An e-commerce front end on Azure

07/13/2018 • 6 minutes to read • Contributors all

In this article

This example scenario walks you through an implementation of an e-commerce front end using Azure platform as a service (PaaS) tools. Many e-commerce websites face seasonality and traffic variability over time. When demand for your products or services takes off, whether predictably or unpredictably, using PaaS tools will allow you to handle more customers and more transactions automatically. Additionally, this scenario takes advantage of cloud economics by paying only for the capacity you use.

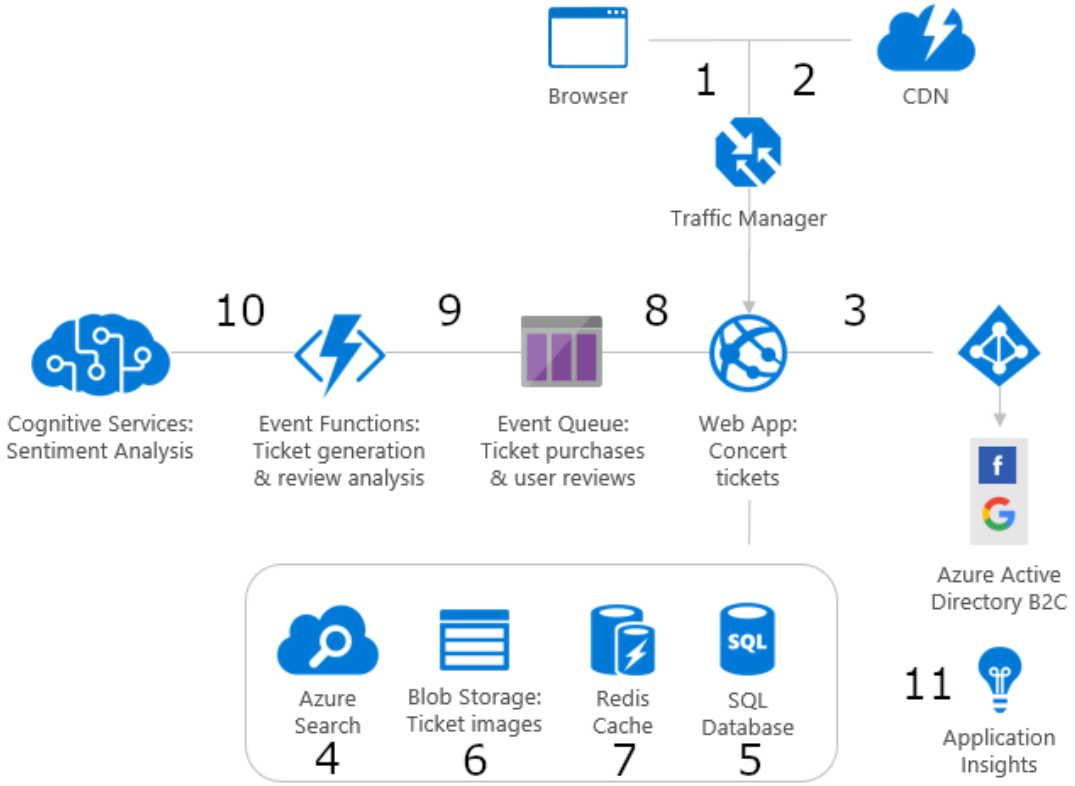
This document will help you will learn about various Azure PaaS components and considerations used to bring together to deploy a sample e-commerce application, *Relecloud Concerts*, an online concert ticketing platform.

Relevant use cases

Other relevant use cases include:

- Building an application that needs elastic scale to handle bursts of users at different times.
- Building an application that is designed to operate at high availability in different Azure regions around the world.

Architecture



This scenario covers purchasing tickets from an e-commerce site, the data flows through the scenario as follows:

1. Azure Traffic Manager routes a user's request to the e-commerce site hosted in Azure App Service.
2. Azure CDN serves static images and content to the user.
3. User signs in to the application through an Azure Active Directory B2C tenant.
4. User searches for concerts using Azure Search.
5. Web site pulls concert details from Azure SQL Database.
6. Web site refers to purchased ticket images in Blob Storage.
7. Database query results are cached in Azure Redis Cache for better performance.
8. User submits ticket orders and concert reviews, which are placed in the queue.
9. Azure Functions processes order payment and concert reviews.
10. Cognitive Services provides an analysis of the concert review to determine the sentiment (positive or negative).
11. Application Insights provides performance metrics for monitoring the health of the web application.

Components

- Azure CDN delivers static, cached content from locations close to users to reduce latency.
- Azure Traffic Manager controls the distribution of user traffic for service endpoints in different Azure regions.
- App Services - Web Apps hosts web applications allowing autoscale and high availability without having to manage infrastructure.
- Azure Active Directory - B2C is an identity management service that enables customization and control over how customers sign up, sign in, and manage their profiles in an application.
- Storage Queues stores large numbers of queue messages that can be accessed by an application.
- Functions are serverless compute options that allow applications to run on-demand without having to manage infrastructure.
- Cognitive Services - Sentiment Analysis uses machine learning APIs and enables developers to easily add intelligent features – such as emotion and video detection; facial, speech, and vision recognition; and speech and language understanding – into applications.
- Azure Search is a search-as-a-service cloud solution that provides a rich search experience over private, heterogenous content in web, mobile, and enterprise applications.
- Storage Blobs are optimized to store large amounts of unstructured data, such as text or binary data.
- Redis Cache improves the performance and scalability of systems that rely heavily on back-end data stores by temporarily copying frequently accessed data to fast storage located close to the application.
- SQL Database is a general-purpose relational database managed service in Microsoft Azure that supports structures such as relational data, JSON, spatial, and XML.
- Application Insights is designed to help you continuously improve performance and usability by automatically detecting performance anomalies through built-in analytics tools to help understand what users do with an app.

Alternatives

Many other technologies are available for building a customer facing application focused on e-commerce at scale. These cover both the front end of the application as well as the data tier.

Other options for the web tier and functions include:

- Service Fabric - A platform focused around building distributed components that benefit from being deployed and run across a cluster with a high degree of control. Service Fabric can also be used to host containers.
- Azure Kubernetes Service - A platform for building and deploying container-based solutions that can be used as one implementation of a microservices architecture. This allows for agility of different components of the application to be able to scale independently on demand.
- Azure Container Instances - A way of quickly deploying and running containers with a short lifecycle. Containers here are deployed to run a quick processing job such as processing a message or performing a calculation and then deprovisioned as soon as they are complete.
- Service Bus could be used in place of a Storage Queue.

Other options for the data tier include:

- Cosmos DB: Microsoft's globally distributed, multi-model database. This service provides a platform to run other data models such as Mongo DB, Cassandra, Graph data, or simple table storage.

Considerations

Availability

- Consider leveraging the typical design patterns for availability when building your cloud application.
- Review the availability considerations in the appropriate App Service web application reference architecture
- For additional considerations concerning availability, see the availability checklist in the Azure Architecture Center.

Scalability

- When building a cloud application be aware of the typical design patterns for scalability.
- Review the scalability considerations in the appropriate App Service web application reference architecture
- For other scalability topics, see the scalability checklist available in the Azure Architecture Center.

Security

- Consider leveraging the typical design patterns for security where appropriate.
- Review the security considerations in the appropriate App Service web application reference architecture.
- Consider following a secure development lifecycle process to help developers build more secure software and address security compliance requirements while reducing development cost.
- Review the blueprint architecture for Azure PCI DSS compliance.

Resiliency

- Consider leveraging the circuit breaker pattern to provide graceful error handling should one part of the application not be available.
- Review the typical design patterns for resiliency and consider implementing these where appropriate.
- You can find a number of recommended practices for App Service in the Azure Architecture Center.
- Consider using active geo-replication for the data tier and geo-redundant storage for images and queues.
- For a deeper discussion on resiliency, see the relevant article in the Azure Architecture Center.

Deploy the scenario

To deploy this scenario, you can follow this step-by-step tutorial demonstrating how to manually deploy each component. This tutorial also provides a .NET sample application that runs a simple ticket purchasing application. Additionally, there is a Resource Manager template to automate the deployment of most of the Azure resources.

Pricing

Explore the cost of running this scenario, all of the services are pre-configured in the cost calculator. To see how the pricing would change for your particular use case change the appropriate variables to match your expected traffic.

We have provided three sample cost profiles based on amount of traffic you expect to get:

- Small: This pricing example represents the components necessary to build the out for a minimum production level instance. Here we are assuming a small number of users, numbering only in a few thousand per month. The app is using a single instance of a standard web app that will be enough to enable autoscaling. The other components

are each scaled to a basic tier that will allow for a minimum amount of cost but still ensure that there is SLA support and enough capacity to handle a production level workload.

- Medium: This pricing example represents the components indicative of a moderate size deployment. Here we estimate approximately 100,000 users using the system over the course of a month. The expected traffic is handled in a single app service instance with a moderate standard tier. Additionally, moderate tiers of cognitive and search services are added to the calculator.
- Large: This pricing example represents an application meant for high scale, at the order of millions of users per month moving terabytes of data. At this level of usage high performance, premium tier web apps deployed in multiple regions fronted by traffic manager is required. Data consists of the following: storage, databases, and CDN, are configured for terabytes of data.

Related resources

- Reference Architecture for Multi-Region Web Application
- eShop on Containers Reference Example