

Welcome to Week 5

Cloud Accelerator Program

Event-Driven Architecture

 **Develop**Intelligence

A PLURALSIGHT COMPANY

Hello

HELLO
my name is

Allen Sanders
with DevelopIntelligence,
a Pluralsight Company.

About me...



- 26+ years in the industry
- 21+ years in teaching
- Certified Cloud architect
- Passionate about learning
- Also, passionate about Reese's Cups!



Why study these subjects?

In modern software engineering, our ability to quickly deploy incremental innovation, ensure its quality, and scale to meet customer demand proves critical to our success

- Cloud is everywhere and it's not going away
- As with many topics in technology, there are multiple options and multiple dimensions to those options
- Building a deeper understanding of Cloud and its offerings helps prepare you for modern IT
- Leveraging event-based integrations (vs. strict point-to-point) can help us design systems that are loosely-coupled, resilient, and highly-performant



My pledge to you

I will...

- Make this interactive
- Ask you questions
- Ensure everyone can speak
- Use an on-screen timer



Agenda

- Event-Driven Architecture – What it is, the problem it solves, and potential “gotchas”
- Common Event-Driven Architecture patterns
- Available services in AWS for building event-based applications



How we're going to work together

- Slides and words to highlight key concepts
- Demos to bring those concepts “to life”
- Lab work (which will take place in sandboxes provided by “A Cloud Guru”) for hands-on reinforcement
- NOTE: I welcome being interrupted – if you need more info, or clarification, or anything else, just break in and ask. I am here to help you.

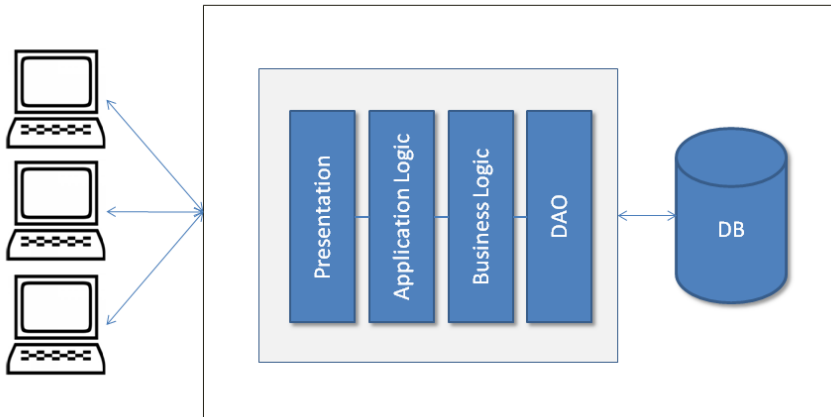


Event-Driven Architecture

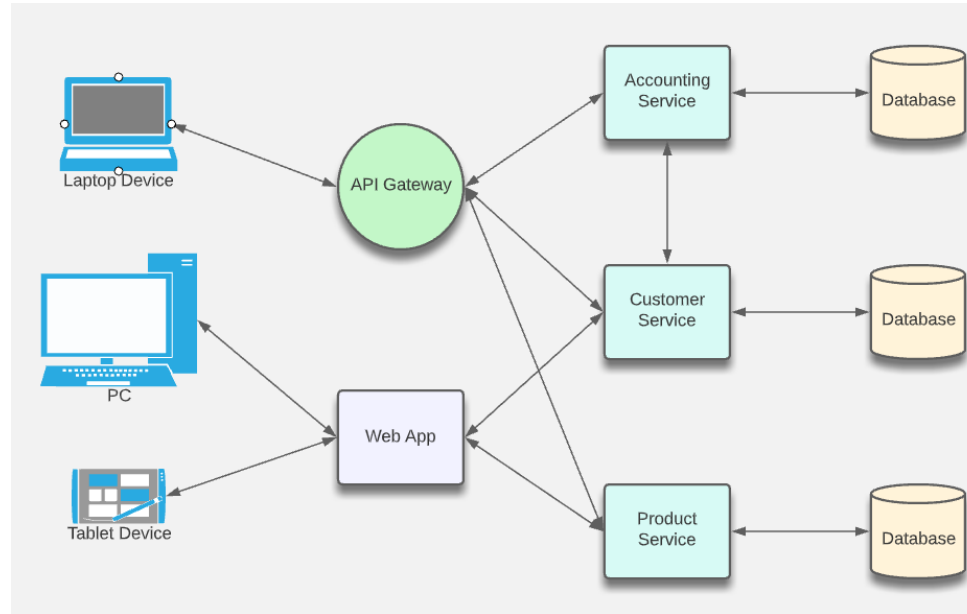
Microservices Architecture

What is a Monolith?

- Typical enterprise application
- Large codebases
- Set technology stack
- Highly coupled elements
- Whole system affected by failures
- Scaling requires the duplication of the entire app
- Minor changes often require full rebuild



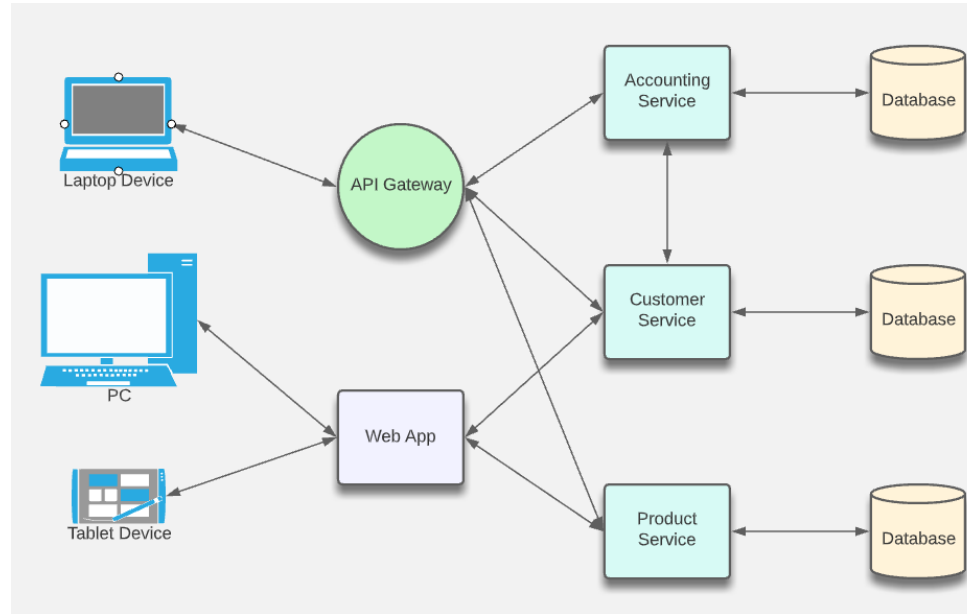
Microservices Architecture



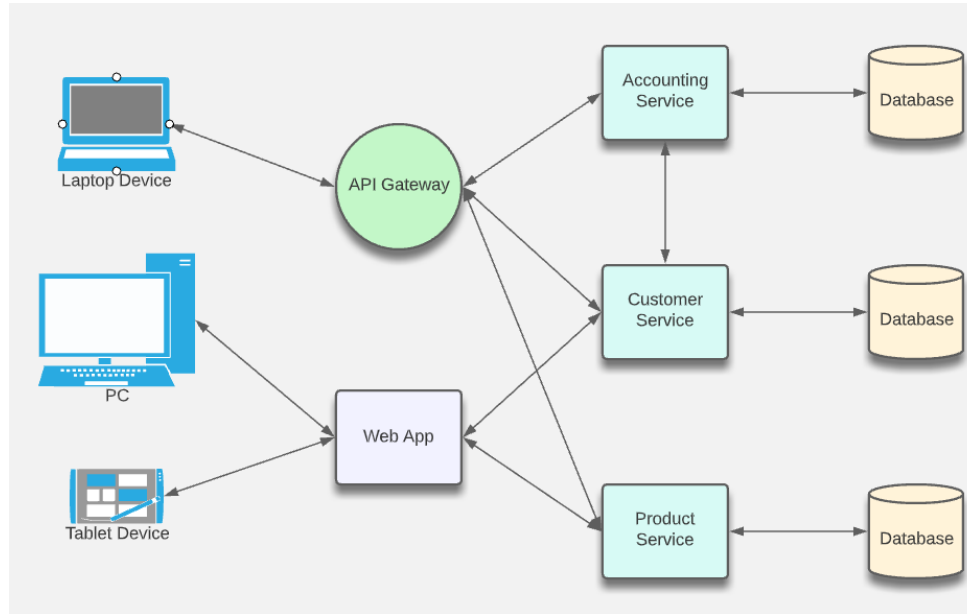
Microservices Architecture

Benefits

- Enables work in parallel
- Promotes organization according to business domain
- Advantages from isolation
- Flexible in terms of deployment and scale
- Flexible in terms of technology
- Allows multiple agile teams to maximize autonomy in effort and technology



Microservices Architecture

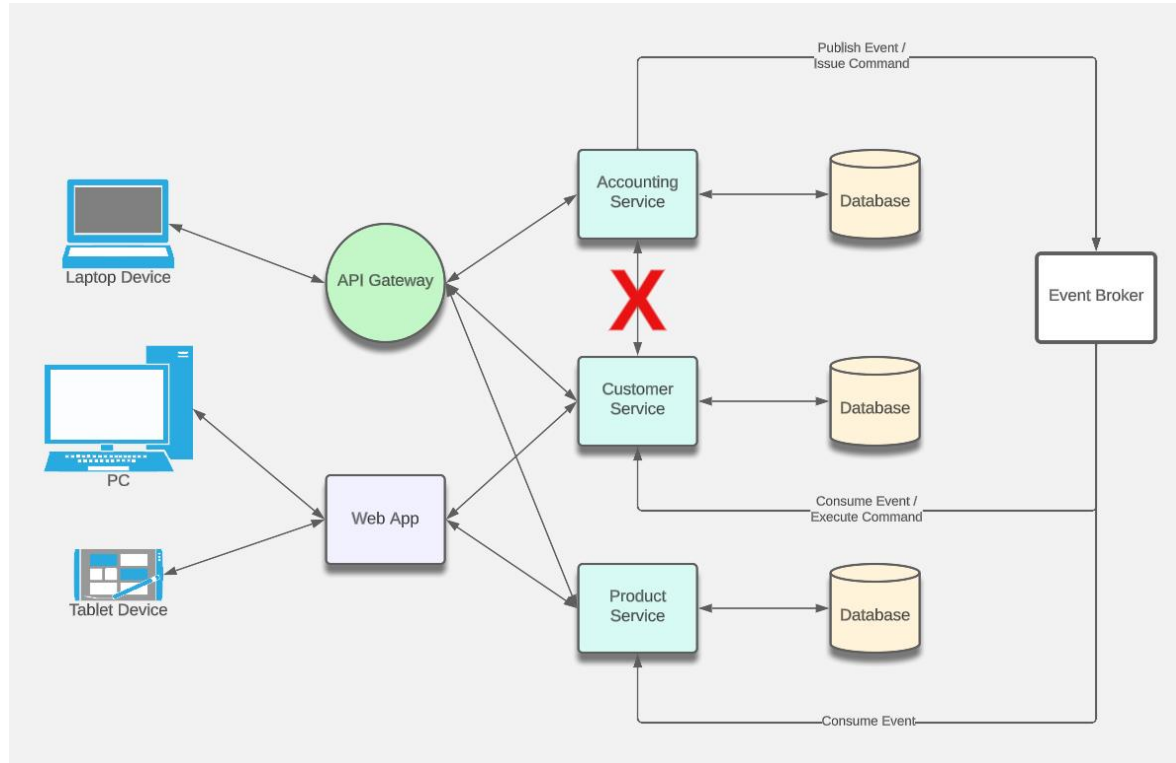


Implications

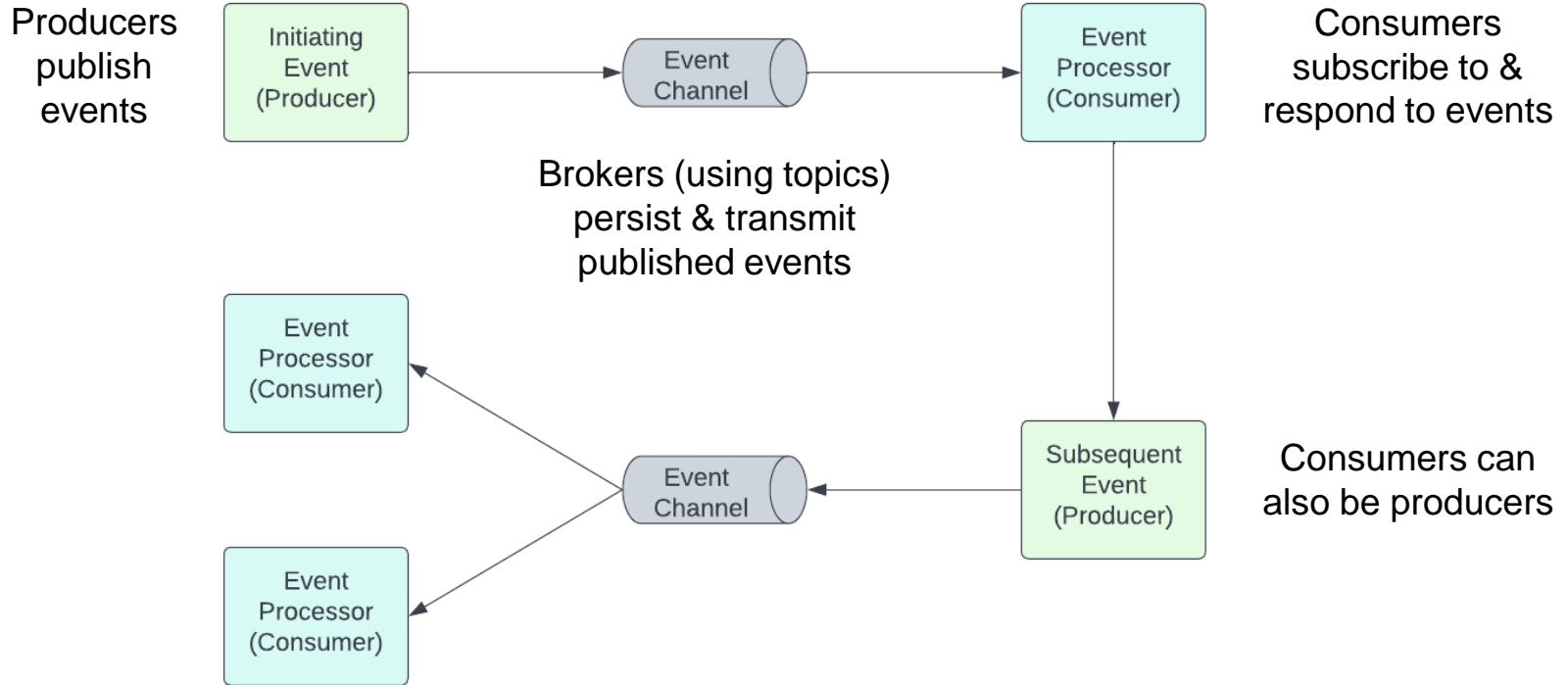
- Requires a different way of thinking
- Complexity moves to the integration layer
- Organization needs to be able to support re-org according to business domain (instead of technology domain)
- With an increased reliance on the network, you may encounter latency and failures at the network layer
- Transactions must be handled differently (across service boundaries)

Event-Driven Architecture (EDA)

Microservices Architecture Using EDA



Event-Driven Architecture



Domain Events vs. Commands (or Messages)

Domain Events

- Notification that something of business significance happened
- Can range from 0 or more processes/components (consumers) that “care”

Commands

- Explicit message intended for a specific target component
- Used to initiate a new step in a flow
- Can be combined with domain events



Event-Driven Architectures

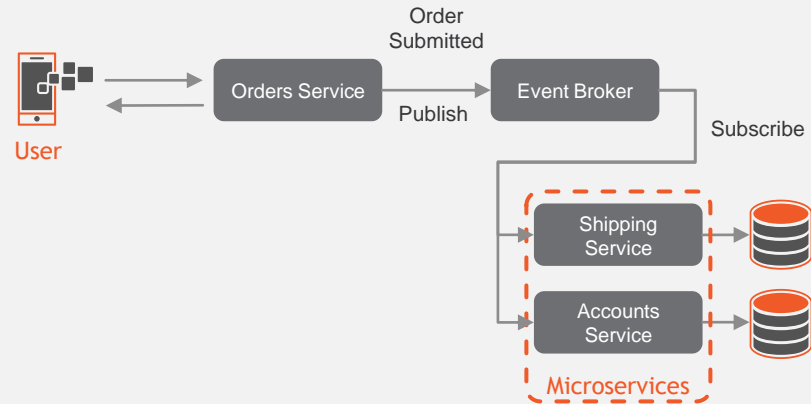


As an architectural style:

- Enables looser coupling between communicating services
- Provides dynamic flexibility in the definition of integrated workflows
- Promotes resiliency as an intermediate component (the broker) is used to separately track and manage the queue events
- Uses concept of domain events – publish of events with business significance



Simplified example



Event-Driven Architectures



Benefits:

- Promotes looser coupling in cases where synchronous calls are not required
- Allows multiple services the option of being notified (vs. point-to-point)
- If given consumer is down, producer can continue to send messages and they won't be lost

Potential “gotchas”:

- Additional complexity
- Can be harder to trace an action end-to-end (but there are ways to handle)
- Not a good fit if specific timing and sequencing required
- Use case might need to tolerate “eventual consistency”

Common EDA Patterns



Event-Driven Architectures

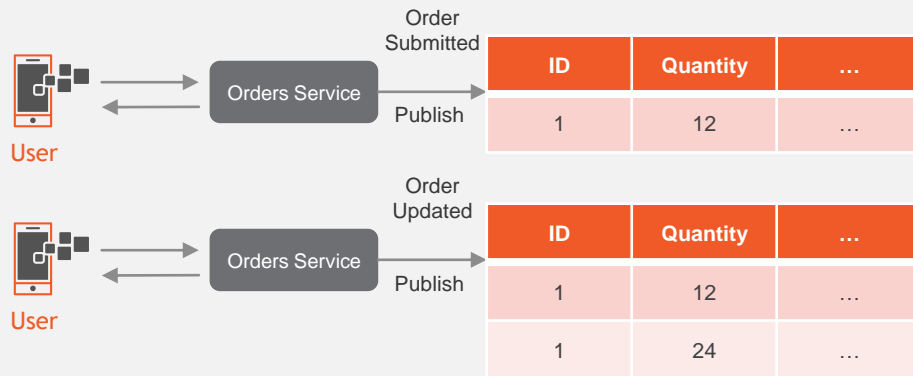


Event Sourcing

- Data changes are captured as immutable events
- When a change is made, rather than overwrite existing record, a new record is created to reflect updated state
- Provides a full picture of what happened during an entity's lifecycle – no data destruction
- Can make reads more difficult



Simplified example





Event-Driven Architectures

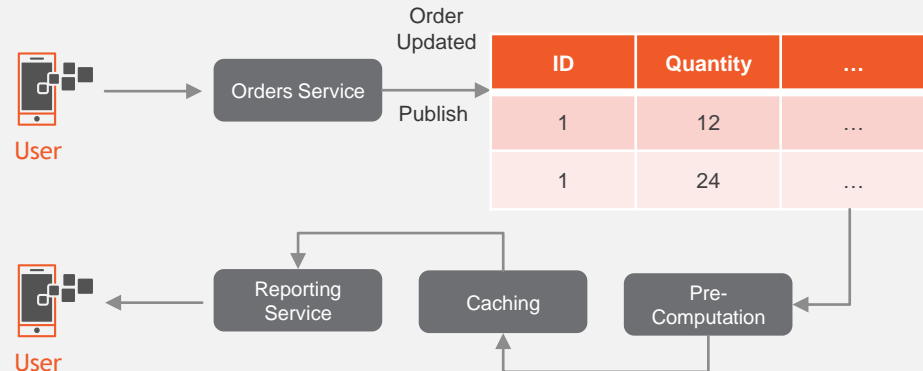


CQRS

- Command-Query Responsibility Segregation
- Can help address challenges with Event Sourcing architectures
- Separate processes used for writes and reads
- Allows write optimization (“raw” events) and read optimization (dedicated aggregation)
- Also, enables denormalization and reformatting for reads separate from write structures



Simplified example

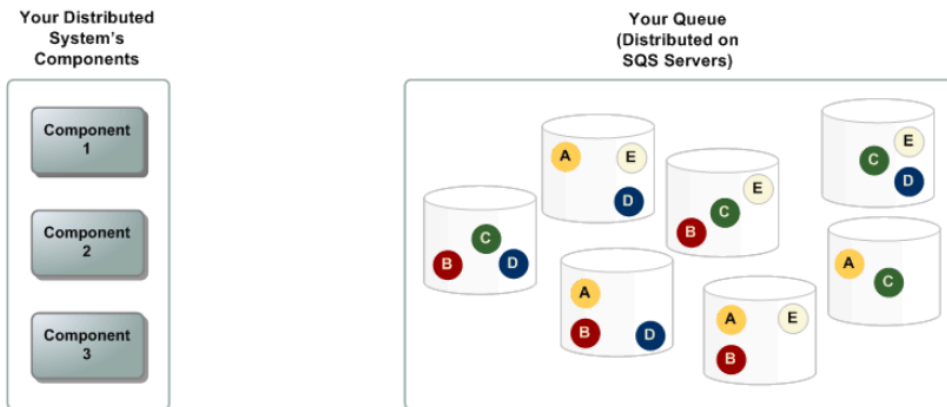


AWS Services for EDA

Simple Queue Service (SQS)

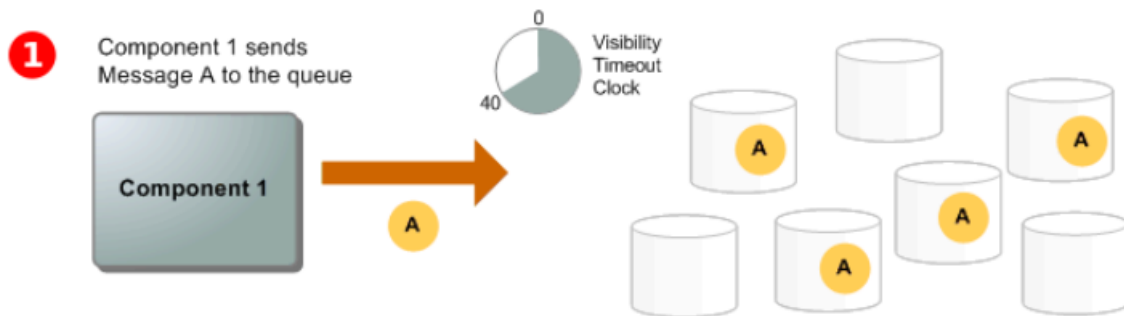
Simple Queue Service (SQS)

SQS provides a hosted and managed queue for communicating across distributed application components (e.g., microservice)



Source: <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-basic-architecture.html>

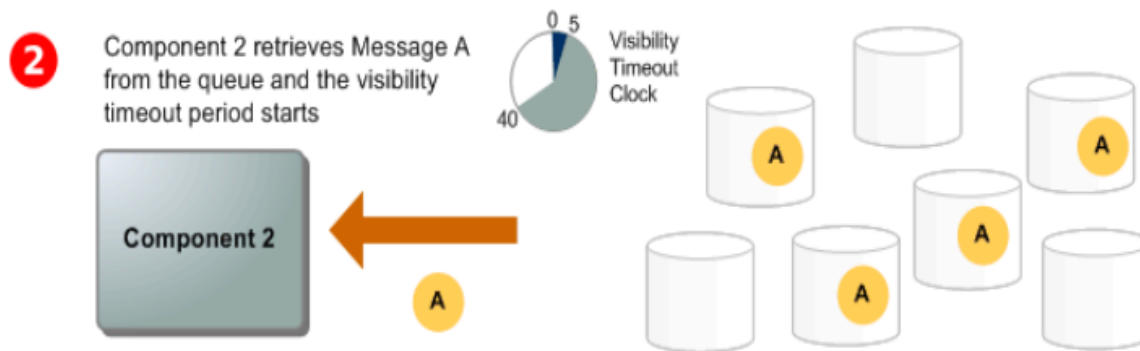
Simple Queue Service (SQS) – Message Lifecycle



- 1 A producer (component 1) sends message A to a queue, and the message is distributed across the Amazon SQS servers redundantly.

Source: <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-basic-architecture.html>

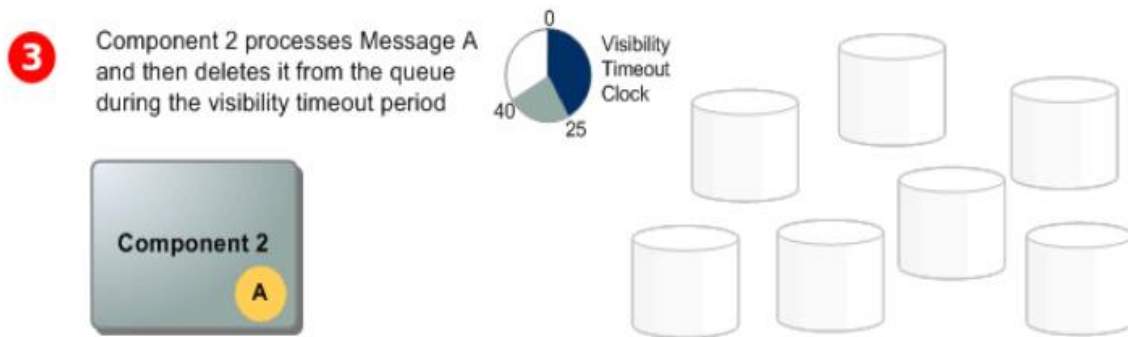
Simple Queue Service (SQS) – Message Lifecycle



- 2** When a consumer (component 2) is ready to process messages, it consumes messages from the queue, and message A is returned. While message A is being processed, it remains in the queue and isn't returned to subsequent receive requests for the duration of the **visibility timeout**.

Source: <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-basic-architecture.html>

Simple Queue Service (SQS) – Message Lifecycle



- 3** The consumer (component 2) deletes message A from the queue to prevent the message from being received and processed again when the visibility timeout expires.

Source: <https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-basic-architecture.html>

LAB:

SQS – Triggering Lambda

Execute the tutorial available at

<https://learn.acloud.guru/handson/c11ca4ba-f942-4a5b-9c75-ad3ff8134a97>

LAB:

SQS – Standard Queues

Execute the tutorial available at

<https://learn.acloud.guru/handson/fdf9217d-1da0-4530-b34b-cd07d96a9194>

LAB:

SQS – FIFO Queues

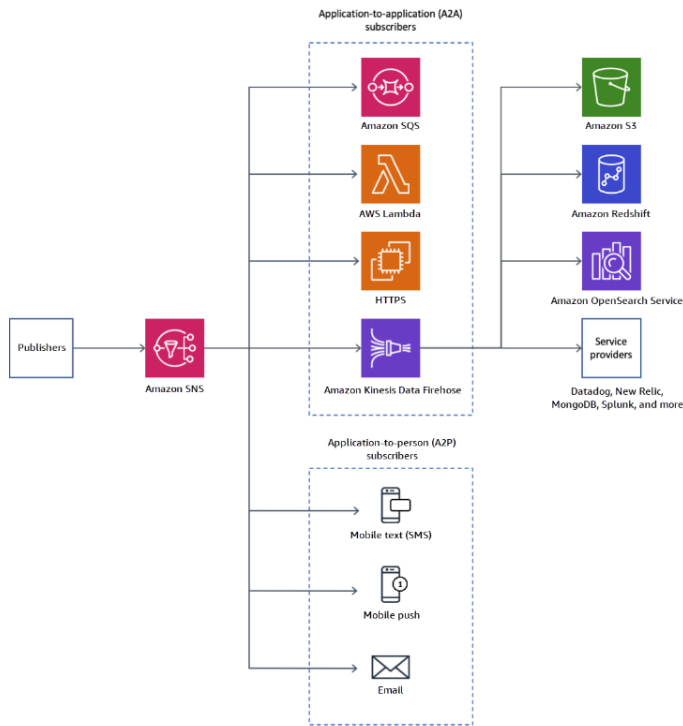
Execute the tutorial available at

<https://learn.acloud.guru/handson/262a58ab-49e1-4102-b2a2-b746c06ba74d>

Simple Notification Service (SNS)

Simple Notification Service (SNS)

SNS provides a managed solution for message delivery using a pub/sub (producer/consumer) model for delivery of important application notifications/messages



LAB:

SNS – Create/Subscribe to Topic

Execute the tutorial available at

<https://learn.acloud.guru/handson/cdfba172-1b7c-4762-a14e-e469ffbe5e54>

LAB:

SNS – Triggering a Lambda

Execute the tutorial available at

<https://learn.acloud.guru/handson/fd65ed62-a3c3-4d6f-8aa0-9051aa038d61>

LAB (If Time Permits):

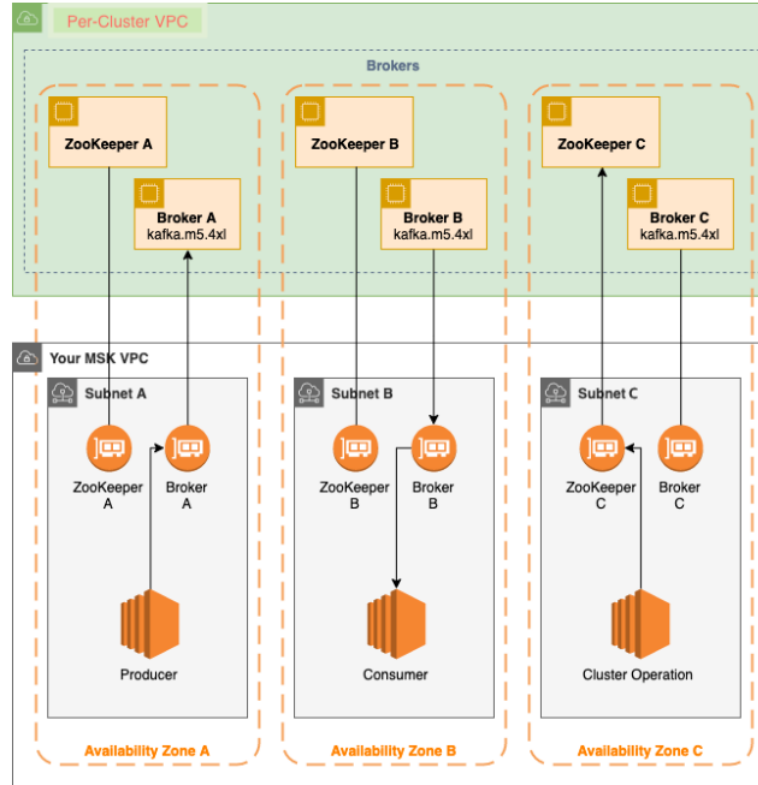
EventBridge

Execute the tutorial available at

<https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-s3-object-created-tutorial.html>

Managed Kafka (MSK)

MSK – Managed Streaming for Apache Kafka



Source: <https://docs.aws.amazon.com/msk/latest/developerguide/what-is-msk.html>

LAB:

MSK – End-to-End

Execute the tutorial available at

<https://docs.aws.amazon.com/msk/latest/developerguide/getting-started.html>



Thank you!

If you have additional questions,
please reach out to me at:
asanders@gamuttechnologysvcs.com