Image classification for insurance claims on Azure

07/05/2018 • 4 minutes to read • Contributors all

In this article

This scenario is relevant for businesses that need to process images.

Potential applications include classifying images for a fashion website, analyzing text and images for insurance claims, or understanding telemetry data from game screenshots. Traditionally, companies would need to develop expertise in machine learning models, train the models, and finally run the images through their custom process to get the data out of the images.

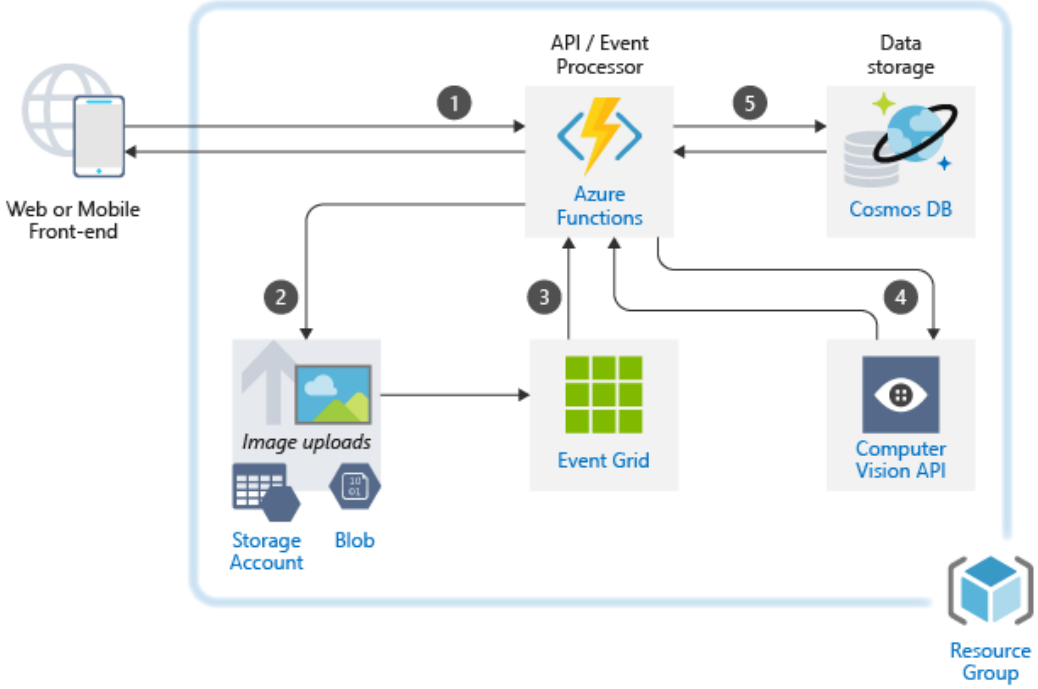
By using Azure services such as the Computer Vision API and Azure Functions, companies can eliminate the need to manage individual servers, while reducing costs and leveraging the expertise that Microsoft has already developed around processing images with Cognitive Services. This example scenario specifically addresses an image-processing use case. If you have different AI needs, consider the full suite of Cognitive Services.

Relevant use cases

Other relevant use cases include:

- Classifying images on a fashion website.
- Classifying telemetry data from screenshots of games.

Architecture



This scenario covers the back-end components of a web or mobile application. Data flows through the scenario as follows:

1. The API layer is built using Azure Functions. These APIs enable the application to upload images and retrieve data from Cosmos DB.
2. When an image is uploaded via an API call, it's stored in Blob storage.
3. Adding new files to Blob storage triggers an Event Grid notification to be sent to an Azure Function.
4. Azure Functions sends a link to the newly uploaded file to the Computer Vision API to analyze.
5. Once the data has been returned from the Computer Vision API, Azure Functions makes an entry in Cosmos DB to persist the results of the analysis along with the image metadata.

Components

- Computer Vision API is part of the Cognitive Services suite and is used to retrieve information about each image.
- Azure Functions provides the back-end API for the web application, as well as the event processing for uploaded images.
- Event Grid triggers an event when a new image is uploaded to blob storage. The image is then processed with Azure functions.
- Blob storage stores all of the image files that are uploaded into the web application, as well any static files that the web application consumes.
- Cosmos DB stores metadata about each image that is uploaded, including the results of the processing from Computer Vision API.

Alternatives

- Custom Vision Service. The Computer Vision API returns a set of taxonomy-based categories. If you need to process information that isn't returned by the Computer Vision API, consider the Custom Vision Service, which lets you build custom image classifiers.
- Azure Search. If your use case involves querying the metadata to find images that meet specific criteria, consider using Azure Search. Currently in preview, Cognitive search seamlessly integrates this workflow.

Considerations

Scalability

The majority of the components used in this example scenario are managed services that will automatically scale. A couple notable exceptions: Azure Functions has a limit of a maximum of 200 instances. If you need to scale beyond this limit, consider multiple regions or app plans.

Cosmos DB doesn't autoscale in terms of provisioned request units (RUs). For guidance on estimating your requirements see request units in our documentation. To fully take advantage of the scaling in Cosmos DB, understand how partition keys work in CosmosDB.

NoSQL databases frequently trade consistency (in the sense of the CAP theorem) for availability, scalability, and partitioning. In this example scenario, a key-value data model is used and transaction consistency is rarely needed as most operations are by definition atomic. Additional guidance to Choose the right data store is available in the Azure Architecture Center. If your implementation requires high consistency, you can choose your consistency level in CosmosDB.

For general guidance on designing scalable solutions, see the scalability checklist in the Azure Architecture Center.

Security

Managed identities for Azure resources are used to provide access to other resources internal to your account and then assigned to your Azure Functions. Only allow access to the requisite resources in those identities to ensure that nothing extra is exposed to your functions (and potentially to your customers).

For general guidance on designing secure solutions, see the Azure Security Documentation.

Resiliency

All of the components in this scenario are managed, so at a regional level they are all resilient automatically.

For general guidance on designing resilient solutions, see Designing resilient applications for Azure.

Pricing

To 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 (we assume all images are 100 kb in size):

- Small: this pricing example correlates to processing < 5000 images a month.
- Medium: this pricing example correlates to processing 500,000 images a month.
- Large: this pricing example correlates to processing 50 million images a month.

Related resources

For a guided learning path, see Build a serverless web app in Azure.

Before deploying this example scenario in a production environment, review recommended practices for optimizing the performance and reliability of Azure Functions.