
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

SVS205-R

Thinking asynchronously: Integration patterns for microservices

Rebekah Kulidzan (she/her)

Go-to-Market Specialist, Workflows
AWS

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Who am I?

- Rebekah Kulidzan (@rkulidzan)
- GTM Specialist, Workflows
- Serverless and event-driven architectures enthusiast
- Mental Health Advocate
- Former Solutions Architect, Data Engineer, Policy Consultant and Social Media Advisor



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Common serverless pattern

Decoupling your application

Enterprise integration patterns

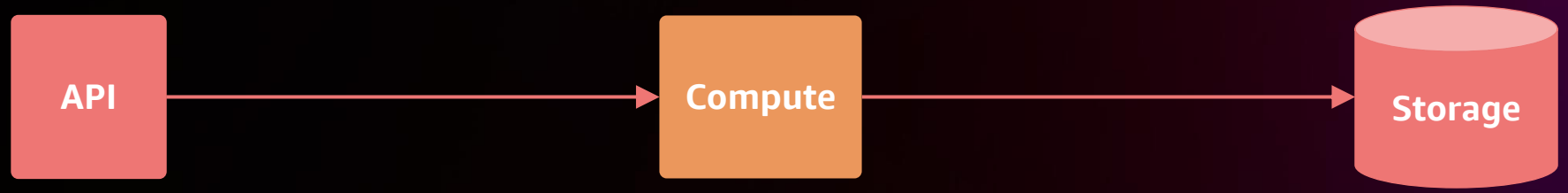
Event-driven architecture

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Common serverless pattern

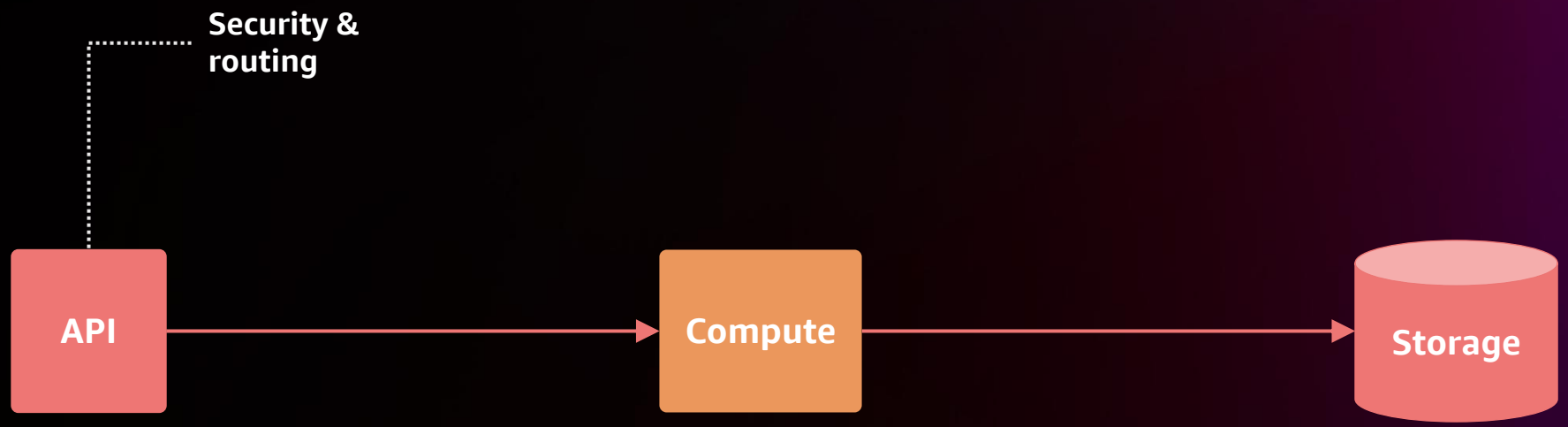
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Application elements



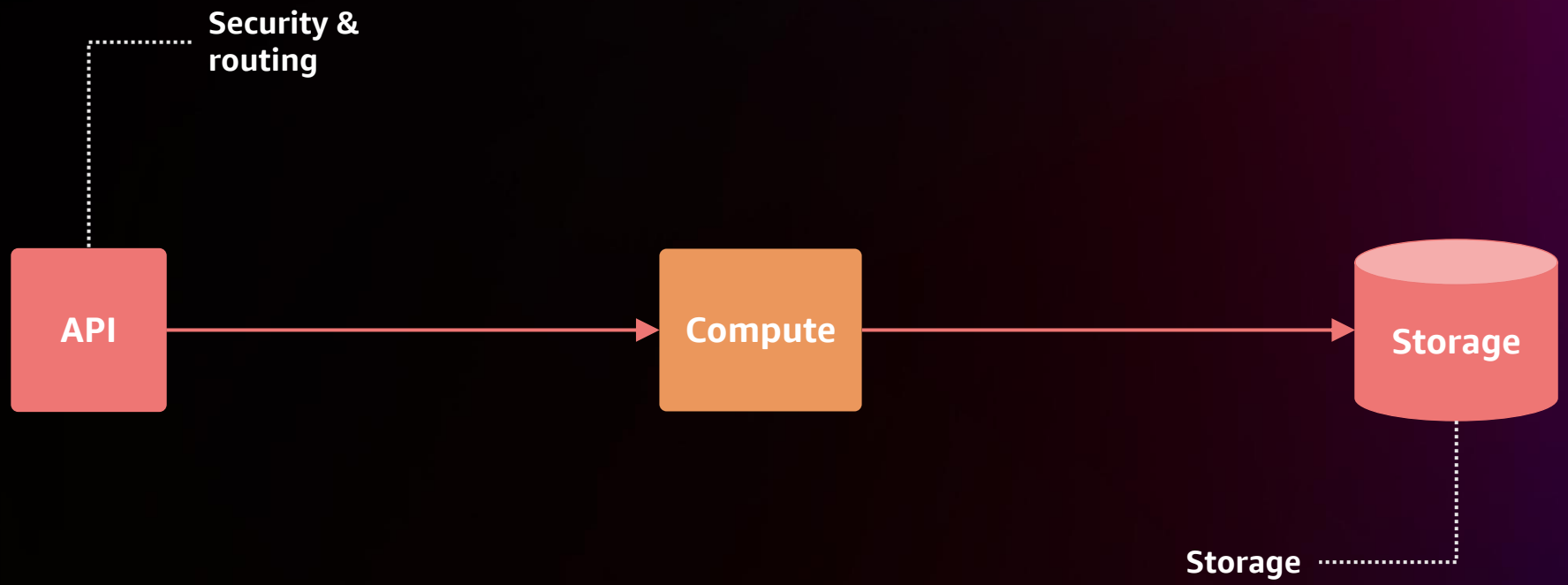
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Application elements



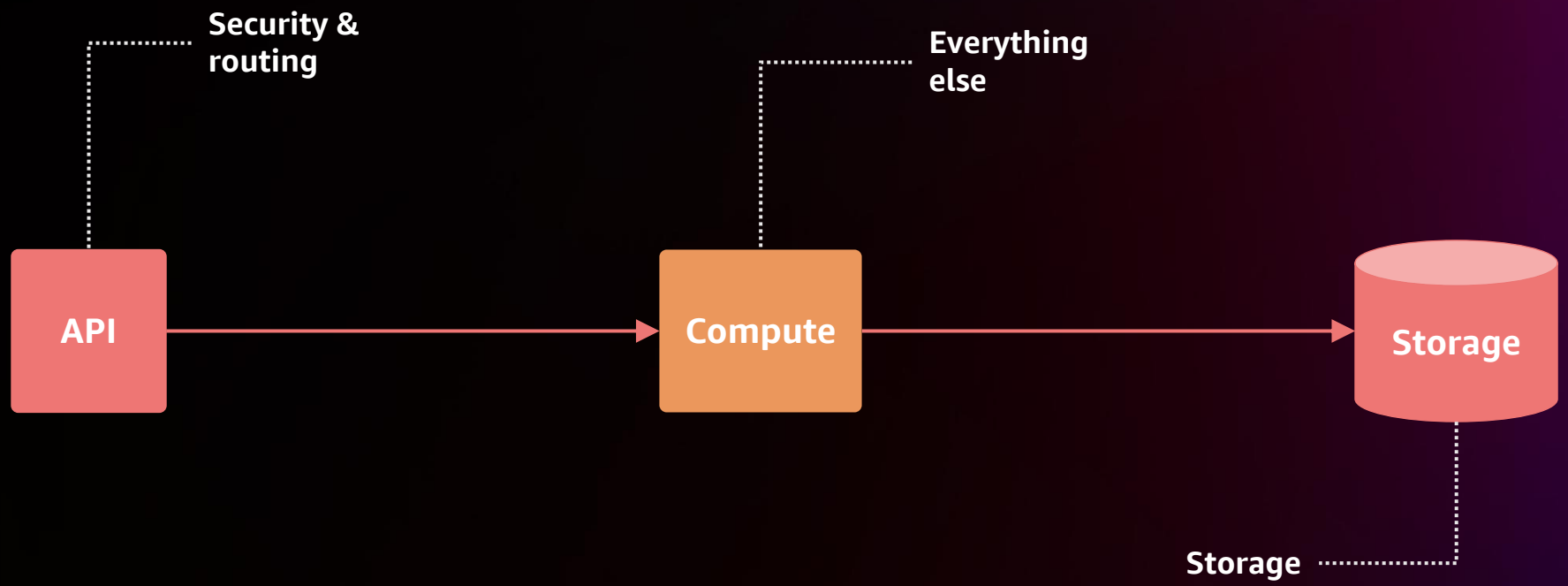
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Application elements



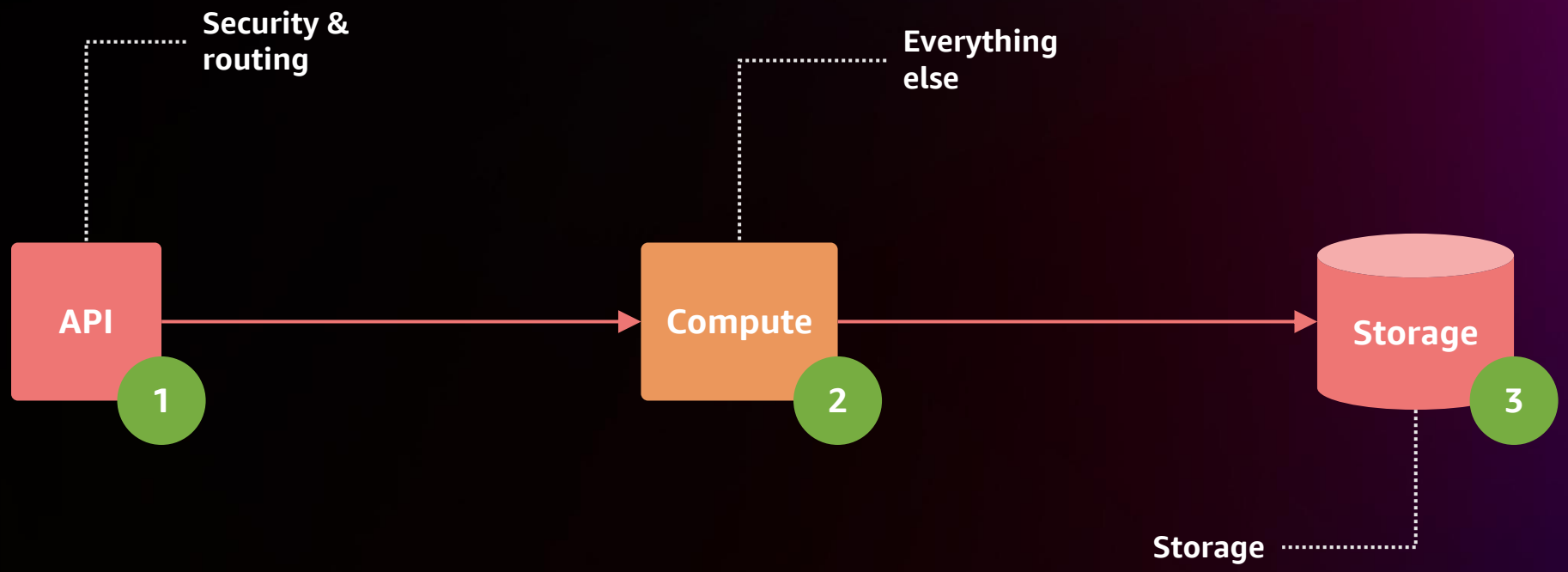
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Application elements



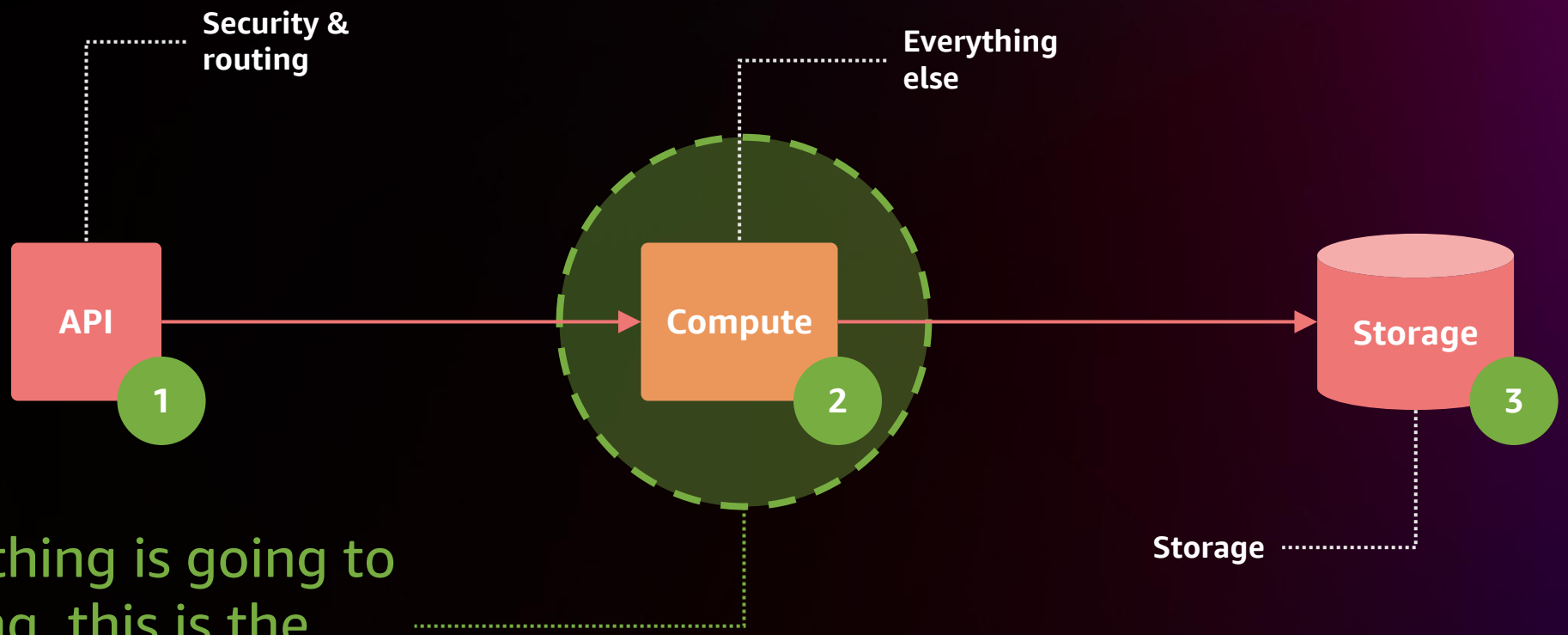
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Application elements



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

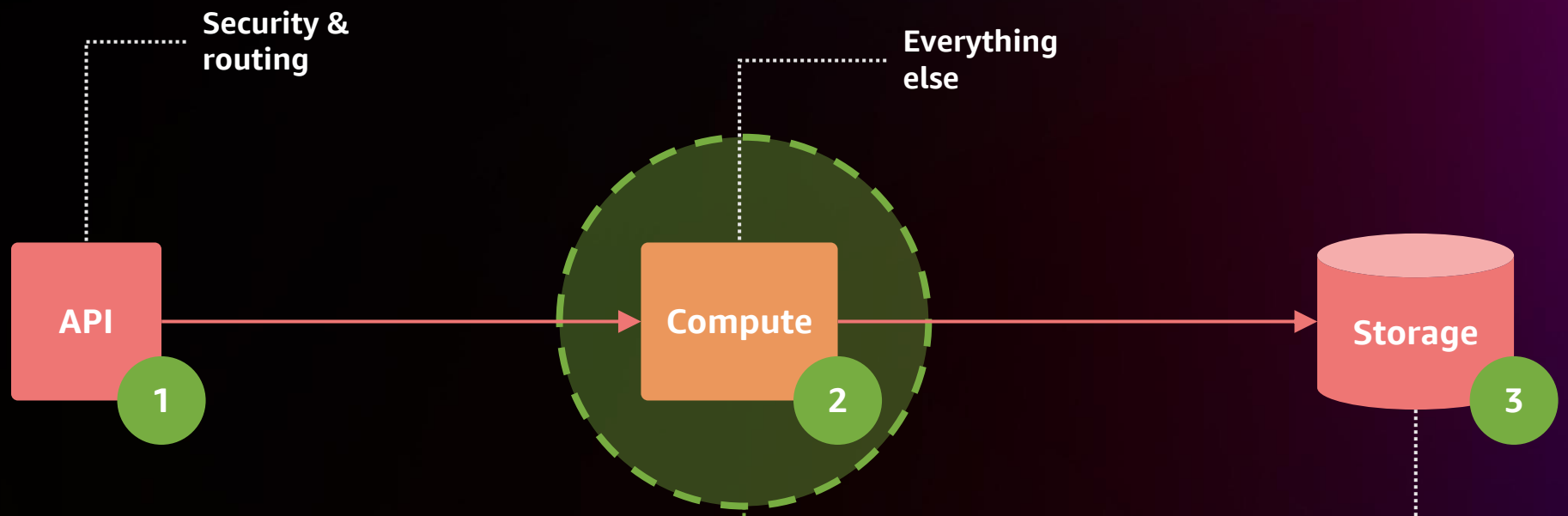
Application elements



If something is going to go wrong, this is the most probable place

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Application elements

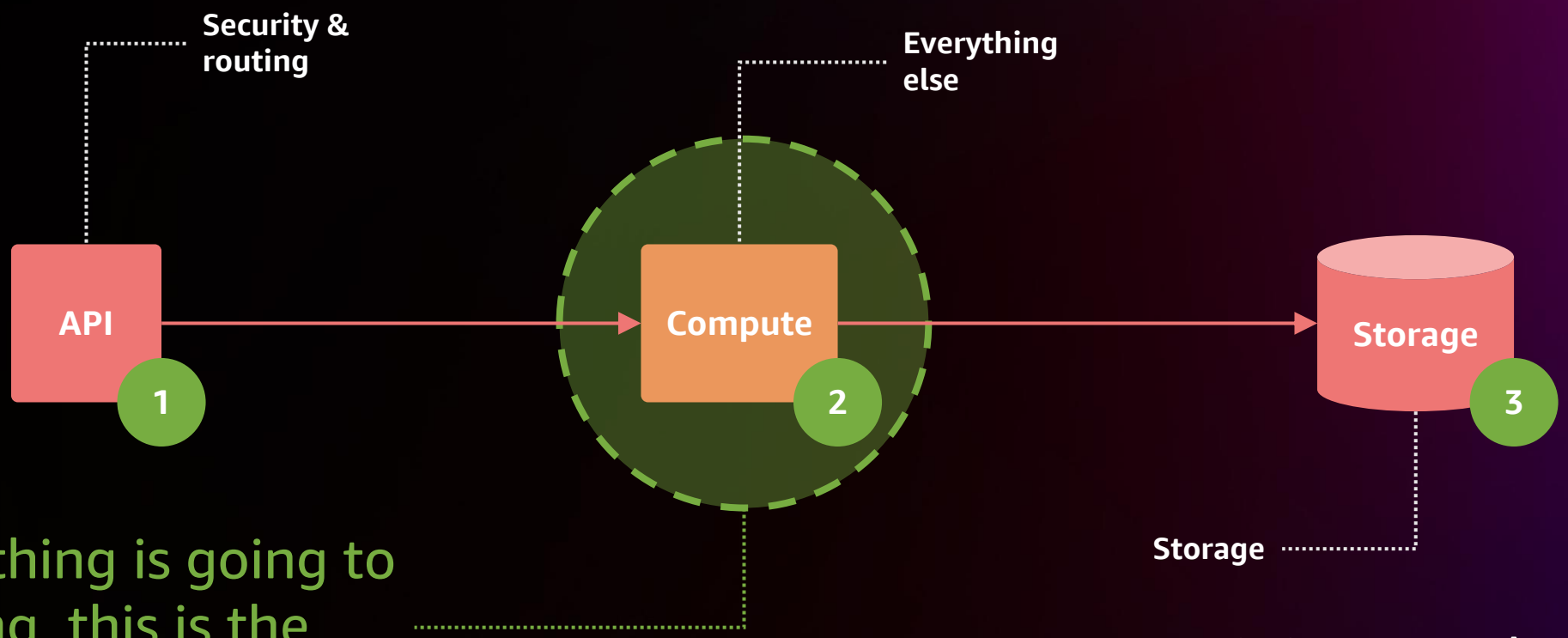


If something is going to go wrong, this is the most probable place

Why?

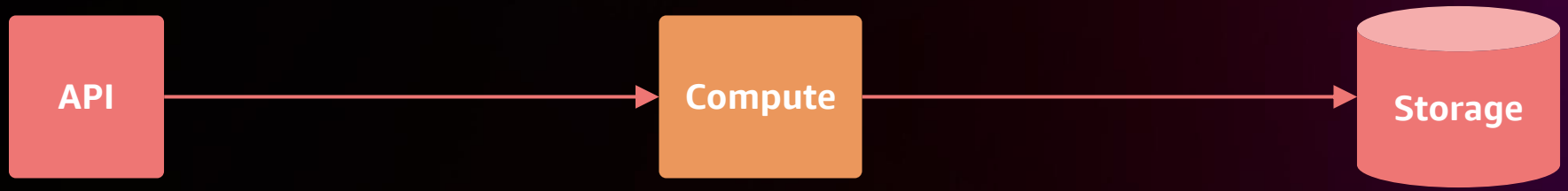
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Application elements



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How do we improve this?

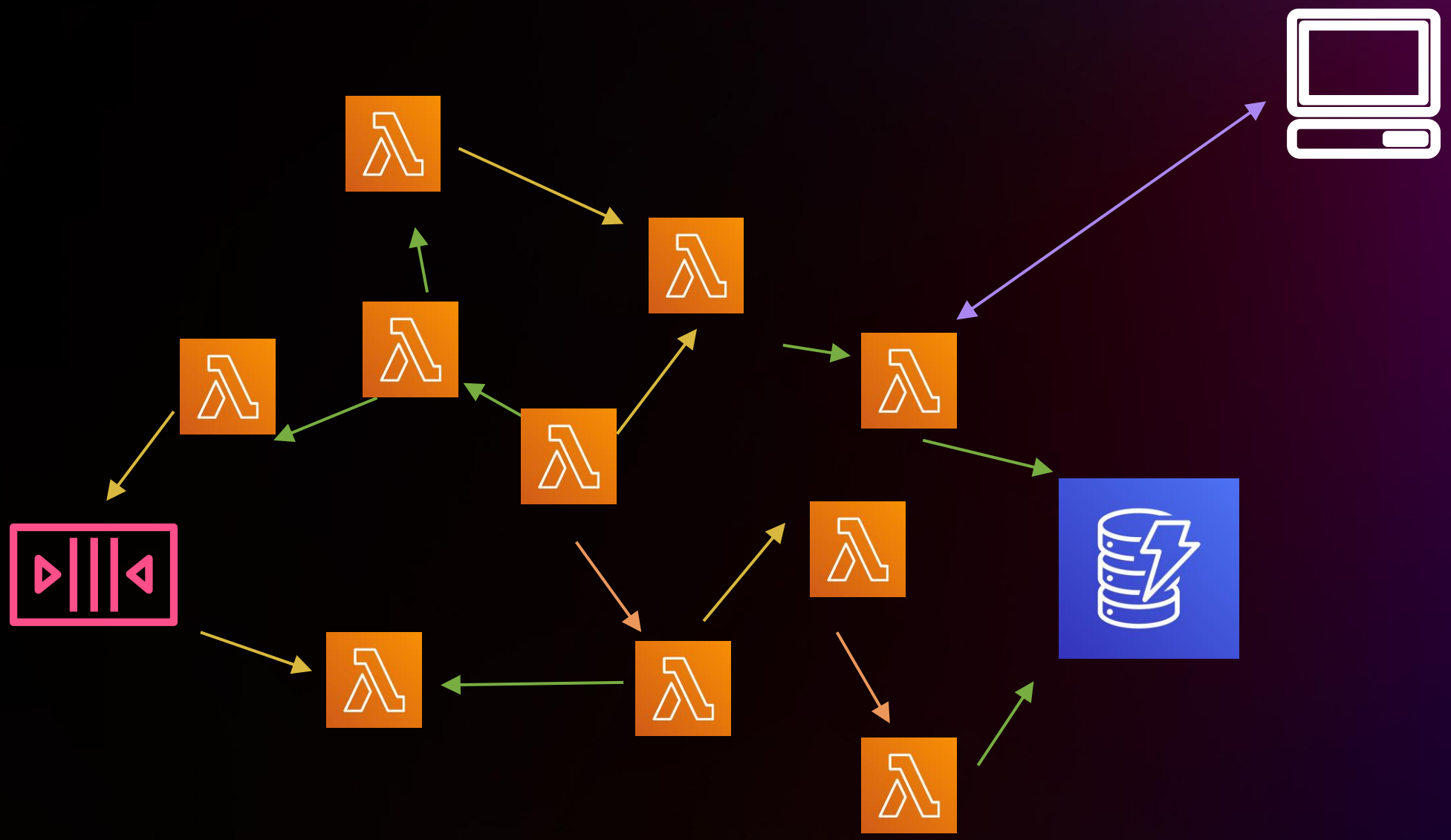


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Chalking and talking…

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

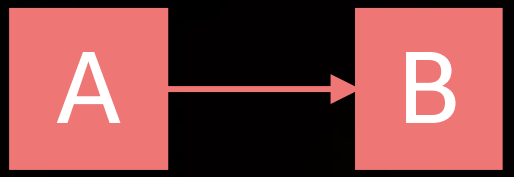
Modern application



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Then you might say the following…

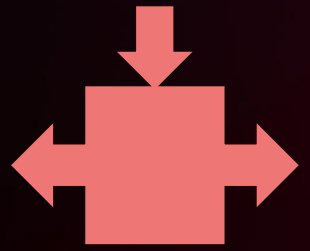
"I want to sequence tasks"



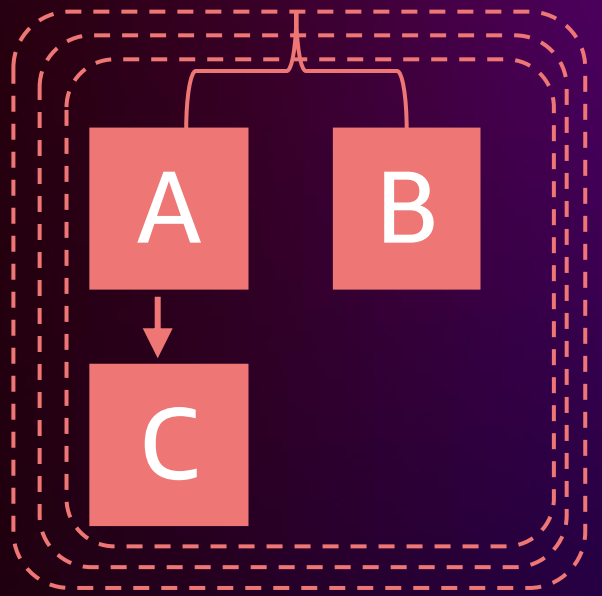
"I want to retry failed tasks"



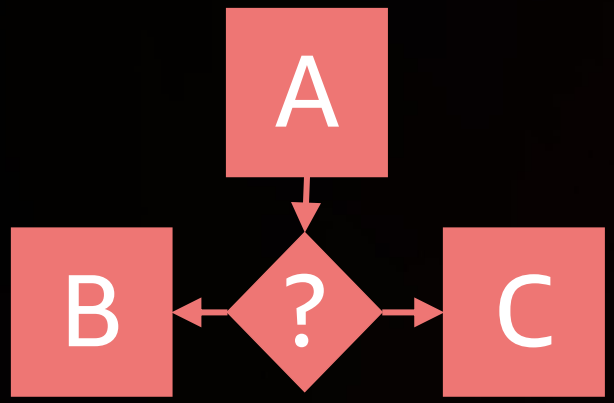
"I want try/catch"



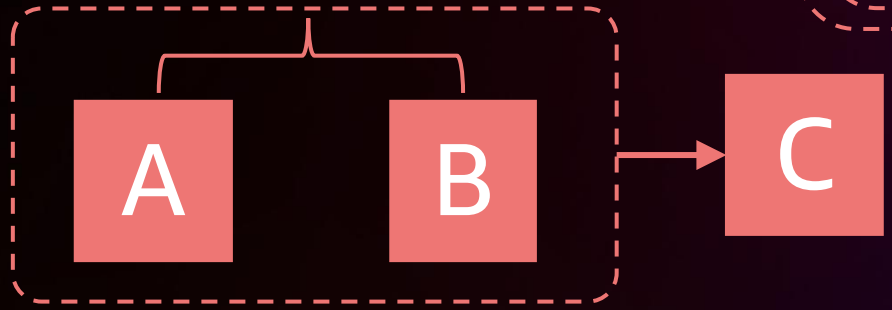
"I want concurrent and iterative tasks"



"I want to select tasks based on data"



"I want to run tasks in parallel"



aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Decoupling your application

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“The appropriate level of coupling depends on the level of control you have over the endpoints.”

Gregor Hohpe

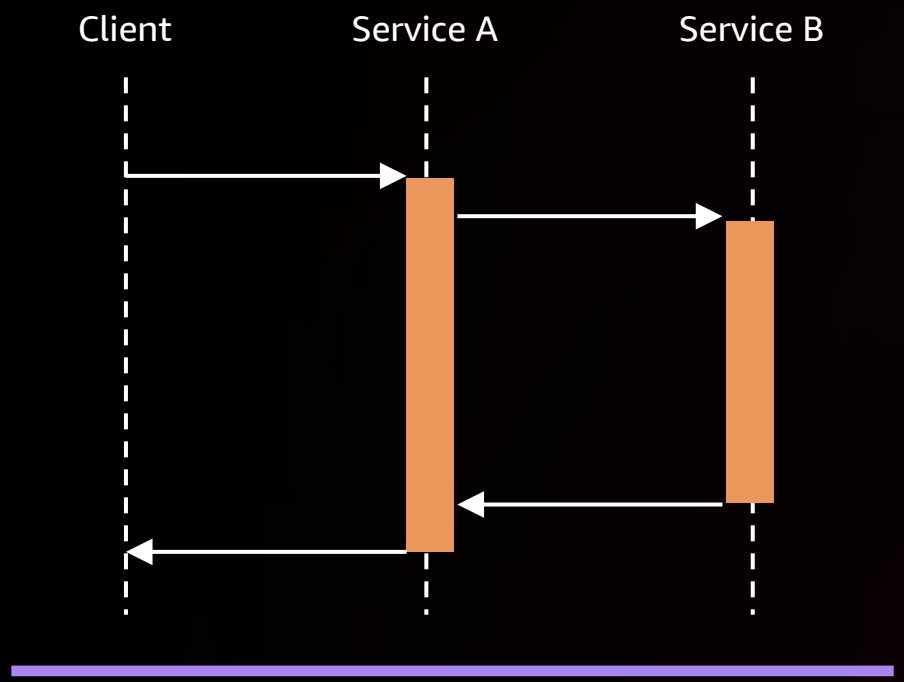
Enterprise Integration Patterns

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

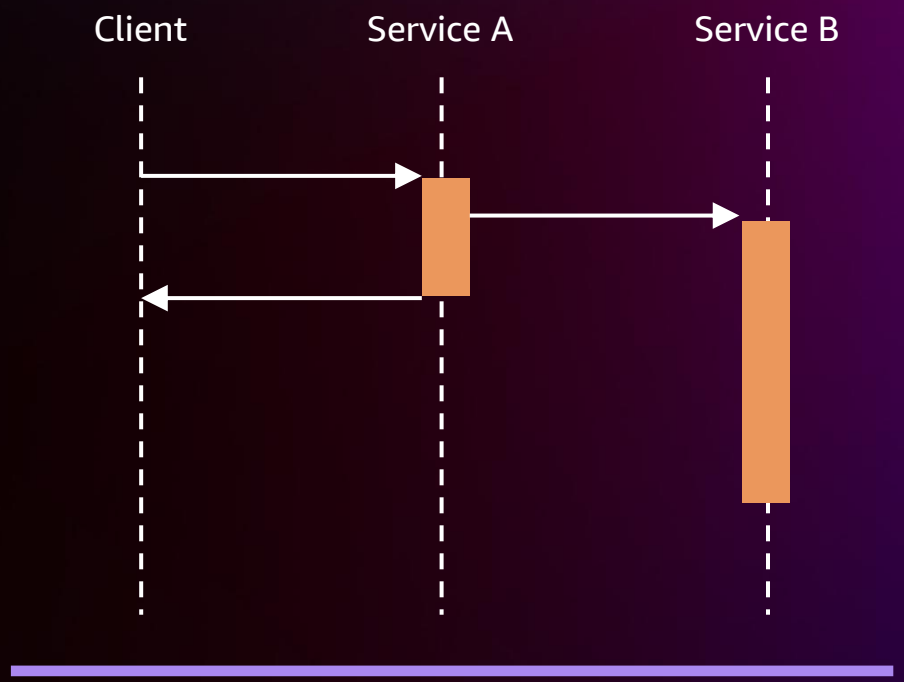
Thinking asynchronously

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Think more asynchronously



Synchronous commands



Asynchronous events

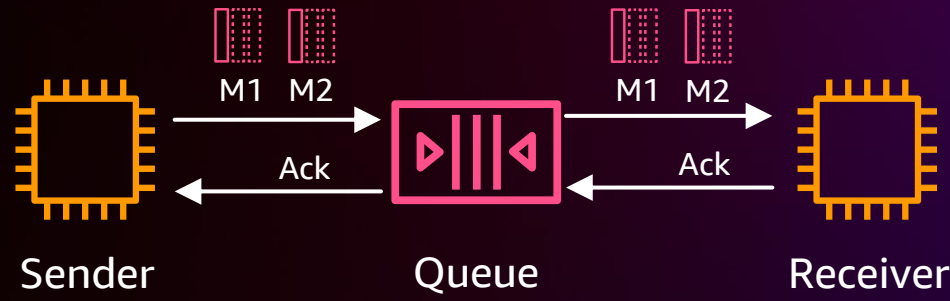
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

If you don’t need a response, execute asynchronously

Synchronous



Asynchronous



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

"If your application is cloud-native, large-scale, or distributed, and doesn't include a messaging component, that's probably a bug."

Tim Bray

Software developer and environmentalist

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Chalking and talking…

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Enterprise integration patterns

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message exchange

One-way

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message exchange

One-way



Sender

Receiver

No response expected

Message intent

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message exchange

One-way



Sender

Receiver

No response expected

Message intent

Asynchronous request-response

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message exchange

One-way



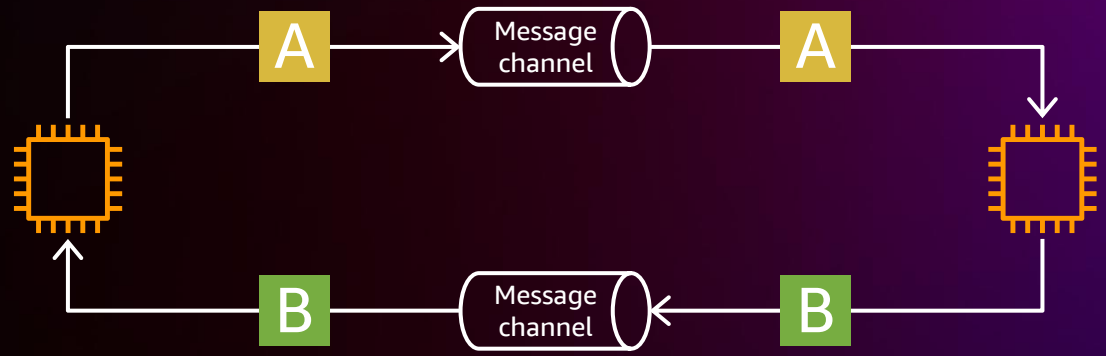
Sender

Receiver

No response expected

Message intent

Asynchronous request-response



Requester

Responder

Response expected

Return address

Correlation ID

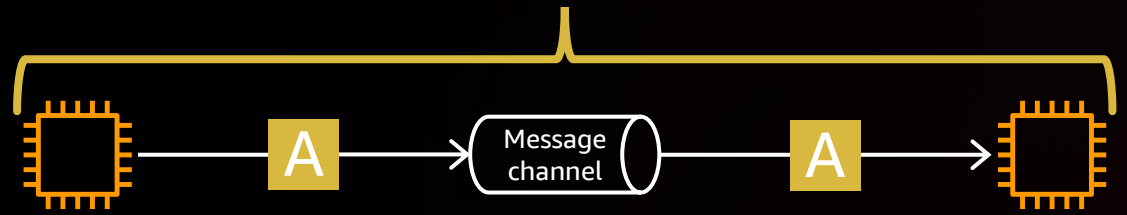
aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message exchange

One-way

Integration pattern



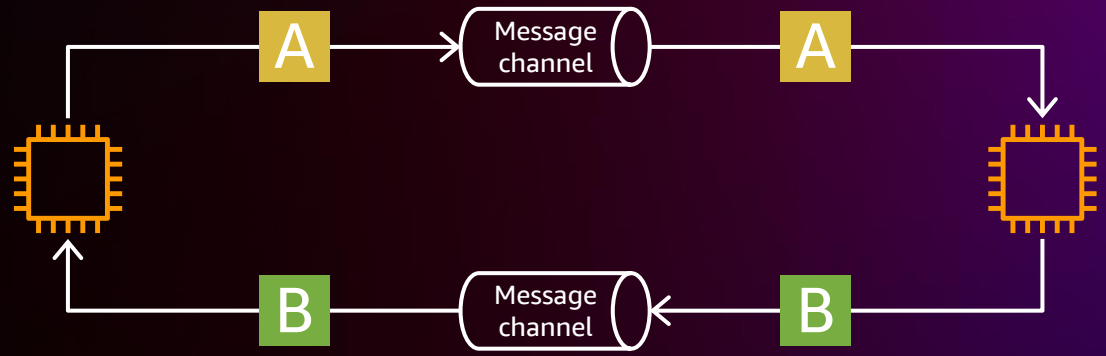
Sender

Receiver

No response expected

Message intent

Asynchronous request-response



Requester

Responder

Response expected

Return address

Correlation ID

aws

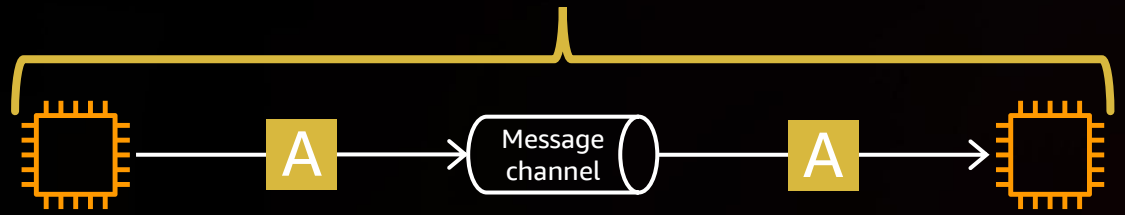
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message exchange

Conversation pattern

One-way

Integration pattern



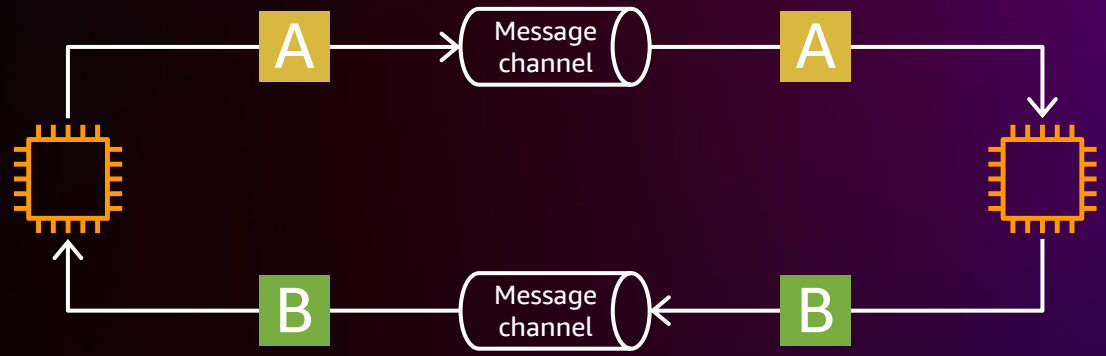
Sender

Receiver

No response expected

Message intent

Asynchronous request-response



Requester

Responder

Response expected

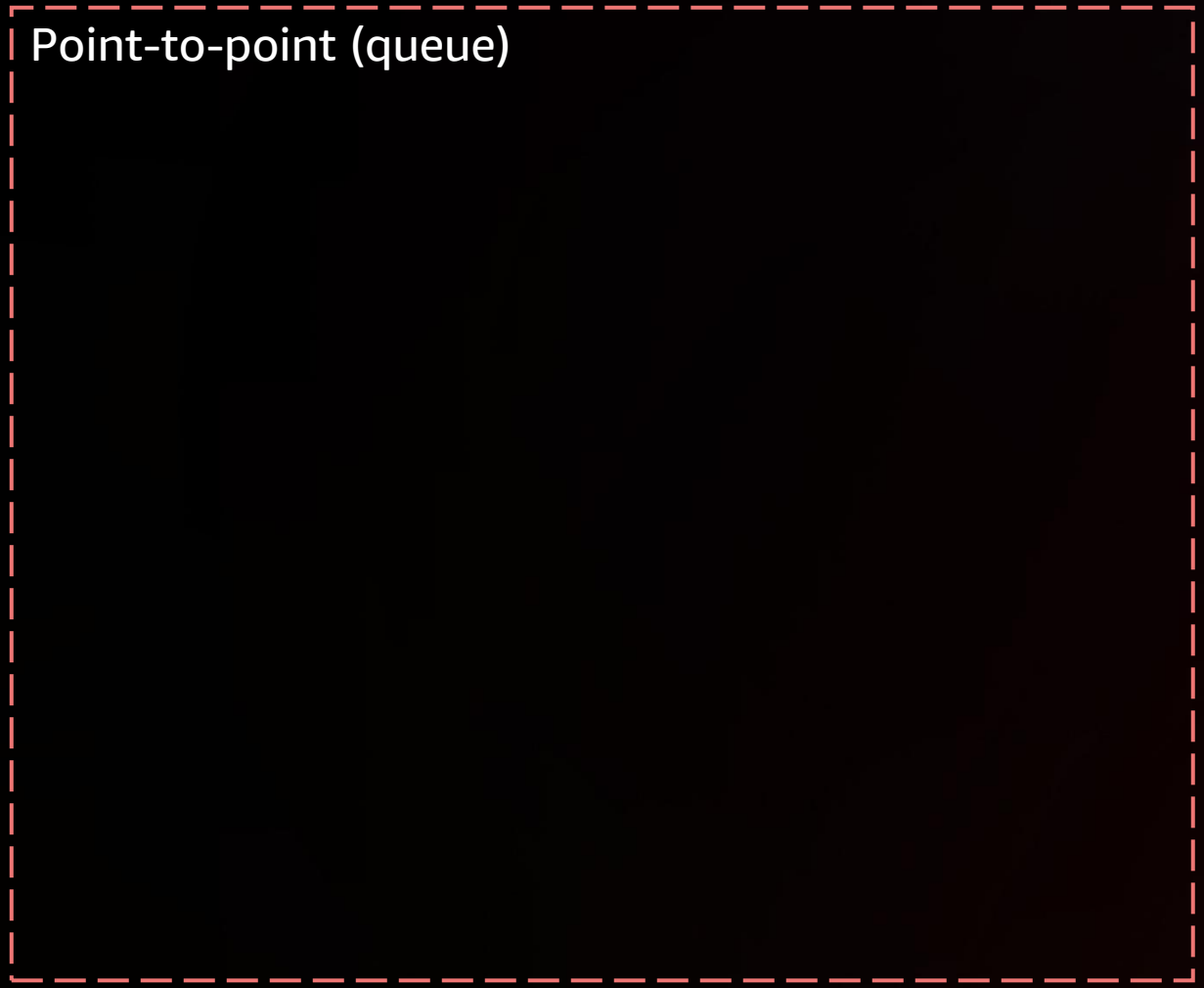
Return address

Correlation ID

aws

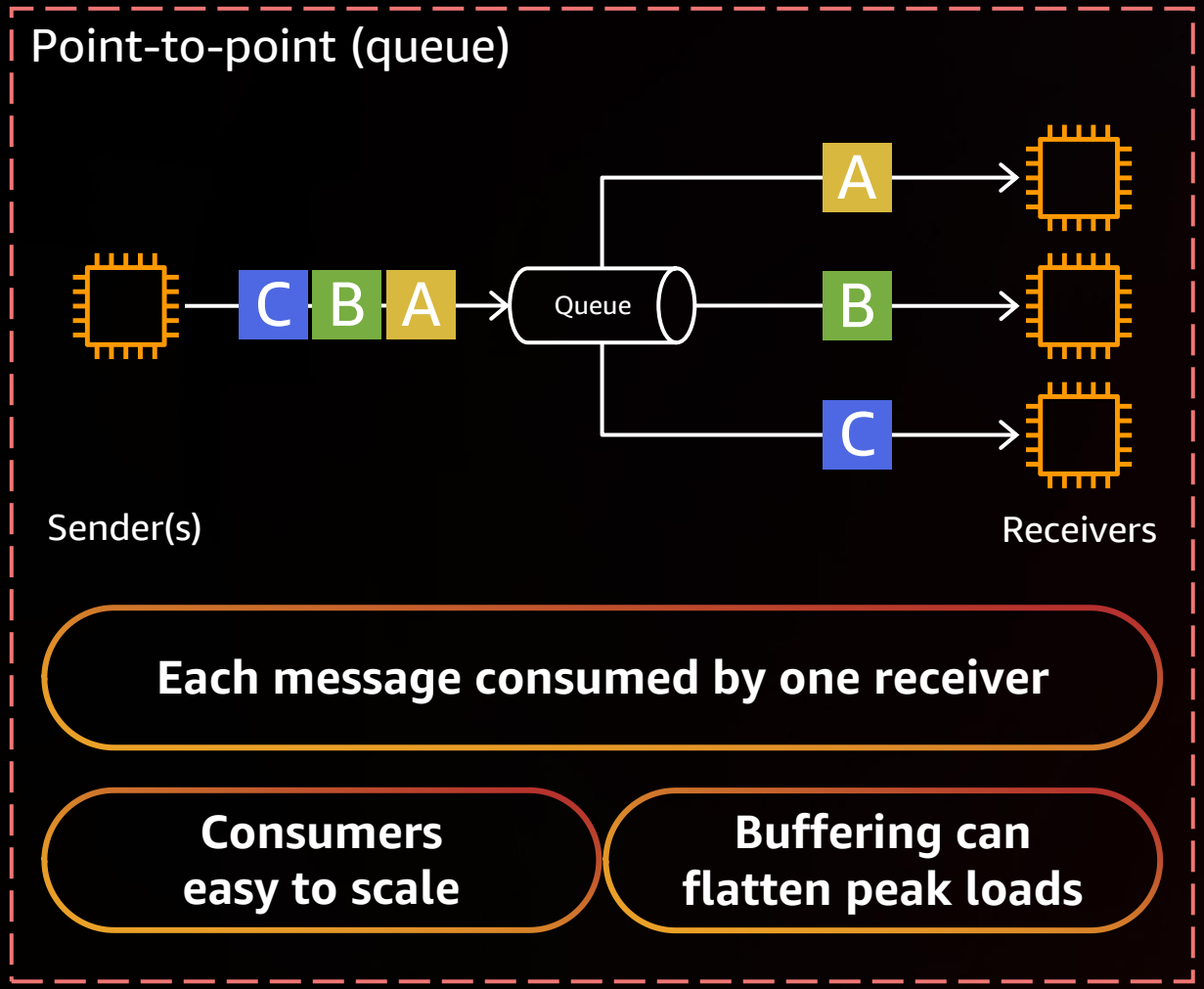
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message channels



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

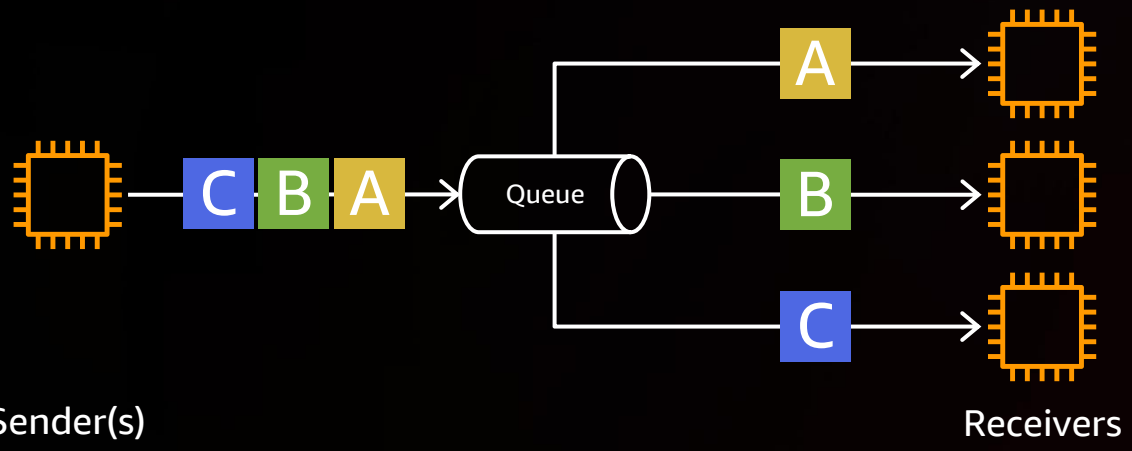
Message channels



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message channels

Point-to-point (queue)



Each message consumed by one receiver

Consumers easy to scale

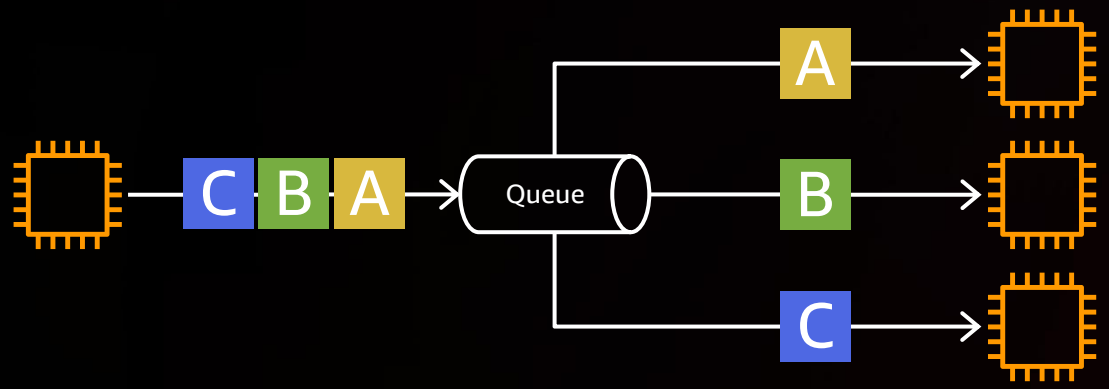
Buffering can flatten peak loads

Publish-subscribe (topic)

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message channels

Point-to-point (queue)

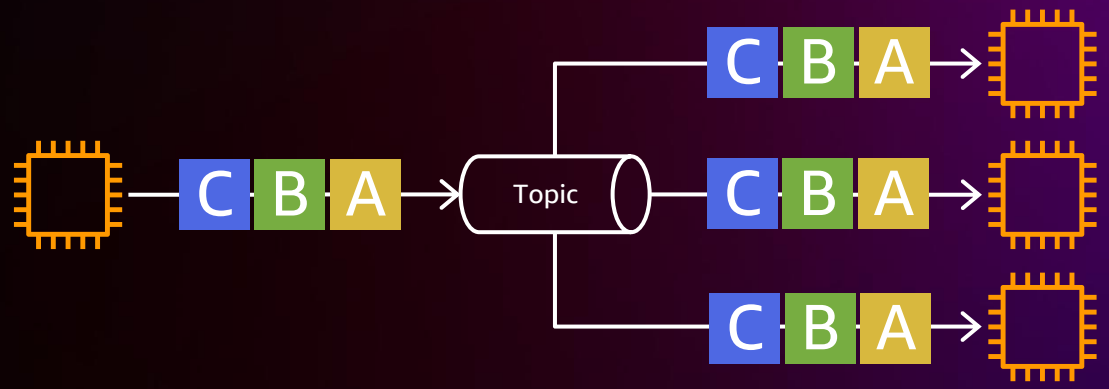


Each message consumed by one receiver

Consumers easy to scale

Buffering can flatten peak loads

Publish-subscribe (topic)



Each message consumed by each subscriber

How to scale consumers?

Conceptually no buffering

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message channels

Dead-letter queue (DLQ)

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message channels

Dead-letter queue (DLQ)



Producer

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message channels

Dead-letter queue (DLQ)



Producer

Consumers

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message channels

Dead-letter queue (DLQ)



Producer

Consumers

Transient failure mitigation

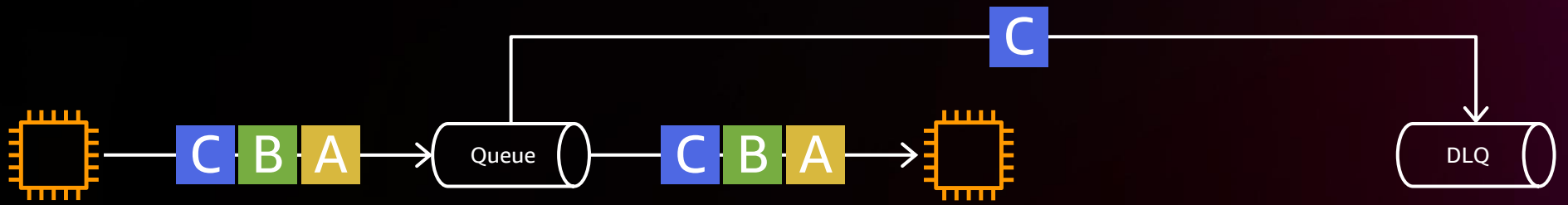
Poison pill handling

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message channels

Dead-letter queue (DLQ)



Producer

Consumers

Transient failure mitigation

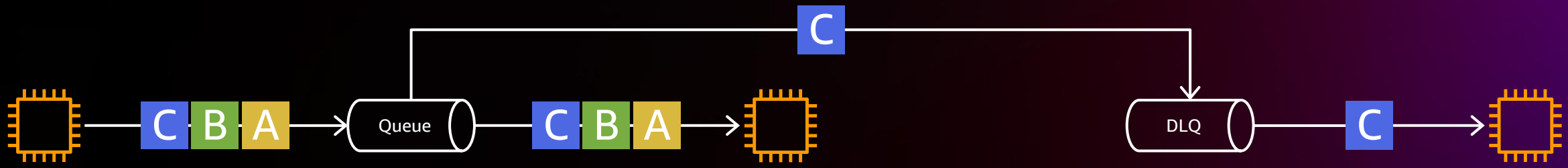
Poison pill handling

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message channels

Dead-letter queue (DLQ)



Producer

Consumers

Application ops

Transient failure mitigation

Poison pill handling

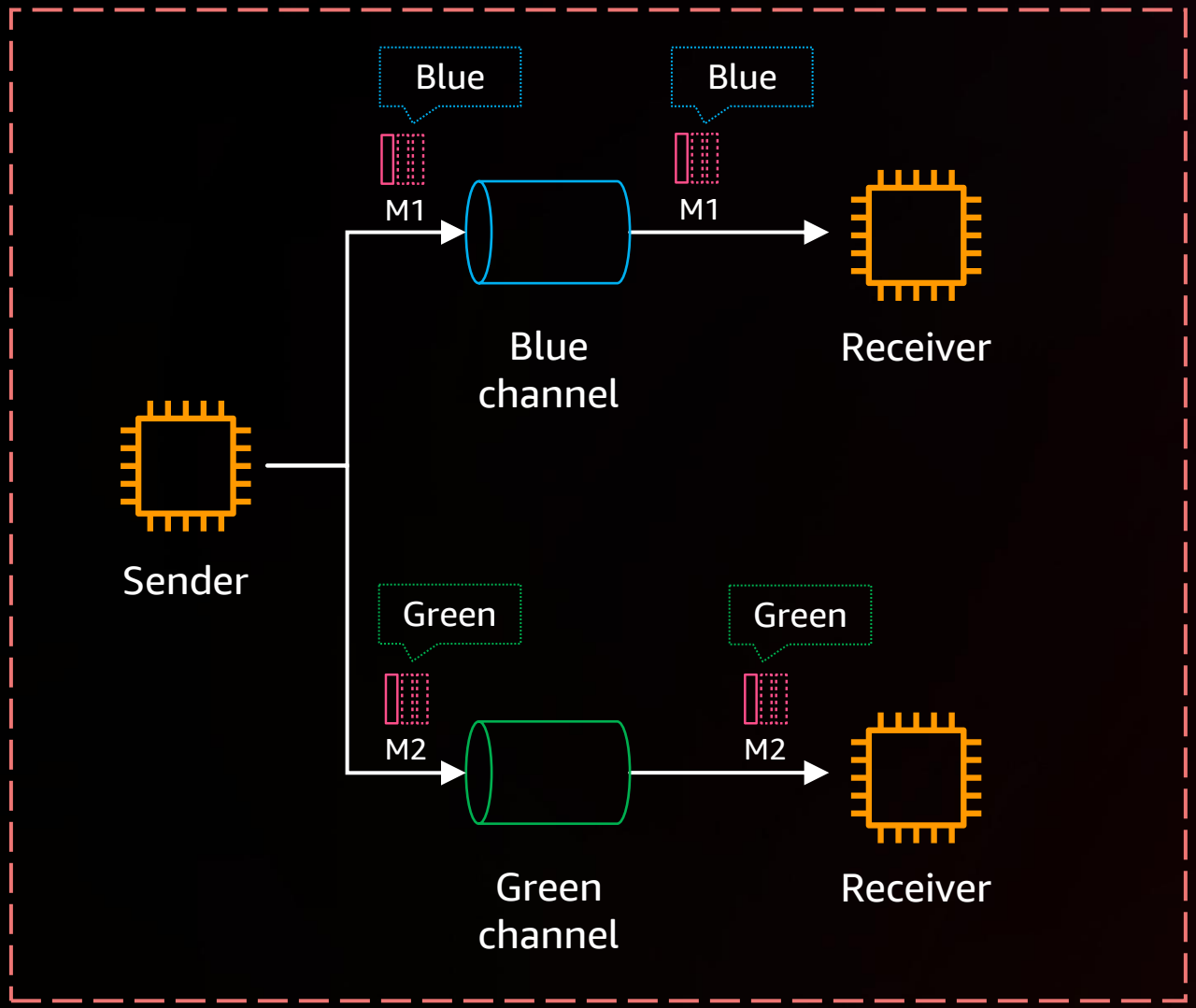
No need for open-heart surgery – investigate without the hassle of production

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Chalking and talking…

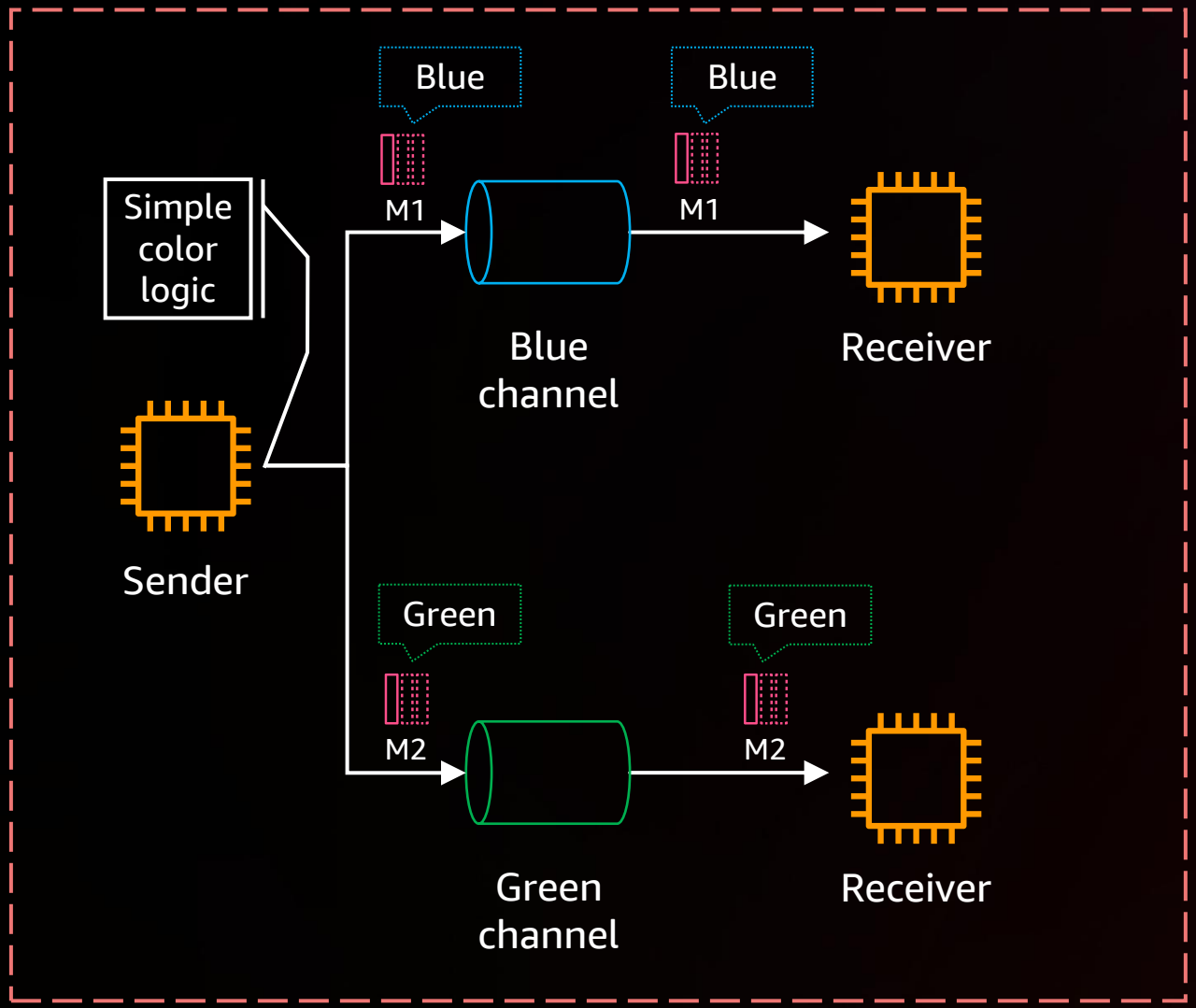
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous point-to-point model (router)



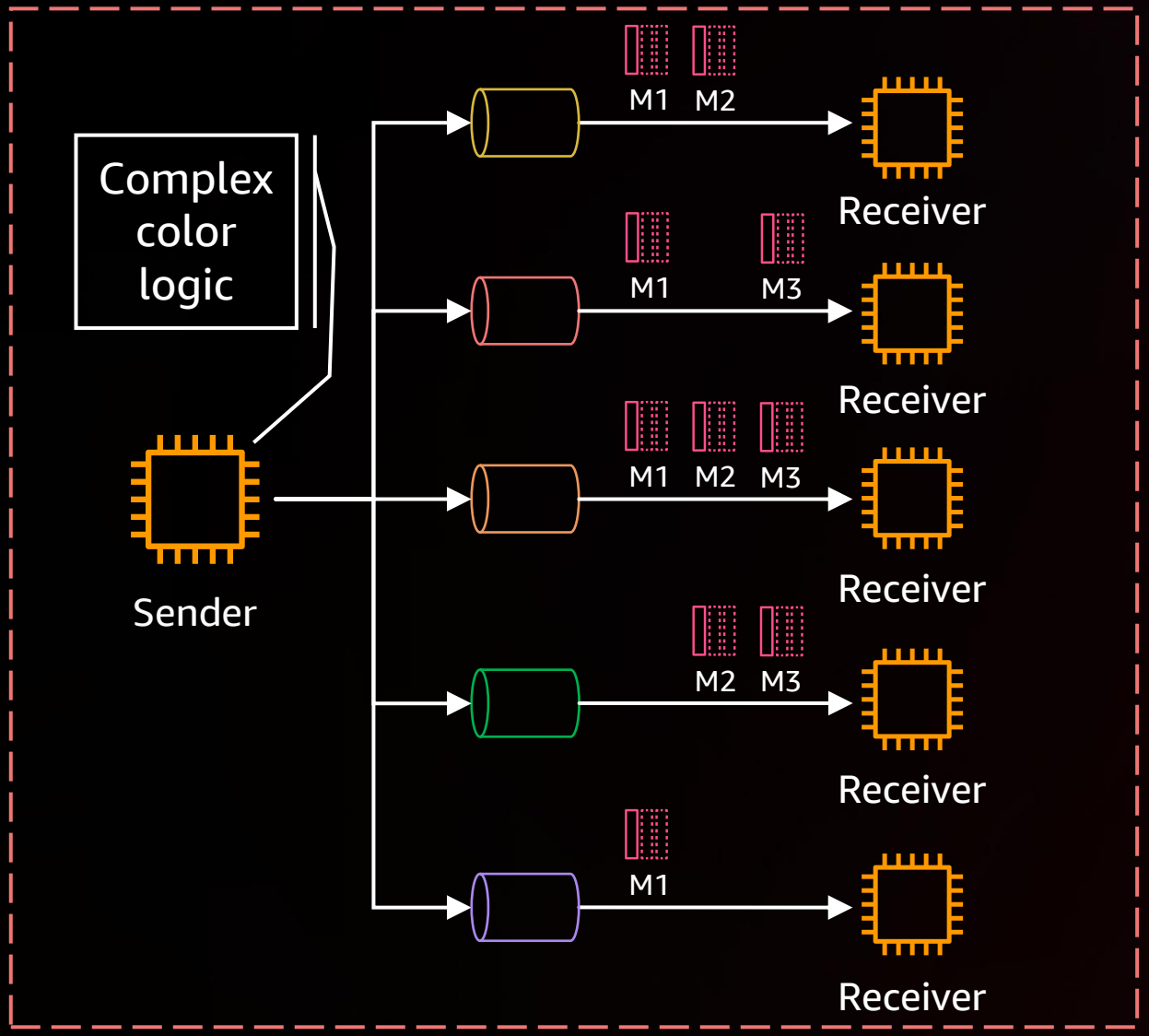
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous point-to-point model (router)



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

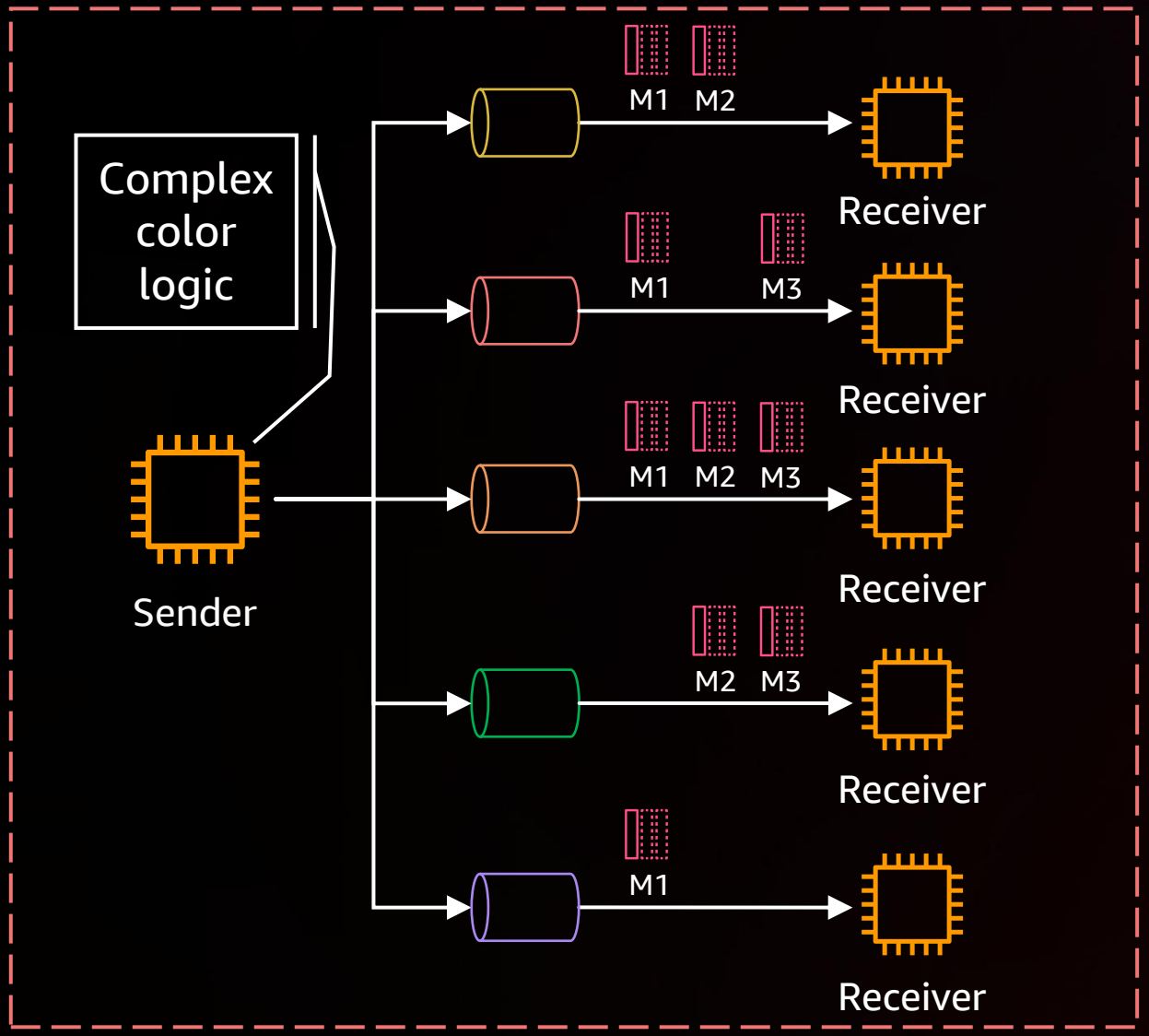
Asynchronous point-to-point model (router)



…Increased complexity, logic, and tighter coupling

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

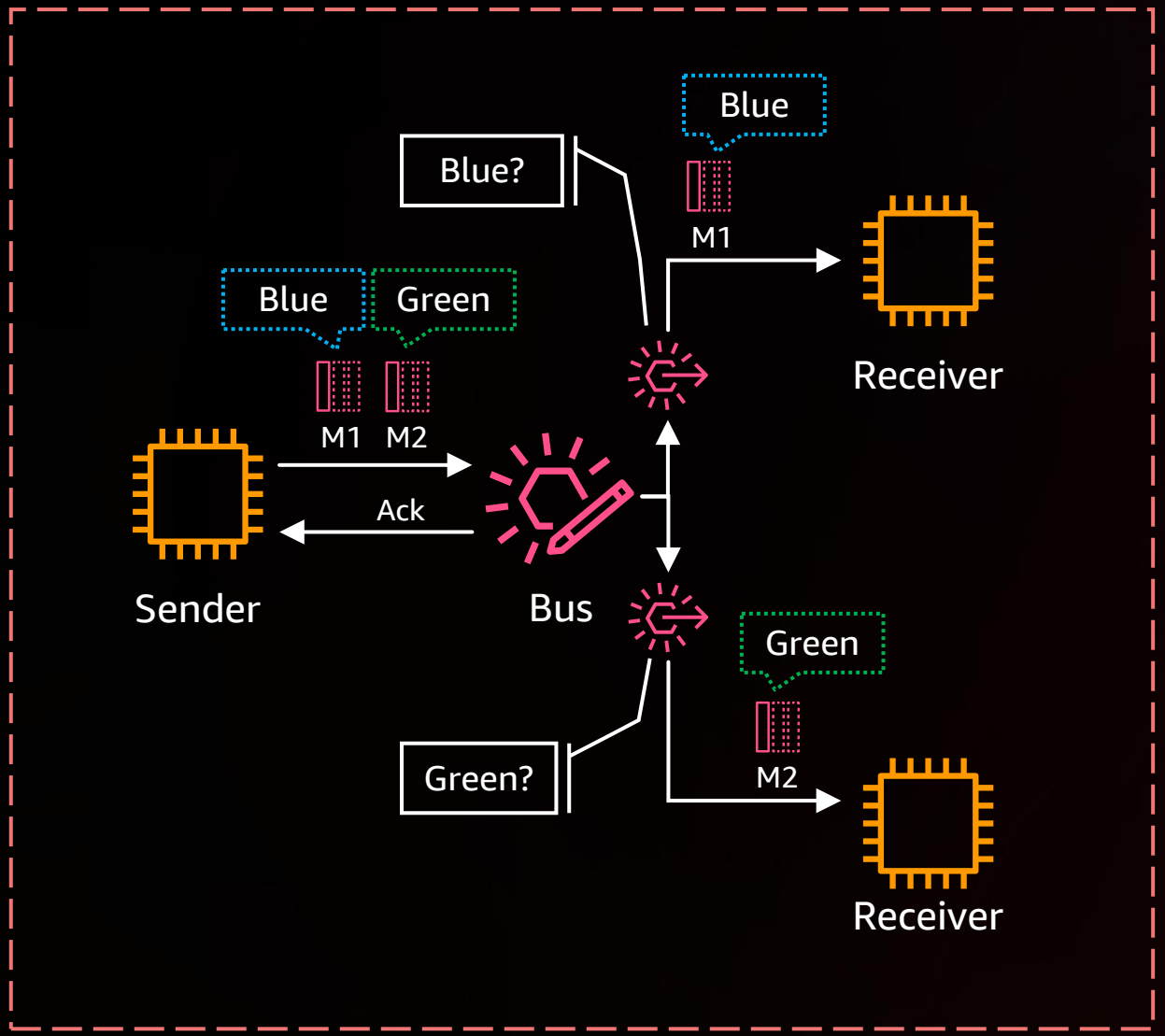
Asynchronous point-to-point model (router)



How can we make this easier to manage?

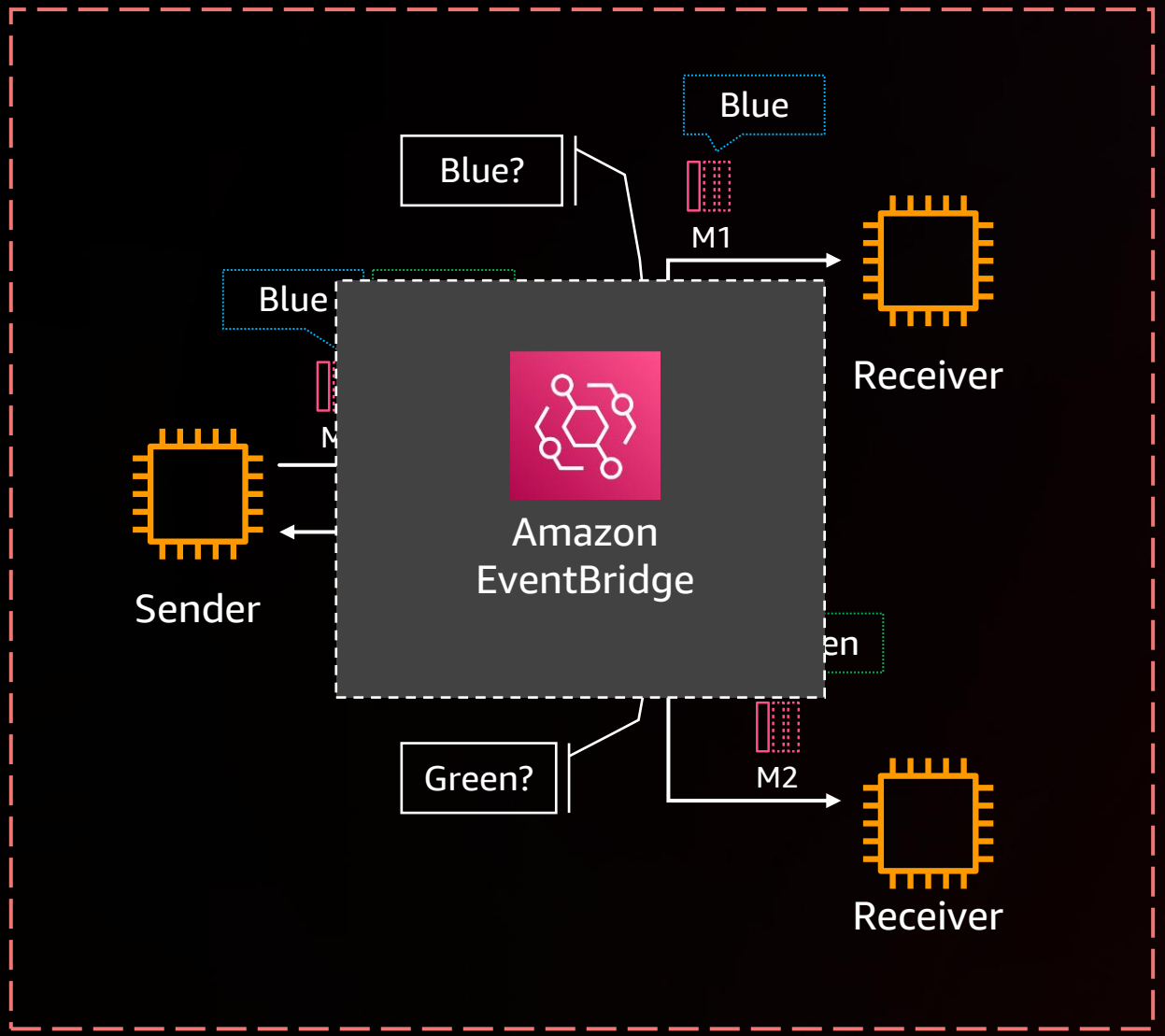
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous message-router (bus)



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous message-router (bus)



Amazon EventBridge is a simple, flexible, fully managed, pay-as-you-go, event bus service that makes it easy to ingest and process data from AWS services, your own applications, and SaaS applications

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Event-driven architecture

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



event

[i-’vent] **noun**

A signal that a system’s state has changed

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Chalking and talking…

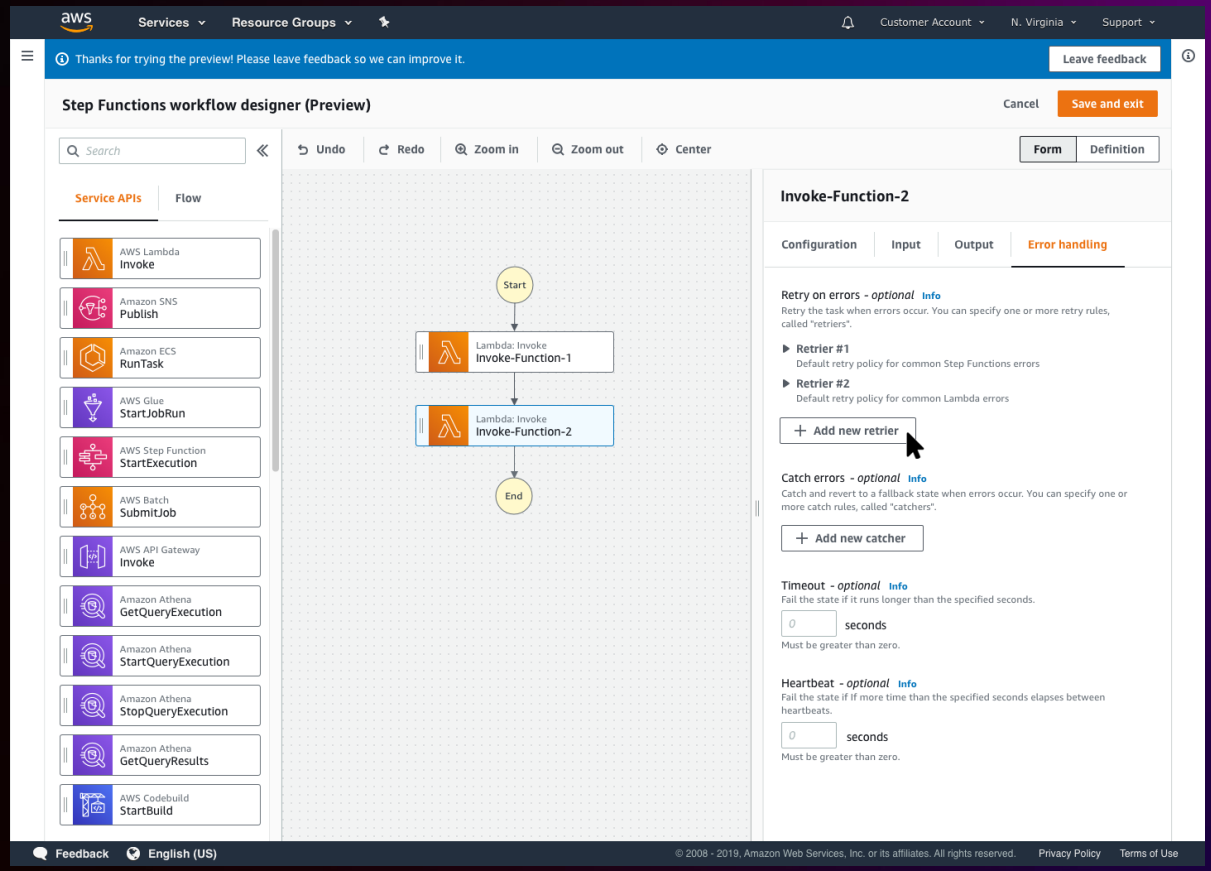
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Step Functions

The workflows you build with Step Functions are called state machines, and each step of your workflow is called a state

When you execute your state machine, each move from one state to the next is called a state transition

You can reuse components, easily edit the sequence of steps, or swap out the code called by task states as your needs change



AWS STEP FUNCTIONS WORKFLOW STUDIO

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Events enable interaction between services

MANAGED SERVICES PROVIDE ROUTING, STORAGE, AND DISTRIBUTION OF EVENTS



Amazon SQS

Messaging

Durable and scalable
Fully managed
Comprehensive security



Amazon EventBridge

Choreography

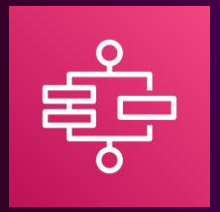
Event filtering
Managed and scalable
SaaS integration



Amazon SNS

Eventing

Performance at scale
Fully managed
Enterprise-ready



AWS Step Functions

Orchestration

Sequencing
Parallel execution
State management

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Orchestration and choreography



Orchestration with Step Functions

- One system controls the flow between components
- Easier to do end-to-end monitoring, timeout, and more
- Centralized business logic



Choreography with EventBridge

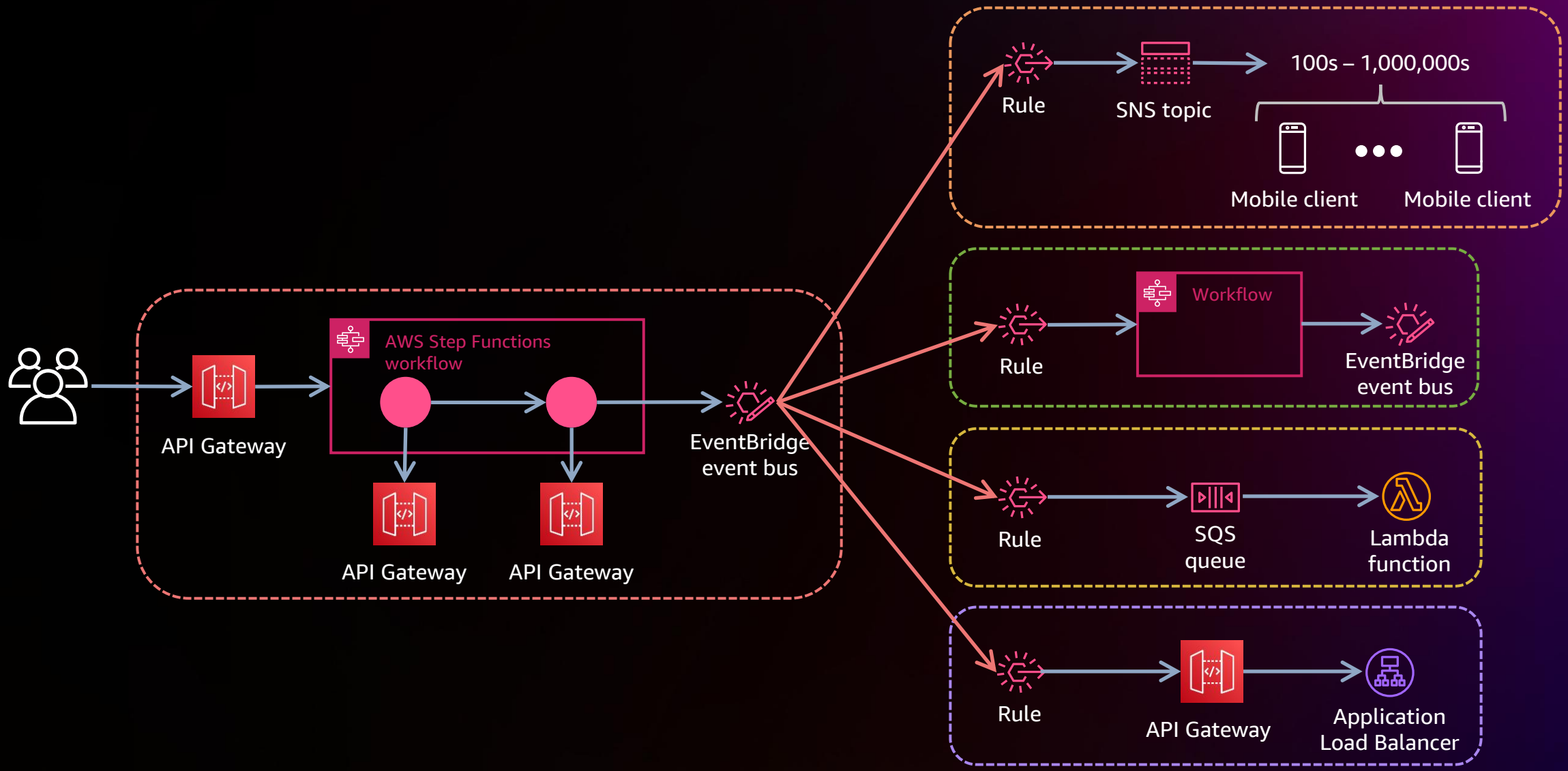
- Pass messages between bounded context of services
- The flow is an emergent property of events being sent
- Easier to extend, modify, and build upon the messages being passed

Source: https://medium.com/theburningmonk-com/choreography-vs-orchestration-in-the-land-of-serverless-8aaf26690889

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Better together: Orchestration + choreography



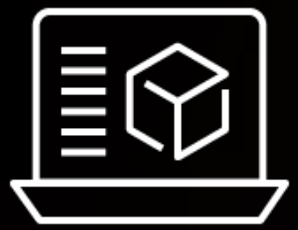
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Choose your pattern and go build!

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Continue your AWS serverless learning

Learn at your own pace



Expand your serverless skills with our Learning Plan on **AWS Skill Builder**

Increase your knowledge



Use our **Ramp-Up Guides** to build your serverless knowledge

Earn AWS serverless badge



Demonstrate your knowledge by achieving **digital badges**

https://s12d.com/serverless-learning

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Related sessions

ID	Title	Type	Schedule
API310	Domain-driven design & event storming	Chalk Talk	Wednesday 10:00 AM
API306	Building event-driven architectures	Workshop	Wednesday 2:30 PM
API303	Application integration patterns for microservices	Breakout	Wednesday 7:45 PM
SVS205-R1	Thinking asynchronously: Integration patterns for microservices [REPEAT]	Chalk Talk	Thursday 11:45 AM
SVS312	Building Serverlesspresso: Creating event-driven architectures	Breakout	Thursday 3:30 PM

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Rebekah Kulidzan
@rkulidzan



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.