✓ 100 XP

Introduction

3 minutes

Imagine you work for a major news organization that reports breaking news alerts. Our company employs a worldwide network of journalists that are constantly sending updates through a web portal and a mobile app. A middle tier web service layer then takes those alert updates and publishes them online through several channels.

However, it's been noticed the system is missing alerts when globally significant events occur. This is a *huge* problem because we're being "scooped" by our competition! You've been hand-selected as the company's top developer to identify and fix the problem.

The middle tier provides plenty of capacity to handle normal loads. However, a look at the server logs revealed the system was overloaded when several journalists tried to upload larger breaking stories at the same time. Some writers complained the portal became unresponsive, and others said they lost their stories altogether. You've spotted a direct correlation between the reported issues and the spike in demand on the middle tier servers.

Clearly, you need a way to handle these unexpected peaks. You don't want to add more instances of the website and middle tier web service because they're expensive and, under normal conditions, redundant. We could dynamically spin up instances, but this takes time and we'd have the issue waiting for new servers to come online.

You can solve this problem by using Azure Queue storage. A storage queue is a high-performance message buffer that can act as a broker between the front-end components (the "producers") and the middle tier (the "consumer").

Our front-end components place a message for each new alert into a queue. The middle tier then retrieves these messages one at a time from the queue for processing. At times of high-demand, the queue may grow in length, but no stories will be lost, and the application will remain responsive. When demand drops back to normal levels, the web service will catch up by working through the queue backlog.

Let's learn how to use Azure Queue storage to handle high demand and improve resilience in your distributed applications.

Learning objectives

- Create an Azure Storage account that supports queues.

- Create a queue using C# and the Azure Storage Client Library for .NET.
- Add, retrieve, and remove messages from a queue using C# and the Azure Storage Client Library for .NET.

Next unit: Create the Azure storage infrastructure

Continue >