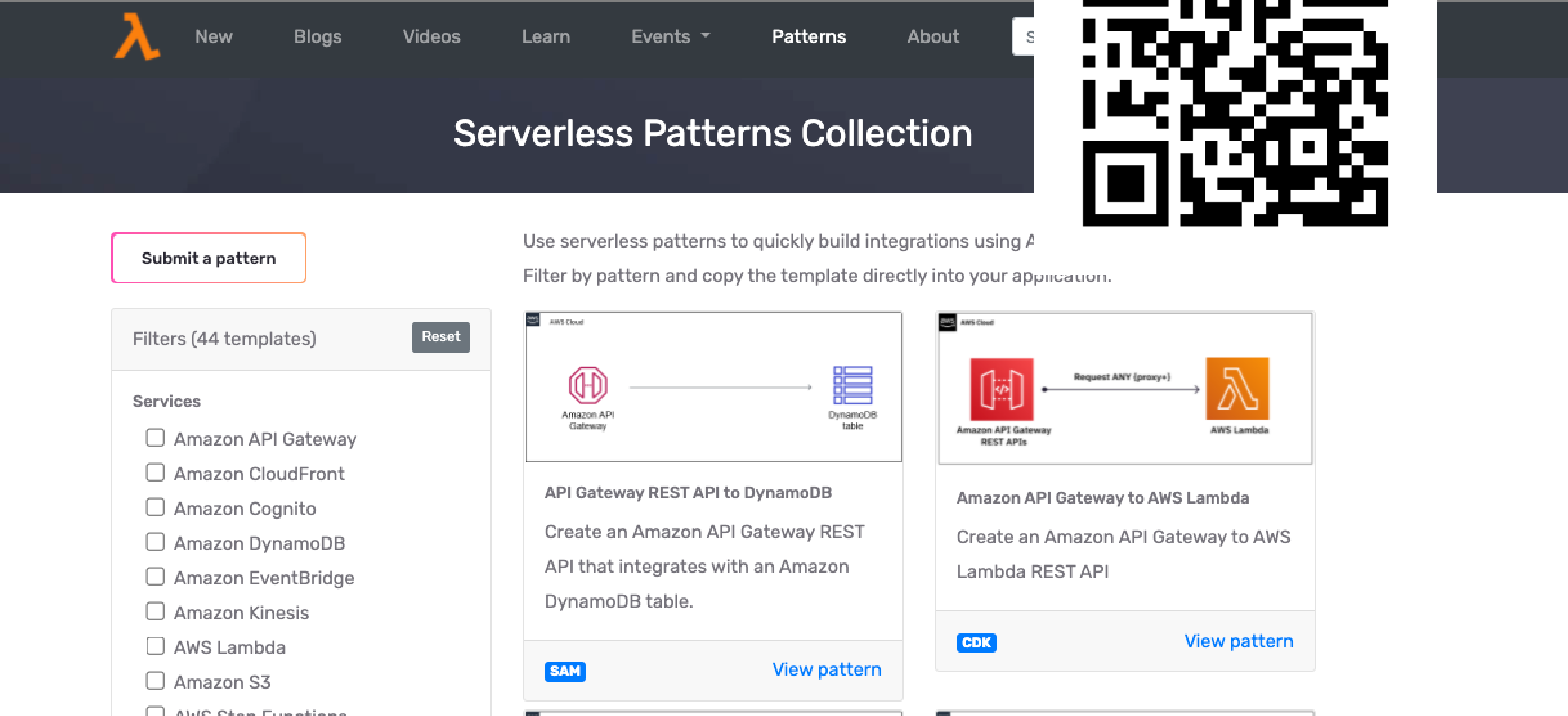
https://serverlessland.com



Welcome to **Serverless** Land

This site brings together all the latest blogs, videos, and training for AWS Serverless. Learn to use and build apps that scale automatically on low-cost, fully-managed serverless architecture.

Serverless Patterns Collection



Serverless Patterns Collection

Submit a pattern

Use serverless patterns to quickly build integrations using A

Filter by pattern and copy the template directly into your application.

Filters (44 templates) Reset

Services

- Amazon API Gateway
- Amazon CloudFront
- Amazon Cognito
- Amazon DynamoDB
- Amazon EventBridge
- Amazon Kinesis
- AWS Lambda
- Amazon S3
- AWS Step Functions

API Gateway REST API to DynamoDB

Create an Amazon API Gateway REST API that integrates with an Amazon DynamoDB table.

SAM View pattern

Amazon API Gateway to AWS Lambda

Create an Amazon API Gateway to AWS Lambda REST API

CDK View pattern

Thank you!

@talia_nassi