COMP5349– Cloud Computing Week 9: CodePipeline & Decoupled Architecture

Dr. Ying Zhou
The University of Sydney

Table of Contents

DevOps and CI/CD

DevOps

- DevOps is a set of practices, cultural philosophies, and tools that aim to integrate and automate the processes between software development (Dev) and IT operations (Ops).
- The ultimate goal is to enable faster and more reliable software delivery with improved collaboration and communication between teams.

DevOps Culture

- Increased transparency, communication and collaboration between teams
- Shared responsibilities
- Autonomous teams
- Fast Feedback
- Automation

https://www.atlassian.com/devops/what-is-devops/devops-culture

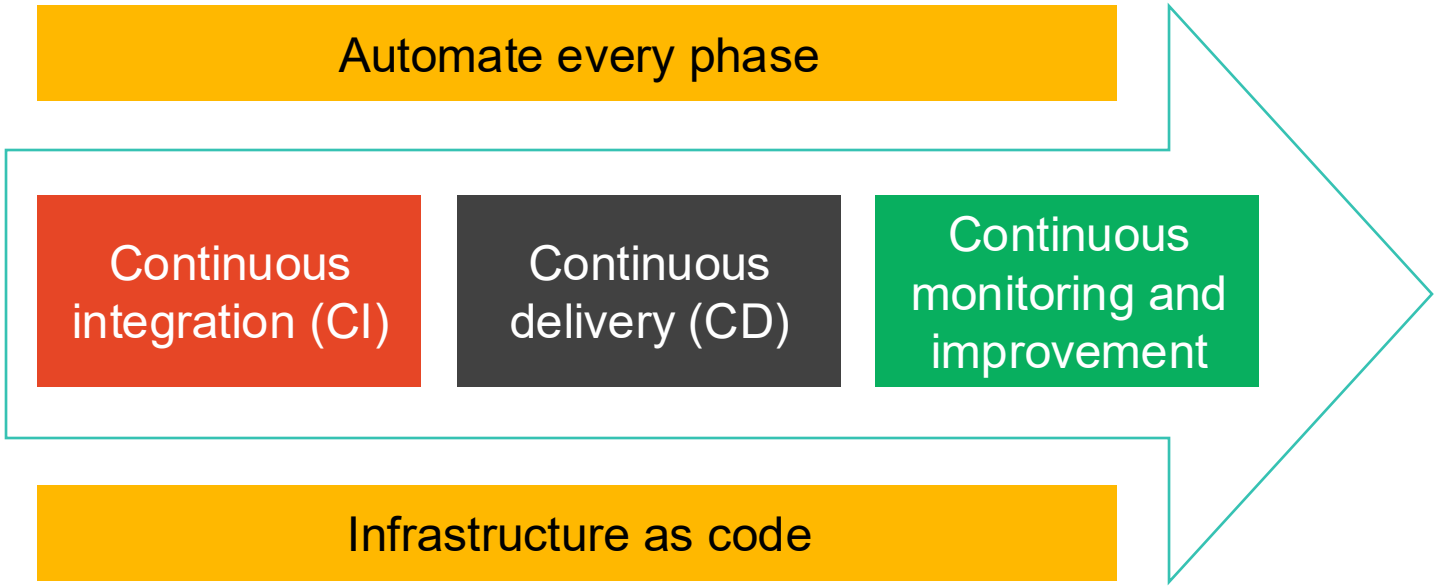
DevOps Best Practice

- Agile Project Management
- Shift left with CI/CD
- Build with the right tools
  - DevOps life cycle: discover, plan, build, test, monitor, operate, continuous feedback
- Implement automation
- Monitor the DevOps pipeline and applications
- Observability
  - three pillars of observability are **logs**, **traces**, and **metrics**.
- Change the culture
  - You build it you run it

https://www.atlassian.com/devops/what-is-devops/devops-best-practices

AWS Supported DevOps practices

- Microservice architecture
- Continuous integration and continuous delivery (CI/CD)
- Continuous monitoring and improvement
- Automation focused
- Infrastructure as code



DevOps tools

CI/CD

- AWS CodePipeline
- AWS CodeBuild
- AWS CodeDeploy
- AWS CodeConnections

Microservices

- Amazon Elastic Container Service (Amazon ECS)
- AWS Lambda
- AWS Fargate

Platform as a Service

- AWS Elastic Beanstalk

Infrastructure as Code

- AWS CloudFormation
- AWS OpsWorks
- AWS Systems Manager

Monitoring and logging

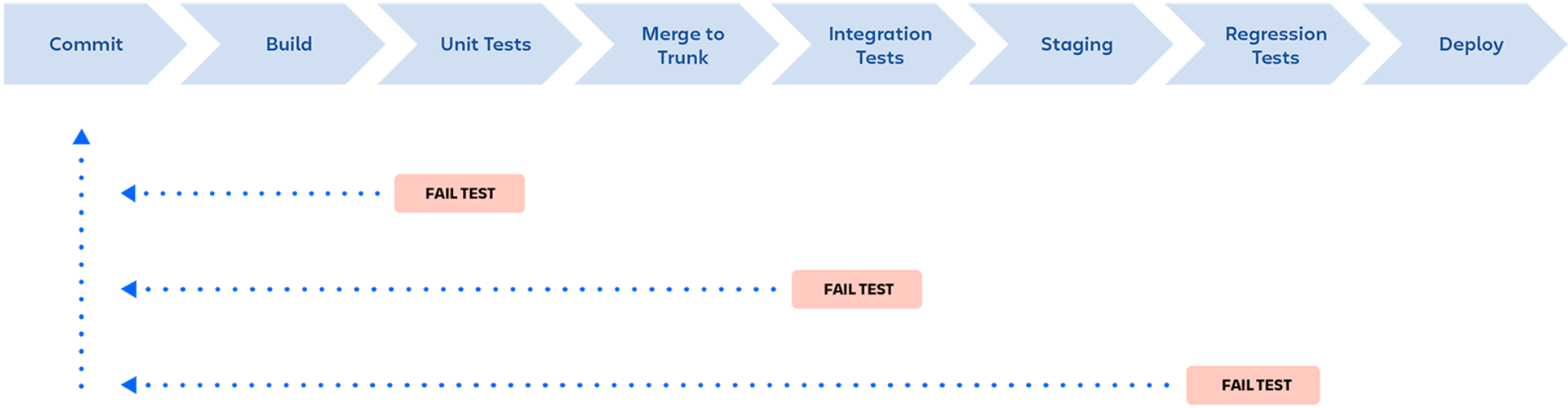
- Amazon CloudWatch
- AWS CloudTrail
- AWS X-Ray
- AWS Config

Continuous Integration

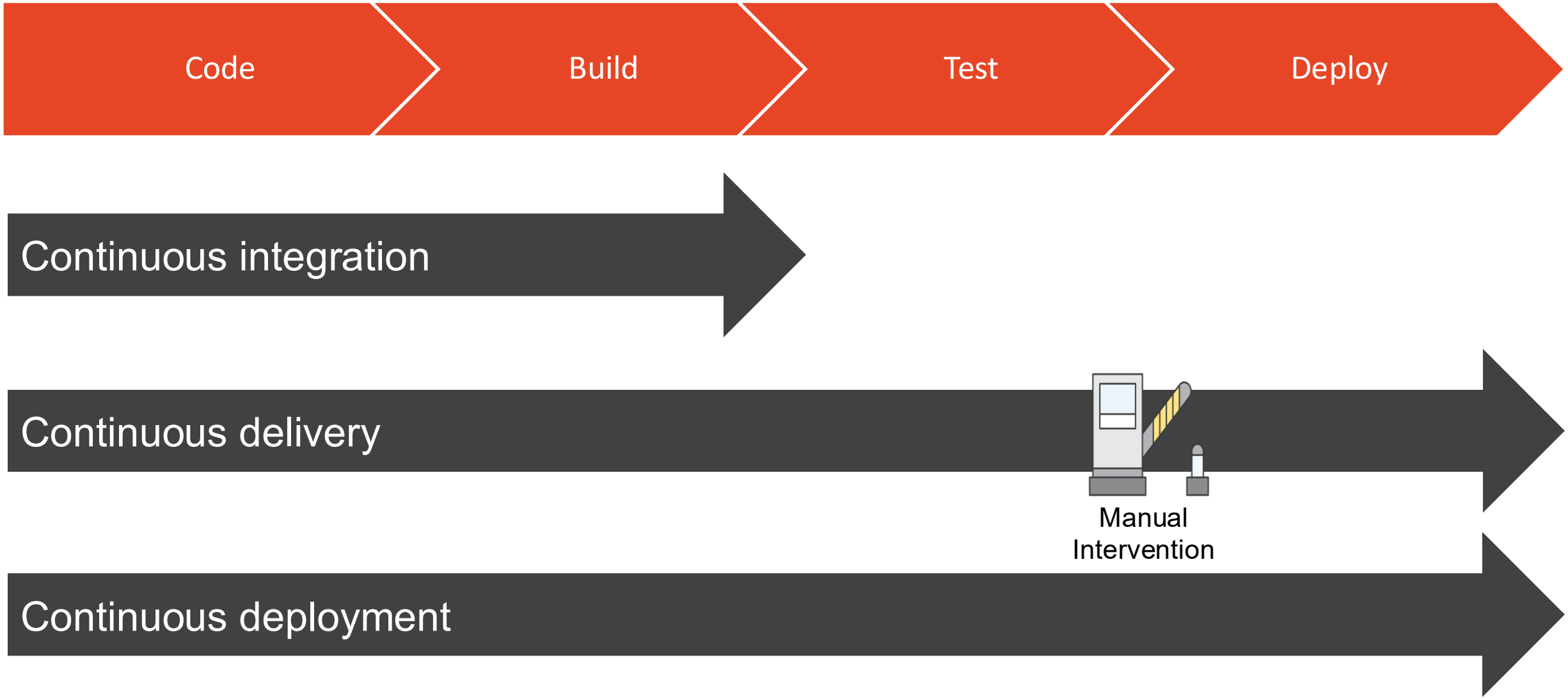
- A software development practice where members of a team use a **version control system** and frequently integrate their work to the same location, such as a main branch. Each change is built and verified to detect integration errors as quickly as possible. Continuous integration is focused on automatically building and testing code
  - Version control system to manage the code change
  - A CI server to manage the automated process of building, testing, and validating those changes

Continuously Delivery/Continuously Deployment

- Continuous Delivery extends CI by including automatically releasing software to a repository.
- Continuous Deployment extends the process further by adding the step of automatically deploying software to production.



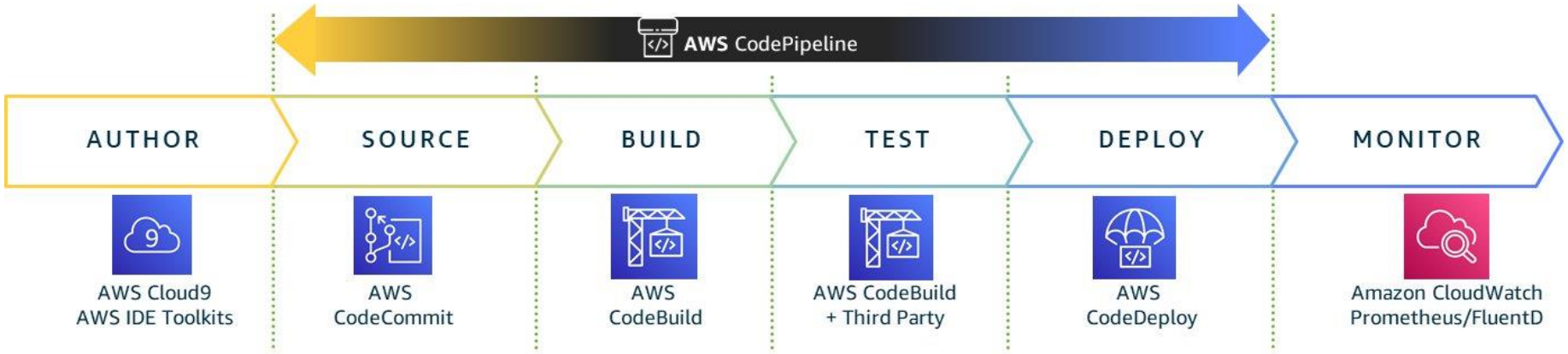
Understanding CI/CD



AWS CodePipeline

CI/CD on AWS

- CI/CD is usually pictured as a pipeline with each stage structured as a logic unit in the delivery process
  - Specialized tool for each stage
  - Manual checking can be added at various points
  - Some stages maybe skipped for certain pipelines

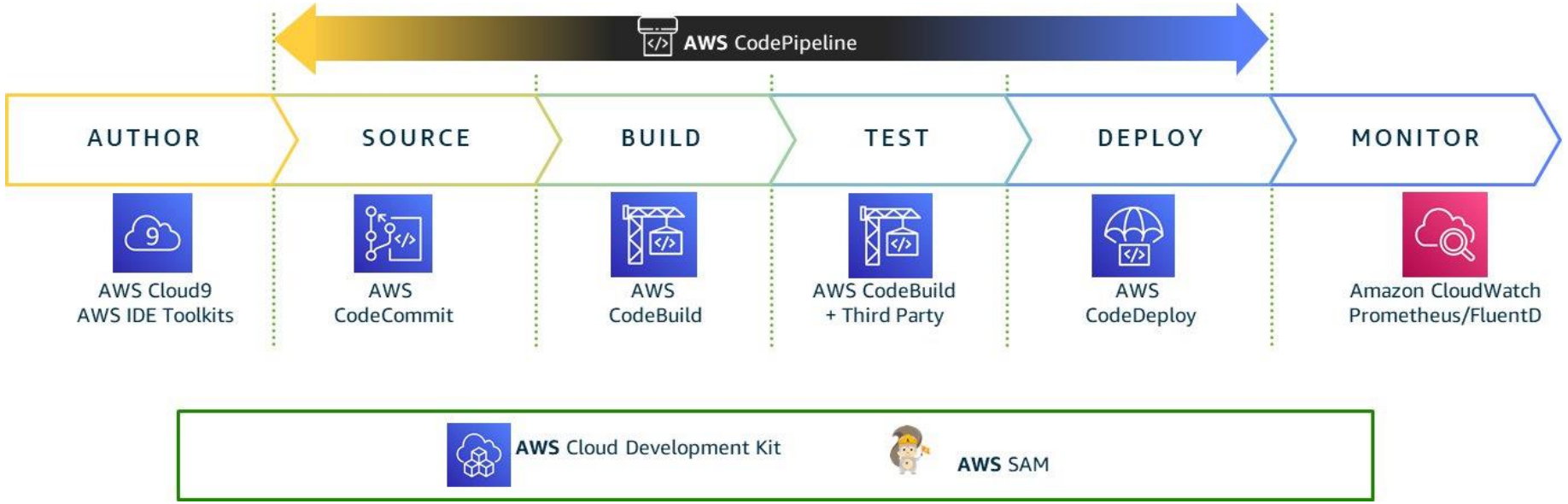


AWS CI/CD pipeline components

- AWS can set up CI/CD pipelines using various development tools
  - AWS CodeCommit
  - AWS CodeBuild
  - AWS CodePipeline
  - AWS Code Deploy
- The pipeline can be used to manage application code and infrastructure code
  - Pipelines for infrastructure code may not need to include all stages
  - The pipelines in the lab contains only two stages

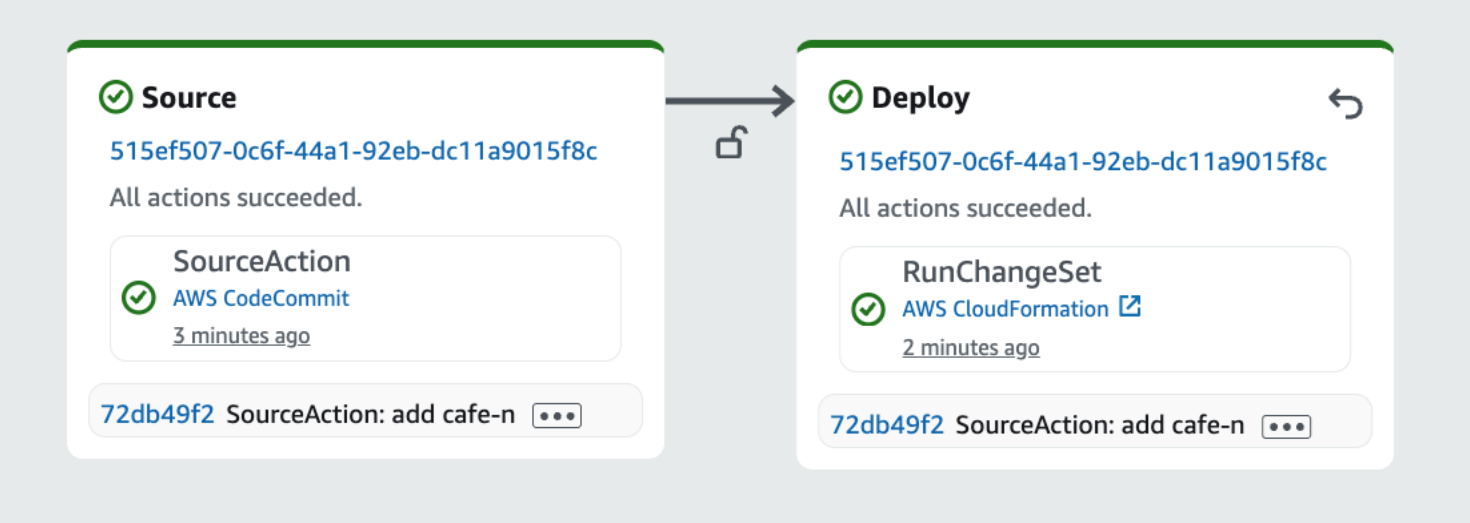
CodePipeline

- A service that allows users to automate steps required to release/deploy software on AWS easily



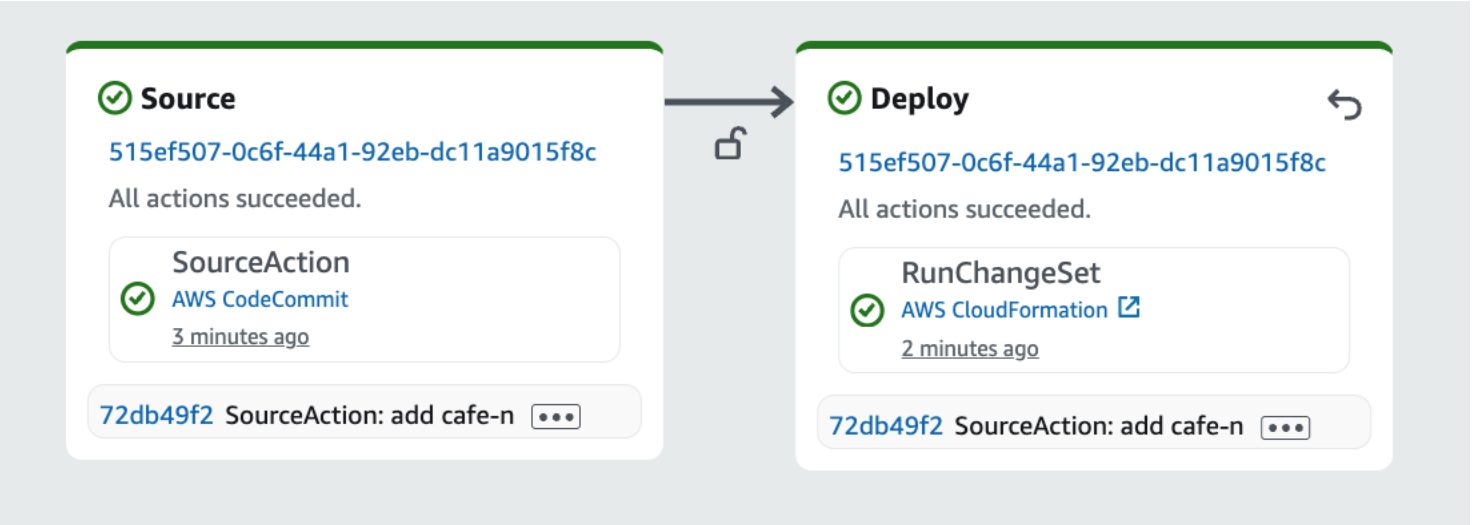
Pipeline

- A *pipeline* is a workflow construct that describes how software changes go through a release process.
- Each pipeline is made up of several stages
  - The simplest pipeline may contain only two stages



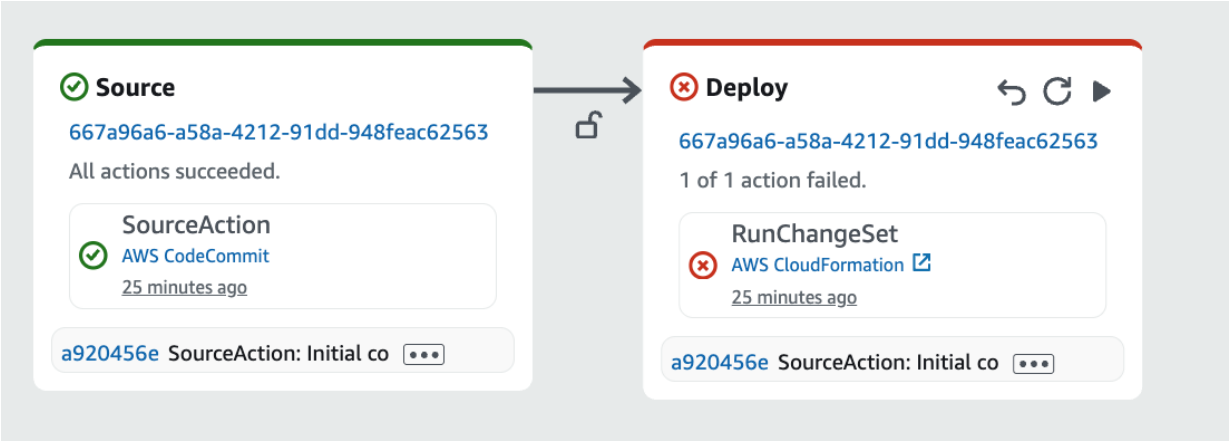
Stages

- Stage is a logic unit representing a relatively independent step in the process where actions can be performed on the artifact, such as source code
- There are stages correspond to typical software development stages
  - Build, test etc



Pipeline execution

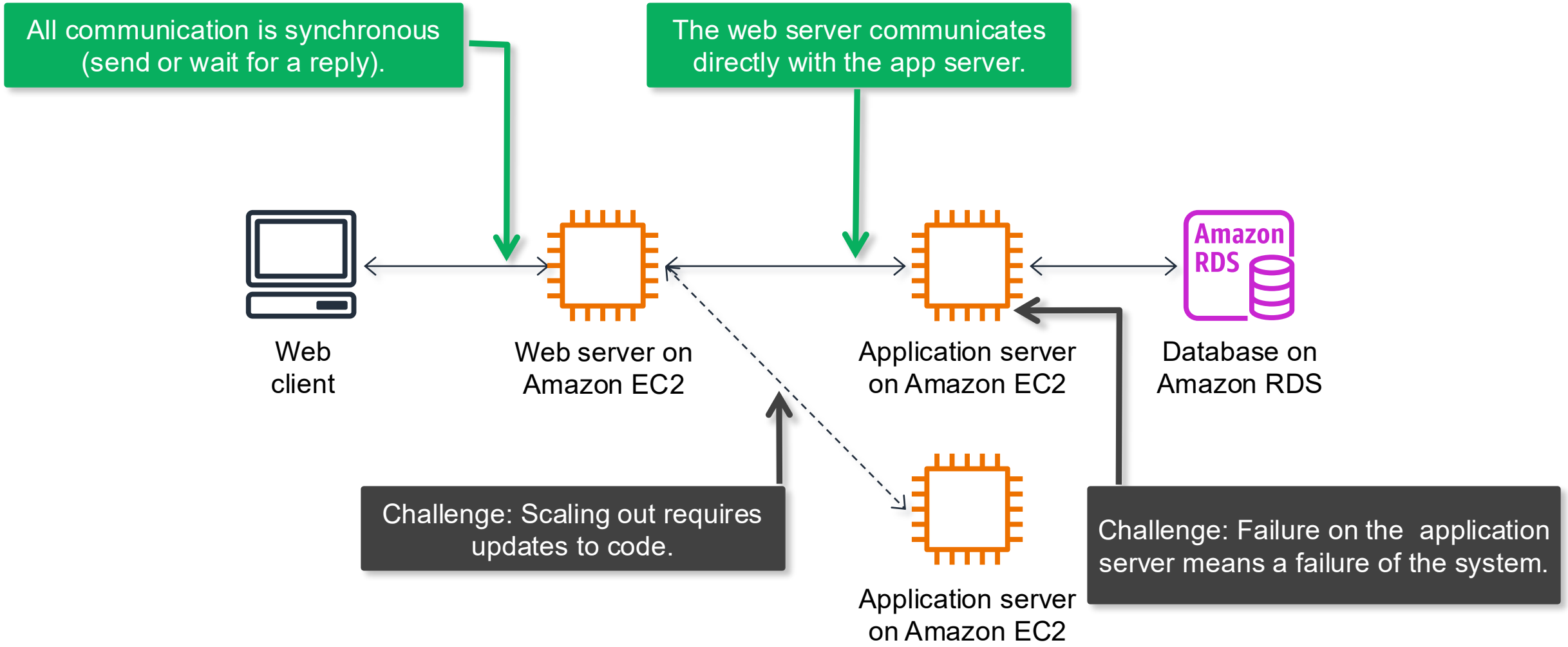
- Traversing the stages in order and complete actions defined in the stages, which may end up with a successful execution or failed execution
- Pipeline can be configured to automatically start by triggers such as commit event.



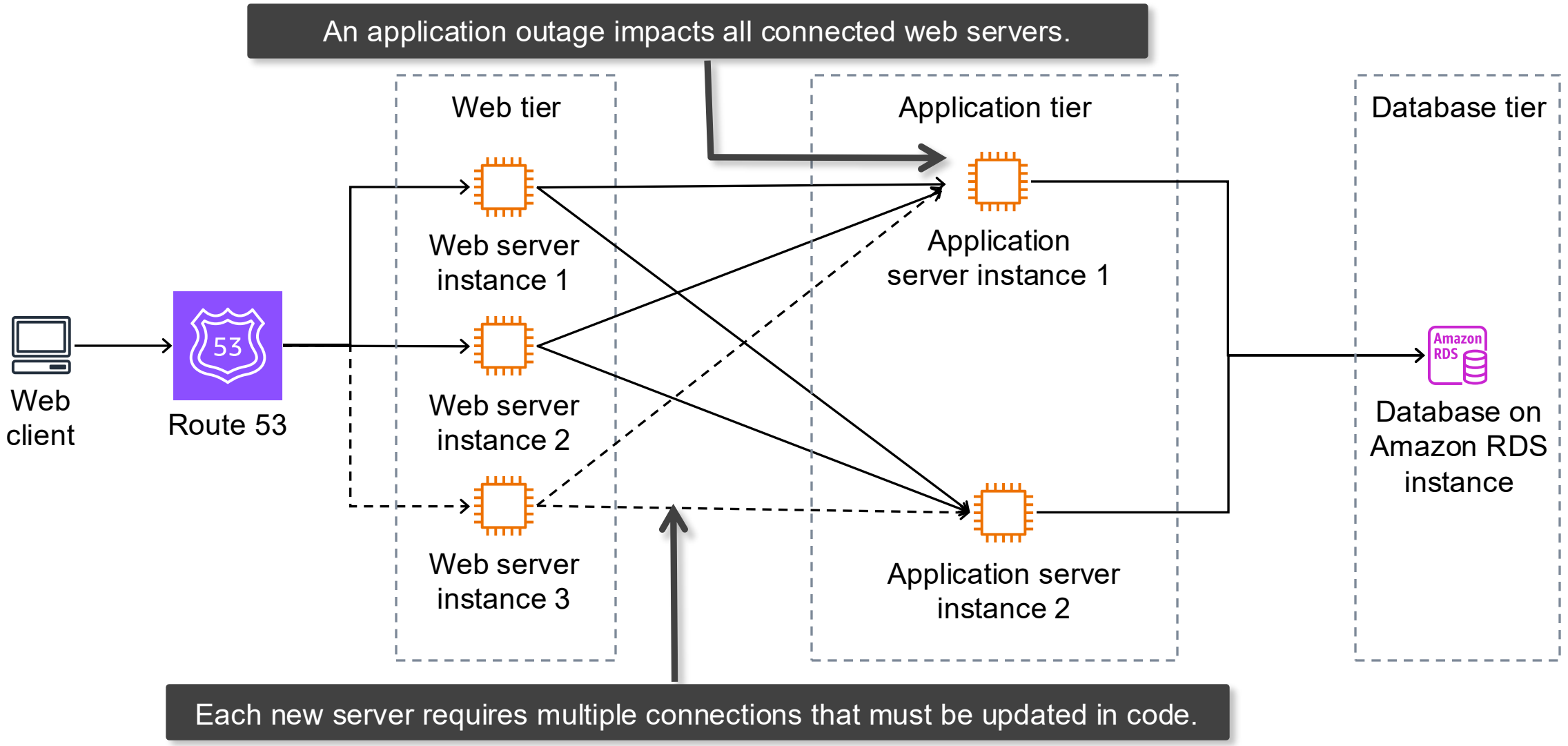
○	CafeAppPipeline	Failed	**SourceAction** – 72db49f2: add cafe-network template	3 minutes ago	View details
○	CafeNetworkPipeline	Succeeded	**SourceAction** – 72db49f2: add cafe-network template	3 minutes ago	View details

Decoupling Software Architecture

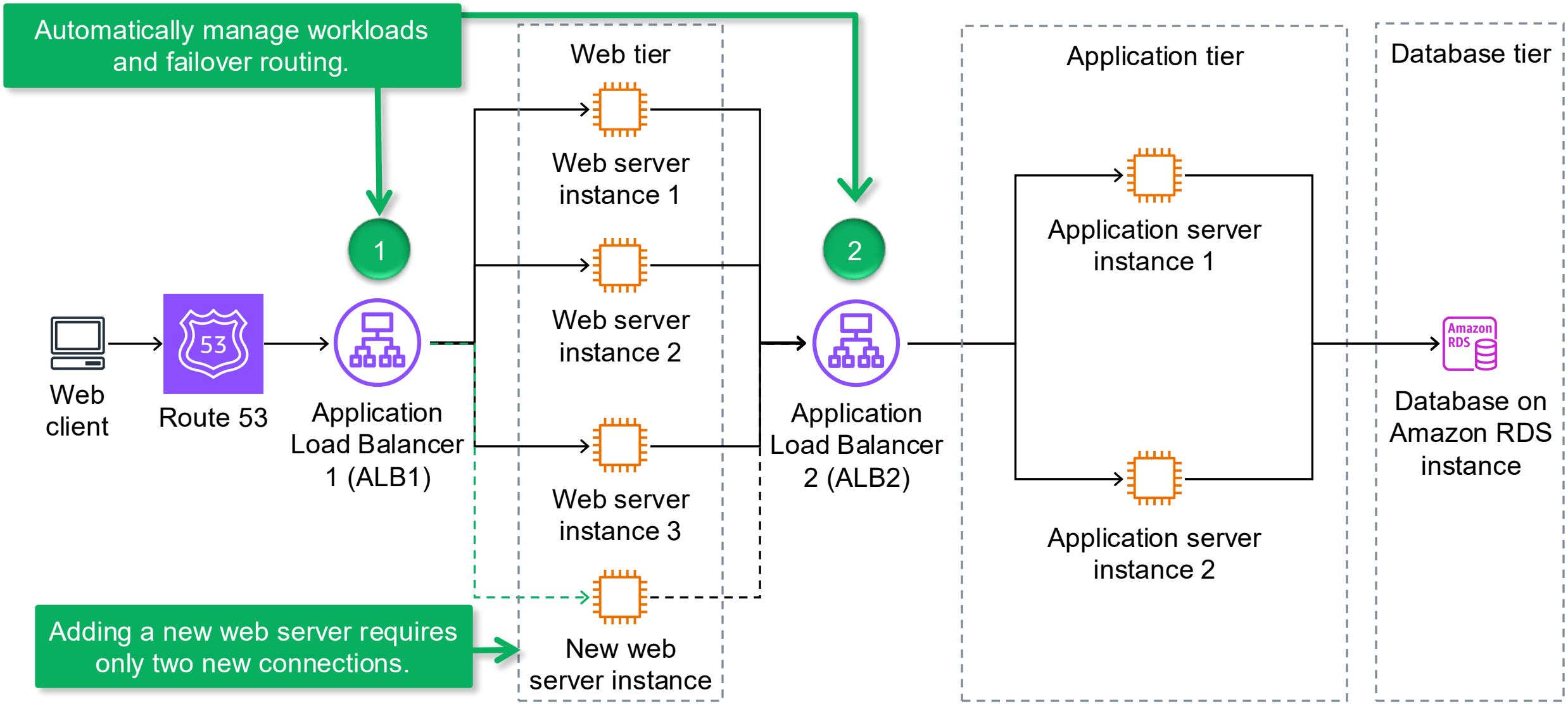
Tight coupling in a three-tier architecture



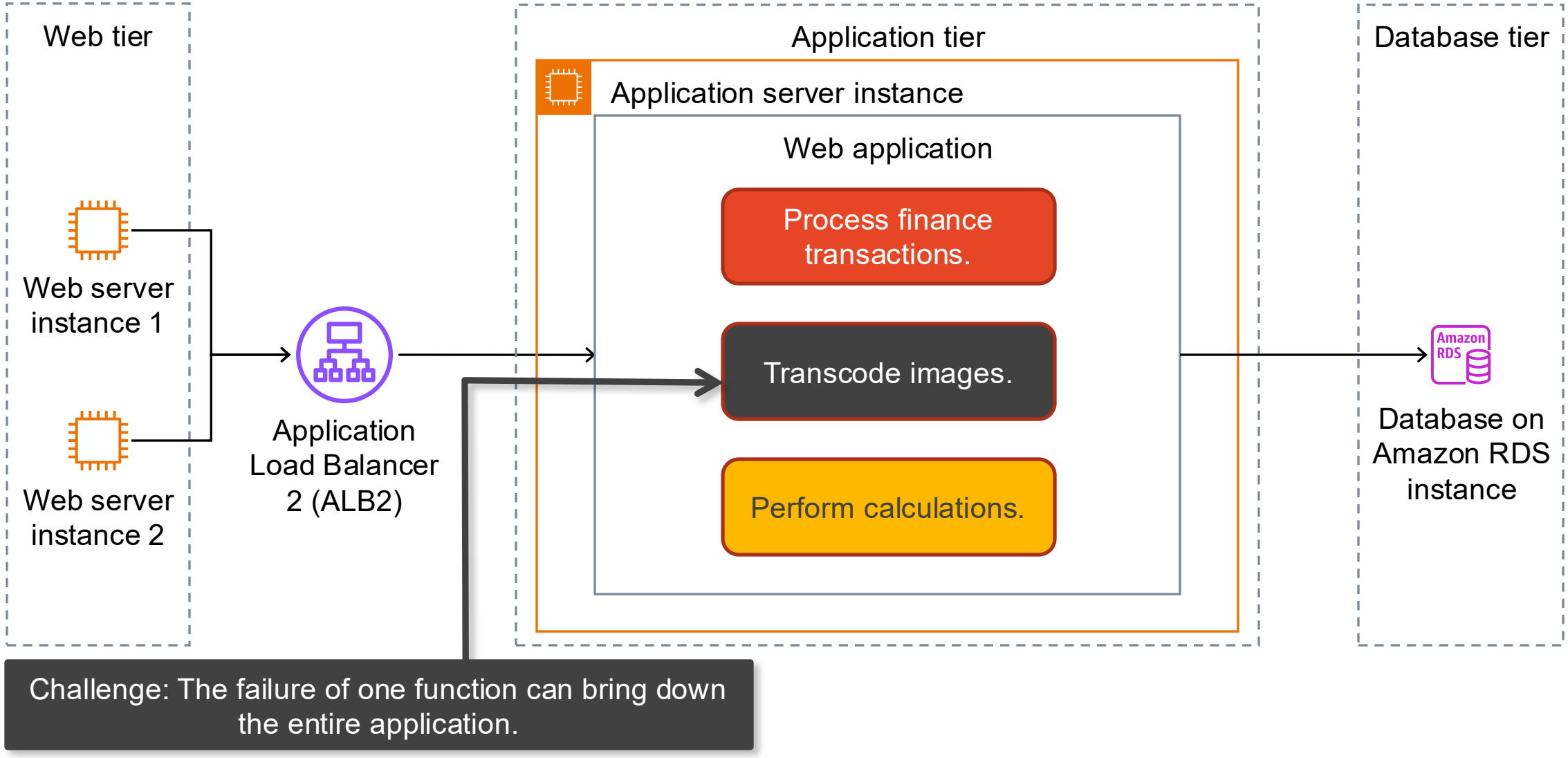
Tight coupling increases the complexity of scaling



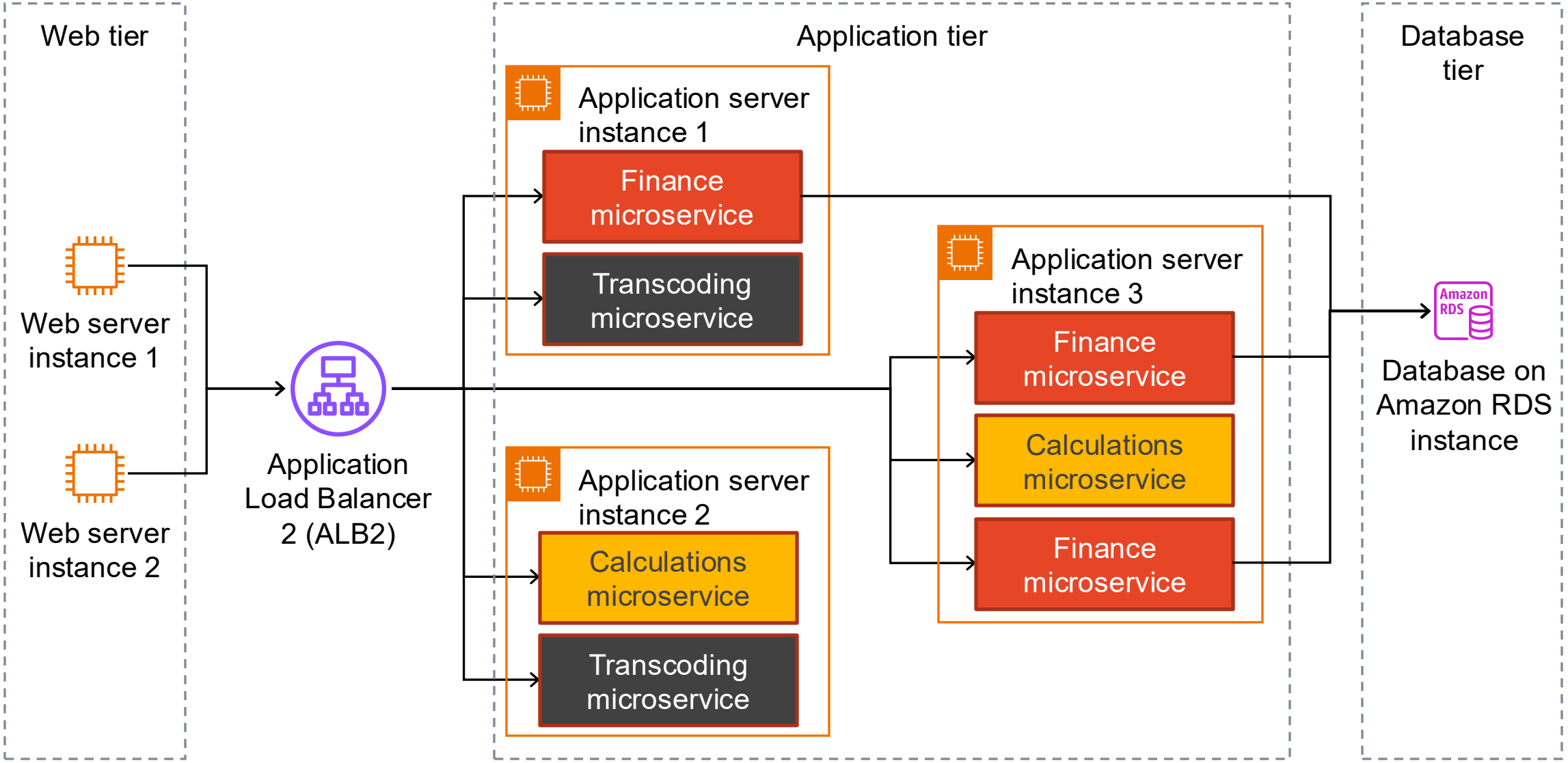
Loose coupling between tiers



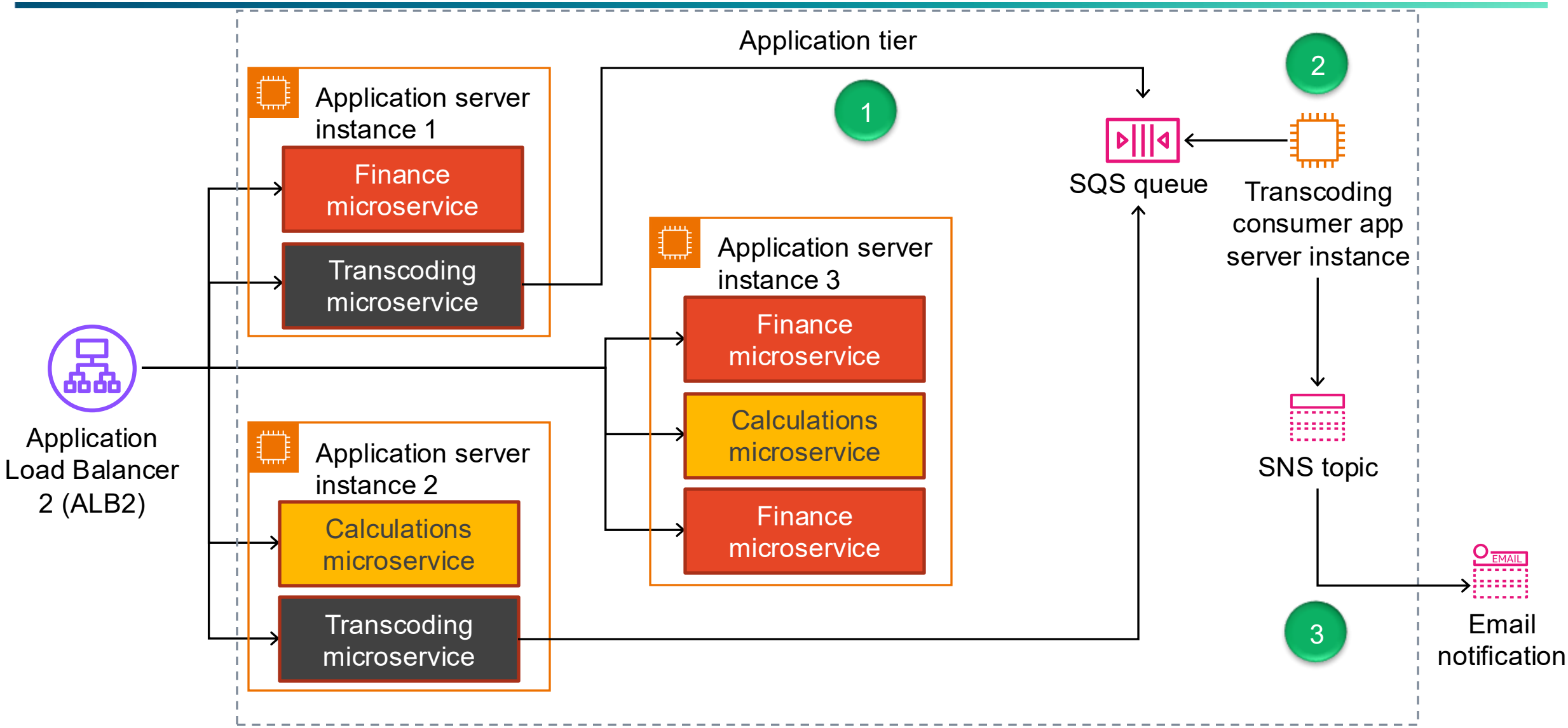
Tight coupling within an application



Loose coupling: Microservice architecture



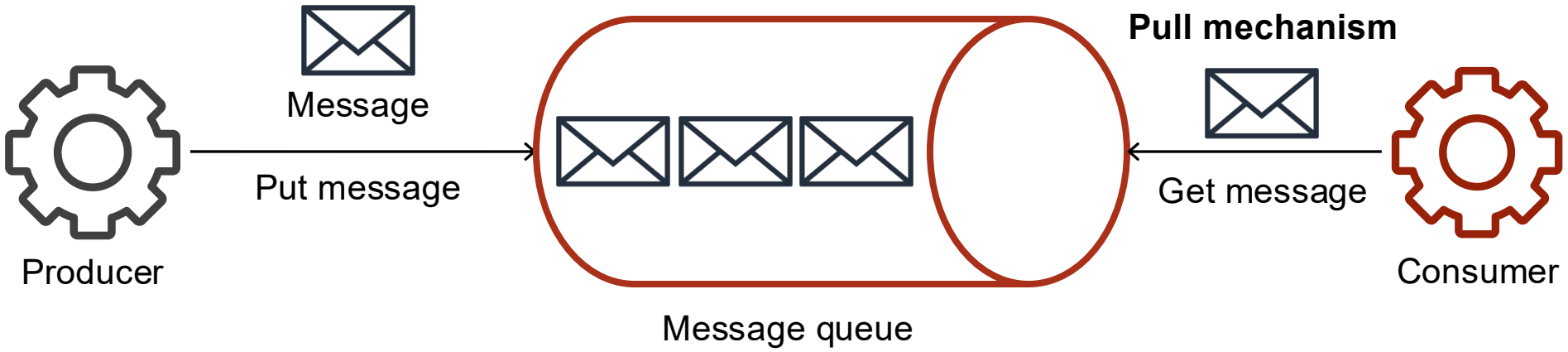
Request offloading: Amazon SQS and Amazon SNS



Amazon SQS

Point-to-point messaging

- You can decouple applications asynchronously by using point-to-point messaging.
- Using point-to-point messaging when the sending application sends a message to only one specific receiving application.
- The sending application is called a *producer*.
- The receiving application is called a *consumer*.
- Point-to-point messaging uses a *message queue* to decouple applications.

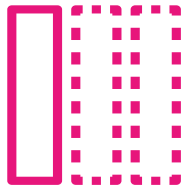


Amazon Simple Queue Service (Amazon SQS)

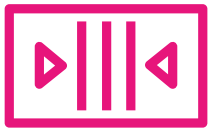
Amazon SQS

- Is a fully managed message queueing service
- Helps integrate and decouple distributed software systems and application components
- Provides highly available, secure, and durable message-queueing capabilities
- Provides an AWS Management Console interface and a web services API

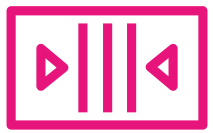
Amazon SQS basic components



Message



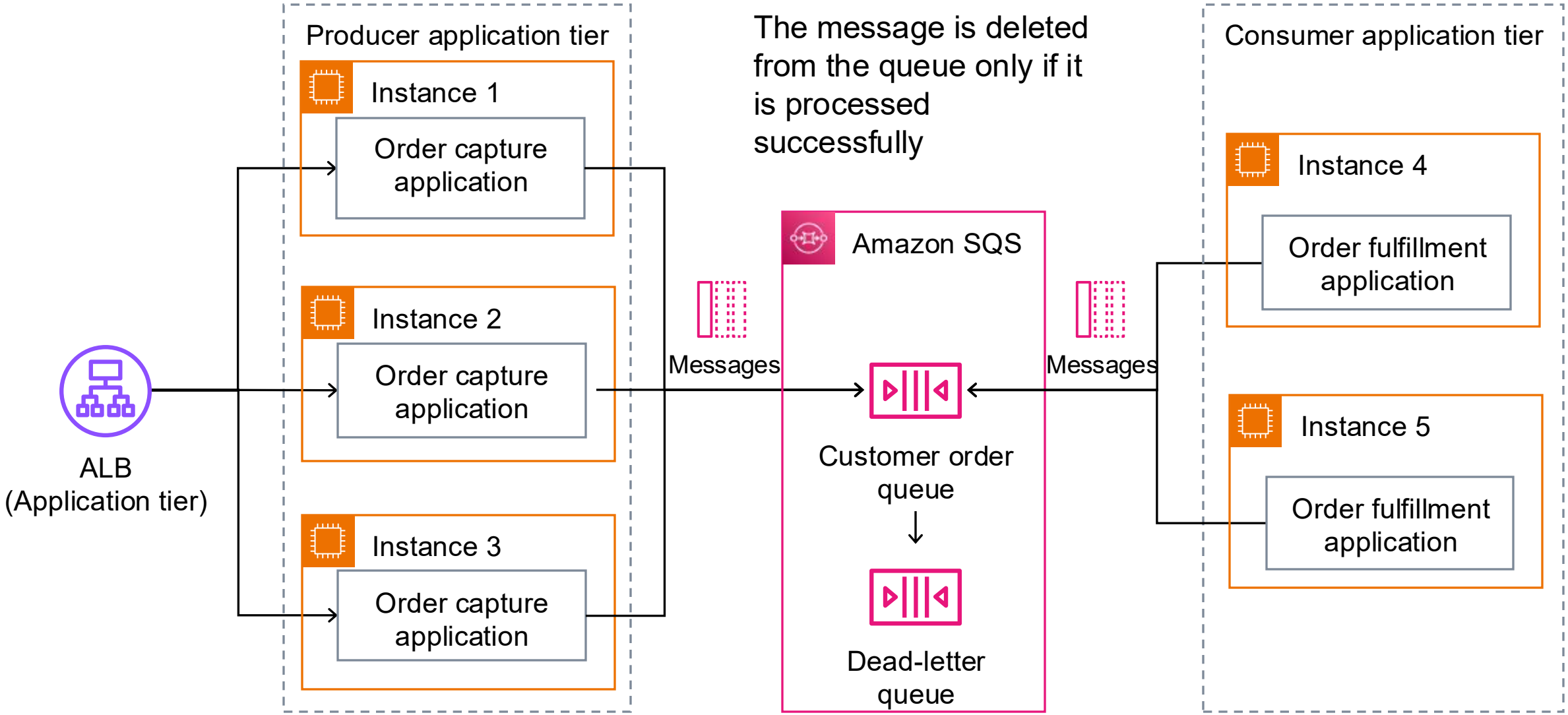
Queue



Dead-letter queue

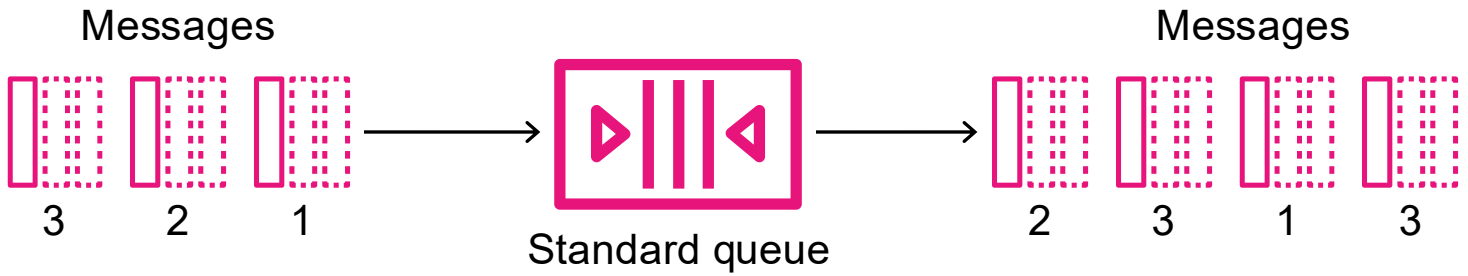
- A message can be up to 256 KB in size.
- A message remains in a queue until it is explicitly deleted or exceeds the queue's message retention period.
- Amazon SQS offers two types of queues: standard and first-in-first-out (FIFO)
- You can configure queue parameters, including the following:
  - Message retention period
  - Visibility timeout
  - Receive message wait time (short polling versus long polling)
- You can associate a dead-letter queue (DLQ) with any queue.
- A DLQ stores messages that cannot be consumed successfully.

Decoupling example: Amazon SQS



Queue types

Standard queue



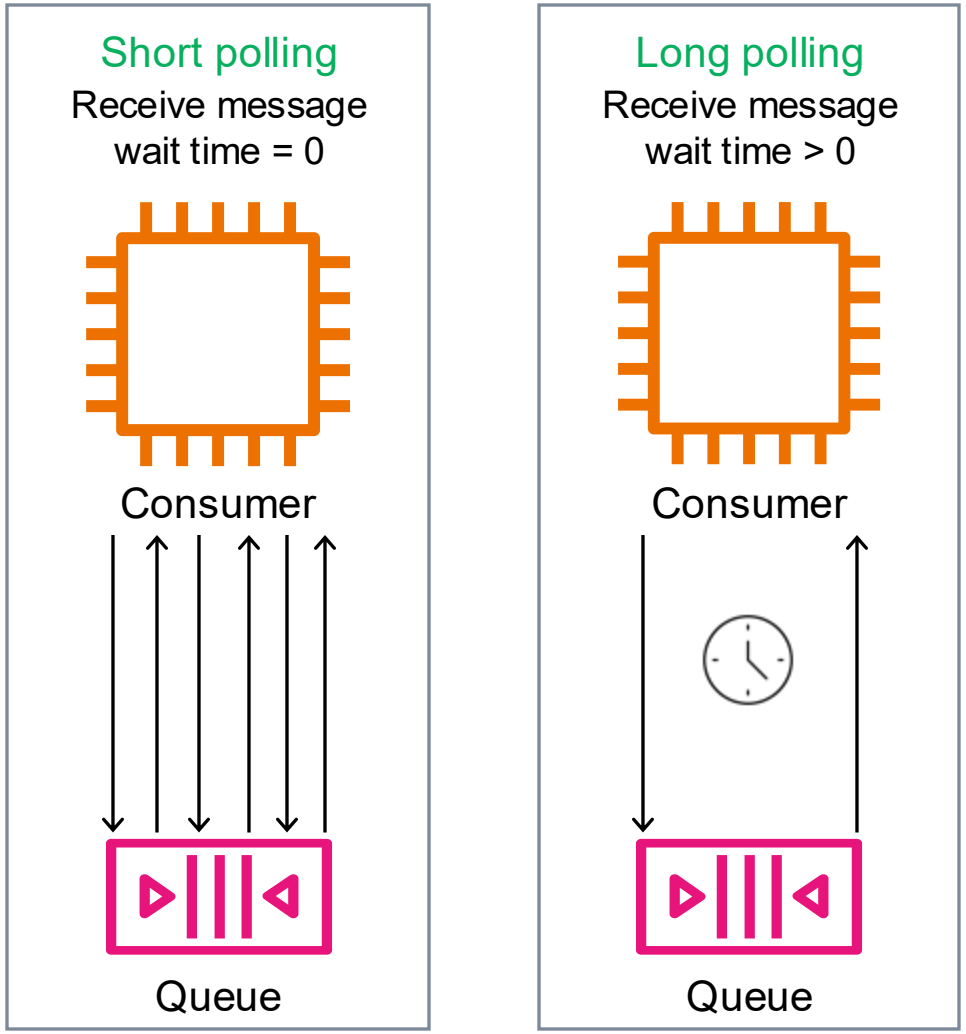
- At-least-once delivery
- Best-effort ordering
- Nearly unlimited throughput

FIFO queue

- FIFO delivery
- Exactly once processing
- High throughput

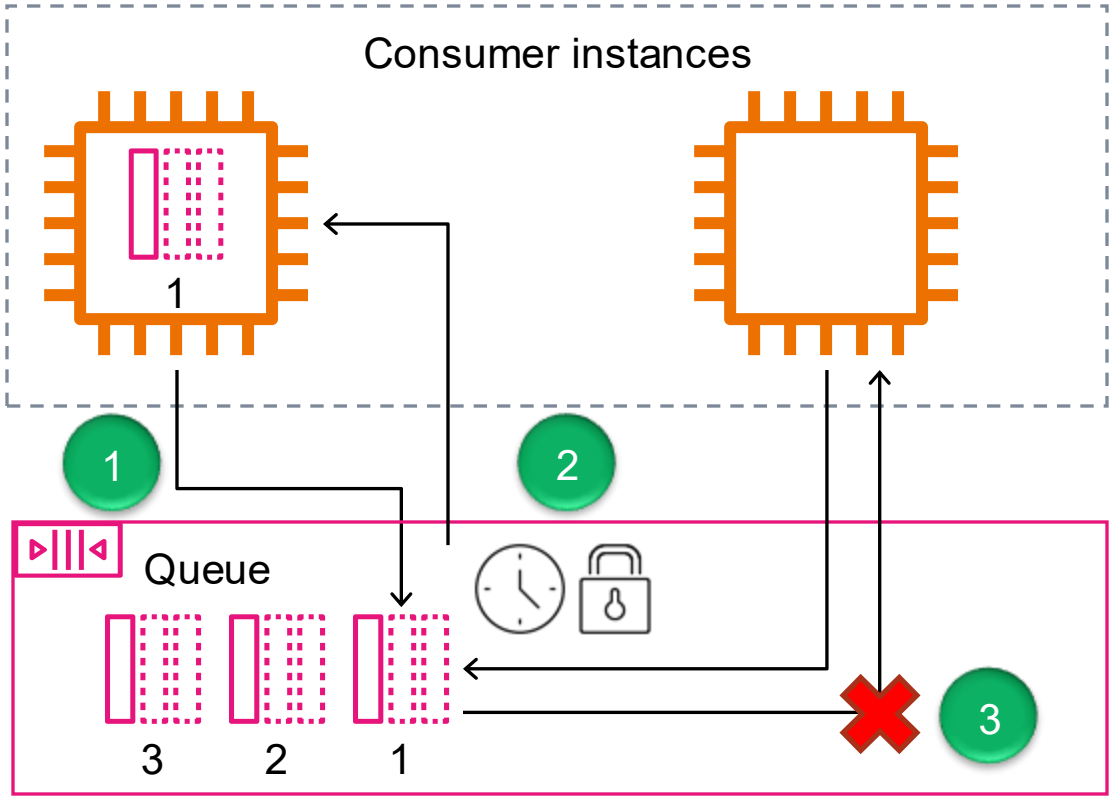
Queue configuration: Polling type

Choose the right polling type.

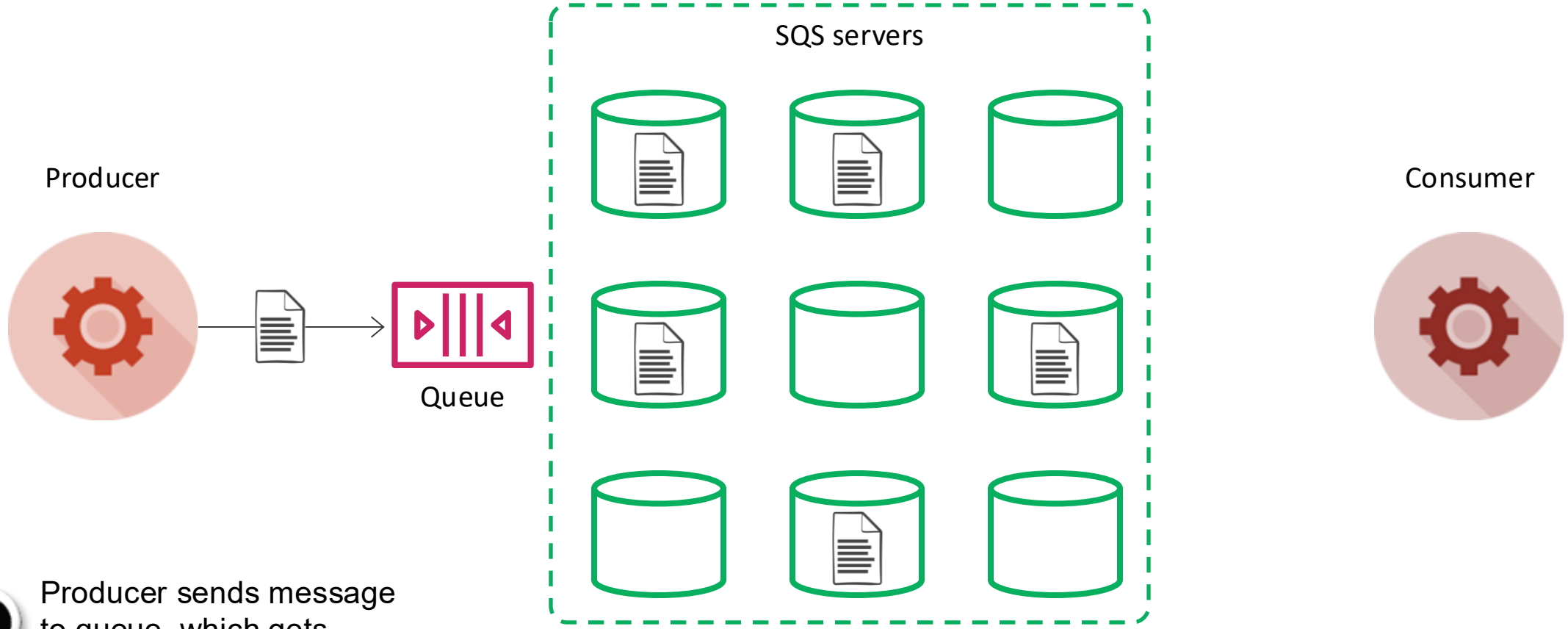


Queue configuration: Message visibility

Tune your visibility timeout.

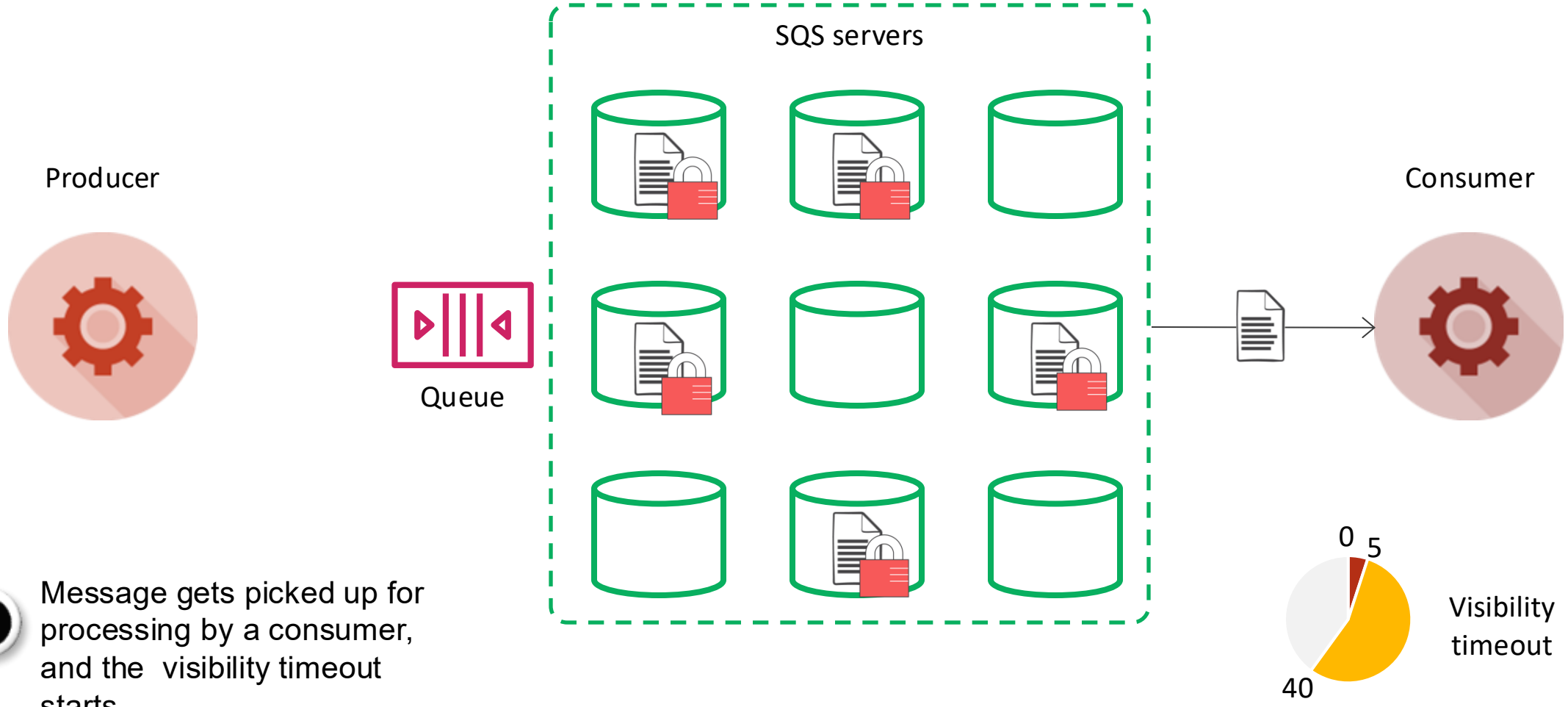


Amazon SQS message lifecycle: Create



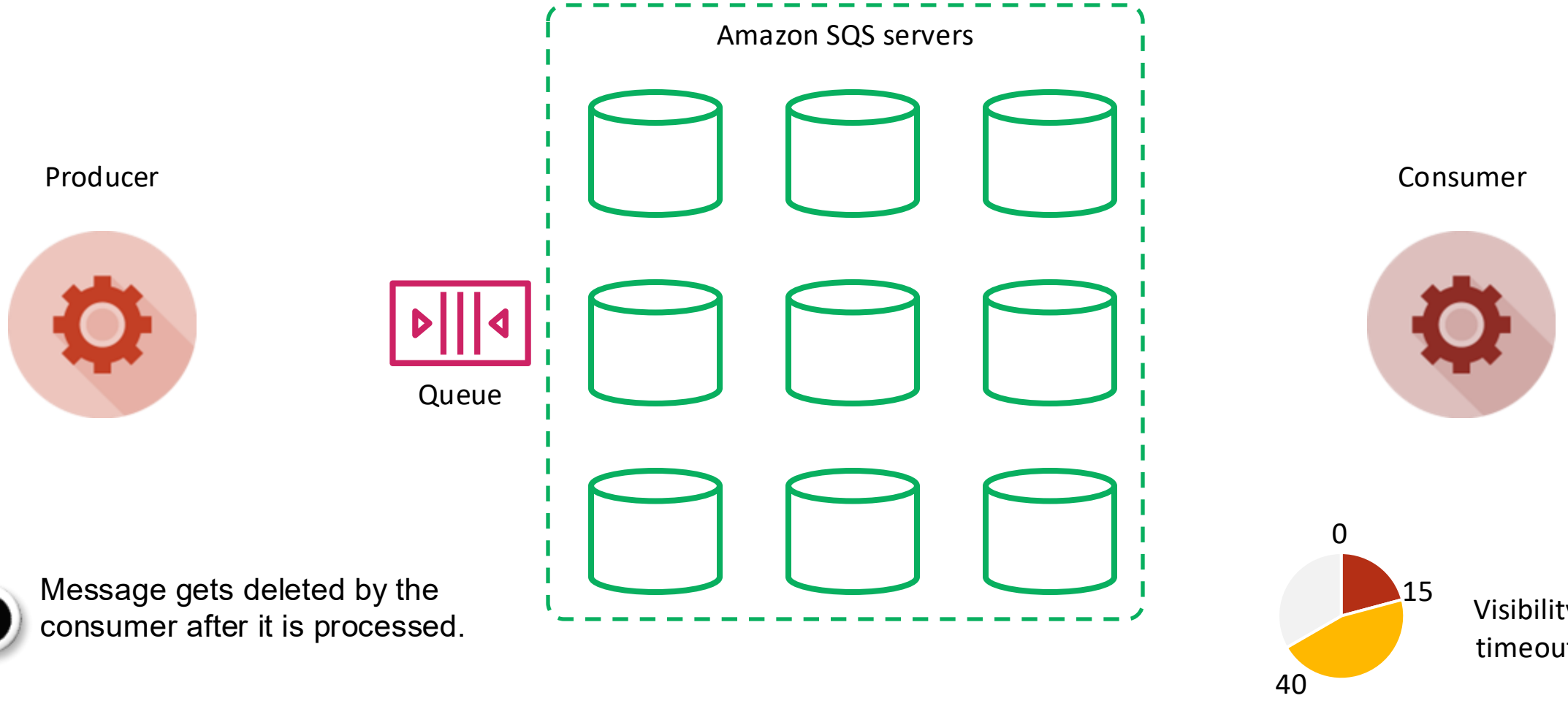
1 Producer sends message to queue, which gets distributed redundantly.

Amazon SQS message lifecycle: Process



2 Message gets picked up for processing by a consumer, and the visibility timeout starts.

Amazon SQS message lifecycle: Delete

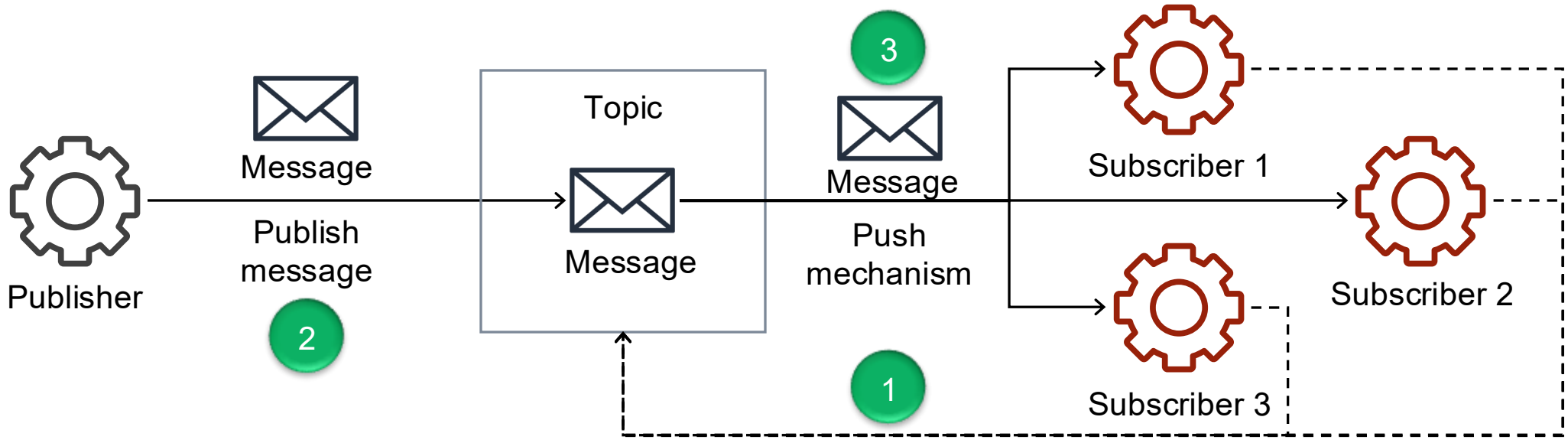


3 Message gets deleted by the consumer after it is processed.

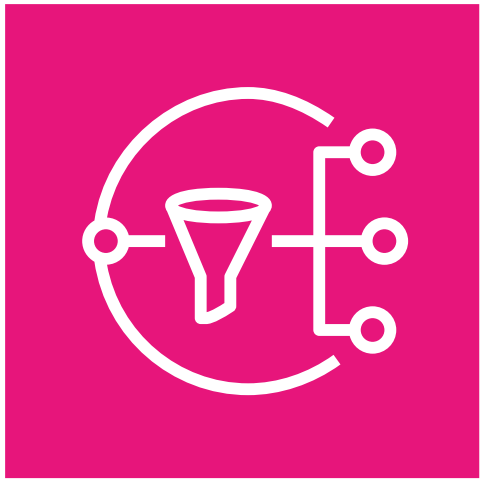
Amazon SNS

Publish/subscribe messaging

- You can decouple applications asynchronously by using publish/subscribe (pub/sub) messaging.
- Use pub/sub messaging when the sending application sends a message to multiple receiving applications and has little or no knowledge about the receiving applications.
- The sending application is called a *publisher*.
- The receiving application is called a *subscriber*.
- Pub/sub messaging uses a *topic* to decouple applications.



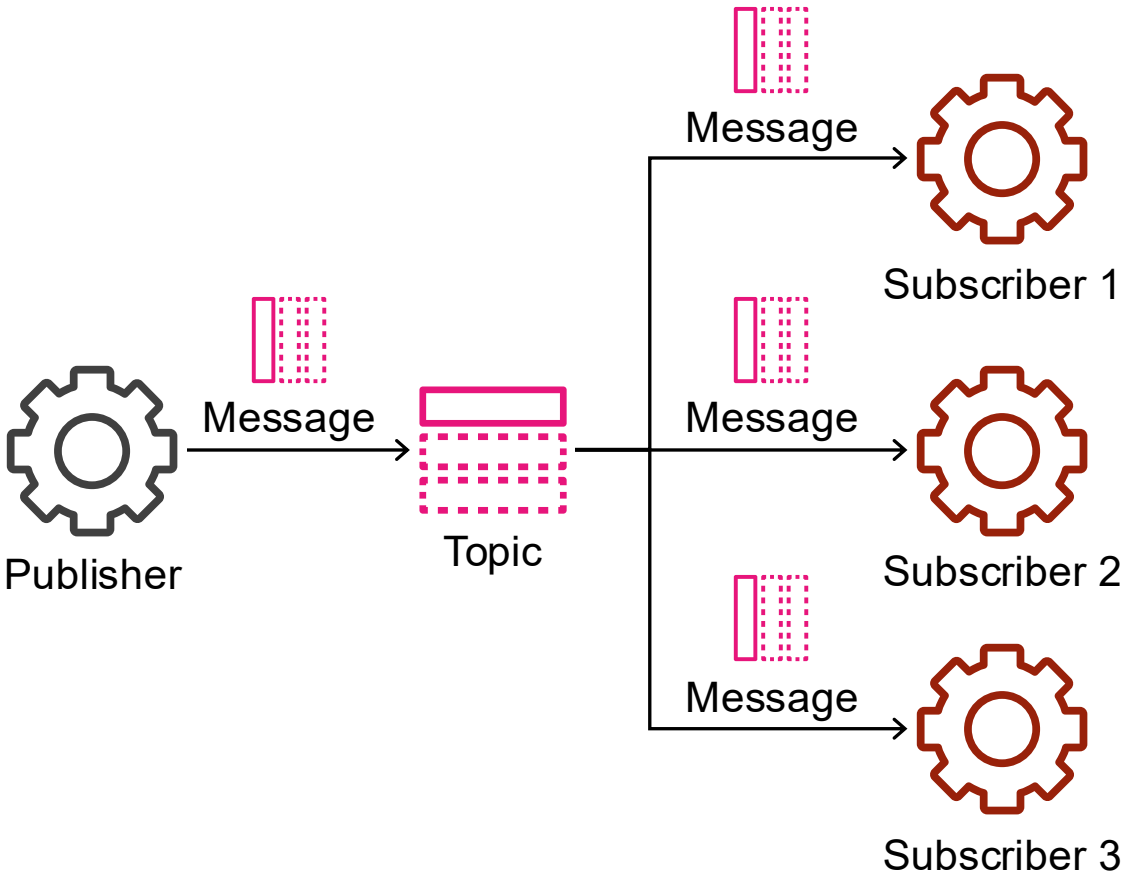
Amazon Simple Notification Service (Amazon SNS)



Amazon SNS

- Is a fully managed pub/sub messaging service
- Helps decouple applications through notifications
- Provides highly scalable, secure, and cost-effective notification capabilities
- Provides an AWS Management Console interface and a web services API

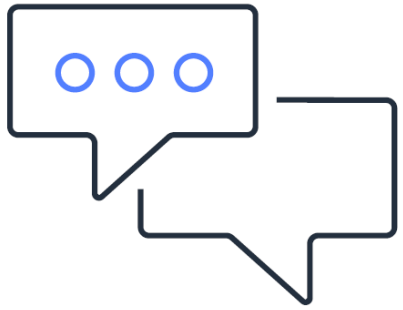
Types of subscribers



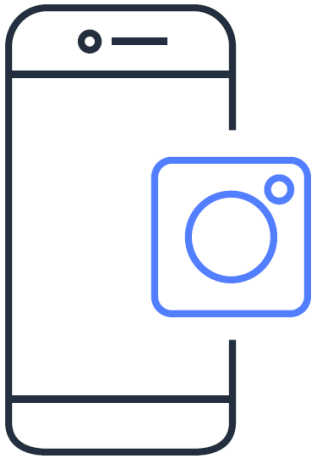
- Email destination
- Mobile text messaging destination
- Mobile push notifications endpoint
- HTTP or HTTPS endpoint
- AWS Lambda function
- SQS queue
- Amazon Kinesis Data Firehose delivery stream

Amazon SNS use cases

Application and system alert notification

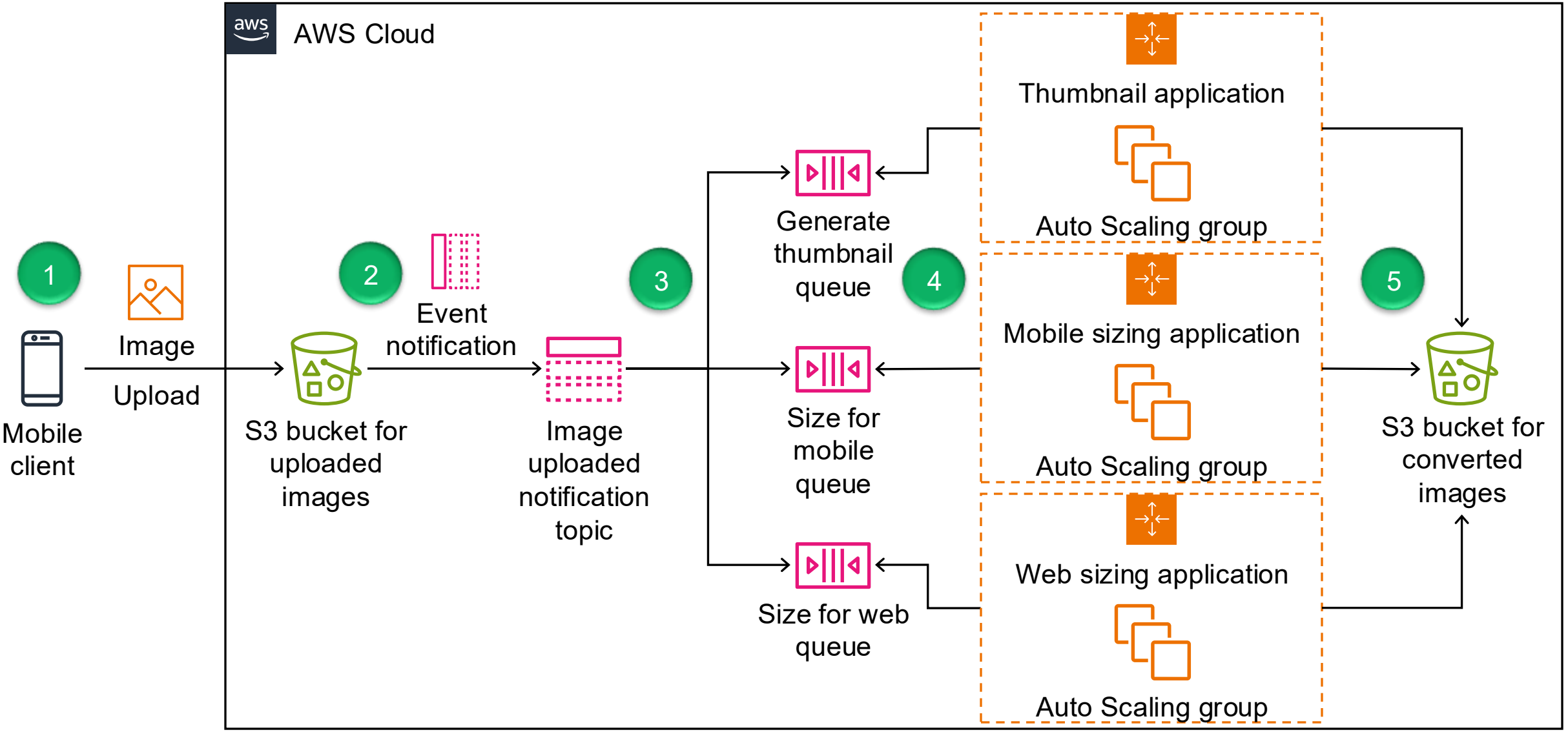


Email and text message notification



Mobile push notification

Decoupling example: Using Amazon SQS with Amazon SNS



Amazon SNS considerations

Message publishing

- Single published message
- No recall options

Message delivery

- Use a *standard* topic if the message delivery order does not matter.
- Use a *FIFO* topic if an exact message delivery order is required.
- Customize the delivery policy of an HTTP or HTTPS endpoint to control the retry behavior.

Amazon SNS and Amazon SQS comparison

	Amazon SNS	Amazon SQS
Messaging Model	Publisher-Subscriber	Producer-Consumer
Distribution Model	One to many	One to one
Delivery Mechanism	Push (passive)	Pull (active)
Message Persistence	No	Yes