

Exam AI—102

Designing and Implementing a
Microsoft Azure AI Solution – Skills
Measured

 **Address**

Level 1, 42 Murray Street, Hobart,
Tasmania 7000 Australia

 **Phone**

03 6234 3883

 **Email**

quill@quill.com.au

 **Web**

<https://www.quill.com.au/>

Audience Profile

Candidates for Exam AI-102: Designing and Implementing a Microsoft Azure AI Solution build, manage, and deploy AI solutions that leverage Azure Cognitive Services and Azure Applied AI services. Their responsibilities include participating in all phases of AI solutions development—from requirements definition and design to development, deployment, maintenance, performance tuning, and monitoring. They work with solution architects to translate their vision and with data scientists, data engineers, IoT specialists, and AI developers to build complete end-to-end AI solutions. Candidates for this exam should be proficient in C# or Python and should be able to use REST based APIs and SDKs to build computer vision, natural language processing, knowledge mining, and conversational AI solutions on Azure. They should also understand the components that make up the Azure AI portfolio and the available data storage options. Plus, candidates need to understand and be able to apply responsible AI principles.



Contents

Audience Profile	2
How to use this guide	5
In the exam	5
Key Learning Objectives	6
Plan and Manage an Azure Cognitive Services Solution (15-20%).....	6
Select the appropriate Cognitive Services resource.....	6
Plan and configure security for a Cognitive Services solution.....	16
Create a Cognitive Services resource	19
Plan and implement Cognitive Services containers	25
Implement Computer Vision Solutions (20-25%)	27
Analyze images by using the Computer Vision API	27
Extract text from images	30
Extract facial information from images	33
Implement image classification by using the Custom Vision service	37
Implement an object detection solution by using the Custom Vision service	42
Analyze video by using Azure Video Analyzer for Media	46

Contents

Implement Natural Language Processing Solutions (20-25%)	51
Analyze text by using the Text Analytics service	51
Manage speech by using the Speech service	54
Translate language	58
Build an initial language model by using Language Understanding Service (LUIS)...	59
Iterate on and optimize a language model by using LUIS	62
Manage a LUIS model	68
 Implement Knowledge Mining Solutions (15-20%)	 72
Implement a Cognitive Search solution	72
Implement an enrichment pipeline	77
Implement a knowledge store	79
Manage a Cognitive Search solution	82
Manage indexing	86
 Implement Conversational AI Solutions (15-20%)	 91
Create a knowledge base by using QnA Maker	91
Design and implement conversation flow	102
Create a bot by using the Bot Framework SDK	105
Create a bot by using the Bot Framework Composer	109
Integrate Cognitive Services into a bot	114

How to use this guide

This guide is here to help you prepare and take the exam. It is designed to complement your existing learning and to help guide you in the areas of focus for the exam. You should use this as a framework to help fill in the blanks on information that you have.

We have developed the following content in direct alignment to the current Learning objectives. These can be viewed directly, by selecting the “Download exam skills outline” from the exam page at: <https://docs.microsoft.com/en-us/learn/certifications/exams/ai-102>

Skills measured

The English language version of this exam will be updated on May 2, 2022. Please download the exam skills outline below to see what’s changing.

- Plan and manage an Azure Cognitive Services solution (15-20%)
- Implement Computer Vision solutions (20-25%)
- Implement natural language processing solutions (20-25%)
- Implement knowledge mining solutions (15-20%)
- Implement conversational AI solutions (15-20%)

[↓ Download exam skills outline](#)

There are loads of exciting and interesting topics we can begin to follow on from these core objectives, but remember for the exam we do need to stay focused and constrain ourselves to these key topics.

In the exam

The exam itself is quite straight forward with no complicated case studies or longwinded questions. The majority of the questions will be “Multiple Choice” or “Choose all that apply” type of questions. You may also come across some “Drag and Drop” questions where you need to place answers in order. The key thing to note is that all of the questions will have the answer in front of you.

Remembering that all of the answers are presented to you, you need to make sure that you answer each question. There is no loss of marks for incorrect answers, so even if you don't know the answer, you should attempt it.

The exam itself will have between 40 and 50 questions, depending on the pool of questions that have been allocated. You will have 60 minutes to complete the exam. As you can see you will need to move at a steady pace throughout. Don't get too stuck on any question, instead select your answer and then mark the question for "Review". Then if you have time at the end of the exam you can go back and review these questions.

Key Learning Objectives

Plan and Manage an Azure Cognitive Services Solution (15-20%)

Select the appropriate Cognitive Services resource

Select the appropriate cognitive service for a vision solution

Azure Cognitive Services are cloud-based artificial intelligence (AI) services that help you build cognitive intelligence into your applications. They are available as REST APIs, client library SDKs, and user interfaces. You can add cognitive features to your applications without having AI or data science skills. Cognitive Services enable you to build cognitive solutions that can see, hear, speak, understand, and even make decisions.

Vision APIs

Service Name	Service Description	Quickstart
Computer Vision	The Computer Vision service provides you with access to advanced cognitive algorithms for processing images and returning information.	Computer Vision quickstart
Custom Vision	The Custom Vision Service lets you build, deploy, and improve your own image classifiers. An image classifier is an AI service that applies labels to images.	Custom Vision quickstart
Face	The Face service provides access to advanced face algorithms, enabling face attribute detection and recognition.	Face quickstart

Computer Vision

Azure's Computer Vision service gives you access to advanced algorithms that process images and return information based on the visual features you're interested in.

- **Optical Character Recognition (OCR)**

The Optical Character Recognition (OCR) service extracts text from images. You can use the new Read API to extract printed and handwritten text from photos and documents. It uses deep-learning-based models and works with text on a variety of surfaces and backgrounds. These include business documents, invoices, receipts, posters, business cards, letters, and whiteboards. The OCR APIs support extracting printed text in several languages.

- **Image Analysis**

The Image Analysis service extracts many visual features from images, such as objects, faces, adult content, and auto-generated text descriptions. Follow the Image Analysis quickstart to get started.

- **Spatial Analysis**

The Spatial Analysis service analyzes the presence and movement of people on a video feed and produces events that other systems can respond to. Install the Spatial Analysis container to get started

<https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview>

Custom Vision

Azure Custom Vision is an image recognition service that lets you build, deploy, and improve your own image identifier models. An image identifier applies labels to images, according to their detected visual characteristics. Each label represents a classification or objects. Unlike the Computer Vision service, Custom Vision allows you to specify your own labels and train custom models to detect them.

What it does

The Custom Vision service uses a machine learning algorithm to analyze images. You, the developer, submit groups of images that have and don't have the characteristics in question. You label the images yourself at the time of submission. Then the algorithm trains to this data and calculates its own accuracy by testing itself on those same images. Once you've trained the algorithm, you can test, retrain, and eventually use it in your image recognition app to classify images. You can also export the model itself for offline use.

Custom Vision functionality can be divided into two features. **Image classification** applies one or more labels to an entire image. **Object detection** is similar, but it returns the coordinates in the image where the applied label(s) can be found.

<https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/overview>

Face

The Azure Face service provides AI algorithms that detect, recognize, and analyze human faces in images. Facial recognition software is important in many different scenarios, such as identity verification, touchless access control, and face blurring for privacy.

Example use cases

Identity verification: Verify someone's identity against a government-issued ID card like a passport or driver's license or other enrollment image. You can use this verification to grant access to digital or physical services or recover an account. Specific access scenarios include opening a new account, verifying a worker, or administering an online assessment. Identity verification can be done once when a person is onboarded, and repeated when they access a digital or physical service.

Touchless access control: Compared to today's methods like cards or tickets, opt-in face identification enables an enhanced access control experience while reducing the hygiene and security risks from card sharing, loss, or theft. Facial recognition assists the check-in process with a human in the loop for check-ins in airports, stadiums, theme parks, buildings, reception kiosks at offices, hospitals, gyms, clubs, or schools.

Face redaction: Redact or blur detected faces of people recorded in a video to protect their privacy.

Face detection is required as a first step in all the other scenarios. The Detect API detects human faces in an image and returns the rectangle coordinates of their locations. It also returns a unique ID that represents the stored face data. This is used in later operations to identify or verify face

Optionally, face detection can extract a set of face-related attributes, such as head pose, age, emotion, facial hair, and glasses. These attributes are general predictions, not actual classifications. Some attributes are useful to ensure that your application is getting high-quality face data when users add themselves to a Face service. For example, your application could advise users to take off their sunglasses if they're wearing sunglasses.

<https://docs.microsoft.com/en-us/azure/cognitive-services/face/overview>

Select the appropriate cognitive service for a language analysis solution

Service Name	Service Description	Quickstart
Language service	Azure Language service provides several Natural Language Processing (NLP) features to understand and analyze text.	Go to the Language documentation to choose a subservice
Translator	Translator provides machine-based text translation in near real time.	Translator quickstart
Language Understanding LUIS	Language Understanding (LUIS) is a cloud-based conversational AI service that applies custom machine-learning intelligence to a user's conversational or natural language text to predict overall meaning and pull	LUIS quickstart
QnA Maker	QnA Maker allows you to build a question and answer service from your semi-structured content.	QnA Maker quickstart

<https://docs.microsoft.com/en-us/azure/cognitive-services/what-are-cognitive-services>

Language Understanding (LUIS)

Language Understanding (LUIS) is a cloud-based conversational AI service that applies custom machine-learning intelligence to a user's conversational, natural language text to predict overall meaning, and pull out relevant, detailed information. LUIS provides access through its custom portal, APIs and SDK client libraries.

What does LUIS Offer

Simplicity: LUIS offloads you from the need of in-house AI expertise or any prior machine learning knowledge. With only a few clicks you can build your own conversational AI application. You can build your custom application by following one of our quickstarts, or you can use one of our prebuilt domain apps.

Security, Privacy and Compliance: Backed by Azure infrastructure, LUIS offers enterprise-grade security, privacy, and compliance. Your data remains yours; you can delete your data at any time. Your data is encrypted while it's in storage. Learn more about this here.

Integration: easily integrate your LUIS app with other Microsoft services like Microsoft Bot framework, QnA Maker, and Speech service.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis>

Translator

Translator is a cloud-based neural machine translation service that is part of the Azure Cognitive Services family of REST APIs. Translator can be used with any operating system and powers many Microsoft products and services used by thousands of businesses worldwide to perform language translation and other language-related operations. In this overview, you'll learn how Translator can enable you to build intelligent, multi-language solutions for your applications across all supported languages.

Translator features and development options

The following features are supported by the Translator service. Use the links in this table to learn more about each feature and browse the API references.

Feature	Description	Development options
<i>Text Translation</i>	Execute text translation between supported source and target languages in real time.	REST API Text translation Docker container —currently in gated preview.
<i>Document Translation</i>	Translate batch and complex files while preserving the structure and format of the original documents.	REST API Client-library SDK
<i>Custom Translator</i>	Build customized models to translate domain- and industry-specific language, terminology, and style.	Custom Translator portal

<https://docs.microsoft.com/en-us/azure/cognitive-services/translator/translator-info-overview>

QnA Maker

QnA Maker is a cloud-based Natural Language Processing (NLP) service that allows you to create a natural conversational layer over your data. It is used to find the most appropriate answer for any input from your custom knowledge base (KB) of information.

QnA Maker is commonly used to build conversational client applications, which include social media applications, chat bots, and speech-enabled desktop applications.

QnA Maker doesn't store customer data. All customer data (question answers and chat logs) is stored in the region the customer deploys the dependent service instances in.

When to use QnA Maker

When you have static information - Use QnA Maker when you have static information in your knowledge base of answers. This knowledge base is custom to your needs, which you've built with documents such as PDFs and URLs.

When you want to provide the same answer to a request, question, or command - when different users submit the same question, the same answer is returned.

When you want to filter static information based on meta-information - add metadata tags to provide additional filtering options relevant to your client application's users and the information. Common metadata information includes chat-chat, content type or format, content purpose, and content freshness.

When you want to manage a bot conversation that includes static information - your knowledge base takes a user's conversational text or command and answers it. If the answer is part of a pre-determined conversation flow, represented in your knowledge base with multi-turn context, the bot can easily provide this flow.

<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/overview/overview>

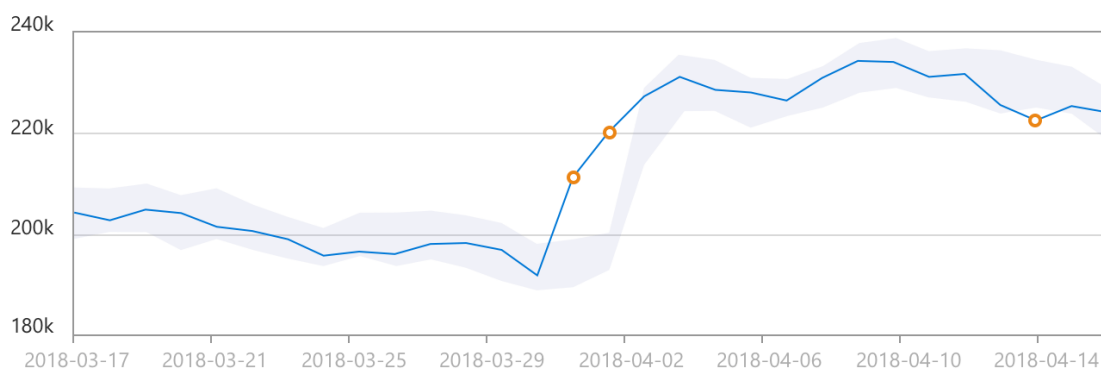
Select the appropriate cognitive Service for a decision support solution

Service Name	Service Description	Quickstart
Anomaly Detector	Anomaly Detector allows you to monitor and detect abnormalities in your time series data.	Anomaly Detector quickstart
Content Moderator	Content Moderator provides monitoring for possible offensive, undesirable, and risky content.	Content Moderator quickstart
Personalizer	Personalizer allows you to choose the best experience to show to your users, learning from their real-time behavior.	Personalizer quickstart

<https://docs.microsoft.com/en-us/azure/cognitive-services/what-are-cognitive-services>

Anomaly Detector

The Anomaly Detector API enables you to monitor and detect abnormalities in your time series data without having to know machine learning. The Anomaly Detector API's algorithms adapt by automatically identifying and applying the best-fitting models to your data, regardless of industry, scenario, or data volume. Using your time series data, the API determines boundaries for anomaly detection, expected values, and which data points are anomalies.



Using the Anomaly Detector doesn't require any prior experience in machine learning, and the REST API enables you to easily integrate the service into your applications and processes.

<https://docs.microsoft.com/en-us/azure/cognitive-services/anomaly-detector/overview>

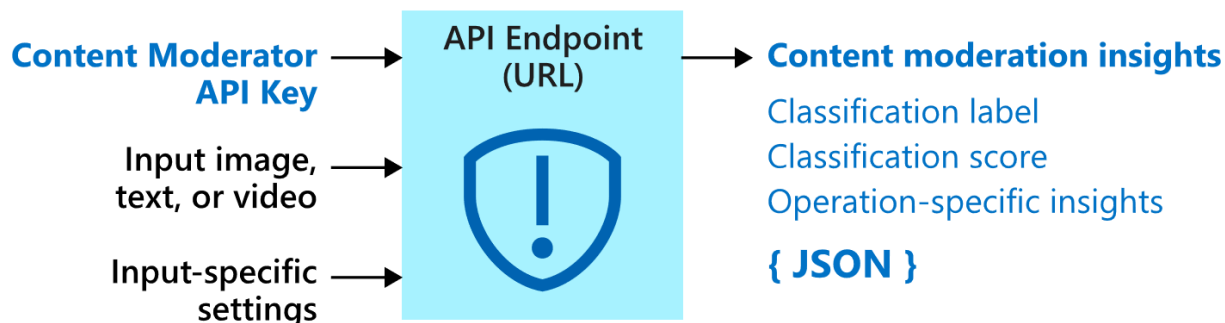
Content Moderator

Azure Content Moderator is an AI service that lets you handle content that is potentially offensive, risky, or otherwise undesirable. It includes the AI-powered content moderation service which scans text, image, and videos and applies content flags automatically.

You may want to build content filtering software into your app to comply with regulations or maintain the intended environment for your users.

The Content Moderator service consists of several web service APIs available through both REST calls and a .NET SDK.

The Content Moderator service includes Moderation APIs, which check content for material that is potentially inappropriate or objectionable.



The following table describes the different types of moderation APIs.

API group	Description
Text moderation	Scans text for offensive content, sexually explicit or suggestive content, profanity, and personal data.
Custom term lists	Scans text against a custom list of terms along with the built-in terms. Use custom lists to block or allow content according to your own content policies.
Image moderation	Scans images for adult or racy content, detects text in images with the Optical Character Recognition (OCR) capability, and detects faces.
Custom image lists	Scans images against a custom list of images. Use custom image lists to filter out instances of commonly recurring content that you don't want to classify again.
Video moderation	Scans videos for adult or racy content

<https://docs.microsoft.com/en-us/azure/cognitive-services/content-moderator/overview>

Personalizer

Azure Personalizer is a cloud-based service that helps your applications choose the best content item to show your users. You can use the Personalizer service to determine what product to suggest to shoppers or to figure out the optimal position for an advertisement. After the content is shown to the user, your application monitors the user's reaction and reports a reward score back to the Personalizer service. This ensures continuous improvement of the machine learning model, and Personalizer's ability to select the best content item based on the contextual information it receives.

Personalizer uses **reinforcement learning** to select the best item (*action*) based on collective behavior and reward scores across all users. Actions are the content items, such as news articles, specific movies, or products.

The **Rank** call takes the action item, along with features of the action, and context features to select the top action item:

- **Actions with features** - content items with features specific to each item
- **Context features** - features of your users, their context or their environment when using your app

The Rank call returns the ID of which content item, **action**, to show to the user, in the **Reward Action ID** field.

The **action** shown to the user is chosen with machine learning models, that try to maximize the total amount of rewards over time.

<https://docs.microsoft.com/en-us/azure/cognitive-services/personalizer/what-is-personalizer>

Select the appropriate cognitive service for a speech solution

The Speech service is the unification of speech-to-text, text-to-speech, and speech translation into a single Azure subscription. It's easy to speech enable your applications, tools, and devices with the Speech CLI, Speech SDK, Speech Studio, or REST APIs.

<https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/overview>

Service Name	Service Description	Quickstart
Speech service	Speech service adds speech-enabled features to applications. Speech service includes various capabilities like speech-to-text, text-to-	Go to the Speech documenta- ti to choose a subservice quickstart.

<https://docs.microsoft.com/en-us/azure/cognitive-services/what-are-cognitive-services>

The following features are part of the Speech service. Use the links in this table to learn more about common use-cases for each feature. You can also browse the API reference.

Service	Feature	Description
Speech-to-text	Real-time speech-to-text	Speech-to-text transcribes or translates audio streams or local files to text in real time that your applications, tools, or devices can consume or display. Use speech-to-text with Language Understanding (LUIS) to derive user intents
	Batch speech-to-text	Batch speech-to-text enables asynchronous speech-to-text transcription of large volumes of speech audio data stored in Azure Blob Storage. In addition to converting speech audio to text, batch speech-to-text allows for diarization and sentiment analysis.

	Multidevice conversation	Connect multiple devices or clients in a conversation to send speech- or text-based messages, with easy support for transcription and translation.
	Conversation transcription	Enables real-time speech recognition, speaker identification, and diarization. It's perfect for transcribing in-person meetings with the ability to distinguish speakers.
	Create custom speech models	If you're using speech-to-text for recognition and transcription in a unique environment, you can create and train custom acoustic, language, and pronunciation models to address ambient noise or industry-specific vocabulary.
	Pronunciation assessment	Pronunciation assessment evaluates speech pronunciation and gives speakers feedback on the accuracy and fluency of spoken audio. With pronunciation assessment, language learners can practice, get instant feedback, and improve their pronunciation so that they can speak and present with confidence.
Text-to-speech	Prebuilt neural voices	Text-to-speech converts input text into humanlike synthesized speech by using the Speech Synthesis Markup Language (SSML). Use neural voices, which are humanlike voices powered by deep neural networks. See Language support.
	Custom neural voices	Create custom neural voice fonts unique to your brand or product.
Speech translation	Speech translation	Speech translation enables real-time, multilanguage translation of speech to your applications, tools, and devices. Use this feature for speech-to-speech and speech-to-text translation.
Language identification	Language identification	Language identification is used to identify languages spoken in audio when compared against a list of supported languages. Use language identification by itself, with speech-to-text recognition, or with speech translation.
Voice assistants	Voice assistants	Voice assistants using the Speech service empower developers to create natural, humanlike conversational interfaces for their applications and experiences.
Speaker recognition	Speaker verification and identification	Speaker recognition provides algorithms that verify and identify speakers by their unique voice characteristics. Speaker recognition is used to answer the question, "Who is speaking?".

Plan and configure security for a Cognitive Services solution

Manage Cognitive Services account keys

Azure Cognitive Services are cloud-based services with REST APIs, and client library SDKs available to help developers build cognitive intelligence into applications without having direct artificial intelligence (AI) or data science skills or knowledge. Azure Cognitive Services enables developers to easily add cognitive features into their applications with cognitive solutions that can see, hear, speak, understand, and even begin to reason.

Get the keys for your resource

1. After your resource is successfully deployed, select Next Steps > Go to resource.

Deployment details [\(Download\)](#)

Resource	Type	Status	Operation details
✓ DocsCognitiveServicesResource	Microsoft.CognitiveServices/accounts	Created	Operation details

Next steps

[Go to resource](#)

2. From the quickstart pane that opens, you can access the resource endpoint and manage keys.
3. If you missed the previous steps or need to find your resource later, go to the Azure services home page. From here you can view recent resources, select **My resources**, or use the search box to find your resource by name.



<https://docs.microsoft.com/en-us/azure/cognitive-services/cognitive-services-apis-create-account>

Manage authentication for a resource

Each request to an Azure Cognitive Service must include an authentication header. This header passes along a subscription key or access token, which is used to validate your subscription for a service or group of services. In this article, you'll learn about three ways to authenticate a request and the requirements for each.

- Authenticate with a single-service or multi-service subscription key
- Authenticate with a token
- Authenticate with Azure Active Directory (AAD)

Authenticate with a single-service subscription key

The first option is to authenticate a request with a subscription key for a specific service, like Translator. The keys are available in the Azure portal for each resource that you've created. To use a subscription key to authenticate a request, it must be passed along as the Ocp-Apim-Subscription-Key header.

Authenticate with a multi-service subscription key

This option also uses a subscription key to authenticate requests. The main difference is that a subscription key is not tied to a specific service, rather, a single key can be used to authenticate requests for multiple Cognitive Services

Authenticate with an authentication token

Some Azure Cognitive Services accept, and in some cases require, an authentication token. Currently, these services support authentication tokens:

- Text Translation API
- Speech Services: Speech-to-text REST API
- Speech Services: Text-to-speech REST API

Authenticate with Azure Active Directory

In the previous sections, we showed you how to authenticate against Azure Cognitive Services using a single-service or multi-service subscription key. While these keys provide a quick and easy path to start development, they fall short in more complex scenarios that require Azure role-based access control (Azure RBAC).

<https://docs.microsoft.com/en-us/azure/cognitive-services/authentication>

Secure Cognitive Services by using Azure Virtual Network

Azure Cognitive Services provides a layered security model. This model enables you to secure your Cognitive Services accounts to a specific subset of networks. When network rules are configured, only applications requesting data over the specified set of networks can access the account. You can limit access to your resources with request filtering. Allowing only requests originating from specified IP addresses, IP ranges or from a list of subnets in Azure Virtual Networks.

An application that accesses a Cognitive Services resource when network rules are in effect requires authorization. Authorization is supported with Azure Active Directory (Azure AD) credentials or with a valid API key.

To secure your Cognitive Services resource, you should first configure a rule to deny access to traffic from all networks (including internet traffic) by default. Then, you should configure rules that grant access to traffic from specific VNets. This configuration enables you to build a secure network boundary for your applications. You can also configure rules to grant access to traffic from select public internet IP address ranges, enabling connections from specific internet or on-premises clients.

Network rules are enforced on all network protocols to Azure Cognitive Services, including REST and WebSocket. To access data using tools such as the Azure test consoles, explicit network rules must be configured. You can apply network rules to existing Cognitive Services resources, or when you create new Cognitive Services resources. Once network rules are applied, they're enforced for all requests.

<https://docs.microsoft.com/en-us/azure/cognitive-services/cognitive-services-virtual-networks>

Plan for a solution that meets responsible AI principles

Microsoft Responsible AI principles in practice

We apply our responsible AI principles with guidance from committees that advise our leadership, engineering, and every team across the company. Learn how responsible AI governance is crucial to guiding AI innovation at Microsoft.

ORA puts Microsoft principles into practice by setting the company-wide rules for responsible AI through the implementation of our governance and public policy work. It has four key functions.

<https://www.microsoft.com/en-us/ai/responsible-ai-resources?activetab=pivot1%3aprimar4>

Create a Cognitive Services Resource

Create a Cognitive Services resource

Azure Cognitive Services are cloud-based services with REST APIs, and client library SDKs available to help developers build cognitive intelligence into applications without having direct artificial intelligence (AI) or data science skills or knowledge. Azure Cognitive Services enables developers to easily add cognitive features into their applications with cognitive solutions that can see, hear, speak, understand, and even begin to reason.

You can access Azure Cognitive Services through two different resources: A multi-service resource, or a single-service one.

- Multi-service resource:
 - ◇Access multiple Azure Cognitive Services with a single key and endpoint.
 - ◇Consolidates billing from the services you use.
- Single-service resource:
 - ◇Access a single Azure Cognitive Service with a unique key and endpoint for each service created.
 - ◇Use the free tier to try out the service.

<https://docs.microsoft.com/en-us/azure/cognitive-services/cognitive-services-apis-create-account>

Configure diagnostic logging for a Cognitive Services resource

To enable diagnostic logging, you'll need somewhere to store your log data. This tutorial uses Azure Storage and Log Analytics.

- **Azure storage** - Retains diagnostic logs for policy audit, static analysis, or backup. The storage account does not have to be in the same subscription as the resource emitting logs as long as the user who configures the setting has appropriate Azure RBAC access to both subscriptions.
- **Log Analytics** - A flexible log search and analytics tool that allows for analysis of raw logs generated by an Azure resource.

Enable diagnostic log collection

1. Navigate to the Azure portal. Then locate and select a Cognitive Services resource. For example, your subscription to Speech Services.
2. Next, from the left-hand navigation menu, locate **Monitoring** and select **Diagnostic**

settings. This screen contains all previously created diagnostic settings for this resource.

3. If there is a previously created resource that you'd like to use, you can select it now. Otherwise, select **+ Add diagnostic setting**.
4. Enter a name for the setting. Then select **Archive to a storage account** and **Send to log Analytics**.
5. When prompted to configure, select the storage account and OMS workspace that you'd like to use to store your diagnostic logs. **Note:** If you don't have a storage account or OMS workspace, follow the prompts to create one.
6. Select **Audit**, **RequestResponse**, and **AllMetrics**. Then set the retention period for your diagnostic log data. If a retention policy is set to zero, events for that log category are stored indefinitely.
7. Click **Save**.

<https://docs.microsoft.com/en-us/azure/cognitive-services/diagnostic-logging>

Manage Cognitive Services costs

This article describes how you plan for and manage costs for Azure Cognitive Services. First, you use the Azure pricing calculator to help plan for Cognitive Services costs before you add any resources for the service to estimate costs. Next, as you add Azure resources, review the estimated costs. After you've started using Cognitive Services resources (for example Speech, Computer Vision, LUIS, Language service, Translator, etc.), use Cost Management features to set budgets and monitor costs. You can also review forecasted costs and identify spending trends to identify areas where you might want to act. Costs for Cognitive Services are only a portion of the monthly costs in your Azure bill. Although this article explains how to plan for and manage costs for Cognitive Services, you're billed for all Azure services and resources used in your Azure subscription, including the third-party services.

Use the Azure pricing calculator to estimate costs before you add Cognitive Services.

Cognitive Services

API: Computer Vision | Region: West US | Resource: \$1

Up to 10 transactions per second.

Features:

Feature	Quantity	Unit Price	Total Price
Tag, Face, GetThumbnail, Color, Image Type	20000	\$0.001	\$20.00
OCR, Adult, Celebrity, and Landmark	10000	\$0.0015	\$15.00
Describe and Recognize Text	100000	\$0.0025	\$250.00

Upfront cost: \$0.00
Monthly cost: \$285.00

Support: Developer \$29.00

Programs and Offers: Microsoft Online Services Agreement

Estimated upfront cost: \$0.00
Estimated monthly cost: \$314.00

Monitor costs

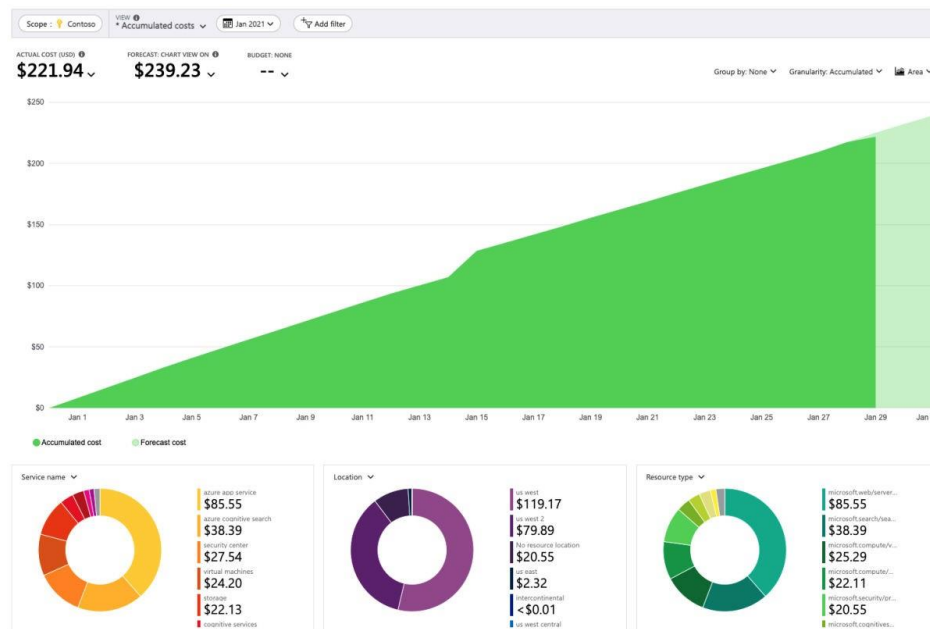
As you use Azure resources with Cognitive Services, you incur costs. Azure resource usage unit costs vary by time intervals (seconds, minutes, hours, and days) or by unit usage (bytes, megabytes, and so on). As soon as use of a Cognitive Service (or Cognitive Services) starts, costs are incurred and you can see the costs in cost analysis.

When you use cost analysis, you view Cognitive Services costs in graphs and tables for different time intervals. Some examples are by day, current and prior month, and year. You also view costs against budgets and forecasted costs. Switching to longer views over time can help you identify spending trends. And you see where overspending might have occurred. If you've created budgets, you can also easily see where they're exceeded.

To view Cognitive Services costs in cost analysis:

1. Sign in to the Azure portal.
2. Open the scope in the Azure portal and select **Cost analysis** in the menu. For example, go to **Subscriptions**, select a subscription from the list, and then select **Cost analysis** in the menu. Select **Scope** to switch to a different scope in cost analysis.
3. By default, cost for services are shown in the first donut chart. Select the area in the chart labeled Cognitive Services.

Actual monthly costs are shown when you initially open cost analysis. Here's an example showing all monthly usage costs.



To narrow costs for a single service, like Cognitive Services, select Add filter and then select Service name. Then, select Cognitive Services.

Create budgets

You can create budgets to manage costs and create alerts that automatically notify stakeholders of spending anomalies and overspending risks. Alerts are based on spending compared to budget and cost thresholds. Budgets and alerts are created for Azure subscriptions and resource groups, so they're useful as part of an overall cost monitoring strategy.

Budgets can be created with filters for specific resources or services in Azure if you want more granularity present in your monitoring. Filters help ensure that you don't accidentally create new resources that cost you more money. |

<https://docs.microsoft.com/en-us/azure/cognitive-services/plan-manage-costs>

Monitor a cognitive service

In Azure portal, the Monitoring tab in the Overview page for each Azure Cognitive Search service includes a brief summary of key metrics, including query volume, latency, and throttled queries. This data is collected automatically, and is available for analysis as soon as you create the resource.

While these metrics are helpful, they are a subset of the monitoring and diagnostic data available for Azure Cognitive Search. You can enable additional types of data collection with some configuration. Additional data collection is supported through Azure Monitor.

What is Azure Monitor?

Azure Cognitive Search creates monitoring data using Azure Monitor which is a full stack monitoring service in Azure that provides a complete set of features to monitor your Azure resources.

If you're not already familiar with monitoring Azure services, start with the article [Monitoring Azure resources with Azure Monitor](#) which describes the following concepts:

- What Azure Monitor is and how it's integrated into the portal for other Azure services
- The types of data collected by Azure Monitor for Azure resources
- Azure Monitor tools used to collect and analyze data

The following sections build on this article by describing the specific data gathered from Azure Cognitive Search and providing examples for configuring data collection and analyzing this data with Azure tools.

Monitoring data

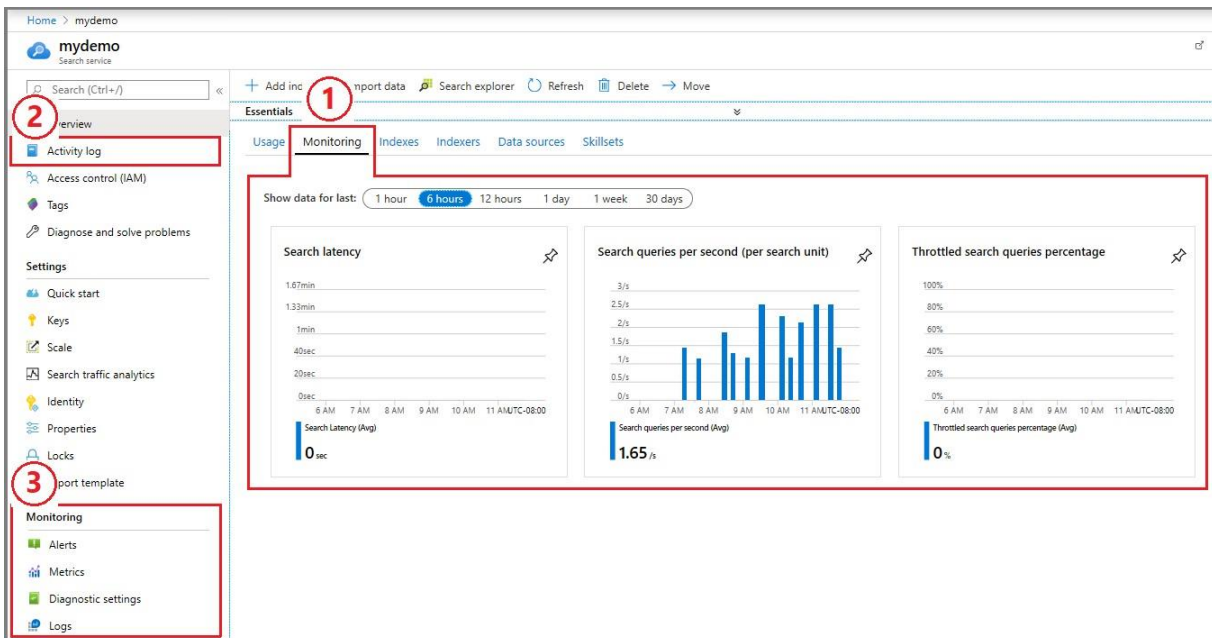
Azure Cognitive Search collects the same kinds of monitoring data as other Azure resources that are described in Monitoring data from Azure resources. See the data reference for detailed information on the metrics and logs metrics created by Azure Cognitive Search.

In addition to the resource logs collected by Azure Monitor, you can also obtain system data from the search service itself, including statistics, counts, and status:

- Service Statistics (REST)
- Index Statistics (REST)
- Document Counts (REST)
- Indexer Status (REST)

The system information above information can also be read from the Azure portal. For REST calls, use an admin API key and Postman or another REST client.

On Azure portal pages, check the Usage and Monitoring tabs for counts and metrics. Commands on the left-navigation provide access to configuration and data exploration pages.



<https://docs.microsoft.com/en-us/azure/search/monitor-azure-cognitive-search>

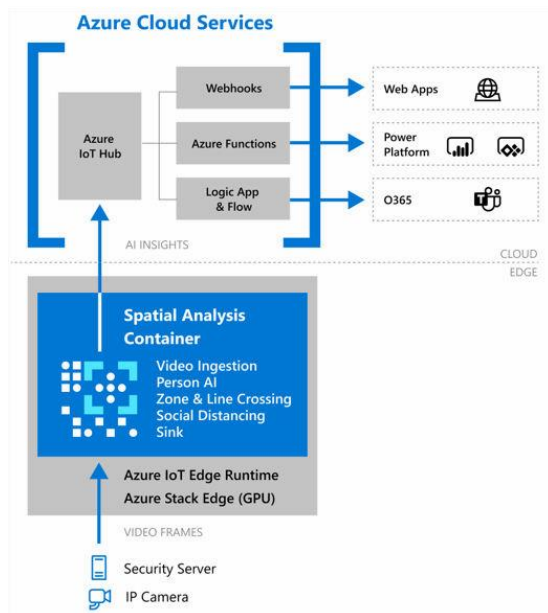
Implement a privacy policy in Cognitive Services

Computer Vision spatial analysis was designed with compliance, privacy, and security in mind, however, the customer is responsible for its use and the implementation of this technology. For example, it is your responsibility to:

- Install and position cameras in your locations, and in doing so take care to avoid sensitive areas and limit collection of data not needed by Spatial Analysis.
- Inform people in your locations with conspicuous disclosure about use of video data and use of AI.
- Comply with all applicable laws and regulations in your jurisdiction.

Each of Microsoft's customers has a different environment and space occupants, and some jurisdictions may have special laws enacted, such as CCTV-related rules and/or licensing requirements or prohibiting use of cameras in sensitive areas. Before using spatial analysis, you must have all the proper rights to use your cameras and video images, including, where required, the necessary consent from individuals for the use of processing images to generate insights in compliance with the laws and regulations of your jurisdiction.

Data and privacy for spatial analysis



Spatial analysis runs in a container customers deploy on an edge compute device. The container runs a pipeline that ingests video frames, detects people in the video, tracks the people as they move around over time, and generates AI insights as people interact with regions of interest. The customer directs where the AI insights are stored. For example, AI insights can flow to the cloud via Azure IoT Hub or Event Grid.

<https://docs.microsoft.com/en-us/legal/cognitive-services/computer-vision/compliance-privacy-security-2>

Plan and Implement Cognitive Services Containers

Identify when to deploy to a container

Azure Cognitive Services provides several Docker containers that let you use the same APIs that are available in Azure, on-premises. Using these containers gives you the flexibility to bring Cognitive Services closer to your data for compliance, security or other operational reasons. Container support is currently available for a subset of Azure Cognitive Services.

Containerization is an approach to software distribution in which an application or service, including its dependencies & configuration, is packaged together as a container image. With little or no modification, a container image can be deployed on a container host. Containers are isolated from each other and the underlying operating system, with a smaller footprint than a virtual machine. Containers can be instantiated from container images for short-term tasks, and removed when no longer needed.

Features and benefits

- **Immutable infrastructure:** Enable DevOps teams' to leverage a consistent and reliable set of known system parameters, while being able to adapt to change. Containers provide the flexibility to pivot within a predictable ecosystem and avoid configuration drift.
- **Control over data:** Choose where your data gets processed by Cognitive Services. This can be essential if you can't send data to the cloud but need access to Cognitive Services APIs. Support consistency in hybrid environments – across data, management, identity, and security.
- **Control over model updates:** Flexibility in versioning and updating of models deployed in their solutions.
- **Portable architecture:** Enables the creation of a portable application architecture that can be deployed on Azure, on-premises and the edge. Containers can be deployed directly to Azure Kubernetes Service, Azure Container Instances, or to a Kubernetes cluster deployed to Azure Stack. For more information, see [Deploy Kubernetes to Azure Stack](#).
- **High throughput / low latency:** Provide customers the ability to scale for high throughput and low latency requirements by enabling Cognitive Services to run physically close to their application logic and data. Containers don't cap transactions per second (TPS) and can be

made to scale both up and out to handle demand if you provide the necessary hardware resources.

- **Scalability:** With the ever growing popularity of containerization and container orchestration software, such as Kubernetes; scalability is at the forefront of technological advancements. Building on a scalable cluster foundation, application development caters to high availability.

<https://docs.microsoft.com/en-us/azure/cognitive-services/cognitive-services-container-support>

Containerize Cognitive Services (including Computer Vision API, Face API, Text Analytics, Speech, Form Recognizer)

By using containers, you can run *some* of the Azure Cognitive Services Speech service APIs in your own environment. Containers are great for specific security and data governance requirements. In this article, you'll learn how to download, install, and run a Speech container.

With Speech containers, you can build a speech application architecture that's optimized for both robust cloud capabilities and edge locality. Several containers are available, which use the same pricing as the cloud-based Azure Speech services.

Container	Features	Latest	Release status
Speech-to-text	Analyzes sentiment and transcribes continuous real-time speech or batch audio recordings with intermediate results.	3.1.0	Generally available
Custom speech-to-text	Using a custom model from the Custom Speech portal, transcribes continuous real-time speech or batch audio recordings into text with intermediate	3.1.0	Generally available
Speech language identification	Detects the language spoken in audio files.	1.5.0	Preview
Neural text-to-speech	Converts text to natural-sounding speech by using deep neural network technology, which allows for more natural synthesized speech.	2.1.1	Generally available

<https://docs.microsoft.com/en-us/azure/cognitive-services/cognitive-services-container-support>

Deploy Cognitive Services Containers in Microsoft Azure

Azure Cognitive Services containers provide the following set of Docker containers, each of which contains a subset of functionality from services in Azure Cognitive Services. You can find instructions and image locations in the tables below. A list of container images is also available.

You must satisfy the following prerequisites before using Azure Cognitive Services containers:

Docker Engine: You must have Docker Engine installed locally. Docker provides packages that configure the Docker environment on macOS, Linux, and Windows. On Windows, Docker must be configured to support Linux containers. Docker containers can also be deployed directly to Azure Kubernetes Service or Azure Container Instances.

Docker must be configured to allow the containers to connect with and send billing data to Azure.

Familiarity with Microsoft Container Registry and Docker: You should have a basic understanding of both Microsoft Container Registry and Docker concepts, like registries, repositories, containers, and container images, as well as knowledge of basic docker commands.

<https://docs.microsoft.com/bs-latn-ba/azure/cognitive-services/cognitive-services-container-support>

Implement Computer Vision Solutions (20-25%)

Analyze images by using the Computer Vision API

Retrieve image descriptions and tags by using the Computer Vision API

Computer Vision can analyze an image and generate a human-readable phrase that describes its contents. The algorithm returns several descriptions based on different visual features, and each description is given a confidence score. The final output is a list of descriptions ordered from highest to lowest confidence.

The image description feature is part of the Analyze Image API. You can call this API through a native SDK or through REST calls. Include Description in the visualFeatures query parameter. Then, when you get the full JSON response, simply parse the string for the contents of the "description" section.

<https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-describing-images>

Computer Vision can return content tags for thousands of recognizable objects, living beings, scenery, and actions that appear in images. Tags are not organized as a taxonomy and do not have inheritance hierarchies. A collection of content tags forms the foundation for an image description displayed as human readable language formatted in complete sentences. When tags are ambiguous or not common knowledge, the API response provides hints to clarify the meaning of the tag in context of a known setting.

After you upload an image or specify an image URL, the Computer Vision algorithm can output tags based on the objects, living beings, and actions identified in the image. Tagging is not limited to the main subject, such as a person in the foreground, but also includes the setting (indoor or outdoor), furniture, tools, plants, animals, accessories, gadgets, and so on.

<https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-tagging-images>

Identify landmarks and celebrities by using the Computer Vision API

In addition to tagging and high-level categorization, Computer Vision also supports further domain-specific analysis using models that have been trained on specialized data.

There are two ways to use the domain-specific models: by themselves (scoped analysis) or as an enhancement to the categorization feature.

Scoped analysis

You can analyze an image using only the chosen domain-specific model by calling the [Models/<model>/Analyze](#) API.

Enhanced categorization analysis

You can also use domain-specific models to supplement general image analysis. You do this as part of high-level categorization by specifying domain-specific models in the *details* parameter of the Analyze API call.

In this case, the 86-category taxonomy classifier is called first. If any of the detected categories have a matching domain-specific model, the image is passed through that model as well and the results are added.

<https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-detecting-domain-content>

Detect brands in images by using the Computer Vision API

Brand detection is a specialized mode of object detection that uses a database of thousands of global logos to identify commercial brands in images or video. You can use this feature, for example, to discover which brands are most popular on social media or most prevalent in media product placement.

The Computer Vision service detects whether there are brand logos in a given image; if there are, it returns the brand name, a confidence score, and the coordinates of a bounding box around the logo.

The built-in logo database covers popular brands in consumer electronics, clothing, and more. If you find that the brand you're looking for is not detected by the Computer Vision service, you could also try creating and training your own logo detector using the Custom Vision service.

<https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-brand-detection>

Moderate content in images by using the Computer Vision API

Computer Vision can detect adult material in images so that developers can restrict the display of these images in their software. Content flags are applied with a score between zero and one so developers can interpret the results according to their own preferences.

Content flag definitions

The "adult" classification contains several different categories:

- **Adult** images are explicitly sexual in nature and often show nudity and sexual acts.
- **Racy** images are sexually suggestive in nature and often contain less sexually explicit content than images tagged as **Adult**.
- **Gory** images show blood/gore.

You can detect adult content with the Analyze Image API. When you add the value of **Adult** to the `visualFeatures` query parameter, the API returns three boolean properties—`isAdultContent`, `isRacyContent`, and `isGoryContent`—in its JSON response. The method also returns corresponding properties—`adultScore`, `racyScore`, and `goreScore`—which represent confidence scores between zero and one for each respective category.

<https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-detecting-adult-content>

Generate thumbnails by using the Computer Vision API

A thumbnail is a reduced-size representation of an image. Thumbnails are used to represent images and other data in a more economical, layout-friendly way. The Computer Vision API uses smart cropping, together with resizing the image, to create intuitive thumbnails for a given image.

The Computer Vision thumbnail generation algorithm works as follows:

1. Remove distracting elements from the image and identify the *area of interest*—the area of the image in which the main object(s) appears.
2. Crop the image based on the identified *area of interest*.
3. Change the aspect ratio to fit the target thumbnail dimensions.

When you upload an image, the Computer Vision API analyzes it to determine the *area of interest*. It can then use this region to determine how to crop the image. The cropping operation, however, will always match the desired aspect ratio if one is specified.

You can also get the raw bounding box coordinates of this same *area of interest* by calling the **areaOfInterest** API instead. You can then use this information to modify the original image however you wish.

<https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-generating-thumbnails>

Extract Text from Images

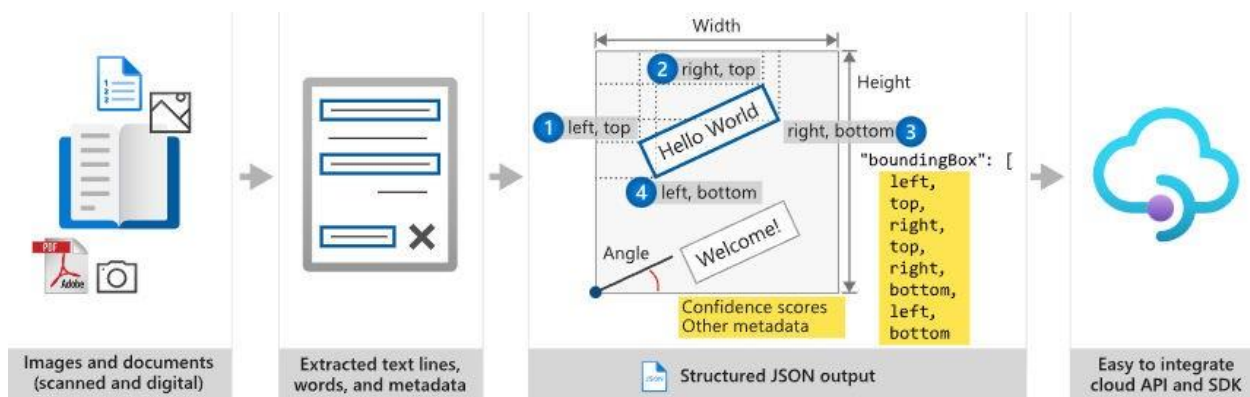
Extract text from images or PDFs by using the Computer Vision service

Optical character recognition (OCR) allows you to extract printed or handwritten text from images, such as photos of street signs and products, as well as from documents—invoices, bills, financial reports, articles, and more. Microsoft's OCR technologies support extracting printed text in several languages.

This documentation contains the following types of articles:

- The quickstarts are step-by-step instructions that let you make calls to the service and get results in a short period of time.
- The how-to guides contain instructions for using the service in more specific or customized ways.

The Computer Vision Read API is Azure's latest OCR technology (learn what's new) that extracts printed text (in several languages), handwritten text (in several languages), digits, and currency symbols from images and multi-page PDF documents. It's optimized to extract text from text-heavy images and multi-page PDF documents with mixed languages. It supports extracting both printed and handwritten text in the same image or document.



<https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview-ocr>

Convert handwritten text by using Ink Recognizer

Use this quickstart to begin sending digital ink strokes to the Ink Recognizer API. This C# application sends an API request containing JSON-formatted ink stroke data, and gets the response.

While this application is written in C#, the API is a RESTful web service compatible with most programming languages.

Typically you would call the API from a digital inking app. This quickstart sends ink stroke data for the following handwritten sample from a JSON file.

Create an Ink Recognizer resource

Azure Cognitive Services are represented by Azure resources that you subscribe to. Create a resource for Ink Recognizer using the Azure portal.

After creating a resource, get your endpoint and key by opening your resource on the Azure portal, and clicking **Quick start**.

Create two environment variables:

- `INK_RECOGNITION_SUBSCRIPTION_KEY` - The subscription key for authenticating your requests.
- `INK_RECOGNITION_ENDPOINT` - The endpoint for your resource. It will look like this: `https://<your-custom-subdomain>.api.cognitive.microsoft.com`

<https://docs.microsoft.com/en-us/previous-versions/azure/cognitive-services/ink-recognizer/quickstarts/csharp>

Extract information from forms or receipts by using the pre-built receipt model in Form Recognizer

The receipt model combines powerful Optical Character Recognition (OCR) capabilities with deep learning models to analyze and extract key information from sales receipts. Receipts can be of various formats and quality including printed and handwritten receipts. The API extracts key information such as merchant name, merchant phone number, transaction date, tax, and transaction total and returns a structured JSON data representation.

Try Form Recognizer

See how data, including time and date of transactions, merchant information, and amount totals, is extracted from receipts using the Form Recognizer Studio or our Sample Labeling tool. You'll need the following resources:

- An Azure subscription—you can create one for free
- A Form Recognizer instance in the Azure portal. You can use the free pricing tier (F0) to try the service. After your resource deploys, select **Go to resource** to get your API key and endpoint.

<https://docs.microsoft.com/en-us/azure/applied-ai-services/form-recognizer/concept-receipt>

Build and optimize a custom model for Form Recognizer

When you use the Form Recognizer custom model, you provide your own training data to the Train Custom Model operation, so that the model can train to your industry-specific forms. Follow this guide to learn how to collect and prepare data to train the model effectively.

- You need at least five filled-in forms of the same type.
- If you want to use manually labeled training data, you must start with at least five filled-in forms of the same type. You can still use unlabeled forms in addition to the required data set.

<https://docs.microsoft.com/en-us/azure/applied-ai-services/form-recognizer/build-training-data-set>

Train a custom model

This section demonstrates how to train a model with your own data. A trained model can output structured data that includes the key/value relationships in the original form document. After you train the model, you can test and retrain it and eventually use it to reliably extract data from more forms according to your needs.

Train a model without labels

Train custom models to analyze all the fields and values found in your custom forms without manually labeling the training documents. The following method trains a model on a given set of documents and prints the model's status to the console.

The returned CustomFormModel object contains information on the form types the model can analyze and the fields it can extract from each form type. The following code block prints this information to the console.

Finally, return the trained model ID for use in later steps.

<https://docs.microsoft.com/en-us/azure/applied-ai-services/form-recognizer/how-to-guides/try-sdk-rest-api>

Extract facial information from images

Detect faces in an image by using the Face API

This guide demonstrates how to use the face detection API to extract attributes like age, emotion, or head pose from a given image. You'll learn the different ways to configure the behavior of this API to meet your needs.

The code snippets in this guide are written in C# by using the Azure Cognitive Services Face client library. The same functionality is available through the REST API.

Submit data to the service

To find faces and get their locations in an image, call the DetectWithUrlAsync or DetectWithStreamAsync method. DetectWithUrlAsync takes a URL string as input, and DetectWithStreamAsync takes the raw byte stream of an image as input.

Besides face rectangles and landmarks, the face detection API can analyze several conceptual attributes of a face. For a full list, see the Face attributes conceptual section.

To analyze face attributes, set the `detectionModel` parameter to `DetectionModel.Detection01` and the `returnFaceAttributes` parameter to a list of `FaceAttributeType` Enum values.

Face landmark results

```
C#  
  
foreach (var face in faces2)  
{  
    var landmarks = face.FaceLandmarks;  
  
    double noseX = landmarks.NoseTip.X;  
    double noseY = landmarks.NoseTip.Y;  
  
    double leftPupilX = landmarks.PupilLeft.X;  
    double leftPupilY = landmarks.PupilLeft.Y;  
  
    double rightPupilX = landmarks.PupilRight.X;  
    double rightPupilY = landmarks.PupilRight.Y;
```

The following code demonstrates how you might retrieve the locations of the nose and pupils:

You also can use face landmark data to accurately calculate the direction of the face. For example, you can define the rotation of the face as a vector from the center of the mouth to the center of the eyes.

When you know the direction of the face, you can rotate the rectangular face frame to align it more properly. To crop faces in an image, you can programmatically rotate the image so the faces always appear upright.

<https://docs.microsoft.com/en-us/azure/cognitive-services/face/face-api-how-to-topics/howtodetectfacesinimage>

Recognize faces in an image by using the Face API

Get started with facial recognition using the Face client library for .NET. Follow these steps to install the package and try out the example code for basic tasks. The Face service provides you with access to advanced algorithms for detecting and recognizing human faces in images.

Use the Face client library for .NET to:

- Detect and analyze faces
- Identify a face
- Find similar faces

<https://docs.microsoft.com/en-us/azure/cognitive-services/face/quickstarts/client-libraries>

Analyze facial attributes by using the Face API

Microsoft Azure provides multiple cognitive services that you can use to detect and analyze faces, including:

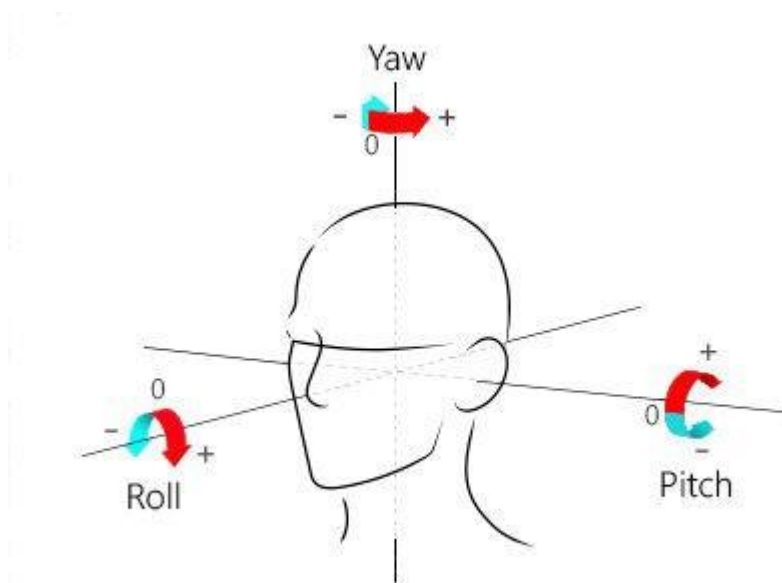
- **Computer Vision**, which offers face detection and some basic face analysis, such as determining age.
- **Video Indexer**, which you can use to detect and identify faces in a video
- **Face**, which offers pre-built algorithms that can detect, recognize, and analyze faces.

<https://docs.microsoft.com/en-in/learn/modules/detect-analyze-faces/2-face-analysis-azure>

Attributes

Attributes are a set of features that can optionally be detected by the Face - Detect API. The following attributes can be detected:

- **Accessories**. Whether the given face has accessories. This attribute returns possible accessories including headwear, glasses, and mask, with confidence score between zero and one for each accessory.
- **Age**. The estimated age in years of a particular face.
- **Blur**. The blurriness of the face in the image. This attribute returns a value between zero and one and an informal rating of low, medium, or high.
- **Emotion**. A list of emotions with their detection confidence for the given face. Confidence scores are normalized, and the scores across all emotions add up to one. The emotions returned are happiness, sadness, neutral, anger, contempt, disgust, surprise, and fear.
- **Exposure**. The exposure of the face in the image. This attribute returns a value between zero and one and an informal rating of underExposure, goodExposure, or overExposure.
- **Facial hair**. The estimated facial hair presence and the length for the given face.
- **Gender**. The estimated gender of the given face. Possible values are male, female, and genderless.
- **Glasses**. Whether the given face has eyeglasses. Possible values are NoGlasses, ReadingGlasses, Sunglasses, and Swimming Goggles.
- **Hair**. The hair type of the face. This attribute shows whether the hair is visible, whether baldness is detected, and what hair colors are detected.
- **Head pose**. The face's orientation in 3D space. This attribute is described by the roll, yaw, and pitch angles in degrees, which are defined according to the right-hand rule. The order of three angles is roll-yaw-pitch, and each angle's value range is from -180 degrees to 180 degrees. 3D orientation of the face is estimated by the roll, yaw, and pitch angles in order. See the following diagram for angle mappings:



Copyright (c) Microsoft. All rights reserved.

- **Makeup.** Whether the face has makeup. This attribute returns a Boolean value for eyeMakeup and lipMakeup.
- **Mask.** Whether the face is wearing a mask. This attribute returns a possible mask type, and a Boolean value to indicate whether nose and mouth are covered.
- **Noise.** The visual noise detected in the face image. This attribute returns a value between zero and one and an informal rating of low, medium, or high.
- **Occlusion.** Whether there are objects blocking parts of the face. This attribute returns a Boolean value for eyeOccluded, foreheadOccluded, and mouthOccluded.
- **Smile.** The smile expression of the given face. This value is between zero for no smile and one for a clear smile.
- **QualityForRecognition** The overall image quality regarding whether the image being used in the detection is of sufficient quality to attempt face recognition on. The value is an informal rating of low, medium, or high. Only "high" quality images are recommended for person enrollment, and quality at or above "medium" is recommended for identification scenarios.

<https://docs.microsoft.com/en-us/azure/cognitive-services/face/concepts/face-detection>

Analyze facial attributes by using the Face API

Given query face's faceId, to search the similar-looking faces from a faceId array, a face list or a large face list. faceId array contains the faces created by Face - Detect With Url or Face - Detect With Stream, which will expire at the time specified by faceIdTimeToLive after creation.

A "faceListId" is created by FaceList - Create containing persistedFaceIds that will not expire. And a "largeFaceListId" is created by LargeFaceList - Create containing persistedFaceIds that will also not expire. Depending on the input the returned similar faces list contains faceIds or

persistedFaceIds ranked by similarity.

Find similar has two working modes, "matchPerson" and "matchFace". "matchPerson" is the default mode that it tries to find faces of the same person as possible by using internal same-person thresholds. It is useful to find a known person's other photos. Note that an empty list will be returned if no faces pass the internal thresholds. "matchFace" mode ignores same-person thresholds and returns ranked similar faces anyway, even the similarity is low. It can be used in the cases like searching celebrity-looking faces.

The 'recognitionModel' associated with the query face's faceId should be the same as the 'recognitionModel' used by the target faceId array, face list or large face list.

<https://docs.microsoft.com/en-us/rest/api/faceapi/face/find-similar>

Implement Image Classification by Using the Custom Vision Service

Label images by using the Computer Vision Portal

When you tag images for a Custom Vision model, the service uses the latest trained iteration of the model to predict the labels of untagged images. It then shows these predictions as suggested tags, based on the selected confidence threshold and prediction uncertainty. You can then either confirm or change the suggestions, speeding up the process of manually tagging the images for training.

Smart Labeler workflow

The following steps show you how to use Smart Labeler:

1. Upload all of your training images to your Custom Vision project.
2. Label part of your data set, choosing an equal number of images for each tag.
3. Start the training process.
4. When training is complete, navigate to the Untagged view and select the Get suggested tags button on the left pane.
5. In the popup window that appears, set the number of images for which you want suggestions. You should only get initial tag suggestions for a portion of the untagged images. You'll get better tag suggestions as you iterate through this process.
6. Confirm the suggested tags, fixing any that aren't correct.
7. Start the training process again.
8. Repeat the preceding steps until you're satisfied with the suggestion quality.

<https://docs.microsoft.com/en-in/azure/cognitive-services/custom-vision-service/suggested-tags>

Train a custom image classification model in the Custom Vision Portal

As a minimum, we recommend you use at least 30 images per tag in the initial training set. You'll also want to collect a few extra images to test your model once it's trained.

In order to train your model effectively, use images with visual variety. Select images that vary by:

- camera angle
- lighting
- background
- visual style
- individual/grouped subject(s)
- size
- Type

Additionally, make sure all of your training images meet the following criteria:

- .jpg, .png, .bmp, or .gif format
- no greater than 6MB in size (4MB for prediction images)

no less than 256 pixels on the shortest edge; any images shorter than this will be automatically scaled up by the Custom Vision Service

Upload and tag images

In this section, you'll upload and manually tag images to help train the classifier.

1. To add images, select **Add images** and then select **Browse local files**. Select **Open** to move to tagging. Your tag selection will be applied to the entire group of images you've selected to upload, so it's easier to upload images in separate groups according to their applied tags. You can also change the tags for individual images after they've been uploaded.
2. To create a tag, enter text in the My Tags field and press Enter. If the tag already exists, it will appear in a dropdown menu. In a multilabel project, you can add more than one tag to your images, but in a multiclass project you can add only one. To finish uploading the images, use the Upload [number] files button.
3. Select Done once the images have been uploaded.

To train the classifier, select the **Train** button. The classifier uses all of the current images to create a model that identifies the visual qualities of each tag. This process can take several minutes.

<https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/getting-started-build-a-classifier>

Train a custom image classification model by using the SDK

Get started with the Custom Vision client library for .NET. Follow these steps to install the package and try out the example code for building an image classification model. You'll create a project, add tags, train the project, and use the project's prediction endpoint URL to programmatically test it. Use this example as a template for building your own image recognition app.

This method creates the first training iteration in the project. It queries the service until training is completed.

```
private static void TrainProject(CustomVisionTrainingClient trainingApi, Project project)
{
    // Now there are images with tags start training the project
    Console.WriteLine("\tTraining");
    iteration = trainingApi.TrainProject(project.Id);

    // The returned iteration will be in progress, and can be queried periodically to see when it has completed
    while (iteration.Status == "Training")
    {
        Console.WriteLine("Waiting 10 seconds for training to complete...");
        Thread.Sleep(10000);

        // Re-query the iteration to get it's updated status
        iteration = trainingApi.GetIteration(project.Id, iteration.Id);
    }
}
```

<https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/quickstarts/image-classification>

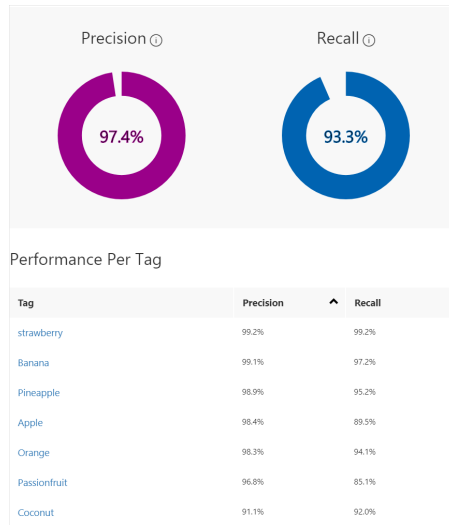
Manage model iterations

Each time you train your classifier, you create a new iteration with updated performance metrics. You can view all of your iterations in the left pane of the Performance tab. You'll also find the Delete button, which you can use to delete an iteration if it's obsolete. When you delete an iteration, you delete any images that are uniquely associated with it.

<https://docs.microsoft.com/en-in/azure/cognitive-services/custom-vision-service/getting-started-build-a-classifier>

After you've trained your model, you can test images programmatically by submitting them to the prediction API endpoint. In this guide, you'll learn how to call the prediction API to score an image. You'll learn the different ways you can configure the behavior of this API to meet your needs.

Evaluate classification model metrics



After training has completed, the model's performance is estimated and displayed. The Custom Vision Service uses the images that you submitted for training to calculate precision and recall, using a process called k-fold cross validation. Precision and recall are two different measurements of the effectiveness of a classifier:

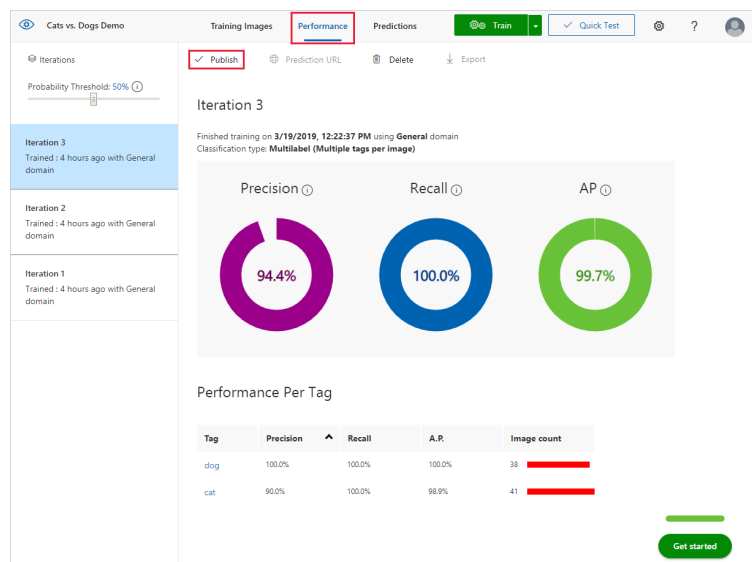
Precision indicates the fraction of identified classifications that were correct. For example, if the model identified 100 images as dogs, and 99 of them were actually of dogs, then the precision would be 99%.

Recall indicates the fraction of actual classifications that were correctly identified. For example, if there were actually 100 images of apples, and the model identified 80 as apples, the recall would be 80%.

Publish a trained iteration of a model

To submit images to the Prediction API, you'll first need to publish your iteration for prediction, which can be done by selecting Publish and specifying a name for the published iteration. This will make your model accessible to the Prediction API of your Custom Vision Azure resource.

Once your model has been successfully published, you'll see a "Published" label appear next to your iteration in the left-hand sidebar, and its name will appear in the description of the iteration.



<https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/use-prediction-api#publish-your-trained-iteration>

Export a model in an appropriate format for a specific target

Custom Vision Service allows classifiers to be exported to run offline. You can embed your exported classifier into an application and run it locally on a device for real-time classification.

Export options

Custom Vision Service supports the following exports:

- **TensorFlow** for **Android**.
- **TensorFlow.js** for JavaScript frameworks like React, Angular, and Vue. This will run on both **Android** and **iOS** devices.
- **CoreML** for **iOS11**.
- **ONNX** for **Windows ML**, **Android**, and **iOS**.
- **Vision AI Developer Kit**.
- A **Docker container** for Windows, Linux, or ARM architecture. The container includes a TensorFlow model and service code to use the Custom Vision API.

<https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/export-your-model>

Consume a classification model from a client application

Deploying an Azure Machine Learning model as a web service creates a REST API endpoint. You can send data to this endpoint and receive the prediction returned by the model. In this document, learn how to create clients for the web service by using C#, Go, Java, and Python.

You create a web service when you deploy a model to your local environment, Azure Container Instances, Azure Kubernetes Service, or field-programmable gate arrays (FPGA). You retrieve the URI used to access the web service by using the Azure Machine Learning SDK. If authentication is enabled, you can also use the SDK to get the authentication keys or tokens.

The general workflow for creating a client that uses a machine learning web service is:

1. Use the SDK to get the connection information.
2. Determine the type of request data used by the model.
3. Create an application that calls the web service.

<https://docs.microsoft.com/en-us/azure/machine-learning/how-to-consume-web-service>

Deploy image classification custom models to containers

Azure IoT Edge can make your IoT solution more efficient by moving workloads out of the cloud and to the edge. This capability lends itself well to services that process a lot of data, like computer vision models. The Custom Vision Service lets you build custom image classifiers and deploy them to devices as containers. Together, these two services enable you to find insights from images or video streams without having to transfer all of the data off site first. Custom Vision provides a classifier that compares an image against a trained model to generate insights.

For example, Custom Vision on an IoT Edge device could determine whether a highway is experiencing higher or lower traffic than normal, or whether a parking garage has available parking spots in a row. These insights can be shared with another service to take action.

Build an image classifier with Custom Vision

To build an image classifier, you need to create a Custom Vision project and provide training images. For more information about the steps that you take in this section, see [How to build a classifier with Custom Vision](#).

Once your image classifier is built and trained, you can export it as a Docker container and deploy it to an IoT Edge device.

<https://docs.microsoft.com/en-us/azure/iot-edge/tutorial-deploy-custom-vision?view=iotedge-2018-06>

Implement an Object Detection Solution by Using the Custom Vision Service

Label images with bounding boxes by using the Computer Vision Portal

Tag images for multi-class classification

If your project is of type "Image Classification Multi-Class," you'll assign a single tag to the entire image. To review the directions at any time, go to the **Instructions** page and select **View detailed instructions**.

If you realize that you made a mistake after you assign a tag to an image, you can fix it. Select the "X" on the label that's displayed below the image to clear the tag. Or, select the image and choose another class. The newly selected value will replace the previously applied tag.

Tag images for multi-label classification

If you're working on a project of type "Image Classification Multi-Label," you'll apply one *or more* tags to an image. To see the project-specific directions, select **Instructions** and go to **View detailed instructions**.

Select the image that you want to label and then select the tag. The tag is applied to all the selected images, and then the images are deselected. To apply more tags, you must reselect the images. The following animation shows multi-label tagging:

1. **Select all** is used to apply the "Ocean" tag.
2. A single image is selected and tagged "Closeup."
3. Three images are selected and tagged "Wide angle."

To correct a mistake, select the "X" to clear an individual tag or select the images and then select the tag, which clears the tag from all the selected images. This scenario is shown here. Selecting "Land" will clear that tag from the two selected images.

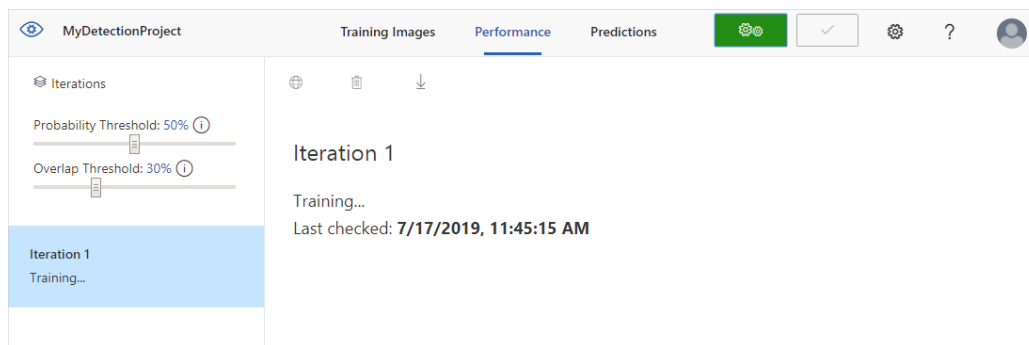
Azure will only enable the Submit button after you've applied at least one tag to each image. Select Submit to save your work.

<https://docs.microsoft.com/en-us/azure/machine-learning/how-to-label-data>

Train a custom object detection model by using the Custom Vision Portal

To train the detector model, select the Train button. The detector uses all of the current images and their tags to create a model that identifies each tagged object. This process can take several minutes.

The training process should only take a few minutes. During this time, information about the training process is displayed in the Performance tab.



<https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/get-started-build-detector>

Train a custom object detection model by using the SDK

Get started with the Custom Vision client library for .NET. Follow these steps to install the package and try out the example code for building an object detection model. You'll create a project, add tags, train the project on sample images, and use the project's prediction endpoint URL to programmatically test it. Use this example as a template for building your own image recognition app.

This method creates the first training iteration in the project. It queries the service until training is completed.

```
private void TrainProject(CustomVisionTrainingClient trainingApi, Project project)
{
    // Now there are images with tags start training the project
    Console.WriteLine("\t\tTraining");
    var iteration = trainingApi.TrainProject(project.Id);

    // The returned iteration will be in progress, and can be queried periodically to see when it has completed
    while (iteration.Status == "Training")
    {
        Thread.Sleep(1000);

        // Re-query the iteration to get its updated status
        iteration = trainingApi.GetIteration(project.Id, iteration.Id);
    }
}
```

<https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/quickstarts/object-detection>

Manage model iterations

Each time you train your detector, you create a new iteration with its own updated performance metrics. You can view all of your iterations in the left pane of the Performance tab. In the left pane you'll also find the Delete button, which you can use to delete an iteration if it's obsolete. When you delete an iteration, you delete any images that are uniquely associated with it.

<https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/get-started-build-detector>

Evaluate object detection model metrics

After training has completed, the model's performance is calculated and displayed. The Custom Vision service uses the images that you submitted for training to calculate precision, recall, and mean average precision. Precision and recall are two different measurements of the effectiveness of a detector:

- Precision indicates the fraction of identified classifications that were correct. For example, if the model identified 100 images as dogs, and 99 of them were actually of dogs, then the precision would be 99%.
- Recall indicates the fraction of actual classifications that were correctly identified. For example, if there were actually 100 images of apples, and the model identified 80 as apples, the recall would be 80%.
- Mean average precision is the average value of the average precision (AP). AP is the area under the precision/recall curve (precision plotted against recall for each prediction made).

<https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/get-started-build-detector>

Publish a trained iteration of a model

This method makes the current iteration of the model available for querying. You can use the model name as a reference to send prediction requests. You need to enter your own value for predictionResourceId. You can find the prediction resource ID on the resource's Properties tab in the Azure portal, listed as Resource ID.

<https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/quickstarts/object-detection>

Consume an object detection model from a client application

Use the object detection model in Power Automate

1. Sign in to Power Automate.
2. Select My flows in the left pane, and then select New flow > Instant cloud flow.
3. Name your flow, select Manually trigger a flow under Choose how to trigger this flow, and then select Create.
4. Expand Manually trigger a flow, and then select +Add an input > File as the input type.
5. Replace File Content with My image (also known as the title).

6. Select **+ New step > AI Builder**, and then select **Detect and count objects in images** in the list of actions.
7. Select the object detection model you want to use.
8. In the **Image** input, select **My Image** from the **Dynamic content** list
9. To retrieve the name of the detected object or objects on the image:
 - a. Select New step.
 - b. Search for the successive action you want your flow to perform, for example add a row into an Excel table or send an email.
 - c. Select any of the successive actions' inputs, and then select Detected object name in the Dynamic content list.

Congratulations! You've created a flow that uses an object detection AI Builder model. Select Save on the top right, and then select Test to try out your flow.

<https://docs.microsoft.com/en-us/ai-builder/object-detection-model-in-flow>

Deploy custom object detection models to containers

You must satisfy the following prerequisites before using Azure Cognitive Services containers:

Docker Engine: You must have Docker Engine installed locally. Docker provides packages that configure the Docker environment on macOS, Linux, and Windows. On Windows, Docker must be configured to support Linux containers. Docker containers can also be deployed directly to Azure Kubernetes Service or Azure Container Instances.

Docker must be configured to allow the containers to connect with and send billing data to Azure.

Familiarity with Microsoft Container Registry and Docker: You should have a basic understanding of both Microsoft Container Registry and Docker concepts, like registries, repositories, containers, and container images, as well as knowledge of basic docker commands.

<https://docs.microsoft.com/en-us/azure/cognitive-services/cognitive-services-container-support>

Analyze video by using Azure Video Analyzer for Media

Process a video

When you're creating a Video Analyzer for Media account, you choose between:

- A free trial account. Video Analyzer for Media provides up to 600 minutes of free indexing

to website users and up to 2,400 minutes of free indexing to API users.

- A paid option where you're not limited by a quota. You create a Video Analyzer for Media account that's connected to your Azure subscription and an Azure Media Services account.
- You pay for indexed minutes.

When you're uploading videos by using the API, you have the following options:

- Upload your video from a URL (preferred).
- Send the video file as a byte array in the request body.
- Use existing an Azure Media Services asset by providing the asset ID. This option is supported in paid accounts only.

Upload and index a video by using the website

Sign in on the Video Analyzer for Media website, and then select Upload.

After your video is uploaded, Video Analyzer for Media starts indexing and analyzing the video. After Video Analyzer for Media is done analyzing, you get an email with a link to your video. The email also includes a short description of what was found in your video (for example: people, topics, optical character recognition).

Upload and index a video by using the API

You can use the Upload Video API to upload and index your videos based on a URL. The code sample that follows includes the commented-out code that shows how to upload the byte array.

<https://docs.microsoft.com/en-us/azure/media-services/video-indexer/upload-index-videos>

Extract insights from a video

Video Analyzer for Media includes predefined models that can recognize well-known celebrities and brands, and transcribe spoken phrases into text. You can extend the recognition capabilities of Video Analyzer by creating custom models for:

- **People.** Add images of the faces of people you want to recognize in videos, and train a model. Video Indexer will then recognize these people in all of your videos.
- **Language.** If your organization uses specific terminology that may not be in common usage, you can train a custom model to detect and transcribe it.
- **Brands.** You can train a model to recognize specific names as brands, for example to identify products, projects, or companies that are relevant to your business.
- **Animated characters.** In addition to recognizing human individuals, you may want to be able to detect the presence of individual animated characters in a video.

<https://docs.microsoft.com/en-us/learn/modules/extract-insights-from-videos-with-video-indexer-service/>

Moderate content in a video

Content moderation is the process of monitoring for possible offensive, undesirable, and risky content. Content Moderator, a Cognitive Services product, combines machine-assisted content moderation APIs and human review tool for images, text, and videos into a complete content moderation solution. In this episode, we will get an overview of Content Moderator and learn about its video moderation capabilities. We will cover the API and the human review capabilities in Content Moderator.

Image moderation

Enhance your ability to detect potentially offensive or unwanted images through machine-learning based classifiers, custom lists and optical character recognition (OCR).

Text moderation

Use content filtering to detect potential profanity in more than 100 languages, flag text that may be deemed inappropriate depending on context (in public preview) and match text against your custom lists. Content Moderator also helps check for personally identifiable information (PII).

Video moderation

Enable machine-assisted detection of possible adult and racy content in videos. The video moderation service (in public preview) is available as part of Azure Media Services.

Human review tool

The best content moderation results come from humans and machines working together. Use the review tool when prediction confidence can be improved or tempered with a real-world context.

<https://channel9.msdn.com/Shows/AI-Show/Video-Moderation-with-Content-Moderator>

Customize the Brands model used by Video Indexer

Azure Video Analyzer for Media (formerly Video Indexer) supports brand detection from speech and visual text during indexing and reindexing of video and audio content. The brand detection feature identifies mentions of products, services, and companies suggested by Bing's brands database. For example, if Microsoft is mentioned in a video or audio content or if it shows up in visual text in a video, Video Analyzer for Media detects it as a brand in the content. Brands are disambiguated from other terms using context.

Brand detection is useful in a wide variety of business scenarios such as contents archive and discovery, contextual advertising, social media analysis, retail compete analysis, and many

more. Video Analyzer for Media brand detection enables you to index brand mentions in speech and visual text, using Bing's brands database as well as with customization by building a custom Brands model for each Video Analyzer for Media account. The custom Brands model feature allows you to select whether or not Video Analyzer for Media will detect brands from the Bing brands database, exclude certain brands from being detected (essentially creating a list of unapproved brands), and include brands that should be part of your model that might not be in Bing's brands database (essentially creating a list of approved brands). The custom Brands model that you create will only be available in the account in which you created the model.

<https://docs.microsoft.com/en-us/azure/azure-video-analyzer/video-analyzer-for-media-docs/customize-brands-model-overview>

Customize the Language model used by Video Indexer by using the Custom Speech Service

Azure Video Analyzer for Media (formerly Video Indexer) supports automatic speech recognition through integration with the Microsoft Custom Speech Service. You can customize the Language model by uploading adaptation text, namely text from the domain whose vocabulary you'd like the engine to adapt to. Once you train your model, new words appearing in the adaptation text will be recognized, assuming default pronunciation, and the Language model will learn new probable sequences of words. See the list of supported by Video Analyzer for Media languages in supported languages.

Let's take a word that is highly specific, like "Kubernetes" (in the context of Azure Kubernetes service), as an example. Since the word is new to Video Analyzer for Media, it is recognized as "communities". You need to train the model to recognize it as "Kubernetes". In other cases, the words exist, but the Language model is not expecting them to appear in a certain context. For example, "container service" is not a 2-word sequence that a non-specialized Language model would recognize as a specific set of words.

You have the option to upload words without context in a list in a text file. This is considered partial adaptation. Alternatively, you can upload text file(s) of documentation or sentences related to your content for better adaptation.

You can use the Video Analyzer for Media APIs or the website to create and edit custom Language models, as described in topics in the Next steps section of this topic.

<https://docs.microsoft.com/en-us/azure/azure-video-analyzer/video-analyzer-for-media-docs/customize-language-model-overview>

Customize the Person model used by Video Indexer

Customize a Person model with the Video Analyzer for Media API

Azure Video Analyzer for Media (formerly Video Indexer) supports face detection and celebrity recognition for video content. The celebrity recognition feature covers about one million faces based on commonly requested data source such as IMDB, Wikipedia, and top LinkedIn influencers. Faces that aren't recognized by the celebrity recognition feature are detected but left unnamed. After you upload your video to Video Analyzer for Media and get results back, you can go back and name the faces that weren't recognized. Once you label a face with a name, the face and name get added to your account's Person model. Video Analyzer for Media will then recognize this face in your future videos and past videos.

You can use the Video Analyzer for Media API to edit faces that were detected in a video, as described in this topic. You can also use the Video Analyzer for Media website, as described in Customize Person model using the Video Analyzer for Media website.

<https://docs.microsoft.com/en-us/azure/azure-video-analyzer/video-analyzer-for-media-docs/customize-person-model-with-api>

Customize a Person model with the Video Analyzer for Media website

You can manage the faces that are detected and people that are recognized in the videos that you index by editing and deleting faces.

Deleting a face removes a specific face from the insights of the video.

Editing a face renames a face that's detected and possibly recognized in your video. When you edit a face in your video, that name is saved as a person entry in the Person model that was assigned to the video during upload and indexing.

If you don't assign a Person model to the video during upload, your edit is saved in your account's Default person model.

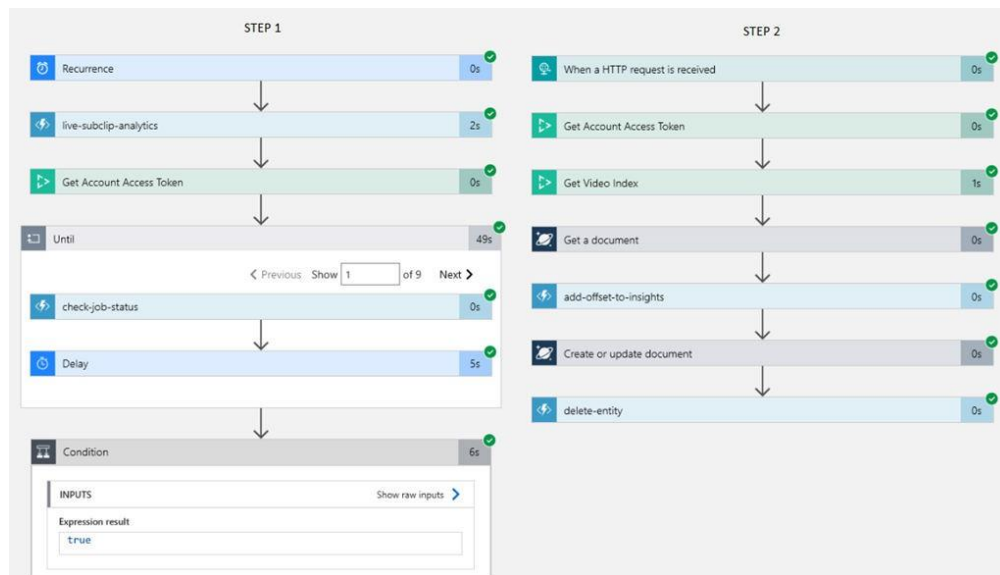
<https://docs.microsoft.com/en-us/azure/azure-video-analyzer/video-analyzer-for-media-docs/customize-person-model-with-website>

Extract insights from a live stream of video data

The stream analysis solution at hand, uses Azure Functions and two Logic Apps to process a live program from a live channel in Azure Media Services with Video Analyzer for Media and displays the result with Azure Media Player showing the near real-time resulted stream.

In high level, it is comprised of two main steps. The first step runs every 60 seconds, and takes a subclip of the last 60 seconds played, creates an asset from it and indexes it via Video Analyzer for Media. Then the second step is called once indexing is complete. The insights captured are processed, sent to Azure Cosmos DB, and the subclip indexed is deleted.

The sample player plays the live stream and gets the insights from Azure Cosmos DB, using a dedicated Azure Function. It displays the metadata and thumbnails in sync with the live video.



<https://docs.microsoft.com/en-us/azure/azure-video-analyzer/video-analyzer-for-media-docs/live-stream-analysis>

Implement Natural Language Processing Solutions (20-25%)

Analyze Text by Using the Text Analytics Service

Retrieve and process key phrases

Key phrase extraction is one of the features offered by Azure Cognitive Service for Language, a collection of machine learning and AI algorithms in the cloud for developing intelligent applications that involve written language. Use key phrase extraction to quickly identify the main concepts in text. For example, in the text "The food was delicious and the staff were wonderful.", key phrase extraction will return the main topics: "food" and "wonderful staff".

<https://docs.microsoft.com/en-in/azure/cognitive-services/language-service/key-phrase-extraction/overview>

Retrieve and process entity information (people, places, urls, etc.)

Named Entity Recognition (NER) is one of the features offered by Azure Cognitive Service for Language, a collection of machine learning and AI algorithms in the cloud for developing intelligent applications that involve written language. The NER feature can identify and categorize entities in unstructured text. For example: people, places, organizations, and quantities.

Typical workflow

To use this feature, you submit data for analysis and handle the API output in your application. Analysis is performed as-is, with no additional customization to the model used on your data.

1. Create an Azure Language resource, which grants you access to the features offered by Azure Cognitive Service for Language. It will generate a password (called a key) and an endpoint URL that you'll use to authenticate API requests.
2. Create a request using either the REST API or the client library for C#, Java, JavaScript, and Python. You can also send asynchronous calls with a batch request to combine API requests for multiple features into a single call.
3. Send the request containing your data as raw unstructured text. Your key and endpoint will be used for authentication.

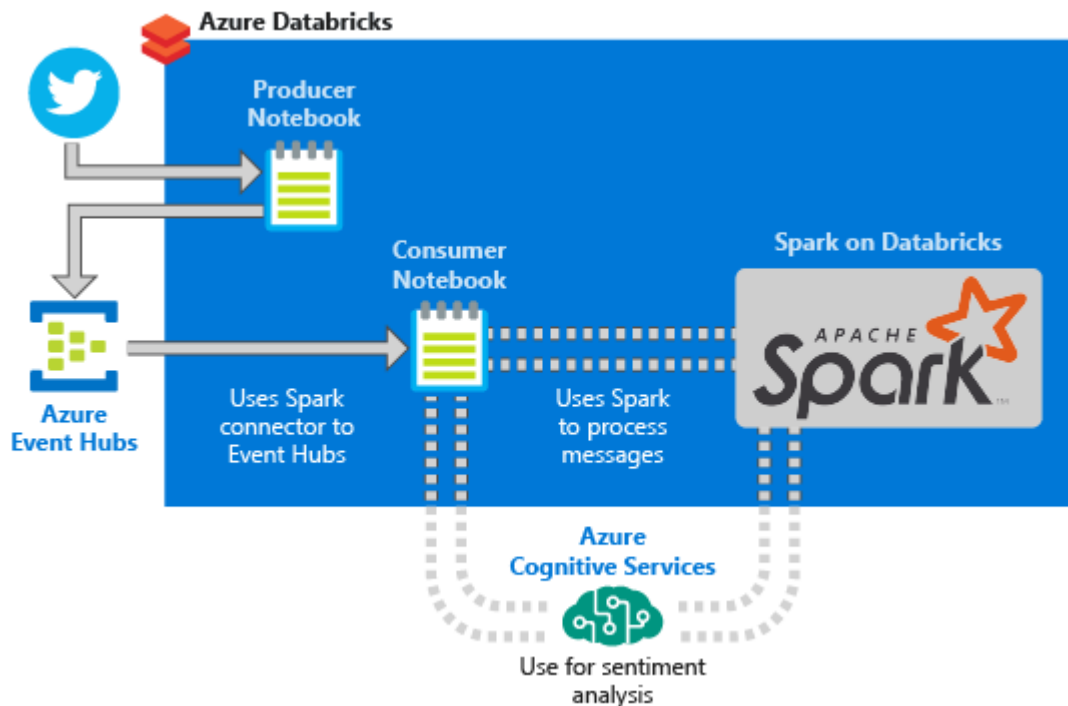
<https://docs.microsoft.com/en-us/azure/cognitive-services/language-service/named-entity-recognition/overview>

Retrieve and process sentiment

In this tutorial, you learn how to run sentiment analysis on a stream of data using Azure Databricks in near real time. You set up data ingestion system using Azure Event Hubs. You consume the messages from Event Hubs into Azure Databricks using the Spark Event Hubs connector. Finally, you use Cognitive Service APIs to run sentiment analysis on the streamed data.

By the end of this tutorial, you would have streamed tweets from Twitter that have the term "Azure" in them and ran sentiment analysis on the tweets.

The following illustration shows the application flow:



<https://docs.microsoft.com/en-us/azure/databricks/scenarios/databricks-sentiment-analysis-cognitive-services>

Detect the language used in text

Language detection is one of the features offered by Azure Cognitive Service for Language, a collection of machine learning and AI algorithms in the cloud for developing intelligent applications that involve written language. Language detection can detect the language a document is written in, and returns a language code for a wide range of languages, variants, dialects, and some regional/cultural languages.

To use this feature, you submit data for analysis and handle the API output in your application. Analysis is performed as-is, with no additional customization to the model used on your data.

1. Create an Azure Language resource, which grants you access to the features offered by Azure Cognitive Service for Language. It will generate a password (called a key) and an endpoint URL that you'll use to authenticate API requests.

2. Create a request using either the REST API or the client library for C#, Java, JavaScript, and Python. You can also send asynchronous calls with a batch request to combine API requests for multiple features into a single call.
3. Send the request containing your data as raw unstructured text. Your key and endpoint will be used for authentication.
4. Stream or store the response locally.

<https://docs.microsoft.com/en-us/azure/cognitive-services/language-service/language-detection/overview>

Manage Speech by Using the Speech Service |

Implement text-to-speech

Text-to-speech enables your applications, tools, or devices to convert text into humanlike synthesized speech. The text-to-speech capability is also known as speech synthesis. Use humanlike prebuilt neural voices out of the box, or create a custom neural voice that's unique to your product or brand.

Text-to-speech includes the following features:

Feature	Summary	Demo
Prebuilt neural voice (called <i>Neural</i> on the pricing page)	Highly natural out-of-the-box voices. Create an Azure account and Speech service subscription, and then use the Speech SDK or visit the Speech Studio portal and select prebuilt neural voices to get started. Check the pricing details.	Check the voice samples and determine the right voice for your business
Custom neural voice (called <i>Custom Neural</i> on the pricing page)	Easy-to-use self-service for creating a natural brand voice, with limited access for responsible use. Create an Azure account and Speech service subscription (with the S0 tier), and apply to use the custom neural feature. After you've been granted access, visit the Speech Studio portal and select Custom Voice to get started. Check the pricing details.	Check the voice samples.

The text-to-speech feature of the Speech service on Azure has been fully upgraded to the neural text-to-speech engine. This engine uses deep neural networks to make the voices of computers nearly indistinguishable from the recordings of people. With the clear articulation of words, neural text-to-speech significantly reduces listening fatigue when users interact with AI systems.

The patterns of stress and intonation in spoken language are called prosody. Traditional text-to-speech systems break down prosody into separate linguistic analysis and acoustic prediction steps that are governed by independent models. That can result in muffled, buzzy voice synthesis.

<https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/text-to-speech>

Customize text-to-speech

Custom Neural Voice is a set of online tools that you use to create a recognizable, one-of-a-kind voice for your brand. All it takes to get started are a handful of audio files and the associated transcriptions. See if Custom Neural Voice supports your language and region.

A Speech service subscription is required before you can use Custom Neural Voice. Follow these instructions to create a Speech service subscription in Azure. If you don't have an Azure account, you can sign up for a new one.

Once you've created an Azure account and a Speech service subscription, you'll need to sign in to Speech Studio and connect your subscription.

1. Get your Speech service subscription key from the Azure portal.
2. Sign in to Speech Studio, and then select **Custom Voice**.
3. Select your subscription and create a speech project.
4. If you want to switch to another Speech subscription, select the **cog** icon at the top.

<https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/how-to-custom-voice>

In Prepare training data, you learned about the different data types you can use to train a custom neural voice, and the different format requirements. After you've prepared your data and the voice talent verbal statement, you can start to upload them to Speech Studio. In this article, you learn how to train a custom neural voice through the Speech Studio portal.

A voice talent is an individual or target speaker whose voices are recorded and used to create neural voice models. Before you create a voice, define your voice persona and select a right voice talent. For details on recording voice samples, see the tutorial.

To train a neural voice, you must create a voice talent profile with an audio file recorded by the voice talent, consenting to the usage of their speech data to train a custom voice model. When you prepare your recording script, make sure you include the statement sentence. You can find the statement in multiple languages on GitHub. The language of the verbal statement must be the same as your recording.

Upload this audio file to the Speech Studio as shown in the following screenshot. You create a voice talent profile, which is used to verify against your training data when you create a voice model.

The following steps assume that you've prepared the voice talent verbal consent files. Go to Speech Studio to select a Custom Neural Voice project, and then follow these steps to create a voice talent profile.

1. Go to **Text-to-Speech > Custom Voice > select a project**, and select **Set up voice talent**.
2. Select **Add voice talent**.
3. Next, to define voice characteristics, select **Target scenario**. Then describe your **Voice characteristics**.
4. Then, go to Upload voice talent statement, and follow the instruction to upload the voice talent statement you've prepared beforehand.
5. Go to Review and create, review the settings, and select Submit.

<https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/how-to-custom-voice-create-voice>

Implement speech-to-text

Speech-to-text, also known as speech recognition, enables real-time transcription of audio streams into text. Your applications, tools, or devices can consume, display, and take action on this text as command input.

This feature uses the same recognition technology that Microsoft uses for Cortana and Office products. It seamlessly works with the translation and text-to-speech offerings of the Speech service. For a full list of available speech-to-text languages, see Language and voice support for the Speech service.

The speech-to-text feature defaults to using the Universal Language Model. This model was trained through Microsoft-owned data and is deployed in the cloud. It's optimal for conversational and dictation scenarios.

When you're using speech-to-text for recognition and transcription in a unique environment,

you can create and train custom acoustic, language, and pronunciation models. Customization is helpful for addressing ambient noise or industry-specific vocabulary.

<https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-to-text>

Improve speech-to-text accuracy

Tenant Model (Custom Speech with Microsoft 365 data) is an opt-in service for Microsoft 365 enterprise customers that automatically generates a custom speech recognition model from your organization's Microsoft 365 data. The model is optimized for technical terms, jargon, and people's names, all in a secure and compliant way.

Get a Speech subscription key

To use your tenant model with the Speech SDK, you need a Speech resource and its associated subscription key.

1. Sign in to the [Azure portal](#).
2. Select **Create a resource**.
3. In the **Search** box, type **Speech**.
4. In the results list, select **Speech**, and then select **Create**.
5. Follow the onscreen instructions to create your resource. Make sure that:
 - **Location** is set to either **eastus** or **westus**.
 - **Pricing tier** is set to **S0**.
6. Select **Create**.
7. After a few minutes, your resource is created. The subscription key is available in the Overview section for your resource

Create a language model

After your admin has enabled Tenant Model for your organization, you can create a language model that's based on your Microsoft 365 data.

1. Sign in to Speech Studio.
2. At the top right, select Settings (gear icon), and then select Tenant Model settings.
3. Select Opt in.

<https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/tutorial-tenant-model>

Translate Language

Translate text by using the Translator service

Custom Translator is a feature of the Microsoft Translator service, which enables Translator enterprises, app developers, and language service providers to build customized neural machine translation (NMT) systems. The customized translation systems seamlessly integrate into existing applications, workflows, and websites.

Translation systems built with Custom Translator are available through the same cloud-based, secure, high performance, highly scalable Microsoft Translator Text API V3, that powers billions of translations every day.

The platform enables users to build and publish custom translation systems to and from English. Custom Translator supports more than three dozen languages that map directly to the languages available for NMT. For a complete list, see Translator language support.

<https://docs.microsoft.com/en-us/azure/cognitive-services/translator/custom-translator/overview>

Translate speech-to-speech by using the Speech service

Create a speech translation configuration

To call the Speech service by using the Speech SDK, you need to create a `SpeechTranslationConfig` instance. This class includes information about your subscription, like your key and associated region, endpoint, host, or authorization token.

You can initialize `SpeechTranslationConfig` in a few ways:

- With a subscription: pass in a key and the associated region.
- With an endpoint: pass in a Speech service endpoint. A key or authorization token is optional.
- With a host: pass in a host address. A key or authorization token is optional.
- With an authorization token: pass in an authorization token and the associated region.

<https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/get-started-speech-translation>

Translate speech-to-text by using the Speech service

Recognize speech from a microphone

Follow these steps to create a new console application and install the Speech SDK.

1. Open a command prompt where you want the new project, and create a console application with the .NET CLI.
2. Install the Speech SDK in your new project with the .NET CLI.
3. Replace the contents of Program.cs with the following code.
4. In Program.cs, replace YourSubscriptionKey with your Speech resource key, and replace YourServiceRegion with your Speech resource region.
5. To change the speech recognition language, replace en-US with another supported language. For example, es-ES for Spanish (Spain). The default language is en-us if you don't specify a language. For details about how to identify one of multiple languages that might be spoken, see language identification.

<https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/get-started-speech-to-text>

Build an Initial Language Model by Using Language Understanding Service (LUIS)

Create intents and entities based on a schema, and then add utterances

Intents in your LUIS app

An intent represents a task or action the user wants to perform. It is a purpose or goal expressed in a user's utterance.

Define a set of intents that corresponds to actions users want to take in your application. For example, a travel app would have several intents:

Travel app intents	Example utterances
BookFlight	"Book me a flight to Rio next week" "Fly me to Rio on the 24th" "I need a plane ticket next Sunday to Rio de Janeiro"
Greeting	"Hi" "Hello" "Good morning"

CheckWeather	"What's the weather like in Boston?"
	"Show me the forecast for this weekend"
None	"Get me a cookie recipe"
	"Did the Lakers win?"

Prebuilt intents

LUIS provides prebuilt intents and their utterances for each of its prebuilt domains. Intents can be added without adding the whole domain. Adding an intent is the process of adding an intent and its utterances to your app. Both the intent name and the utterance list can be modified.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/intents>

Add intents to determine user intention of utterances

Add an intent to your app

1. Sign in to the LUIS portal, and select your **Subscription** and **Authoring resource** to see the apps assigned to that authoring resource.
2. Open your app by selecting its name on the **My Apps** page.
3. Select **Build** from the top navigation bar, then select **Intents** from the left panel.
4. On the **Intents** page, select **+ Create**.
5. In the **Create new intent** dialog box, enter the intent name, for example *ModifyOrder*, and select **Done**.

Add an example utterance

Example utterances are text examples of user questions or commands. To teach Language Understanding (LUIS) when to predict the intent, you need to add example utterances. Carefully consider each utterance you add. Each utterance added should be different than the examples that are already added to the intent..

On the intent details page, enter a relevant utterance you expect from your users, such as "I want to change my pizza order to large please" in the text box below the intent name, and then press Enter.

LUIS converts all utterances to lowercase and adds spaces around tokens, such as hyphens.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/how-to/intents>

Create complex hierarchical entities

An entity is an item or an element that is relevant to the user's intent. Entities define data that can be extracted from the utterance and is essential to complete a user's required action.

Utterance	Intent predicted	Entities	Explanation
Hello, how are you?	Greeting	-	Nothing to extract.
I want to order a small	orderPizza	'small'	'Size' entity is extracted as 'small'.
Turn off bedroom light	turnOff	'bedroom'	'Room' entity is extracted as 'bedroom'.
Check balance in my savings account ending in	checkBalance	'savings', '4406'	'accountType' entity is extracted as 'savings' and 'accountNumber' entity is
Buy 3 tickets to New York	buyTickets	'3', 'New York'	'ticketsCount' entity is extracted as '3' and 'Destination' entity is extracted as

Entities are optional but recommended. You don't need to create entities for every concept in your app, only when:

- The client application needs the data, or
- The entity acts as a hint or signal to another entity or intent. To learn more about entities as Features go to Entities as features.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/entities>

use this instead of roles

An app owner can add contributors to apps. These contributors can modify the model, train, and publish the app. Once you have migrated your account, contributors are managed in the Azure portal for the authoring resource, using the Access control (IAM) page. Add a user, using the collaborator's email address and the contributor role.

Add contributor to Azure authoring resource

You have migrated if your LUIS authoring experience is tied to an Authoring resource on the Manage -> Azure resources page in the LUIS portal.

In the Azure portal, find your Language Understanding (LUIS) authoring resource. It has the type LUIS.Authoring. In the resource's Access Control (IAM) page, add the role

of contributor for the user that you want to contribute. For detailed steps, see Assign Azure roles using the Azure portal.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-how-to-collaborate>

Train and deploy a model

Training is the process of teaching your Language Understanding (LUIS) app to extract intent and entities from user utterances. Training comes after you make updates to the model, such as: adding, editing, labeling, or deleting entities, intents, or utterances.

Training and testing an app is an iterative process. After you train your LUIS app, you test it with sample utterances to see if the intents and entities are recognized correctly. If they're not, you should make updates to the LUIS app, then train and test again.

Training is applied to the active version in the LUIS portal.

How to train interactively

Before you start training your app in the LUIS portal, make sure every intent has at least one utterance. You must train your LUIS app at least once to test it.

1. Access your app by selecting its name on the My Apps page.
2. In your app, select Train in the top-right part of the screen.
3. When training is complete, a notification appears at the top of the browser.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/how-to/train-test>

Iterate on and optimize a language model by using LUIS

Implement phrase lists

Features are a necessary part of your schema design. LUIS supports both phrase lists and models as features:

- Phrase list feature
- Model (intent or entity) as a feature

Create a phrase list for a concept

A phrase list is a list of words or phrases that describe a concept. A phrase list is applied as a

case-insensitive match at the token level.

When adding a phrase list, you can set the feature to global. A global feature applies to the entire app.

When to use a phrase list

Use a phrase list when you need your LUIS app to generalize and identify new items for the concept. Phrase lists are like domain-specific vocabulary. They enhance the quality of understanding for intents and entities.

How to use a phrase list

With a phrase list, LUIS considers context and generalizes to identify items that are similar to, but aren't, an exact text match. Follow these steps to use a phrase list:

1. Start with a machine-learning entity:
2. Add example utterances.
3. Label with a machine-learning entity.
4. Add a phrase list:
5. Add words with similar meaning. Don't add every possible word or phrase. Instead, add a few words or phrases at a time. Then retrain and publish.
6. Review and add suggested words.

A typical scenario for a phrase list

A typical scenario for a phrase list is to boost words related to a specific idea.

Medical terms are a good example of words that might need a phrase list to boost their significance. These terms can have specific physical, chemical, therapeutic, or abstract meanings. LUIS won't know the terms are important to your subject domain without a phrase list.

For example, to extract the medical terms:

1. Create example utterances and label medical terms within those utterances.
2. Create a phrase list with examples of the terms within the subject domain. This phrase list should include the actual term you labeled and other terms that describe the same concept. Add the phrase list to the entity or subentity that extracts the concept used in the phrase list. The most common scenario is a component (child) of a machine-learning entity. If the phrase list should be applied across all intents or entities, mark the phrase list as a global phrase list. The **enabledForAllModels** flag controls this model scope in the API.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/patterns-features>

Implement a model as a feature (i.e. prebuilt entities)

Model as a feature helps another model

You can add a model (intent or entity) as a feature to another model (intent or entity). By adding an existing intent or entity as a feature, you're adding a well-defined concept that has labeled examples.

When adding a model as a feature, you can set the feature as:

- **Required.** A required feature must be found for the model to be returned from the prediction endpoint
- **Global.** A global feature applies to the entire app.

Required features

A required feature has to be found in order for the model to be returned from the prediction endpoint. Use a required feature when you know your incoming data must match the feature.

If the utterance text doesn't match the required feature, it won't be extracted.

A required feature uses a non-machine-learning entity:

- Regular-expression entity
- List entity
- Prebuilt entity

If you're confident that your model will be found in the data, set the feature as required. A required feature doesn't return anything if it isn't found.

Continuing with the example of the shipping address:

Shipping address (machine learned entity)

- Street number (subentity)
- Street address (subentity)
- Street name (subentity)
- City (subentity)
- State or Province (subentity)
- Country/Region (subentity)
- Postal code (subentity)

Required feature using prebuilt entities

Prebuilt entities such as city, state, and country/region are generally a closed set of lists,

meaning they don't change much over time. These entities could have the relevant recommended features and those features could be marked as required. However, the `isRequired` flag is only related to the entity it is assigned to and doesn't affect the hierarchy. If the prebuilt sub-entity feature is not found, this will not affect the detection and return of the parent entity.

As an example of a required feature, consider you want to detect addresses. You might consider making a street number a requirement. This would allow a user to enter "1 Microsoft Way" or "One Microsoft Way", and both would resolve to the numeral "1" for the street number sub-entity. See the prebuilt entity article for more information.

Required feature using list entities

A list entity is used as a list of canonical names along with their synonyms. As a required feature, if the utterance doesn't include either the canonical name or a synonym, then the entity isn't returned as part of the prediction endpoint.

Suppose that your company only ships to a limited set of countries/regions. You can create a list entity that includes several ways for your customer to reference the country/region. If LUIS doesn't find an exact match within the text of the utterance, then the entity (that has the required feature of the list entity) isn't returned in the prediction.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/patterns-features>

Manage punctuation and diacritics

Punctuation normalization

The following utterances show how punctuation impacts utterances

With punctuation set to False

Hmm..... I will take the cappuccino

With punctuation set to True

Hmm I will take the cappuccino

Diacritics normalization

The following utterances show how diacritics normalization impacts utterances:

With diacritics set to false

quiero tomar una piña colada

With diacritics set to true

quiero tomar una pina colada

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-reference-application-settings#diacritics-normalization>

Implement active learning

Active Learning

The process of reviewing endpoint utterances for correct predictions is called Active learning. Active learning captures queries that are sent to the endpoint, and selects user utterances that it is unsure of. You review these utterances to select the intent and mark the entities for these real-world utterances. Then you can accept these changes into your app's example utterances, then train and publish the app. This helps LUIS identify utterances more accurately.

To enable active learning, you must log user queries. This is accomplished by calling the endpoint query with the log=true query string parameter and value.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/how-to/improve-application>

Monitor and correct data imbalances

Review incorrect predictions

The **incorrect prediction** intent list shows intents that have utterances, which are used as examples for a specific intent, but are predicted for different intents.

To fix this issue:

- Edit utterances to be more specific to the intent and train again.
- Combine intents if utterances are too closely aligned and train again.

Review unclear predictions

The **unclear prediction** intent list shows intents with utterances with prediction scores that are not far enough way from their nearest rival, that the top intent for the utterance may change on the next training, due to negative sampling.

To fix this issue;

- Edit utterances to be more specific to the intent and train again.
- Combine intents if utterances are too closely aligned and train again.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-how-to-use-dashboard#review-data-imbalance>

Implement patterns

After a LUIS app receives endpoint utterances, use a pattern to improve prediction accuracy for utterances that reveal a pattern in word order and word choice. Patterns use specific syntax to indicate the location of: entities, entity roles, and optional text.

Create a pattern.any entity

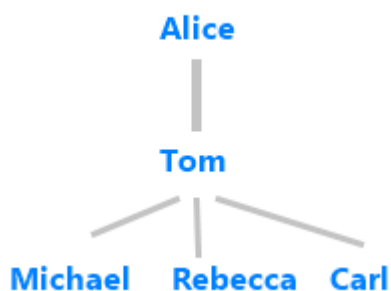
Pattern.any entities are only valid in patterns, not intents' example utterances. This type of entity helps LUIS find the end of entities of varying length and word choice. Because this entity is used in a pattern, LUIS knows where the end of the entity is in the utterance template.

1. Sign in to the LUIS portal, and select your **Subscription** and **Authoring resource** to see the apps assigned to that authoring resource.
2. Open your app by selecting its name on **My Apps** page.
3. From the **Build** section, select **Entities** in the left panel, and then select **+ Create**.
4. In the **Choose an entity type** dialog box, enter the entity name in the **Name** box, and select **Pattern.Any** as the **Type** then select **Create**.
5. Once you create a pattern utterance using this entity, the entity is extracted with a combined machine-learning and text-matching algorithm.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-how-to-model-intent-pattern>

Patterns are designed to improve accuracy when multiple utterances are very similar. A pattern allows you to gain more accuracy for an intent without providing several more utterances.

Consider a Human Resources app that reports on the organizational chart in relation to an employee. Given an employee's name and relationship, LUIS returns the employees involved. Consider an employee, Tom, with a manager named Alice, and a team of subordinates named: Michael, Rebecca, and Carl.



Patterns solve the following situations:

The intent score is low

The correct intent is not the top score but too close to the top score.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/patterns-features>

Manage a LUIS model

Manage collaborators

Azure Active Directory resources

If you use Azure Active Directory (Azure AD) in your organization, Language Understanding (LUIS) needs permission to the information about your users' access when they want to use LUIS. The resources that LUIS requires are minimal.

You see the detailed description when you attempt to sign up with an account that has admin consent or does not require admin consent, such as administrator consent:

- Allows you to sign in to the app with your organizational account and let the app read your profile. It also allows the app to read basic company information. This gives LUIS permission to read basic profile data, such as user ID, email, name
- Allows the app to see and update your data, even when you are not currently using the app. The permission is required to refresh the access token of the user.

Azure Active Directory tenant user

LUIS uses standard Azure Active Directory (Azure AD) consent flow.

The tenant admin should work directly with the user who needs access granted to use LUIS in the Azure AD.

- First, the user signs into LUIS, and sees the pop-up dialog needing admin approval. The user contacts the tenant admin before continuing.
- Second, the tenant admin signs into LUIS, and sees a consent flow pop-up dialog. This is the dialog the admin needs to give permission for the user. Once the admin accepts the permission, the user is able to continue with LUIS. If the tenant admin will not sign in to LUIS, the admin can access consent for LUIS. On this page you can filter the list to items that include the name LUIS.

If the tenant admin only wants certain users to use LUIS, there are a couple of possible solutions:

- Giving the "admin consent" (consent to all users of the Azure AD), but then set to "Yes" the "User assignment required" under Enterprise Application Properties, and finally assign/add only the wanted users to the Application. With this method, the Administrator is still providing "admin consent" to the App, however, it's possible to control the users that can access it.
- A second solution, is by using the Azure AD identity and access management API in Microsoft Graph to provide consent to each specific user.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-how-to-collaborate>

Manage versioning

Versions allow you to build and publish different models. A good practice is to clone the current active model to a different version of the app before making changes to the model.

The active version is the version you are editing in the LUIS portal **Build** section with intents, entities, features, and patterns. When using the authoring APIs, you don't need to set the active version because the version-specific REST API calls include the version in the route.

To work with versions, open your app by selecting its name on **My Apps** page, and then select **Manage** in the top bar, then select **Versions** in the left navigation.

The list of versions shows which versions are published, where they are published, and which version is currently active.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-how-to-manage-versions>

Publish a model through the portal or in a container

When you finish building, training, and testing your active LUIS app, you make it available to your client application by publishing it to an endpoint.


Publishing


1. Sign in to the LUIS portal, and select your **Subscription** and **Authoring resource** to see the apps assigned to that authoring resource.
2. Open your app by selecting its name on **My Apps** page.
3. To publish to the endpoint, select **Publish** in the top-right corner of the panel.

DASHBOARD

BUILD

MANAGE

 Train

 Test

 Publish

4. Select your settings for the published prediction endpoint, then select Publish.

Publishing slots

Select the correct slot when the pop-up window displays:

- Staging
- Production

By using both publishing slots, you can have two different versions of your app available at the published endpoints, or the same version on two different endpoints.

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/how-to/publish>

Export a LUIS package

Export packaged app from LUIS

The LUIS container requires a trained or published LUIS app to answer prediction queries of user utterances. In order to get the LUIS app, use either the trained or published package API.

The default location is the input subdirectory in relation to where you run the docker run command.

Place the package file in a directory and reference this directory as the input mount when you run the docker container.

Package types

The input mount directory can contain the **Production**, **Staging**, and **Versioned** models of the app simultaneously. All the packages are mounted.

Package Type	Query Endpoint API	Query availability	Package filename format
Versioned	GET, POST	Container only	{APP_ID}_v{APP_VERSION}.gz
Staging	GET, POST	Azure and container	{APP_ID}_STAGING.gz
Production	GET, POST	Azure and container	{APP_ID}_PRODUCTION.gz

<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-container-howto>

Deploy a LUIS package to a container

With the following steps, scale Azure Cognitive Services applications in the cloud easily with Azure Container Instances. Containerization helps you focus on building your applications instead of managing the infrastructure. For more information on using containers, see features and benefits.

The recipe works with any Cognitive Services container. The Cognitive Service resource must be created before using the recipe. Each Cognitive Service that supports containers has a "How to install" article for installing and configuring the service for a container. Some services require a file or set of files as input for the container, it is important that you understand and have used the container successfully before using this solution.

- An Azure resource for the Azure Cognitive Service you're using.

- Cognitive Service endpoint URL - review your specific service's "How to install" for the container, to find where the endpoint URL is from within the Azure portal, and what a correct example of the URL looks like. The exact format can change from service to service.
- Cognitive Service key - the keys are on the Keys page for the Azure resource. You only need one of the two keys. The key is a string of 32 alpha-numeric characters.
- A single Cognitive Services Container on your local host (your computer). Make sure you can:
 - ◇ Pull down the image with a docker pull command.
 - ◇ Run the local container successfully with all required configuration settings with a docker run command.
 - ◇ Call the container's endpoint, getting a response of HTTP 2xx and a JSON response back.

All variables in angle brackets, <>, need to be replaced with your own values. This replacement includes the angle brackets.

<https://docs.microsoft.com/en-us/azure/cognitive-services/containers/azure-container-instance-recipe>

Integrate Bot Framework (LUDown) to run outside of the LUIS portal

.lu files help describe language understanding components for your bot. LUDown is a command line tool that helps convert .lu file(s) into JSON files that you can then use to create your LUIS app or QnAMaker knowledge base.

Language Understanding (LUIS.ai) allows your application to understand what a person wants in their own words. LUIS uses machine learning to allow developers to build applications that can receive user input in natural language and extract meaning from it.

QnA Maker enables you to go from FAQ to bot in minutes. With QnA Maker you can build, train and publish a simple question and answer bot based on FAQ URLs, structured documents or editorial content in minutes.

The .lu file and the LUDown tool serve several use cases:

- Quick, simple and easy way to bootstrap language understanding for your bot.
- Describe intents, entities, utterances and patterns for your bot in simple markdown documents.

- The language understanding definition for your bot lives with the rest of your code. You can use other CLI tools like luis to create a new LUIS app using the .lu files.
- Group language understanding documents by language and locale to easily manage localization.
- Define question and answer pairs for your bot.
- Works in conjunction with other Microsoft Bot Builder tools, like LUIS CLI and QnAMaker CLI.

<https://github.com/Microsoft/botbuilder-tools/tree/master/packages/Ludown#ludown>

Implement Knowledge Mining Solutions (15-20%)

Implement a Cognitive Search Solution

Create data sources

In Azure Cognitive Search, a data source is used with indexers, providing the connection information for ad hoc or scheduled data refresh of a target index, pulling data from supported Azure data sources.

You can use either POST or PUT on the request. For either one, the JSON document in the request body provides the object definition.

Alternatively, you can use PUT and specify the name on the URI.

HTTPS is required for all service requests. If the object doesn't exist, it is created. If it already exists, it is updated to the new definition

<https://docs.microsoft.com/en-us/rest/api/searchservice/create-data-source>

Define an index

In Azure Cognitive Search, a search index is your searchable content, available to the search engine for indexing, full text search, and filtered queries. An index is defined by a schema and saved to the search service, with data import following as a second step. This content exists within your search service, apart from your primary data stores, which is necessary for the millisecond response times expected in modern applications. Except for specific indexing scenarios, the search service will never connect to or query your local data.

If you're creating and managing a search index, this article will help you understand the following:

- Content (documents and schema)
- Physical representation
- Basic operations

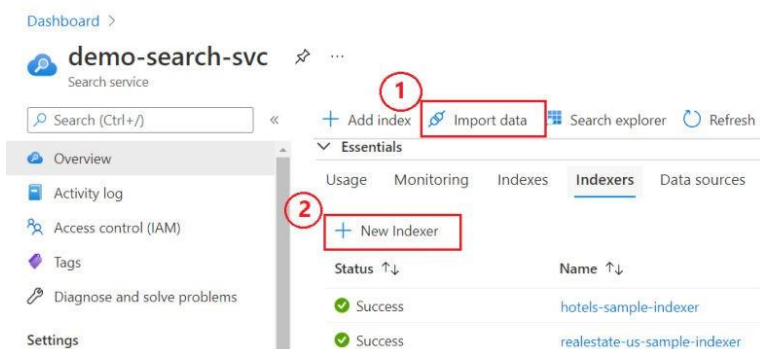
In Cognitive Search, indexes contain search documents. Conceptually, a document is a single unit of searchable data in your index. For example, a retailer might have a document for each product, a news organization might have a document for each article, a travel site might have a document for each hotel and destination, and so forth. Mapping these concepts to more familiar database equivalents: a search index equates to a table, and documents are roughly equivalent to rows in a table.

The structure of a document is determined by the index schema, as illustrated below. The "fields" collection is typically the largest part of an index, where each field is named, assigned a data type, and attributed with allowable behaviors that determine how it is used.

<https://docs.microsoft.com/en-us/azure/search/search-what-is-an-index>

Create and run an indexer

When you're ready to create an indexer on a remote search service, you'll need a search client. A search client can be the Azure portal, Postman or another REST client, or code that instantiates an indexer client. We recommend the Azure portal or REST APIs for early development and proof-of-concept testing.



1. Sign in to Azure portal.
2. On the search service Overview page, choose from two options:
 - **Import data wizard.** The wizard is unique in that it creates all of the required elements. Other approaches require that you have predefined a data source and index.
 - **New Indexer,** a visual editor for specifying an indexer definition.

3. The following screenshot shows where you can find these features in the portal.

Run the indexer

By default, an indexer runs immediately when you create it on the search service. You can override this behavior by setting "disabled" to true in the indexer definition. Indexer execution is the moment of truth where you'll find out if there are problems with connections, field mappings, or skillset construction.

There are several ways to run an indexer:

- Run on indexer creation or update (default).
- Run on demand when there are no changes to the definition, or precede with reset for full indexing. For more information, see [Run or reset indexers](#).
- Schedule indexer processing to invoke execution at regular intervals.

Scheduled execution is usually implemented when you have a need for incremental indexing so that you can pick up the latest changes. As such, scheduling has a dependency on change detection.

<https://docs.microsoft.com/en-us/azure/search/search-howto-create-indexers>

Query an index

Azure Cognitive Search offers a rich query language to support a broad range of scenarios, from free text search, to highly-specified query patterns. This article describes query requests, and what kinds of queries you can create.

In Cognitive Search, a query is a full specification of a round-trip search operation, with parameters that both inform query execution and shape the response coming back.

Parameters and parsers determine the type of query request.

Types of queries

With a few notable exceptions, a query request iterates over inverted indexes that are structured for fast scans, where a match can be found in potentially any field, within any number of search documents. In Cognitive Search, the primary methodology for finding matches is either full text search or filters, but you can also implement other well-known search experiences like autocomplete, or geo-location search. The rest of this article summarizes queries in Cognitive Search and provides links to more information and examples.

<https://docs.microsoft.com/en-us/azure/search/search-query-overview>

Configure an index to support autocomplete and autosuggest

Search-as-you-type is a common technique for improving query productivity. In Azure Cognitive Search, this experience is supported through *autocomplete*, which finishes a term or phrase based on partial input (completing "micro" with "microsoft"). A second user experience is *suggestions*, or a short list of matching documents (returning book titles with an ID so that you can link to a detail page about that book). Both autocomplete and suggestions are predicated on a match in the index. The service won't offer queries that return zero results.

To implement these experiences in Azure Cognitive Search, you will need:

- A *suggester* definition that's embedded in the index schema.
- A *query* specifying Autocomplete or Suggestions API on the request.
- A *UI control* to handle search-as-you-type interactions in your client app. We recommend using an existing JavaScript library for this purpose.

In Azure Cognitive Search, autocompleted queries and suggested results are retrieved from the search index, from selected fields that you have registered with a suggester. A suggester is part of the index, and it specifies which fields will provide content that either completes a query, suggests a result, or does both. When the index is created and loaded, a suggester data structure is created internally to store prefixes used for matching on partial queries. For suggestions, choosing suitable fields that are unique, or at least not repetitive, is essential to the experience.

The remainder of this article is focused on queries and client code. It uses JavaScript and C# to illustrate key points. REST API examples are used to concisely present each operation.

<https://docs.microsoft.com/en-us/azure/search/search-add-autocomplete-suggestions>

Boost results based on relevance

For full text search queries, the search engine computes a search score for each matching document, which allows results to be ranked from high to low. Azure Cognitive Search uses a default scoring algorithm to compute an initial score, but you can customize the calculation through a scoring profile.

Scoring profiles are embedded in index definitions and include properties for boosting the score of matches, where additional criteria found in the profile provides the boosting logic. For

example, you might want to boost matches based on their revenue potential, promote newer items, or perhaps boost items that have been in inventory too long.

Unfamiliar with relevance concepts? The following video segment fast-forwards to how scoring profiles work in Azure Cognitive Search, but the video also covers basic concepts. You might also want to review Similarity ranking and scoring for more background.

What is a scoring profile?

A scoring profile is part of the index definition and is composed of weighted fields, functions and parameters. The purpose of a scoring profile is to boost or amplify matching documents based on criteria you provide.

The following definition shows a simple profile named 'geo'. This one boosts results that have the search term in the hotelName field. It also uses the distance function to favor results that are within ten kilometers of the current location. If someone searches on the term 'inn', and 'inn' happens to be part of the hotel name, documents that include hotels with 'inn' within a 10 KM radius of the current location will appear higher in the search results.

<https://docs.microsoft.com/en-us/azure/search/index-add-scoring-profiles>

Implement synonyms

Within a search service, synonym maps are a global resource that associate equivalent terms, expanding the scope of a query without the user having to actually provide the term. For example, assuming "dog", "canine", and "puppy" are mapped synonyms, a query on "canine" will match on a document containing "dog".

Create synonyms

A synonym map is an asset that can be created once and used by many indexes. The service tier determines how many synonym maps you can create, ranging from three synonym maps for Free and Basic tiers, up to 20 for the Standard tiers.

You might create multiple synonym maps for different languages, such as English and French versions, or lexicons if your content includes technical or obscure terminology. Although you can create multiple synonym maps in your search service, a field can only use one of them.

A synonym map consists of name, format, and rules that function as synonym map entries. The only format that is supported is solr, and the solr format determines rule construction.

<https://docs.microsoft.com/en-us/azure/search/search-synonyms>

Implement an Enrichment Pipeline

Attach a Cognitive Services account to a skillset

When configuring an optional AI enrichment pipeline in Azure Cognitive Search, you can enrich a limited number of documents free of charge. For larger and more frequent workloads, you should attach a billable **multi-service Cognitive Services resource**.

A multi-service resource references "Cognitive Services" as the offering, rather than individual services, with access granted through a single API key. This key is specified in a **skillset** and allows Microsoft to charge you for using these APIs:

- Computer Vision for image analysis and optical character recognition (OCR)
- Language service for language detection, entity recognition, sentiment analysis, and key phrase extraction
- Translator for machine text translation

<https://docs.microsoft.com/en-us/azure/search/cognitive-search-attach-cognitive-services>

Select and include built-in skills for documents

Built-in cognitive skills for text & image processing

This article describes the skills provided with Azure Cognitive Search that you can include in a skillset to extract content and structure from raw unstructured text and image files. A *skill* is an atomic operation that transforms content in some way. Often, it is an operation that recognizes or extracts text, but it can also be a utility skill that reshapes the enrichments that are already created. Typically, the output is text-based so that it can be used in full text queries.

Built-in skills

Built-in skills are based on pre-trained models from Microsoft, which means you cannot train the model using your own training data. Skills that call the Cognitive Resources APIs have a dependency on those services and are billed at the Cognitive Services pay-as-you-go price when you attach a resource. Other skills are metered by Azure Cognitive Search, or are utility skills that are available at no charge.

<https://docs.microsoft.com/en-us/azure/search/cognitive-search-predefined-skills>

Document Extraction cognitive skill

The **Document Extraction** skill extracts content from a file within the enrichment pipeline. This allows you to take advantage of the document extraction step that normally happens before the skillset execution with files that may be generated by other skills.

This skill isn't bound to Cognitive Services and has no Cognitive Services key requirement. This skill extracts text and images. Text extraction is free. Image extraction is metered by Azure Cognitive Search. On a free search service, the cost of 20 transactions per indexer per day is absorbed so that you can complete quickstarts, tutorials, and small projects at no charge. For Basic, Standard, and above, image extraction is billable.

<https://docs.microsoft.com/en-us/azure/search/cognitive-search-skill-document-extraction>

Implement custom skills and include them in a skillset

Custom skills might sound complex but can be simple and straightforward in terms of implementation. If you have existing packages that provide pattern matching or classification models, the content you extract from blobs could be passed to these models for processing. Since AI enrichment is Azure-based, your model should be on Azure also. Some common hosting methodologies include using Azure Functions or Containers.

If you are building a custom skill, this article describes the interface you'll use to integrate the skill into the pipeline. The primary requirement is the ability to accept inputs and emit outputs in ways that are consumable within the skillset as a whole. As such, the focus of this article is on the input and output formats that the enrichment pipeline requires.

Benefits of custom skills

Building a custom skill gives you a way to insert transformations unique to your content. A custom skill executes independently, applying whatever enrichment step you require. For example, you could build custom classification models to differentiate business and financial contracts and documents, or add a speech recognition skill to reach deeper into audio files for relevant content. For a step-by-step example, see Example: Creating a custom skill for AI enrichment.

<https://docs.microsoft.com/en-us/azure/search/cognitive-search-custom-skill-interface>

Implement a Knowledge Store

Define file projections

Projecting to file

File projections are always binary, normalized images, where normalization refers to potential resizing and rotation for use in skillset execution. File projections, similar to object projections, are created as blobs in Azure Storage, and contain binary data (as opposed to JSON).

To define a file projection, use the files array in the projections property. A files projection has three required properties:

Property	Description
storageContainer	Determines the name of a new container created in Azure Storage.
generatedKeyName	Column name for the key that uniquely identifies each row. The value is system-generated. If you omit this property, a column will be created automatically that uses the table name and "key" as the naming convention.
source	A path to a node in an enrichment tree that is the root of the projection. For images files, the source is always /document/normalized_images/*. File projections only act on the normalized_images collection. Neither indexers nor a skillset will pass through the original non-normalized image.

The destination is always a blob container, with a folder prefix of the base64 encoded value of the document ID. If there are multiple images, they will be placed together in the same folder. File projections cannot share the same container as object projections and need to be projected into a different container.

<https://docs.microsoft.com/en-us/azure/search/knowledge-store-projections-examples>

Define object projections

Projecting to objects

Object projections are JSON representations of the enrichment tree that can be sourced from any node. In comparison with table projections, object projections are simpler to define and are used when projecting whole documents. Object projections are limited to a single projection in a container and cannot be sliced.

To define an object projection, use the objects array in the projections property. An object projection has three required properties:

Property	Description
storageContainer	Determines the name of a new container created in Azure Storage.
generatedKeyName	Column name for the key that uniquely identifies each row. The value is system-generated. If you omit this property, a column will be created automatically that uses the table name and "key" as the naming convention.
source	A path to a node in an enrichment tree that is the root of the projection. The node is usually a reference to a complex data shape that determines blob structure.

<https://docs.microsoft.com/en-us/azure/search/knowledge-store-projections-examples>

Define table projections

Projecting to tables

Table projections are recommended for scenarios that call for data exploration, such as analysis with Power BI or workloads that consume data frames. The tables section of a projections array is a list of tables that you want to project.

To define a table projection, use the tables array in the projections property. A table projection has three required properties:

Property	Description
tableName	Determines the name of a new table created in Azure Table Storage.
generatedKeyName	Column name for the key that uniquely identifies each row. The value is system-generated. If you omit this property, a column will be created automatically that uses the table name and "key" as the naming convention.
source	A path to a node in an enrichment tree. The node should be a reference to a complex shape that determines which columns are created in the table.

In table projections, "source" is usually the output of a Shaper skill that defines the shape of the table. Tables have rows and columns, and shaping is the mechanism by which rows and columns are specified. You can use a Shaper skill or inline shapes. The Shaper skill produces valid JSON, but the source could be the output from any skill, if valid JSON.

<https://docs.microsoft.com/en-us/azure/search/knowledge-store-projections-examples>

Query projections

Knowledge store “projections” in Azure Cognitive Search

Projections are the physical tables, objects, and files in a knowledge store that accept content from a Cognitive Search AI enrichment pipeline. If you're creating a knowledge store, defining and shaping projections is most of the work.

This article introduces projection concepts and workflow so that you have some background before you start coding.

Projections are defined in Cognitive Search skillsets, but the end results are the table, object, and image file projections in Azure Storage.

Projection definition

Projections are specified under the "knowledgeStore" property of a skillset. Projection definitions are used during indexer invocation to create and load objects in Azure Storage with enriched content. If you are unfamiliar with these concepts, start with AI enrichment for an introduction.

The following example illustrates the placement of projections under knowledgeStore, and the basic construction. The name, type, and content source make up a projection definition.

<https://docs.microsoft.com/en-us/azure/search/knowledge-store-projection-overview>

Manage a Cognitive Search Solution

Provision Cognitive Search

Create an Azure Cognitive Search service

1. Sign in to the Azure portal.
2. Click the plus sign (" + Create Resource") in the top-left corner.
3. Use the search bar to find "Azure Cognitive Search" or navigate to the resource through Web > Azure Cognitive Search.

Choose a subscription

If you have more than one subscription, choose one for your search service. If you are implementing double encryption or other features that depend on managed service identities, choose the same subscription as the one used for Azure Key Vault or other services for which managed identities are used.

Set a resource group

A resource group is a container that holds related resources for your Azure solution. It's required for the search service. It's also useful for managing resources all-up, including costs. A resource group can consist of one service, or multiple services used together. For example, if you're using Azure Cognitive Search to index an Azure Cosmos DB database, you could make both services part of the same resource group for management purposes.

If you aren't combining resources into a single group, or if existing resource groups are filled with resources used in unrelated solutions, create a new resource group just for your Azure Cognitive Search resource.

<https://docs.microsoft.com/en-us/azure/search/search-create-service-portal>

Configure security for Cognitive Search

Security in Azure Cognitive Search

Data flow (network traffic patterns)

A Cognitive Search service is hosted on Azure and is typically accessed by client applications over public network connections. While that pattern is predominant, it's not the only traffic pattern that you need to care about. Understanding all points of entry as well as outbound traffic is necessary background for securing your development and production environments.

Cognitive Search has three basic network traffic patterns:

- Inbound requests made by a client to the search service (the predominant pattern)
- Outbound requests issued by the search service to other services on Azure and elsewhere
- Internal service-to-service requests over the secure Microsoft backbone network

Network security

Network security protects resources from unauthorized access or attack by applying controls to network traffic. Azure Cognitive Search supports networking features that can be your first line of defense against unauthorized access.

<https://docs.microsoft.com/en-us/azure/search/search-security-overview>

Configure keys for data encryption in Azure Cognitive Search

Key Vault tips

If you're new to Azure Key Vault, review this quickstart to learn about basic tasks: Set and retrieve a secret from Azure Key Vault using PowerShell. Here are some tips for using Key Vault:

- Use as many key vaults as you need. Managed keys can be in different key vaults. A search service can have multiple encrypted objects, each one encrypted with a different customer-managed encryption key, stored in different key vaults.
- Enable logging on Key Vault so that you can monitor key usage.

- Remember to follow strict procedures during routine rotation of key vault keys and Active Directory application secrets and registration. Always update all encrypted content to use new secrets and keys before deleting the old ones. If you miss this step, your content can't be decrypted.

Encrypt content

Encryption keys are added when you create an object. To add a customer-managed key on an index, synonym map, indexer, data source, or skillset, use the Search REST API or an Azure SDK to create an object that has encryption enabled. The portal does not allow encryption properties on object creation.

1. Call the Create APIs to specify the encryptionKey property:

- Create Index
- Create Synonym Map
- Create Indexer
- Create Data Source
- Create Skillset.

<https://docs.microsoft.com/en-us/azure/search/search-security-manage-encryption-keys>

Configure IP firewall for Azure Cognitive Search

Azure Cognitive Search supports IP rules for inbound access through a firewall, similar to the IP rules you'll find in an Azure virtual network security group. By leveraging IP rules, you can restrict search service access to an approved set of machines and cloud services. Access to data stored in your search service from the approved sets of machines and services will still require the caller to present a valid authorization token.

You can set IP rules in the Azure portal, as described in this article, on search services provisioned at the Basic tier and above. Alternatively, you can use the Management REST API version 2020-03-13, Azure PowerShell, or Azure CLI.

Set IP ranges in Azure portal

To set the IP access control policy in the Azure portal, go to your Azure Cognitive Search service page and select Networking on the left navigation pane. Endpoint networking connectivity must be Public Access. If your connectivity is set to Private Access or Shared Private Access, you can only access your search service via a Private Endpoint.

The Azure portal provides the ability to specify IP addresses and IP address ranges in the CIDR

format. An example of CIDR notation is 8.8.8.0/24, which represents the IPs that range from 8.8.8.0 to 8.8.8.255.

After you enable the IP access control policy for your Azure Cognitive Search service, all requests to the data plane from machines outside the allowed list of IP address ranges are rejected.

<https://docs.microsoft.com/en-us/azure/search/service-configure-firewall>

Configure scalability for Cognitive Search

Scale for performance on Azure Cognitive Search

Availability and business continuity in Azure Cognitive Search

In Cognitive Search, availability is achieved through multiple replicas, whereas business continuity (and disaster recovery) is achieved through multiple search services. This article provides guidance that you can use as a starting point for developing a strategy that meets your business requirements for both availability and continuous operations.

High availability

In Cognitive Search, replicas are copies of your index. Having multiple replicas allows Azure Cognitive Search to do machine reboots and maintenance against one replica, while query execution continues on other replicas. For more information about adding replicas, see [Add or reduce replicas and partitions](#).

For each individual search service, Microsoft guarantees at least 99.9% availability for configurations that meet these criteria:

- Two replicas for high availability of read-only workloads (queries)
- Three or more replicas for high availability of read-write workloads (queries and indexing)

No SLA is provided for the Free tier. For more information, see [SLA for Azure Cognitive Search](#).

<https://docs.microsoft.com/en-us/azure/search/search-performance-optimization>

Manage Indexing

Manage re-indexing

Update index

Modifying an existing Azure Cognitive Search index typically requires an index drop and rebuild, with the exception of the following schema changes:

- Add new fields
- Add or change scoring profiles
- Change CORS options
- Change existing fields with any of the following three modifications:
 - Show or hide fields (retrievable: true | false)
 - Change the analyzer used at query time (searchAnalyzer)
 - Add or edit the synonymMap used at query time (synonymMaps)

To make any of these schema changes to an existing index, specify the name of the index on the request URI, and then include a fully-specified index definition with the new or changed elements.

Although existing fields cannot be deleted and most attributes cannot be changed, new fields can be added to an existing index at any time. The same applies to a suggester. New fields may be added to a suggester at the same time fields are added, but existing fields cannot be removed from nor added to suggesters without an index rebuild.

When a new field is added, all existing documents in the index automatically have a null value for that field. No additional storage space is consumed until one of two things occur: a value is provided for the new field (using merge), or new documents are added.

Once an analyzer, a tokenizer, a token filter or a char filter is defined, it cannot be modified. New ones can be added to an existing index only if the allowIndexDowntime flag is set to true in the index update request:

This operation takes your index offline for at least a few seconds, causing your indexing and query requests to fail. Performance and write availability of the index can be impaired for several minutes after the index is updated, or longer for indexes.

<https://docs.microsoft.com/en-us/rest/api/searchservice/update-index>

Rebuild indexes

Rebuild an index in Azure Cognitive Search

During development, the index schema changes frequently. You can plan for it by creating indexes that can be deleted, recreated, and reloaded quickly with a small representative data set.

For applications already in production, we recommend creating a new index that runs side by side an existing index to avoid query downtime. Your application code provides redirection to the new index.

1. Determine whether a rebuild is required. If you are just adding fields, or changing some part of the index that is unrelated to fields, you might be able to simply update the definition without deleting, recreating, and fully reloading it.
2. Get an index definition in case you need it for future reference.
3. Drop the existing index, assuming you are not running new and old indexes side by side.
Any queries targeting that index are immediately dropped. Remember that deleting an index is irreversible, destroying physical storage for the fields collection and other constructs. Pause to think about the implications before dropping it.
4. Create a revised index, where the body of the request includes changed or modified field definitions.
5. Load the index with documents from an external source.

When you create the index, physical storage is allocated for each field in the index schema, with an inverted index created for each searchable field. Fields that are not searchable can be used in filters or expressions, but do not have inverted indexes and are not full-text or fuzzy searchable. On an index rebuild, these inverted indexes are deleted and recreated based on the index schema you provide.

When you load the index, each field's inverted index is populated with all of the unique, tokenized words from each document, with a map to corresponding document IDs. For example, when indexing a hotels data set, an inverted index created for a City field might contain terms for Seattle, Portland, and so forth. Documents that include Seattle or Portland in the City field would have their document ID listed alongside the term. On any Add, Update or Delete operation, the terms and document ID list are updated accordingly.

<https://docs.microsoft.com/en-us/azure/search/search-howto-reindex>

Schedule indexing

Schedule indexers in Azure Cognitive Search

Indexers can be configured to run on a schedule when you set the "schedule" property in the indexer definition. By default, an indexer runs once, immediately after it is created.

Afterwards, you can run it again on demand or on a schedule. Some situations where indexer scheduling is useful include:

- Source data is changing over time, and you want the indexer to automatically process the difference.
- A search index is populated from multiple data sources, and you want to stagger the indexer jobs to reduce conflicts.
- Source data is very large and you want to spread the indexer processing over time.

Indexer jobs are subject to a maximum running time of 24 hours for regular data sources and 2 hours for indexers with skillsets. If indexing cannot complete within the maximum interval, you can configure a schedule that runs every 2 hours. Indexers can automatically pick up where they left off, based on an internal high water mark that marks where indexing last ended. Running an indexer on a recurring 2-hour schedule allows it to process a very large data set (many millions of documents) beyond the 24-interval allowed for a single job. For more information about indexing large data volumes, see [How to index large data sets in Azure Cognitive Search](https://docs.microsoft.com/en-us/azure/search/search-howto-schedule-indexers).

<https://docs.microsoft.com/en-us/azure/search/search-howto-schedule-indexers>

Monitor indexing

Monitor Azure Cognitive Search indexer status

You can monitor indexer processing in the Azure portal, or programmatically through REST calls or an Azure SDK. In addition to status about the indexer itself, you can review start and end times, and detailed errors and warnings from a particular run.

Monitor using Get Indexer Status (REST API)

You can retrieve the status and execution history of an indexer using the Get Indexer Status command:

The response contains overall indexer status, the last (or in-progress) indexer invocation, and the history of recent indexer invocations.

Execution history contains up to the 50 most recent runs, which are sorted in reverse chronological order (most recent first).

Note there are two different status values. The top level status is for the indexer itself. A indexer status of running means the indexer is set up correctly and available to run, but not that it's currently running.

Each run of the indexer also has its own status that indicates whether that specific execution is ongoing (running), or already completed with a success, transientFailure, or persistentFailure status.

When an indexer is reset to refresh its change tracking state, a separate execution history entry is added with a Reset status.

<https://docs.microsoft.com/en-us/azure/search/search-howto-monitor-indexers>

Implement incremental indexing

Incremental enrichment and caching

Incremental enrichment refers to the use of cached enrichments during skillset execution so that only new and changed skills and documents incur AI processing. The cache contains the output from document cracking, plus the outputs of each skill for every document. Although caching is billable (it uses Azure Storage), the overall cost of enrichment is reduced because the costs of storage are less than image extraction and AI processing.

When you enable caching, the indexer evaluates your updates to determine whether existing enrichments can be pulled from the cache. Image and text content from the document cracking phase, plus skill outputs that are upstream or orthogonal to your edits, are likely to be reusable.

After performing the incremental enrichments as indicated by the skillset update, refreshed results are written back to the cache, and also to the search index or knowledge store.

<https://docs.microsoft.com/en-us/azure/search/cognitive-search-incremental-indexing-conceptual>

Manage concurrency

Manage concurrency in Azure Cognitive Search

When managing Azure Cognitive Search resources such as indexes and data sources, it's important to update resources safely, especially if resources are accessed concurrently by different components of your application. When two clients concurrently update a resource without coordination, race conditions are possible. To prevent this, Azure Cognitive Search offers an *optimistic concurrency model*. There are no locks on a resource. Instead, there is an ETag for every resource that identifies the resource version so that you can formulate requests that avoid accidental overwrites.

<https://docs.microsoft.com/en-us/azure/search/search-howto-concurrency>

Push data to an index

Pushing data to an index

The push model, used to programmatically send your data to Azure Cognitive Search, is the most flexible approach for the following reasons:

- First, there are no restrictions on data source type. The dataset must be composed of JSON documents that map to your index schema, but the data can come from anywhere.
- Second, there are no restrictions on frequency of execution. You can push changes to an index as often as you like. For applications having low latency requirements (for example, if you need search operations to be in sync with dynamic inventory databases), the push model is your only option.
- Third, you can upload documents individually or in batches up to 1000 per batch, or 16 MB per batch, whichever limit comes first.

<https://docs.microsoft.com/en-us/azure/search/search-what-is-data-import#pushing-data-to-an-index>

Troubleshoot indexing for a pipeline

Troubleshooting common indexer issues

Occasionally, indexers run into problems and there is no error to help with diagnosis. This article covers problems and potential resolutions when indexer results are unexpected and there is limited information to go on. If you have an error to investigate, see Troubleshooting common indexer errors and warnings instead.

Troubleshoot connections to restricted resources

For data sources that are secured by Azure network security mechanisms, indexers have a limited set of options for making the connection. Currently, indexers can access restricted data sources behind an IP firewall or on a virtual network through a private endpoint.

<https://docs.microsoft.com/en-us/azure/search/search-indexer-troubleshooting>

Implement Conversational AI Solutions (15-20%)

Create a Knowledge Base by Using QnA Maker

Create a QnA Maker service

Create a new QnA Maker service

This procedure creates the Azure resources needed to manage the knowledge base content. After you complete these steps, you'll find the *subscription* keys on the **Keys** page for the resource in the Azure portal.

1. Sign in to the Azure portal and create a QnA Maker resource.
2. Select **Create** after you read the terms and conditions.
3. In **QnA Maker**, select the appropriate tiers and regions
 - In the **Name** field, enter a unique name to identify this QnA Maker service. This name also identifies the QnA Maker endpoint that your knowledge bases will be associated with.

- Choose the Subscription under which the QnA Maker resource will be deployed.
 - Select the Pricing tier for the QnA Maker management services (portal and management APIs). See more details about SKU pricing.
 - Create a new Resource group (recommended) or use an existing one in which to deploy this QnA Maker resource. QnA Maker creates several Azure resources. When you create a resource group to hold these resources, you can easily find, manage, and delete these resources by the resource group name.
 - Select a Resource group location.
 - Choose the Search pricing tier of the Azure Cognitive Search service. If the Free tier option is unavailable (appears dimmed), it means you already have a free service deployed through your subscription. In that case, you'll need to start with the Basic tier. See Azure Cognitive Search pricing details.
 - Choose the Search location where you want Azure Cognitive Search indexes to be deployed. Restrictions on where customer data must be stored will help determine the location you choose for Azure Cognitive Search.
 - In the App name field, enter a name for your Azure App Service instance.
 - By default, App Service defaults to the standard (S1) tier. You can change the plan after creation. Learn more about App Service pricing.
 - Choose the Website location where App Service will be deployed.
 - Choose whether or not you want to enable Application Insights. If Application Insights is enabled, QnA Maker collects telemetry on traffic, chat logs, and errors.
 - Choose the App insights location where the Application Insights resource will be deployed.
 - For cost savings measures, you can share some but not all Azure resources created for QnA Maker.
4. After all the fields are validated, select **Create**. The process can take a few minutes to complete.
 5. After deployment is completed, you'll see the following resources created in your subscription:

<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/how-to/set-up-qnamaker-service-azure>

Create a knowledge base

Create your QnA Maker knowledge base

1. Sign in to the QnAMaker.ai portal with your Azure credentials.
2. In the QnA Maker portal, select Create a knowledge base.
3. On the Create page, skip Step 1 if you already have your QnA Maker resource.

If you haven't created the service yet, select Stable and Create a QnA service. You are directed to the Azure portal to set up a QnA Maker service in your subscription. Remember your Azure Active Directory ID, Subscription, QnA resource name you selected when you created the resource.

When you are done creating the resource in the Azure portal, return to the QnA Maker portal, refresh the browser page, and continue to Step 2.

4. In Step 2, select your Active directory, subscription, service (resource), and the language for all knowledge bases created in the service.
5. In Step 3, name your knowledge base My Sample QnA KB.
6. In Step 4, configure the settings with the following table.
7. In Step 5, Select Create your KB.

The extraction process takes a few moments to read the document and identify questions and answers.

After QnA Maker successfully creates the knowledge base, the Knowledge base page opens. You can edit the contents of the knowledge base on this page.

<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/quickstarts/create-publish-knowledge-base>

Import a knowledge base

Migrate a knowledge base from QnA Maker (Step 6)

Export a knowledge base

1. Sign in to QnA Maker portal.
2. Select the knowledge base you want to move.

3. On the **Settings** page, you have the options to export QnAs, Synonyms, or Knowledge Base Replica. You can choose to download the data in .tsv/.xlsx.
 - a. QnAs: When exporting QnAs, all QnA pairs (with questions, answers, metadata, follow-up prompts, and the data source names) are downloaded. The QnA IDs that are exported with the questions and answers may be used to update a specific QnA pair using the update API. The QnA ID for a specific QnA pair remains unchanged across multiple export operations.
 - b. Synonyms: You can export Synonyms that have been added to the knowledge base.
 - c. Knowledge Base Replica: If you want to download the entire knowledge base with synonyms and other settings, you can choose this option.

Import a knowledge base

1. Click Create a knowledge base from the top menu of the qnamaker.ai portal and then create an *empty* knowledge base by not adding any URLs or files. Set the name of your choice for the new knowledge base and then Click Create your KB.
2. In this new knowledge base, open the Settings tab and under *Import knowledge base* select one of the following options: QnAs, Synonyms, or Knowledge Base Replica.
 - a. QnAs: This option imports all QnA pairs. The QnA pairs created in the new knowledge base shall have the same QnA ID as present in the exported file. You can refer SampleQnAs.xlsx, SampleQnAs.tsv to import QnAs.
 - b. Synonyms: This option can be used to import synonyms to the knowledge base. You can refer SampleSynonyms.xlsx, SampleSynonyms.tsv to import synonyms.
 - c. Knowledge Base Replica: This option can be used to import KB replica with QnAs, Synonyms and Settings. You can refer KBReplicaSampleExcel, KBReplicaSampleTSV for more details. If you also want to add unstructured content to the replica, refer CustomQnAKBReplicaSample.

Either QnAs or Unstructured content is required when importing replica. Unstructured documents are only valid for Custom question answering. Synonyms file is not mandatory when importing replica. Settings file is mandatory when importing replica.

3. **Test** the new knowledge base using the Test panel. Learn how to test your knowledge base.
4. **Publish** the knowledge base and create a chat bot. Learn how to publish your knowledge base.

<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/tutorials/export-knowledge-base>

Train and test a knowledge base

Train the knowledge base

Save and train

In the upper right, select **Save and train** to save your edits and train QnA Maker. Edits aren't kept unless they're saved.

Test the knowledge base

1. In the QnA Maker portal, in the upper right, select **Test** to test that the changes you made took effect.
2. Enter an example user query in the textbox.
3. Select **Inspect** to examine the response in more detail. The test window is used to test your changes to the knowledge base before publishing your knowledge base.
4. Select **Test** again to close the **Test** panel.

<https://docs.microsoft.com/en-us/azure/cognitive-services/gnamaker/quickstarts/create-publish-knowledge-base>

How to test a knowledge base?

Testing your QnA Maker knowledge base is an important part of an iterative process to improve the accuracy of the responses being returned. You can test the knowledge base through an enhanced chat interface that also allows you make edits.

The QnA Maker service is being retired on the 31st of March, 2025. A newer version of the question and answering capability is now available as part of Azure Cognitive Service for Language. For question answering capabilities within the Language Service, see question answering. Starting 1st October, 2022 you won't be able to create new QnA Maker resources. For information on migrating existing QnA Maker knowledge bases to question answering, consult the migration guide.

Interactively test in QnA Maker portal

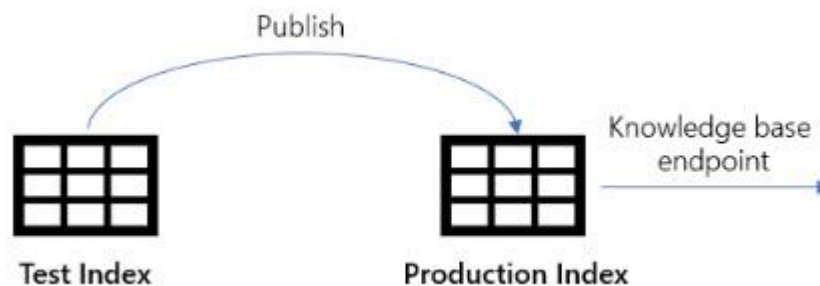
1. Access your knowledge base by selecting its name on the My knowledge bases page.
2. To access the Test slide-out panel, select Test in your application's top panel.
3. Enter a query in the text box and select Enter.
4. The best-matched answer from the knowledge base is returned as the response.

<https://docs.microsoft.com/en-us/azure/cognitive-services/gnamaker/how-to/test-knowledge-base>

Publish a knowledge base

Publish the knowledge base

When you publish a knowledge base, the contents of your knowledge base move from the test index to a prod index in Azure search.



1. In the QnA Maker portal, select **Publish**. Then to confirm, select **Publish** on the page.

The QnA Maker service is now successfully published. You can use the endpoint in your application or bot code.

<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/quickstarts/create-publish-knowledge-base>

Create a multi-turn conversation

Use follow-up prompts to create multiple turns of a conversation

Add question pair with follow-up prompts

To help users solve issues with their Surface Pen, we add follow-up prompts:

- Add a new question pair with two follow-up prompts
 - Add a follow-up prompt to one of the newly added prompts
1. Add a new QnA pair with two follow-up prompts **Check compatibility** and **Check Pen Settings** Using the editor, we add a new QnA pair with a follow-up prompt by clicking on **Add question pair**
 2. We then add a follow-up prompt to the newly created question pair by choosing **Add follow-up prompts**.

We provide **Check Compatibility** as the “Display text” for the prompt and try to link it to a QnA. Since, no related QnA pair is available to link to the prompt, when we search “Check your Surface Pen Compatibility”, we create a new question pair by clicking on **Create link to new pair** and select **Done**. Then select **Save changes**.

3. Similarly, we add another prompt **Check Pen Settings** to help the user troubleshoot the Surface Pen and add question pair to it.
4. Add another follow-up prompt to the newly created prompt. We now add “Replace Pen tips” as a follow-up prompt to the previously created prompt “Check Pen Settings”.
5. Finally, save the changes and test these prompts in the **Test** pane:

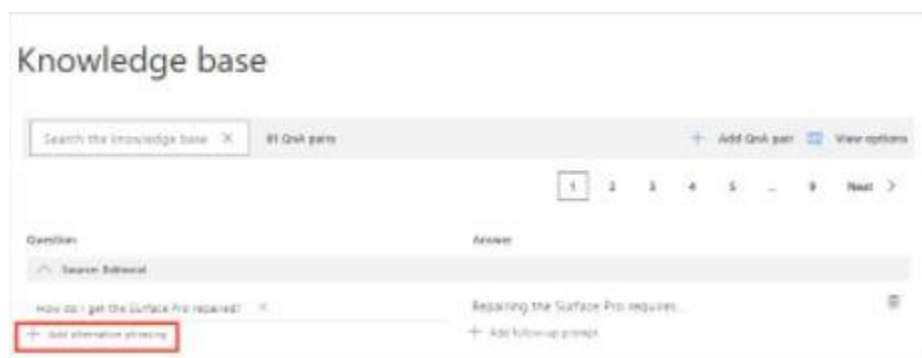
For a user query **Issues with Surface Pen**, the system returns an answer and presents the newly added prompts to the user. The user then selects one of the prompts **Check Pen Settings** and the related answer is returned to the user with another prompt **Replace Pen Tips**, which when selected further provides the user with more information. So, multi-turn is used to help and guide the user to the desired answer.

<https://docs.microsoft.com/en-us/azure/cognitive-services/language-service/question-answering/tutorials/guided-conversations>

Add alternate phrasing

Add alternate questions

Add alternate questions to an existing QnA pair to improve the likelihood of a match to a user query.



<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/how-to/edit-knowledge-base>

Add additional alternatively-phrased questions

The current knowledge base has the QnA Maker troubleshooting QnA pairs. These pairs were created when the URL was added to the knowledge base during the creation process.

When this URL was imported, only one question with one answer was created. In this procedure, add additional questions.

1. From the **Edit** page, use the search textbox above the question and answer pairs, to find the question How large a knowledge base can I create?
2. In the **Question** column, select **+ Add alternative phrasing** then add each new phrasing, provided in the following table.
3. Select **Save and train** to retrain the knowledge base.
4. Select **Test**, then enter a question that is close to one of the new alternative phrasings but isn't exactly the same wording:

What GB size can a knowledge base be?

The correct answer is returned in markdown format:

The size of the knowledge base depends on the SKU of Azure search you choose when creating the QnA Maker service. Read [here](../concepts/azure-resources.md) for more details.

If you select **Inspect** under the returned answer, you can see more answers met the question but not with the same high level of confidence.

Do not add every possible combination of alternative phrasing. When you turn on QnA Maker's active learning, this finds the alternative phrasings that will best help your knowledge base meet your users' needs.

5. Select **Test** again to close the test window.

<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/quickstarts/add-question-metadata-portal>

Get suggested alternate questions

Active Learning alters the Knowledge Base or Search Service after you approve the suggestion, then save and train. If you approve the suggestion, it will be added as an alternate question.

View suggested questions

1. In order to see the suggested questions, on the **Edit** knowledge base page, select **View Options**, then select **Show active learning suggestions**. This option will be disabled if there are no suggestions present for any of the question and answer pairs.
2. Filter the knowledge base with question and answer pairs to show only suggestions by selecting **Filter by Suggestions**.
3. Each QnA pair suggests the new question alternatives with a check mark, ☐, to accept the question or an x to reject the suggestions. Select the check mark to add the question.
4. Select **Save and Train** to save the changes to the knowledge base.
5. Select **Publish** to allow the changes to be available from the GenerateAnswer API.

When 5 or more similar queries are clustered, every 30 minutes, QnA Maker suggests the alternate questions for you to accept or reject.

<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/how-to/improve-knowledge-base>

Add chit-chat to a knowledge base

Adding chit-chat to your bot makes it more conversational and engaging. The chit-chat feature in QnA maker allows you to easily add a pre-populated set of the top chit-chat, into your knowledge base (KB). This can be a starting point for your bot's personality, and it will save you the time and cost of writing them from scratch.

This dataset has about 100 scenarios of chit-chat in the voice of multiple personas, like Professional, Friendly and Witty. Choose the persona that most closely resembles your bot's voice. Given a user query, QnA Maker tries to match it with the closest known chit-chat QnA.

Some examples of the different personalities are below. You can see all the personality datasets along with details of the personalities.

Personality	Example
Professional	Age doesn't really apply to me.
Friendly	I don't really have an age.
Witty	I'm age-free.
Caring	I don't have an age.
Enthusiastic	I'm a bot, so I don't have an age.

<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/how-to/chit-chat-knowledge-base>

Export a knowledge base

Migrate a knowledge base using export-import

Export a knowledge base

1. Sign in to QnA Maker portal.
2. Select the knowledge base you want to move.
3. On the **Settings** page, you have the options to export **QnAs**, **Synonyms**, or **Knowledge Base Replica**. You can choose to download the data in .tsv/.xlsx.
 - a. **QnAs**: When exporting QnAs, all QnA pairs (with questions, answers, metadata, follow-up prompts, and the data source names) are downloaded. The QnA IDs that are exported with the questions and answers may be used to update a specific QnA pair using the update API. The QnA ID for a specific QnA pair remains unchanged across multiple export operations.
 - b. **Synonyms**: You can export Synonyms that have been added to the knowledge base.
 - c. **Knowledge Base Replica**: If you want to download the entire knowledge base with synonyms and other settings, you can choose this option.

Import a knowledge base

1. Click **Create a knowledge base** from the top menu of the qnamaker.ai portal and then create an *empty* knowledge base by not adding any URLs or files. Set the name of your choice for the new knowledge base and then Click **Create your KB**.
2. In this new knowledge base, open the **Settings** tab and under *Import knowledge base* select one of the following options: **QnAs**, **Synonyms**, or **Knowledge Base Replica**.
 - a. **QnAs**: This option imports all QnA pairs. **The QnA pairs created in the new knowledge base shall have the same QnA ID as present in the exported file.** You can refer SampleQnAs.xlsx, SampleQnAs.tsv to import QnAs.
 - b. **Synonyms**: This option can be used to import synonyms to the knowledge base. You can refer SampleSynonyms.xlsx, SampleSynonyms.tsv to import synonyms.
 - c. **Knowledge Base Replica**: This option can be used to import KB replica with QnAs, Synonyms and Settings. You can refer KBReplicaSampleExcel, KBReplicaSampleTSV for more details. If you also want to add unstructured content to the replica, refer CustomQnAKBReplicaSample.

Either QnAs or Unstructured content is required when importing replica. Unstructured documents are only valid for Custom question answering. Synonyms file is not mandatory when importing replica. Settings file is mandatory when importing replica.

3. **Test** the new knowledge base using the Test panel. Learn how to test your knowledge base.
4. **Publish** the knowledge base and create a chat bot. Learn how to publish your knowledge base.

<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/tutorials/export-knowledge-base>

Add active learning to a knowledge base

The *Active learning suggestions* feature allows you to improve the quality of your knowledge base by suggesting alternative questions, based on user-submissions, to your question and answer pair. You review those suggestions, either adding them to existing questions or rejecting them.

Your knowledge base doesn't change automatically. In order for any change to take effect, you must accept the suggestions. These suggestions add questions but don't change or remove existing questions.

How active learning works

Active learning is triggered based on the scores of the top few answers returned by QnA Maker. If the score differences between QnA pairs that match the query lie within a small range, then the query is considered a possible suggestion (as an alternate question) for each of the possible QnA pairs. Once you accept the suggested question for a specific QnA pair, it is rejected for the other pairs. You need to remember to save and train, after accepting suggestions.

Active learning gives the best possible suggestions in cases where the endpoints are getting a reasonable quantity and variety of usage queries. When five or more similar queries are clustered, every 30 minutes, QnA Maker suggests the user-based questions to the knowledge base designer to accept or reject. All the suggestions are clustered together by similarity and top suggestions for alternate questions are displayed based on the frequency of the particular queries by end users.

<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/how-to/use-active-learning>

Manage collaborators

Collaborate with other authors and editors

Collaborate with other authors and editors using Azure role-based access control (Azure RBAC) placed on your QnA Maker resource.

Authenticate by QnA Maker portal

If you author and collaborate using the QnA Maker portal, after you add the appropriate role to the resource for a collaborator, the QnA Maker portal manages all the access permissions.

Authenticate by QnA Maker APIs and SDKs

If you author and collaborate using the APIs, either through REST or the SDKs, you need to create a service principal to manage the authentication.

<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/concepts/role-based-access-control>

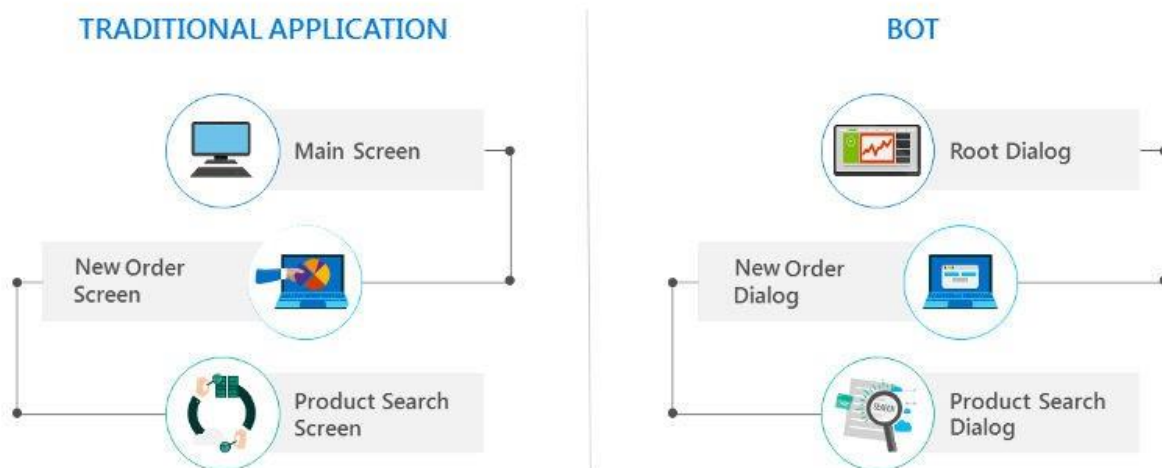
Design and implement conversation flow

Design conversation logic for a bot

Conversations with a bot are generally focused on the task a bot is trying to achieve, which is called a procedural flow. This is where the bot asks the user a series of questions to gather all the information it needs before processing the task.

In a procedural conversation flow, you define the order of the questions and the bot will ask the questions in the order you defined. You can organize the questions into logical groups to keep the code centralized while staying focused on guiding the conversation. For example, you may design one module to contain the logic that helps the user browse for products and a separate module to contain the logic that helps the user create a new order.

You can structure these modules to flow in any way you like, ranging from free form to sequential. The Bot Framework SDK provides a dialogs library that allows you to construct any conversational flow your bot needs. The library includes waterfall dialogs for creating a sequence of steps and prompts for asking users questions.



In a traditional application, everything begins with the main screen. The main screen invokes the new order screen. The new order screen remains in control until it either closes or invokes other screens, such as the product search screen. If the new order screen closes, the user is returned to the main screen.

In a bot that uses dialogs, everything begins with the root dialog. The root dialog invokes the new order dialog. At that point, the new order dialog takes control of the conversation and remains in control until it either closes or invokes another dialog, such as the product search dialog. If the new order dialog closes, control of the conversation returns back to the root dialog.

<https://docs.microsoft.com/en-us/azure/bot-service/bot-service-design-conversation-flow?view=azure-bot-service-4.0>

Create and evaluate *.chat file conversations by using the Bot Framework Emulator

One of the keys to successful testing and debugging a bot is your ability to record and examine the set of conditions that occur when running your bot. This article discusses the creation and use of a bot transcript file to provide a detailed set of user interactions and bot responses for testing and debugging.

A bot transcript file is a specialized JSON file that preserves the interactions between a user and your bot. A transcript file preserves not only the contents of a message, but also interaction details such as the user ID, channel ID, channel type, channel capabilities, time of

the interaction, etc. All of this information can then be used to help find and resolve issues when testing or debugging your bot.

Creating/Storing a bot transcript file

This article shows how to create bot transcript files using Microsoft's Bot Framework Emulator. Transcript files may also be created programmatically; see Blob transcript storage to read more concerning that approach. In this article we will use the Bot Framework sample code for Multi Turn Prompt Bot that requests a user's mode of transportation, name and age, but any code that can be accessed using Microsoft's Bot Framework Emulator may be used to create a transcript file.

To begin this process ensure that the bot code you want to test is running within your development environment. Start the bot framework Emulator, select the *Open Bot* button, then enter the address of *localhost:port* shown in your browser followed by */api/messages* as shown in the image below. Now click the *Connect* button to connect the Emulator to your bot.

After connecting the Emulator to your running code, test your code by sending simulated user interactions to the bot. For this example we have passed in the user's mode of transportation, name and age. After you have entered all of the user interactions you want to preserve, use the bot framework Emulator to create and save a transcript file containing this conversation.

Choose a location and name for your transcript file and then select the save button.

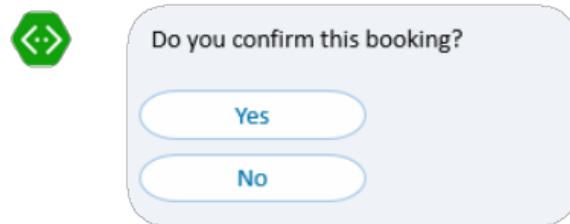
All of the user interactions and bot responses that you entered to test your code with the Emulator have now been saved into a transcript file that you can later reload to help debug interactions between your user and your bot.

<https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-debug-transcript?view=azure-bot-service-4.0>

Choose an appropriate conversational model for a bot, including activity handlers and dialogs

It may be tempting to assume that users will perform procedural tasks one by one in a neat and orderly way. For example, in a procedural conversation flow using dialogs, the user will start at the root dialog and invoke the new order dialog. From the new order dialog, they invoke the product search dialog. Then when selecting one of the results listed in product search dialog, they invoke the new order dialog. After completing the order, they arrive back at the root dialog.

Although it would be great if users always traveled such a linear, logical path, it seldom occurs. People do not always communicate in sequential order. They tend to frequently change their minds. Consider the following example:



Actually... what was the time of that movie again?

While your bot may be procedural centric, the user may decide to do something entirely different or ask a question that may be unrelated to the current topic. In the example above, the user asks a question rather than providing the yes/no response that the bot expects. How should your bot respond?

- Insist that the user answer the question first.
- Disregard everything that the user had done previously, reset the whole dialog stack, and start from the beginning by attempting to answer the user's question.
- Attempt to answer the user's question and then return to that yes/no question and try to resume from there.

There is no *right* answer to this question, as the best solution will depend upon the specifics of your scenario and how the user would reasonably expect the bot to respond.

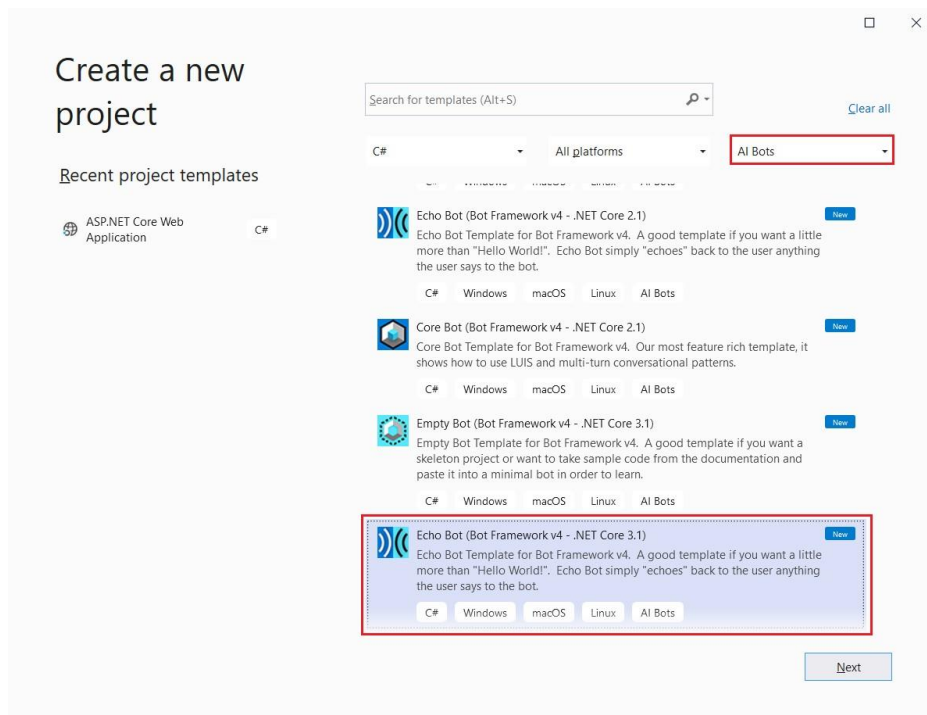
<https://docs.microsoft.com/en-us/azure/bot-service/bot-service-design-conversation-flow?view=azure-bot-service-4.0>

Create a bot by using the Bot Framework SDK

Use the Bot Framework SDK to create a bot from a template

Creating your first bot doesn't require an Azure subscription or an Azure Bot Service resource. This quickstart focuses on creating your first bot locally.

In Visual Studio, create a new bot project using the **Echo Bot (Bot Framework v4 - .NET Core 3.1)** template. To see only bot templates, choose **AI Bots** from the project types.



Thanks to the template, your project contains all the necessary code to create the bot in this quickstart. You don't need any additional code to test your bot.

<https://docs.microsoft.com/en-us/azure/bot-service/bot-service-quickstart-create-bot?view=azure-bot-service-4.0>

Implement activity handlers and dialogs

This provides an extensible class for handling incoming activities in an event-driven way. You can register an arbitrary set of handlers for each event type.

To register a handler for an event, use the corresponding *on event* method. If multiple handlers are registered for an event, they are run in the order in which they were registered.

This object emits a series of *events* as it processes an incoming activity. A handler can stop the propagation of the event by not calling the continuation function.

Event type

Turn

Type-specific

Sub-type

Dialog

Description

Emitted first for every activity.

Emitted for the specific activity type, before emitting an

Emitted for certain specialized events, based on activity con-

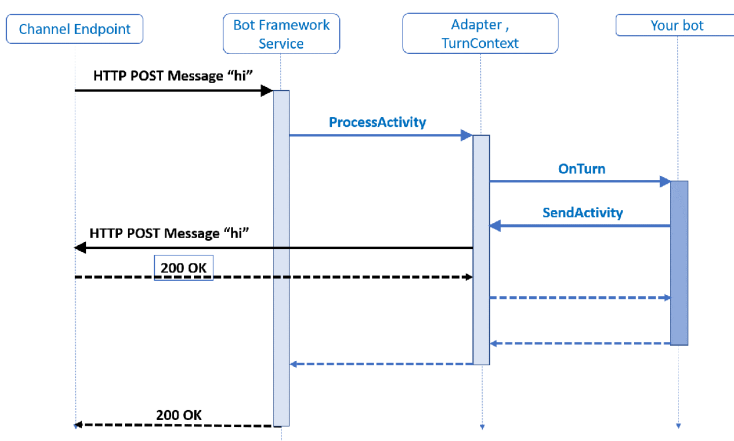
Emitted as the final activity processing event.

<https://docs.microsoft.com/en-us/javascript/api/botbuilder-core/activityhandler?view=botbuilder-ts-latest>

Use Turn Context

The *turn context* object provides information about the activity such as the sender and receiver, the channel, and other data needed to process the activity. It also allows for the addition of information during the turn across various layers of the bot.

The turn context is one of the most important abstractions in the SDK. Not only does it carry the inbound activity to all the middleware components and the application logic but it also provides the mechanism whereby the middleware components and the bot logic can send outbound activities.



In the example above, the bot replied to the message activity with another message activity containing the same text message. Processing starts with the HTTP POST request, with the activity information carried as a JSON payload, arriving at the web server. In C# this will typically be an ASP.NET project, in a JavaScript Node.js project this is likely to be one of the popular frameworks such as Express or restify.

The *adapter*, an integrated component of the SDK, is the core of the SDK runtime. The activity is carried as JSON in the HTTP POST body. This JSON is deserialized to create the *activity* object that is then handed to the adapter through its *process activity* method. On receiving the activity, the adapter creates a *turn context* and calls the middleware.

As mentioned above, the turn context provides the mechanism for the bot to send outbound activities, most often in response to an inbound activity. To achieve this, the turn context

provides *send*, *update*, and *delete activity* response methods. Each response method runs in an asynchronous process.

<https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-basics?view=azure-bot-service-4.0>

Test a bot using the Bot Framework Emulator

Bot Framework Emulator is a desktop application that allows bot developers to test and debug bots, either locally or remotely. Using the Emulator, you can chat with your bot and inspect the messages that your bot sends and receives. The Emulator displays messages as they would appear in a web chat UI and logs JSON requests and responses as you exchange messages with your bot. Before you deploy your bot to the cloud, run it locally and test it using the Emulator. You can test your bot using the Emulator even if you haven't yet created it with Azure Bot Service or configured it to run on any channels.

Run a bot locally

Before connecting your bot to the Bot Framework Emulator, you need to run your bot locally. You can use Visual Studio or Visual Studio Code to run your bot, or use command line. To run a bot using command line, do the following:

- Go to the command prompt and change directory to your bot project directory.
- Start the bot by running the following command:
dotnet run
- Copy the port number in the line before *Application started*. Press CTRL+C to shut down.

<https://docs.microsoft.com/en-us/azure/bot-service/bot-service-debug-emulator?view=azure-bot-service-4.0>

Deploy a bot to Azure

Starting with Bot Framework SDK 4.14.1.2, when you create a bot from a VSIX or Yeoman template, the resulting project contains ARM templates. The samples and any new bots created from a Bot Framework template contain a *deployment templates* directory that contains the ARM templates. You'll use the ARM templates to provision many of your bot resources.

<https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-deploy-az-cli?view=azure-bot-service-4.0>

Create a bot by using the Bot Framework Composer

Implement dialogs

Modern conversational software has many different components. Bot Framework Composer integrates these pieces into *dialogs*, a single interface for constructing the building blocks of bot functionality.

Each dialog represents a portion of the bot's functionality and contains instructions for how the bot will react to the input. Dialogs can include custom business logic, calls to cloud APIs, training data for language processing systems, and importantly, the actual content used in conversations with the bot's end users. Simple bots will have just a few dialogs. Sophisticated bots might have dozens or hundreds of individual dialogs.

In Composer, dialogs are functional components offered in a visual interface that reduces the need to write code. The dialog system supports building an extensible model that integrates all of the building blocks of a bot's functionality. Composer helps you focus on conversation modeling rather than the mechanics of dialog management.

Dialog types

You create a dialog in Composer to manage a conversation task. There are two types of dialogs in Composer: *main dialog* and *child dialog*. The main dialog is initialized by default when you create a new bot. You can create one or more child dialogs to keep the dialog system organized. Each bot has one main dialog and can have zero or more child dialogs. Refer to the Create a bot article on how to create a bot and its main dialog in Composer. Refer to the Add a dialog article on how to create a child dialog and wire it up in the dialog system.

<https://docs.microsoft.com/en-us/composer/concept-dialog>

Maintain state

For many bots, state (where the conversation left off or data previously received about the user) is necessary for the bot to have a useful conversation. Composer creates for your bot various *properties* and *memory scopes* in which to track your bot's active state. Some properties are maintained for you, such as which dialog is active, but you can also save and retrieve your own properties.

With the memory system, you can:

- Use properties in adaptive expressions.

- Use properties in bot responses and response templates.
- Store user profiles and preferences.
- Remember things between sessions, such as the last search query or a list of recently mentioned locations.
- Pass information between dialogs.

<https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-concept-state?view=azure-bot-service-4.0>

Implement logging for a bot conversation

Add trace activities to your bot

A *trace activity* is an activity that your bot can send to the Bot Framework Emulator. You can use trace activities to interactively debug a bot, as they allow you to view information about your bot while it runs locally.

Trace activities are sent only to the Emulator and not to any other client or channel. The Emulator displays them in the log but not the main chat panel.

- Trace activities sent via the turn context are sent through the *send activity handlers* registered on the turn context.
- Trace activities sent via the turn context are associated with the inbound activity, by applying the conversation reference, if there was one. For a proactive message, the *reply to ID* will be a new GUID.
- Regardless of how it is sent, a trace activity never sets the *responded* flag.

<https://docs.microsoft.com/en-us/azure/bot-service/using-trace-activities?view=azure-bot-service-4.0>

Implement a prompt for user input

Create your own prompts to gather user input

A conversation between a bot and a user often involves asking (prompting) the user for information, parsing the user's response, and then acting on that information. Your bot should track the context of a conversation, so that it can manage its behavior and remember answers

to previous questions. A bot's *state* is information it tracks to respond appropriately to incoming messages.

<https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-primitive-prompts?view=azure-bot-service-4.0>

Troubleshoot a conversational bot

How can I troubleshoot issues with my bot?

1. Debug your bot's source code with Visual Studio Code or Visual Studio.
2. Test your bot using the Emulator before you deploy it to the cloud.
3. Deploy your bot to a cloud hosting platform such as Azure and then test connectivity to your bot by using the built-in web chat control on your bot's dashboard in the Azure portal. If you encounter issues with your bot after you deploy it to Azure, you might consider using this blog article: Understanding Azure troubleshooting and support.
4. Rule out authentication as a possible issue.
5. Test your bot on Web Chat, Teams, or any other channel you intend to use with your bot. This will help you to validate the end-to-end user experience.
6. Consider testing your bot on channels that have additional authentication requirements such as Direct Line or Web Chat.
7. Review the how-to debug a bot and the other debugging articles in that section.

I'm using the Bot Framework SDK for .NET. How can I troubleshoot issues with my bot?

Look for exceptions. In Visual Studio 2019, go to **Debug > Windows > Exception Settings**. In the **Exceptions Settings** window, select the **Break When Thrown** checkbox next to **Common Language Runtime Exceptions**. You may also see diagnostics output in your Output window when there are thrown or unhandled exceptions.

Look at the call stack. In Visual Studio, you can choose whether or you are debugging Just My Code or not. Examining the full call stack may provide additional insight into any issues.

Ensure all dialog methods end with a plan to handle the next message. All dialog steps need to feed into the next step of the waterfall, or end the current dialog to pop it off the stack. If a step is not correctly handled, the conversation will not continue like you expect. Take a look at the concept article for dialogs for more on dialogs.

<https://docs.microsoft.com/en-us/azure/bot-service/bot-service-troubleshoot-general-problems?view=azure-bot-service-4.0>

Test a bot

Test and debug with the Emulator

Bot Framework Emulator is a desktop application that allows bot developers to test and debug bots, either locally or remotely. Using the Emulator, you can chat with your bot and inspect the messages that your bot sends and receives. The Emulator displays messages as they would appear in a web chat UI and logs JSON requests and responses as you exchange messages with your bot. Before you deploy your bot to the cloud, run it locally and test it using the Emulator. You can test your bot using the Emulator even if you haven't yet created it with Azure Bot Service or configured it to run on any channels.

Use bot credentials

When you open the bot, set the **Microsoft App ID** and **Microsoft App password** if your bot is running with credentials. If you created your bot with the Azure Bot Service, the credentials are available on the bot's App Service, under the **Settings -> Configuration** section. If you do not know the values, you can remove those from the locally running bot's configuration file, then run the bot in the Emulator. If the bot isn't running with these settings, you don't need to run the Emulator with the settings either.

When creating an AD identity provider application, remember the following:

- When the supported account types is set to single tenant, if you use a personal subscription instead of a Microsoft account, the Emulator would issue the error: *The bot's Microsoft App ID or Microsoft App Password is incorrect..*
- In this case, the supported account types must be set to *Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Xbox).*

<https://docs.microsoft.com/en-us/azure/bot-service/bot-service-debug-emulator?view=azure-bot-service-4.0>

Publish a bot

Publish a bot to Azure from Composer

This article describes how to sign in to Azure, provision Azure resources, import existing Azure resources, and publish a basic bot to *Azure Web App (Preview)*, all from within Bot Framework Composer.

Sign in to Azure

1. Select **Publish** in the navigation pane.
2. In the **Publish your bots** pane, select the **Publish** tab.
3. Select the bot you want to publish.
4. In the Publish target section, select Manage profiles from the drop-down menu.

<https://docs.microsoft.com/en-us/composer/how-to-publish-bot>

Add language generation for a response

Language generation lets you define multiple variations of a phrase, execute simple expressions based on context, and refer to conversational memory. At the core of language generation lies template expansion and entity substitution. You can provide one-of variation for expansion as well as conditionally expand a template. The output from language generation can be a simple text string, multi-line response, or a complex object payload that a layer above language generation will use to construct a complete activity. Bot Framework Composer natively supports language generation to produce output activities using the LG templating system.

You can use Language generation to:

- Achieve a coherent personality, tone of voice for your bot.
- Separate business logic from presentation.
- Include variations and sophisticated composition for any of your bot's replies.

Construct cards, suggested actions and attachments using a structured response template.

Language generation is achieved through:

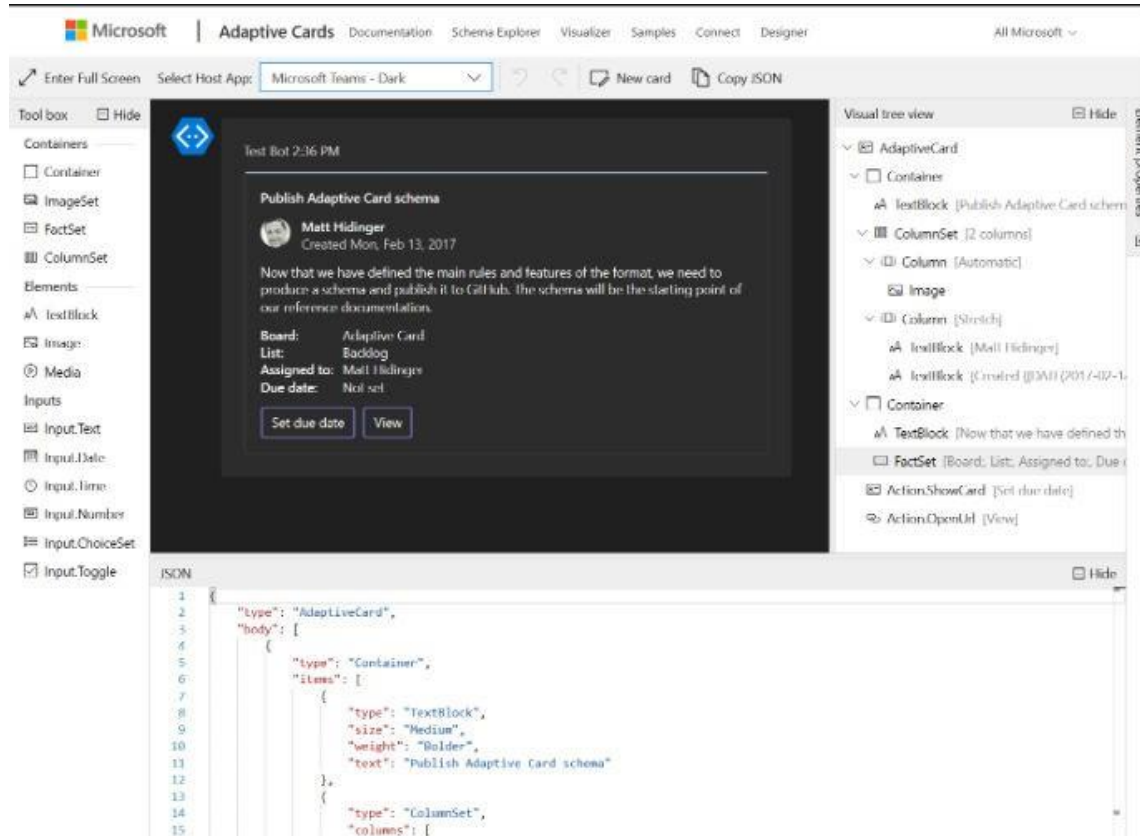
- A Markdown based .lg file that contains the templates and their composition.
- Full access to the current bot's memory so you can data bind language to the state of memory.
- Parser and runtime libraries that help achieve runtime resolution

<https://docs.microsoft.com/en-us/composer/concept-language-generation>

Design and implement adaptive cards

The Adaptive Card Designer provides a rich, interactive design-time experience for authoring adaptive cards.

Try it out at <https://adaptivecards.io/designer>



This SDK allows you to easily integrate the adaptive cards designer into your own experiences.

<https://docs.microsoft.com/en-us/adaptive-cards/sdk/designer>

Integrate Cognitive Services into a bot

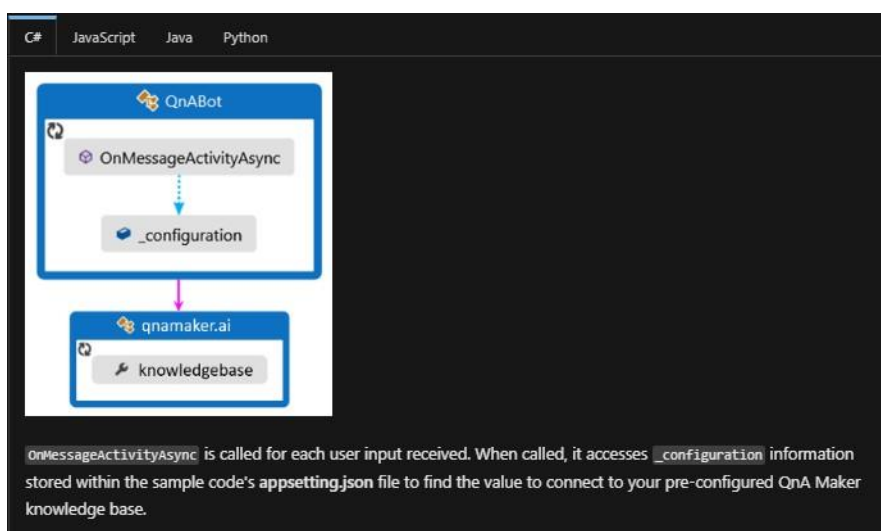
Integrate a QnA Maker service

QnA Maker provides a conversational question and answer layer over your data. This allows your bot to send a question to the QnA Maker and receive an answer without needing to parse and interpret the question intent.

One of the basic requirements in creating your own QnA Maker service is to populate it with questions and answers. In many cases, the questions and answers already exist in content like FAQs or other documentation; other times, you may want to customize your answers to questions in a more natural, conversational way.

About this sample

To use QnA Maker in your bot, you need to create a knowledge base in the QnA Maker portal, as shown in the next section. Your bot then can use the knowledge base to answer the user's questions.



<https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-howto-qna?view=azure-bot-service-4.0>

Integrate a LUIS service

The ability to understand what your user means conversationally and contextually can be a difficult task, but can provide your bot a more natural conversation feel. *Language Understanding (LUIS)* is a cloud-based API service that enables you to do just that so that your bot can recognize the intent of user messages, allow for more natural language from your user, and better direct the conversation flow.

This topic walks you through adding LUIS to a flight booking application to recognize different intents and entities contained within user input.

<https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-howto-v4-luis?view=azure-bot-service-4.0>

Integrate a Speech service

If you are building a bot for a speech-enabled channel, you can construct messages that specify the text to be spoken by your bot. You can also attempt to influence the state of the client's microphone by specifying an input hint to indicate whether your bot is accepting, expecting, or ignoring user input.

You can configure your bot to allow client applications to communicate with it through Direct Line Speech channel.

<https://docs.microsoft.com/en-us/azure/bot-service/rest-api/bot-framework-rest-connector-text-to-speech?view=azure-bot-service-4.0>

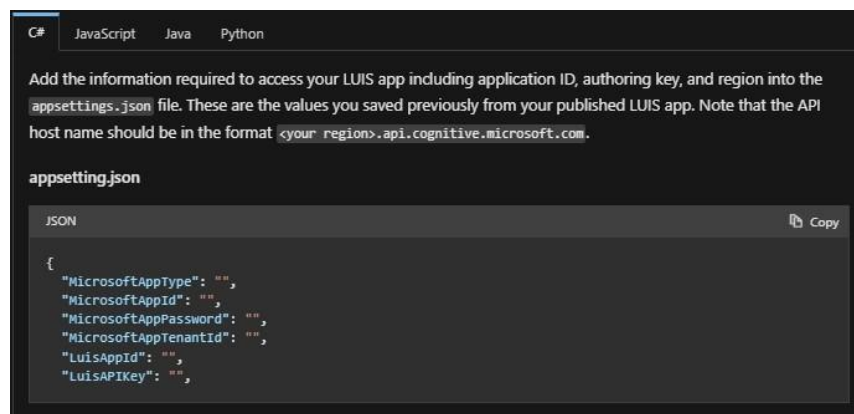
Integrate Orchestrator for multiple language models

If a bot uses multiple LUIS models and QnA Maker knowledge bases (knowledge bases), you can use Orchestrator to determine which LUIS model or QnA Maker knowledge base best matches the user input. The bf orchestrator CLI tool does this by creating a Orchestrator snapshot file that will be used to route user input to the correct model at run time. For more information about Orchestrator, including the CLI commands.

<https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-tutorial-orchestrator?view=azure-bot-service-4.0>

Manage keys in app settings file

Update the settings file



```
C# JavaScript Java Python

Add the information required to access your LUIS app including application ID, authoring key, and region into the
appsettings.json file. These are the values you saved previously from your published LUIS app. Note that the API
host name should be in the format <your region>.api.cognitive.microsoft.com.

appsetting.json

JSON Copy

{
  "MicrosoftAppType": "",
  "MicrosoftAppId": "",
  "MicrosoftAppPassword": "",
  "MicrosoftAppTenantId": "",
  "LuisAppId": "",
  "LuisAPIKey": ""
}
```

<https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-howto-v4-luis?view=azure-bot-service-4.0>