Scale your API quickly using Message Queues on AWS

Introduction

- Nathan Westfall
- Work at Tyler Technologies in the ERP/School divisions
- Love open source (github.com/nwestfall)
- .NET Developer working with Xamarin and .NET Core

tyler technologies

Overview

- Review what message queues are
- Look at different architectures
- Build API in AWS

tyler technologies

What is a Message Queue?

tyler technologies

Message Queue

A message queue is a way to asynchronously communicate between applications or services, commonly used in serverless or microservice architectures.

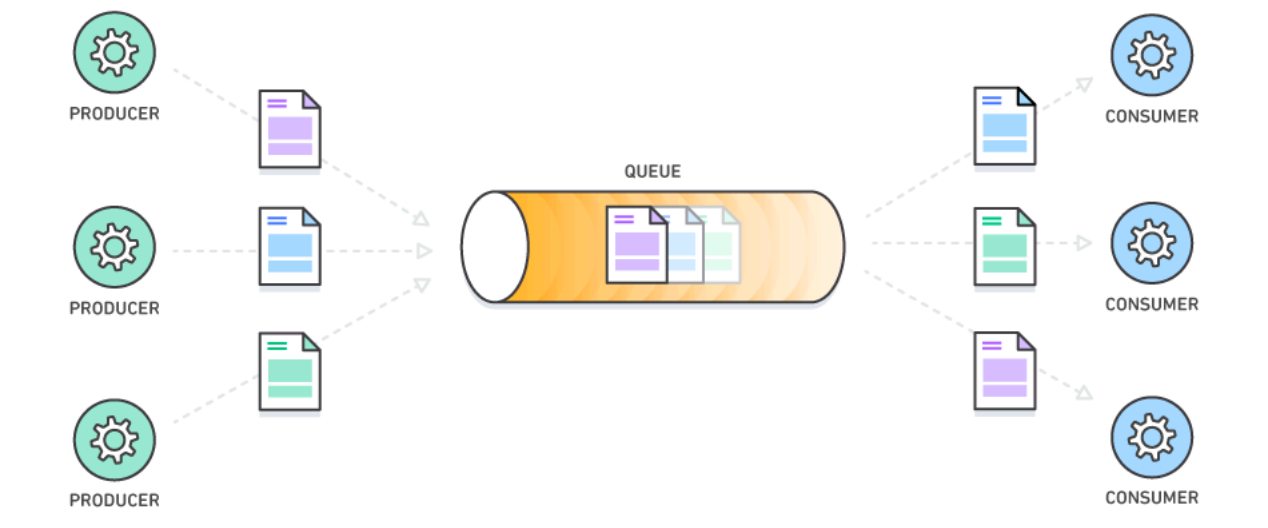
Message Queue Concepts

Producer/Publisher

- An application/client that creates and sends a message

Consumer/Subscriber

- An application/client that receives and reads a message



Message Queue Types

Standard Queue

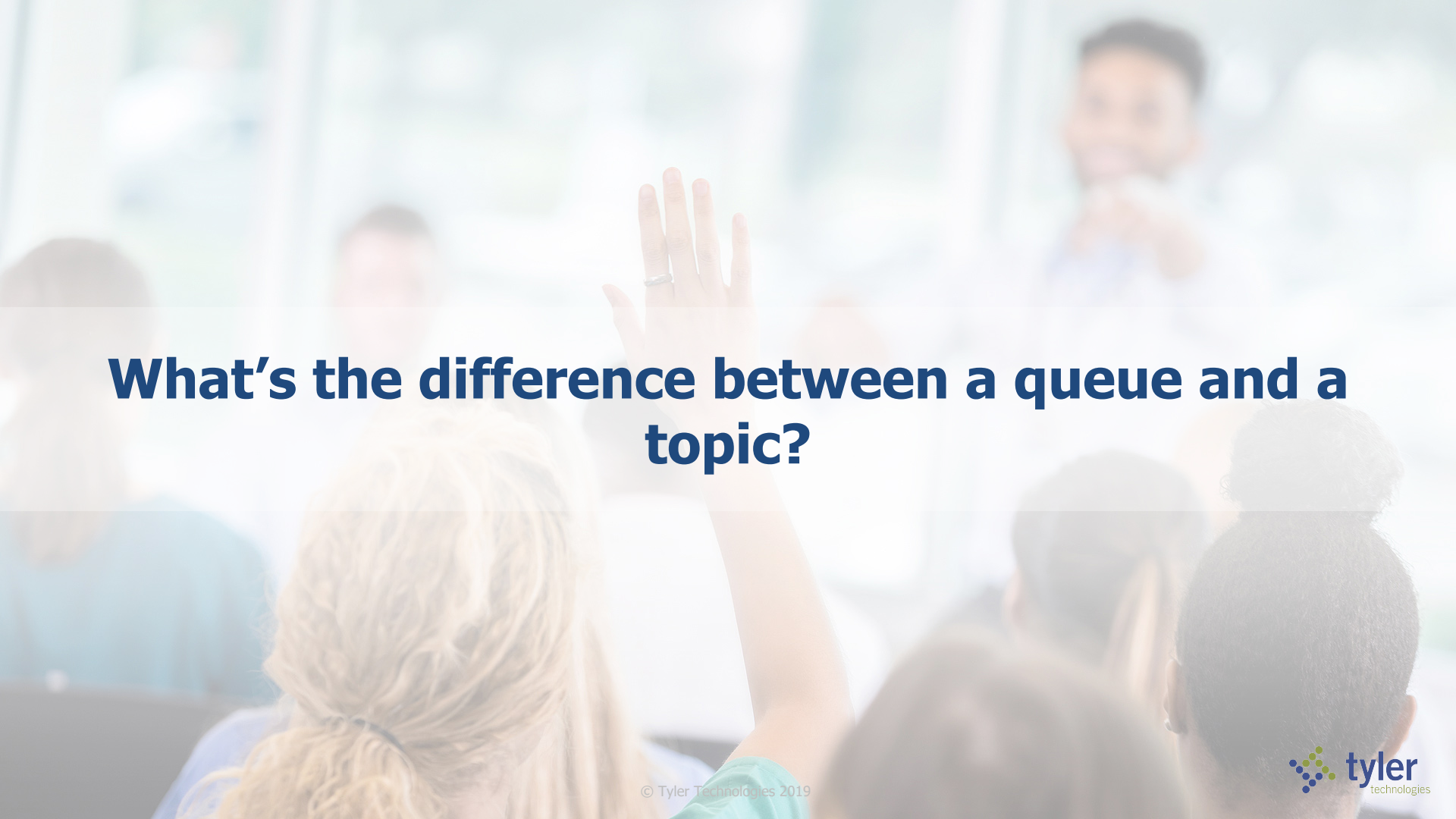
- Messages are sorted by best effort (can be unordered)
- High throughput

FIFO Queue (First-In-First-Out)

- Messages are guaranteed “first-in-first-out” order
- Lower throughput

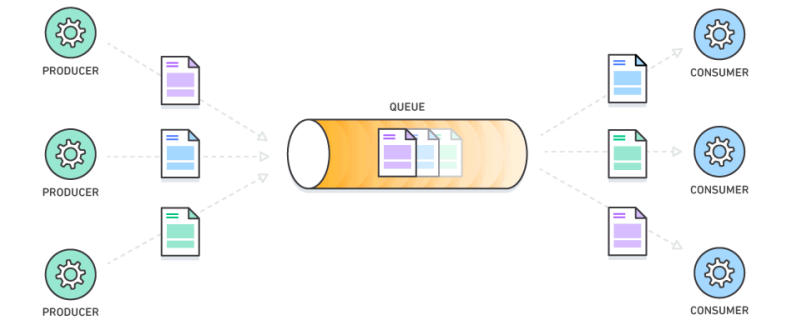
Message Queue Solutions

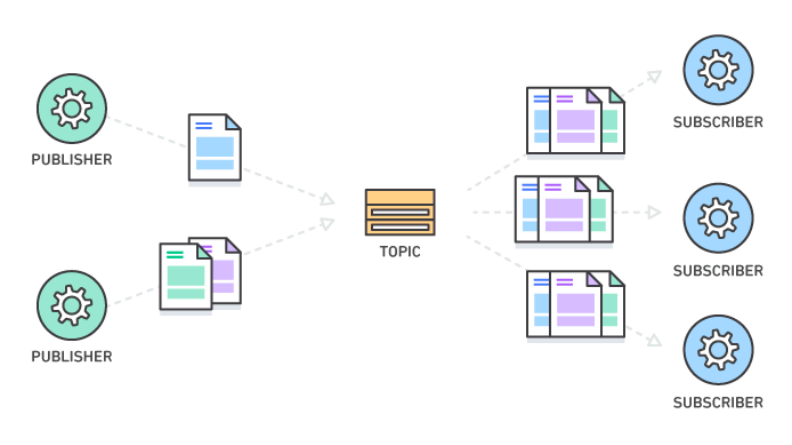
- RabbitMQ
- ActiveMQ
- Azure Service Bus
- Google Cloud Pub/Sub
- AWS SQS



What's the difference between a queue and a topic?

Queue vs. Topic





Why should I use one?

Why should I use a message queue?

- Allows a "decoupled" system
  - Example: Your API doesn't need to directly talk to your database. Your API can put data to process in a message queue and you can have another process save that data.
- Quicker scaling
  - Message queues are designed for high throughput
  - Scale the "producer" or "consumer" independently
- Redundancy/Resiliency
  - If part of your system goes down, your whole system doesn't fail

Design your API

tyler technologies

Goal

A REST API that saves bus locations to a database. We also want to take these bus locations and process them later for stop arrival performance, notifications, and reporting.

Bus Location API

- .NET Core REST API
- AWS RDS (MySql)
- AWS SQS
- Node.JS subscriber

Demo

VS Live Share - http://bit.ly/VTAWSDEMO

Recap

- Built an API in dotnet core locally and ECS
- Connected it to an RDS instance and SQS
- Created a lambda function that listened to our SQS
- With 0 code changes moved the location of the API and scaled it accordingly all while still processing data

tyler technologies

Recap

- Used a standard queue to process data
- If we did FIFO, you can't use lambda
  - github.com/nwestfall/MessageDelivery

Final Thoughts

- Message Queues should be used… when appropriate
- Decoupling of services allows you to work in any cloud/environment without attached networking or configuration
- Easily to scale either side of the queue without code changes

Questions

Scale your API quickly using Message Queues on AWS

Thank you!

Nathan Westfall

nathan.westfall@tylertech.com

tyler technologies

https://blog.iron.io/top-10-uses-for-message-queue/

https://dev.to/matteojoliveau/microservices-communications-why-you-should-switch-to-message-queues--48ia

https://aws.amazon.com/message-queue/

https://aws.amazon.com/blogs/compute/building-scalable-applications-and-microservices-adding-messaging-to-your-toolbox/