Contents

What are Azure Cognitive Services?

10/4/2020 • 5 minutes to read • Edit Online

Azure Cognitive Services are cloud-base 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.

The catalog of AI services, which provide cognitive understanding are categorized into five main pillars:

- Vision
- Speech
- Language
- Web Search
- Decision

The current list of new documentation is available at What's new in Cognitive Services docs.

Vision APIs

SERVICE NAME	SERVICE DESCRIPTION
Computer Vision	The Computer Vision service provides you with access to advanced cognitive algorithms for processing images and returning information.
Custom Vision Service	The Custom Vision Service allows you to build custom image classifiers.
Face	The Face service provides access to advanced face algorithms, enabling face attribute detection and recognition.
Form Recognizer (Preview)	Form Recognizer identifies and extracts key-value pairs and table data from form documents; then outputs structured data including the relationships in the original file.
Ink Recognizer (Retiring)	Ink Recognizer allows you to recognize and analyze digital ink stroke data, shapes and handwritten content, and output a document structure with all recognized entities.
Video Indexer	Video Indexer enables you to extract insights from your video.

Speech APIs

SERVICE NAME	SERVICE DESCRIPTION
Speech service	Speech service adds speech-enabled features to applications.
Speaker Recognition API (Preview)	The Speaker Recognition API provides algorithms for speaker identification and verification.

SERVICE NAME	SERVICE DESCRIPTION
Bing Speech (Retiring)	The Bing Speech API provides you with an easy way to create speech-enabled features in your applications.
Translator Speech (Retiring)	Translator Speech is a machine translation service.

NOTE

Looking for Azure Cognitive Search? Although it uses Cognitive Services for some tasks, it's a different search technology that supports other scenarios.

Language APIs

SERVICE NAME	SERVICE DESCRIPTION
Language Understanding LUIS	Language Understanding service (LUIS) allows your application to understand what a person wants in their own words.
QnA Maker	QnA Maker allows you to build a question and answer service from your semi-structured content.
Text Analytics	Text Analytics provides natural language processing over raw text for sentiment analysis, key phrase extraction and language detection.
Translator	Translator provides machine-based text translation in near real-time.

Search APIs

SERVICE NAME	SERVICE DESCRIPTION
Bing News Search	Bing News Search returns a list of news articles determined to be relevant to the user's query.
Bing Video Search	Bing Video Search returns a list of videos determined to be relevant to the user's query.
Bing Web Search	Bing Web Search returns a list of search results determined to be relevant to the user's query.
Bing Autosuggest	Bing Autosuggest allows you to send a partial search query term to Bing and get back a list of suggested queries.
Bing Custom Search	Bing Custom Search allows you to create tailored search experiences for topics that you care about.
Bing Entity Search	Bing Entity Search returns information about entities that Bing determines are relevant to a user's query.

SERVICE NAME	SERVICE DESCRIPTION
Bing Image Search	Bing Image Search returns a display of images determined to be relevant to the user's query.
Bing Visual Search	Bing Visual Search provides returns insights about an image such as visually similar images, shopping sources for products found in the image, and related searches.
Bing Local Business Search	Bing Local Business Search API enables your applications to find contact and location information about local businesses based on search queries.
Bing Spell Check	Bing Spell Check allows you to perform contextual grammar and spell checking.

Decision APIs

SERVICE NAME	SERVICE DESCRIPTION
Anomaly Detector (Preview)	Anomaly Detector allows you to monitor and detect abnormalities in your time series data.
Content Moderator	Content Moderator provides monitoring for possible offensive, undesirable, and risky content.
Metrics Advisor (Preview)	Metrics Advisor provides customizable anomaly detection on multi-variate time series data, and a fully featured web portal to help you use the service.
Personalizer	Personalizer allows you to choose the best experience to show to your users, learning from their real-time behavior.

Learn with the Quickstarts

Learn about creating a Cognitive Services resource with hands-on quickstarts, using the:

- Azure portal
- Azure CLI
- Azure SDK client libraries
- Azure Resource Manager (ARM) templates

Subscription management

Once you are signed in with your Microsoft Account, you can access My subscriptions to show the products you are using, the quota remaining, and the ability to add additional products to your subscription.

Upgrade to unlock higher limits

All APIs have a free tier, which has usage and throughput limits. You can increase these limits by using a paid offering and selecting the appropriate pricing tier option when deploying the service in the Azure portal. Learn more about the offerings and pricing. You'll need to set up an Azure subscriber account with a credit card and a phone number. If you have a special requirement or simply want to talk to sales, click "Contact us" button at the top the pricing page.

Regional availability

The APIs in Cognitive Services are hosted on a growing network of Microsoft-managed data centers. You can find the regional availability for each API in Azure region list.

Looking for a region we don't support yet? Let us know by filing a feature request on our UserVoice forum.

Supported cultural languages

Cognitive Services supports a wide range of cultural languages at the service level. You can find the language availability for each API in the supported languages list.

Securing resources

Azure Cognitive Services provides a layered security model, including authentication via Azure Active Directory credentials, a valid resource key, and Azure Virtual Networks.

Container support

Cognitive Services provides containers for deployment in the Azure cloud or on-premises. Learn more about Cognitive Services Containers.

Certifications and compliance

Cognitive Services has been awarded certifications such as CSA STAR Certification, FedRAMP Moderate, and HIPAA BAA.

You can download certifications for your own audits and security reviews.

To understand privacy and data management, go to the Trust Center.

Support

Cognitive Services provides several support options.

Next steps

- Create a Cognitive Services account

Cognitive Services docs: What's new for August 1, 2020 - August 31, 2020

10/4/2020 • 2 minutes to read • Edit Online

Welcome to what's new in the Cognitive Services docs from August 1, 2020 through August 31, 2020. This article lists some of the major changes to docs during this period.

Computer Vision

New articles

- Upgrade to v3.0 of Computer Vision API from v2.0 and v2.1

Containers

Updated articles

- Azure Cognitive Services container image tags

Form Recognizer

New articles

- Business card concepts
- Receipt concepts
- Quickstart: Extract business card data using the Form Recognizer REST API with Python

Updated articles

- Quickstart: Train a Form Recognizer model and extract form data by using the REST API with cURL
- Quickstart: Train a Form Recognizer model with labels using REST API and Python
- Quickstart: Extract text and layout information using the Form Recognizer REST API with Python
- Quickstart: Train a Form Recognizer model and extract form data by using the REST API with Python

Speech Service

New articles

- Speech Services quotas and limits

Updated articles

- Long Audio API (Preview)

Community contributors

The following people contributed to the Cognitive Services docs during this period. Thank you!

- hyoshioka0128 - Hiroshi Yoshioka (2)
- anwesh-b - Anwesh Budhathoki (1)
- jangelfdez - José Ángel Fernández (1)
- xhan742 (1)

Service-specific updates

- Computer Vision
- Custom Vision
- Form Recognizer
- Language Understanding (LUIS)
- Personalizer
- QnA Maker
- Speech service
- Text Analytics

Cognitive Services and machine learning

10/4/2020 • 5 minutes to read • Edit Online

Cognitive Services provides machine learning capabilities to solve general problems such as analyzing text for emotional sentiment or analyzing images to recognize objects or faces. You don't need special machine learning or data science knowledge to use these services.

Cognitive Services is a group of services, each supporting different, generalized prediction capabilities. The services are divided into different categories to help you find the right service.

SERVICE CATEGORY	PURPOSE	
Decision	Build apps that surface recommendations for informed and efficient decision-making.	
Language	Allow your apps to process natural language with pre-built scripts, evaluate sentiment and learn how to recognize what users want.	
Search	Add Bing Search APIs to your apps and harness the ability to comb billions of webpages, images, videos, and news with a single API call.	
Speech	Convert speech into text and text into natural-sounding speech. Translate from one language to another and enable speaker verification and recognition.	
Vision	Recognize, identify, caption, index, and moderate your pictures, videos, and digital ink content.	

Use Cognitive Services when you:

- Can use a generalized solution.
- Access solution from a programming REST API or SDK.

Use another machine-learning solution when you:

- Need to choose the algorithm and need to train on very specific data.

What is machine learning?

Machine learning is a concept where you bring together data and an algorithm to solve a specific need. Once the data and algorithm are trained, the output is a model that you can use again with different data. The trained model provides insights based on the new data.

The process of building a machine learning system requires some knowledge of machine learning or data science.

Machine learning is provided using Azure Machine Learning (AML) products and services.

What is a Cognitive Service?

A Cognitive Service provides part or all of the components in a machine learning solution: data, algorithm, and trained model. These services are meant to require general knowledge about your data without needing experience with machine learning or data science. These services provide both REST API(s) and language-based SDKs. As a result, you need to have programming language knowledge to use the services.

How are Cognitive Services and Azure Machine Learning (AML) similar?

Both have the end-goal of applying artificial intelligence (AI) to enhance business operations, though how each provides this in the respective offerings is different.

Generally, the audiences are different:

- Cognitive Services are for developers without machine-learning experience.
- Azure Machine Learning is tailored for data scientists.

How is a Cognitive Service different from machine learning?

A Cognitive Service provides a trained model for you. This brings data and an algorithm together, available from a REST API(s) or SDK. You can implement this service within minutes, depending on your scenario. A Cognitive Service provides answers to general problems such as key phrases in text or item identification in images.

Machine learning is a process that generally requires a longer period of time to implement successfully. This time is spent on data collection, cleaning, transformation, algorithm selection, model training, and deployment to get to the same level of functionality provided by a Cognitive Service. With machine learning, it is possible to provide answers to highly specialized and/or specific problems. Machine learning problems require familiarity with the specific subject matter and data of the problem under consideration, as well as expertise in data science.

What kind of data do you have?

Cognitive Services, as a group of services, can require none, some, or all custom data for the trained model.

No additional training data required

Services that provide a fully-trained model can be treated as a *opaque box*. You don't need to know how they work or what data was used to train them. You bring your data to a fully trained model to get a prediction.

Some or all training data required

Some services allow you to bring your own data, then train a model. This allows you to extend the model using the Service's data and algorithm with your own data. The output matches your needs. When you bring your own data, you may need to tag the data in a way specific to the service. For example, if you are training a model to identify flowers, you can provide a catalog of flower images along with the location of the flower in each image to train the model.

A service may *allow* you to provide data to enhance its own data. A service may *require* you to provide data.

Real-time or near real-time data required

A service may need real-time or near-real time data to build an effective model. These services process significant amounts of model data.

Service requirements for the data model

The following data categorizes each service by which kind of data it allows or requires.

COGNITIVE SERVICE	NO TRAINING DATA REQUIRED	YOU PROVIDE SOME OR ALL TRAINING DATA	REAL-TIME OR NEAR REAL-TIME DATA COLLECTION
Anomaly Detector	x	x	x
Bing Search	x		
Computer Vision	x		
Content Moderator	x		x
Custom Vision		x	
Face	x	x	
Form Recognizer		x	
Immersive Reader	x		
Ink Recognizer	x	x	
Language Understanding (LUIS)		x	
Personalizer	x*	x*	x
QnA Maker		x	
Speaker Recognizer		x	
Speech Text-to-speech (TTS)	x	x	
Speech Speech-to-text (STT)	x	x	
Speech Translation	x		
Text Analytics	x		
Translator	x		
Translator - custom translator		x	

*Personalizer only needs training data collected by the service (as it operates in real-time) to evaluate your policy and data. Personalizer does not need large historical datasets for up-front or batch training.

Where can you use Cognitive Services?

The services are used in any application that can make REST API(s) or SDK calls. Examples of applications include web sites, bots, virtual or mixed reality, desktop and mobile applications.

How is Azure Cognitive Search related to Cognitive Services?

Azure Cognitive Search is a separate cloud search service that optionally uses Cognitive Services to add image and natural language processing to indexing workloads. Cognitive Services is exposed in Azure Cognitive Search through built-in skills that wrap individual APIs. You can use a free resource for walkthroughs, but plan on creating and attaching a billable resource for larger volumes.

How can you use Cognitive Services?

Each service provides information about your data. You can combine services together to chain solutions such as converting speech (audio) to text, translating the text into many languages, then using the translated languages to get answers from a knowledge base. While Cognitive Services can be used to create intelligent solutions on their own, they can also be combined with traditional machine learning projects to supplement models or accelerate the development process.

Cognitive Services that provide exported models for other machine learning tools:

COGNITIVE SERVICE	MODEL INFORMATION
Custom Vision	Export for Tensorflow for Android, CoreML for iOS11, ONNX for Windows ML

Learn more

- Architecture Guide - What are the machine learning products at Microsoft?
- Machine learning - Introduction to deep learning vs. machine learning

Next steps

- Create your Cognitive Service account in the Azure portal or with Azure CLI.
- Learn how to authenticate to a Cognitive Service.
- Use diagnostic logging for issue identification and debugging.
- Deploy a Cognitive Service in a Docker container.
- Keep up to date with service updates.

Custom subdomain names for Cognitive Services

5/19/2020 • 2 minutes to read • Edit Online

Azure Cognitive Services use custom subdomain names for each resource created through the Azure portal, Azure Cloud Shell, or Azure CLI. Unlike regional endpoints, which were common for all customers in a specific Azure region, custom subdomain names are unique to the resource. Custom subdomain names are required to enable features like Azure Active Directory (Azure AD) for authentication.

How does this impact existing resources?

Cognitive Services resources created before July 1, 2019 will use the regional endpoints for the associated service. These endpoints will work with existing and new resources.

If you'd like to migrate an existing resource to leverage custom subdomain names, so that you can enable features like Azure AD, follow these instructions:

1. Sign in to the Azure portal and locate the Cognitive Services resource that you'd like to add a custom subdomain name to.
2. In the **Overview** blade, locate and select **Generate Custom Domain Name**.
3. This opens a panel with instructions to create a unique custom subdomain for your resource.

> **WARNING**
> After you've created a custom subdomain name it **cannot** be changed.

Do I need to update my existing resources?

No. The regional endpoint will continue to work for new and existing Cognitive Services and the custom subdomain name is optional. Even if a custom subdomain name is added the regional endpoint will continue to work with the resource.

What if an SDK asks me for the region for a resource?

> **WARNING**
> The Speech Services **do not** support custom subdomains at this time. Please use the regional endpoints when using the Speech Services and associated SDKs.

Regional endpoints and custom subdomain names are both supported and can be used interchangeably. However, the full endpoint is required.

Region information is available in the **Overview** blade for your resource in the Azure portal. For the full list of regional endpoints, see Is there a list of regional endpoints?

Are custom subdomain names regional?

Yes. Using a custom subdomain name doesn't change any of the regional aspects of your Cognitive Services resource.

What are the requirements for a custom subdomain name?

A custom subdomain name is unique to your resource. The name can only include alphanumeric characters and the `-` character; it must be between 2 and 64 characters in length and cannot end with a `-`.

Can I change a custom domain name?

No. After a custom subdomain name is created and associated with a resource it cannot be changed.

Can I reuse a custom domain name?

Each custom subdomain name is unique, so in order to reuse a custom subdomain name that you've assigned to a Cognitive Services resource, you'll need to delete the existing resource. After the resource has been deleted, you can reuse the custom subdomain name.

Is there a list of regional endpoints?

Yes. This is a list of regional endpoints that you can use with Azure Cognitive Services resources.

> **NOTE**
> The Translator service and Bing Search APIs use global endpoints.

ENDPOINT TYPE	REGION	ENDPOINT
Public	Global (Translator & Bing)	`https://api.cognitive.microsoft.com`
	Australia East	`https://australiaeast.api.cognitive.microsoft.`
	Brazil South	`https://brazilsouth.api.cognitive.microsoft.co`
	Canada Central	`https://canadacentral.api.cognitive.microsoft.`
	Central US	`https://centralus.api.cognitive.microsoft.com`
	East Asia	`https://eastasia.api.cognitive.microsoft.com`
	East US	`https://eastus.api.cognitive.microsoft.com`
	East US 2	`https://eastus2.api.cognitive.microsoft.com`
	France Central	`https://francecentral.api.cognitive.microsoft.`
	India Central	`https://centralindia.api.cognitive.microsoft.c`
	Japan East	`https://japaneast.api.cognitive.microsoft.com`
	Korea Central	`https://koreacentral.api.cognitive.microsoft.c`
	North Central US	`https://northcentralus.api.cognitive.microsoft`
	North Europe	`https://northeurope.api.cognitive.microsoft.co`
	South Africa North	`https://southafricanorth.api.cognitive.microso`
	South Central US	`https://southcentralus.api.cognitive.microsoft`
	Southeast Asia	`https://southeastasia.api.cognitive.microsoft.`
	UK South	`https://uksouth.api.cognitive.microsoft.com`
	West Central US	`https://westcentralus.api.cognitive.microsoft.`
	West Europe	`https://westeurope.api.cognitive.microsoft.com`

ENDPOINT TYPE	REGION	ENDPOINT
	West US	`https://westus.api.cognitive.microsoft.com`
	West US 2	`https://westus2.api.cognitive.microsoft.com`
US Gov	US Gov Virginia	`https://virginia.api.cognitive.microsoft.us`
China	China East 2	`https://chinaeast2.api.cognitive.azure.cn`
	China North	`https://chinanorth.api.cognitive.azure.cn`

See also

- What are the Cognitive Services?
- Authentication

Natural language support for Azure Cognitive Services

10/4/2020 • 2 minutes to read • Edit Online

Azure Cognitive Services enable you to build applications that see, hear, speak with, and understand your users. Between these services, more than three dozen languages are supported, allowing users to communicate with your application in natural ways. Use the links below to view language availability by service.

These Cognitive Services are language agnostic and don't have limitations based on human language.

- Anomaly Detector (Preview)
- Custom Vision
- Face
- Personalizer

Vision

- Computer Vision
- Form Recognizer (Preview)
- Ink Recognizer (Preview)
- Video Indexer

Language

- Immersive Reader
- Language Understanding (LUIS)
- QnA Maker
- Text Analytics
- Translator

Speech

- Speech Service: Speech-to-Text
- Speech Service:Text-to-Speech
- Speech Service: Speech Translation

Search

- Bing Custom Search
- Bing Image Search
- Bing News Search
- Bing Autosuggest
- Bing Spell Check
- Bing Visual Search
- Bing Web Search

Decision

- Content Moderator

See also

- What are the Cognitive Services?
- Create an account

Quickstart: Create a Cognitive Services resource using the Azure portal

10/4/2020 • 4 minutes to read • Edit Online

Use this quickstart to start using Azure Cognitive Services. After creating a Cognitive Service resource in the Azure portal, you'll get an endpoint and a key for authenticating your applications.

Azure Cognitive Services are cloud-base 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.

Prerequisites

- A valid Azure subscription - Create one for free.

Create a new Azure Cognitive Services resource

1. Create a resource.

   - Multi-service resource
   - Single-service resource

   The multi-service resource is named **Cognitive Services** in the portal. Create a Cognitive Services resource.

   At this time, the multi-service resource enables access to the following Cognitive Services:

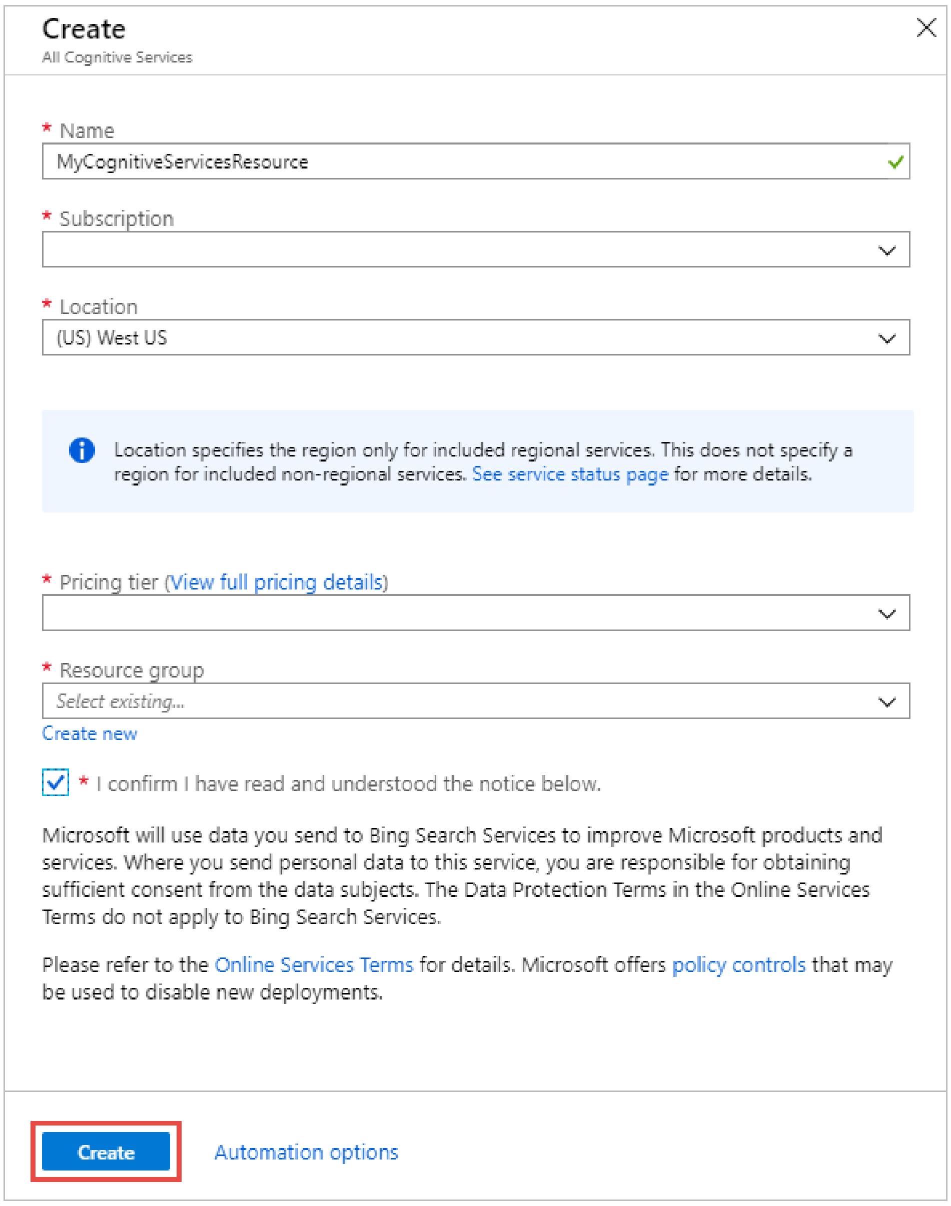
   - Computer Vision
   - Content Moderator
   - Face
   - Language Understanding (LUIS)
   - Text Analytics
   - Translator
   - Bing Search v7
     (Web, Image, News, Video, Visual)
   - Bing Custom Search
   - Bing Entity Search
   - Bing Autosuggest

- Bing Spell Check

2. On the **Create** page, provide the following information:

   - Multi-service resource
   - Single-service resource

Name	A descriptive name for your cognitive services resource. For example, *MyCognitiveServicesResource*.
Subscription	Select one of your available Azure subscriptions.
Location	The location of your cognitive service instance. Different locations may introduce latency, but have no impact on the runtime availability of your resource.
Pricing tier	The cost of your Cognitive Services account depends on the options you choose and your usage. For more information, see the API pricing details.
Resource group	The Azure resource group that will contain your Cognitive Services resource. You can create a new group or add it to a pre-existing group.


Create

All Cognitive Services

* Name

MyCognitiveServicesResource

* Subscription

* Location

(US) West US

Location specifies the region only for included regional services. This does not specify a region for included non-regional services. See service status page for more details.

* Pricing tier (View full pricing details)

* Resource group

Select existing...

Create new

✓ * I confirm I have read and understood the notice below.

Microsoft will use data you send to Bing Search Services to improve Microsoft products and services. Where you send personal data to this service, you are responsible for obtaining sufficient consent from the data subjects. The Data Protection Terms in the Online Services Terms do not apply to Bing Search Services.

Please refer to the Online Services Terms for details. Microsoft offers policy controls that may be used to disable new deployments.

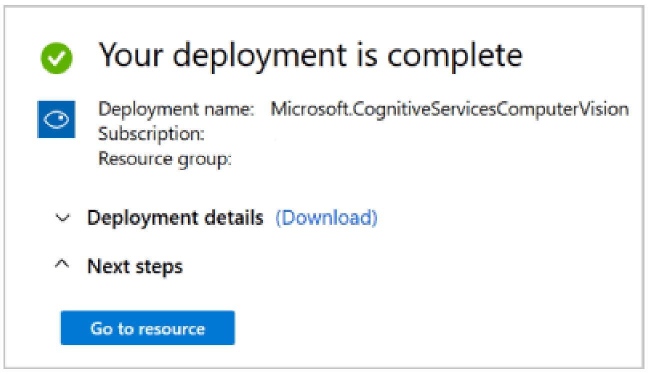
Click **Create**.

TIP

If your subscription doesn't allow you to create a Cognitive Service resource, you may need to enable that ability of the Azure resource provider with the Azure portal, PowerShell command or an Azure CLI command. If you are not the subscription owner, ask the *Subscription Owner* or someone with a role of *admin* to complete the registration for you or ask for the /register/action privileges granted to your account.

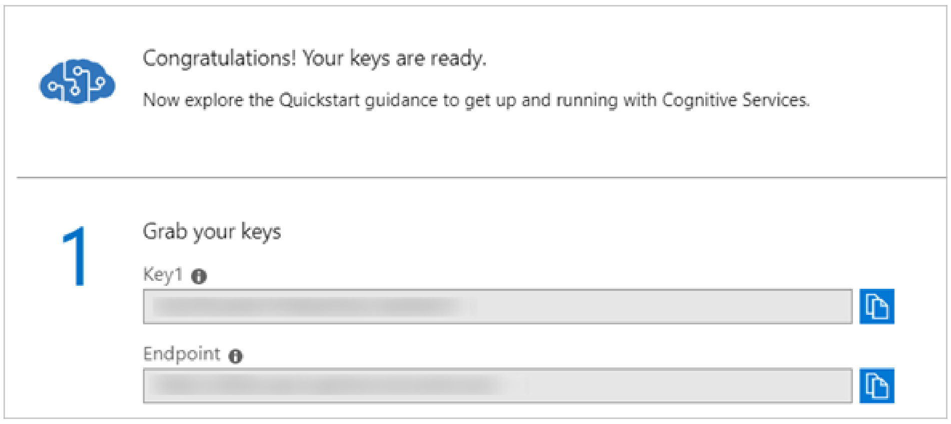
Get the keys for your resource

1. After your resource is successfully deployed, click on **Go to resource** under **Next Steps**.


Your deployment is complete

Deployment name: Microsoft.CognitiveServicesComputerVision
Subscription:
Resource group:

2. From the quickstart pane that opens, you can access your key and endpoint.


Congratulations! Your keys are ready.

Now explore the Quickstart guidance to get up and running with Cognitive Services.

Configure an environment variable for authentication

Applications need to authenticate access to the Cognitive Services they use. To authenticate, we recommend creating an environment variable to store the keys for your Azure Resources.

After you have your key, write it to a new environment variable on the local machine running the application. To set the environment variable, open a console window, and follow the instructions for your operating system. Replace `your-key` with one of the keys for your resource.

- Windows
- Linux
- macOS

```
setx COGNITIVE_SERVICE_KEY "your-key"
```

After you add the environment variable, you may need to restart any running programs that will need to read the environment variable, including the console window. For example, if you are using Visual Studio as your editor, restart Visual Studio before running the example.

Clean up resources

If you want to clean up and remove a Cognitive Services subscription, you can delete the resource or resource group. Deleting the resource group also deletes any other resources contained in the group.

1. In the Azure portal, expand the menu on the left side to open the menu of services, and choose **Resource Groups** to display the list of your resource groups.
2. Locate the resource group containing the resource to be deleted
3. Right-click on the resource group listing. Select **Delete resource group**, and confirm.

See also

- Authenticate requests to Azure Cognitive Services
- What is Azure Cognitive Services?
- Create a new resource using the Azure Management client library
- Natural language support
- Docker container support

Quickstart: Create a Cognitive Services resource using the Azure Command-Line Interface(CLI)

10/4/2020 • 6 minutes to read • Edit Online

Use this quickstart to get started with Azure Cognitive Services using the Azure Command Line Interface(CLI).

Azure Cognitive Services are cloud-base 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.

Cognitive Services are represented by Azure resources that you create in your Azure subscription. After creating the resource, Use the keys and endpoint generated for you to authenticate your applications.

In this quickstart, you'll learn how to sign up for Azure Cognitive Services and create an account that has a single-service or multi-service subscription, Using the Azure Command Line Interface(CLI). These services are represented by Azure resources, which enable you to connect to one or more of the Azure Cognitive Services APIs.

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.

Prerequisites

- A valid Azure subscription - Create one for free.
- The Azure Command Line Interface(CLI)

Install the Azure CLI and sign in

Install the Azure CLI. To sign into your local installation of the CLI, run the az login command:

```
az login
```

You can also use the green **Try It** button to run these commands in your browser.

Create a new Azure Cognitive Services resource group

Before creating a Cognitive Services resource, you must have an Azure resource group to contain the resource. When you create a new resource, you have the option to either create a new resource group, or use an existing one. This article shows how to create a new resource group.

Choose your resource group location

To create a resource, you'll need one of the Azure locations available for your subscription. You can retrieve a list of available locations with the az account list-locations command. Most Cognitive Services can be accessed from several locations. Choose the one closest to you, or see which locations are available for the service.

IMPORTANT

- Remember your Azure location, as you will need it when calling the Azure Cognitive Services.
- The availability of some Cognitive Services can vary by region. For more information, see Azure products by region.

```
az account list-locations \
    --query "[].{Region:name}" \
    --out table
```

After you have your Azure location, create a new resource group in the Azure CLI using the az group create command.

In the example below, replace the Azure location `westus2` with one of the Azure locations available for your subscription.

```
az group create \
    --name cognitive-services-resource-group \
    --location westus2
```

Create a Cognitive Services resource

Choose a cognitive service and pricing tier

When creating a new resource, you will need to know the "kind" of service you want to use, along with the pricing tier (or sku) you want. You will use this and other information as parameters when creating the resource.

Multi-service

SERVICE	KIND
Multiple services. See the pricing page for more details.	`CognitiveServices`

NOTE

Many of the Cognitive Services below have a free tier you can use to try the service. To use the free tier, use `F0` as the sku for your resource.

Vision

SERVICE	KIND
Computer Vision	`ComputerVision`
Custom Vision - Prediction	`CustomVision.Prediction`
Custom Vision - Training	`CustomVision.Training`
Face	`Face`

SERVICE	KIND
Form Recognizer	`FormRecognizer`
Ink Recognizer	`InkRecognizer`

Search

SERVICE	KIND
Bing Autosuggest	`Bing.Autosuggest.v7`
Bing Custom Search	`Bing.CustomSearch`
Bing Entity Search	`Bing.EntitySearch`
Bing Search	`Bing.Search.v7`
Bing Spell Check	`Bing.SpellCheck.v7`

Speech

SERVICE	KIND
Speech Services	`SpeechServices`
Speech Recognition	`SpeakerRecognition`

Language

SERVICE	KIND
Form Understanding	`FormUnderstanding`
LUIS	`LUIS`
QnA Maker	`QnAMaker`
Text Analytics	`TextAnalytics`
Text Translation	`TextTranslation`

Decision

SERVICE	KIND
Anomaly Detector	`AnomalyDetector`
Content Moderator	`ContentModerator`
Personalizer	`Personalizer`

You can find a list of available Cognitive Service "kinds" with the az cognitiveservices account list-kinds command:

```
az cognitiveservices account list-kinds
```

Add a new resource to your resource group

To create and subscribe to a new Cognitive Services resource, use the az cognitiveservices account create command. This command adds a new billable resource to the resource group created earlier. When creating your new resource, you will need to know the "kind" of service you want to use, along with its pricing tier (or sku) and an Azure location:

You can create an F0 (free) resource for Anomaly Detector, named `anomaly-detector-resource` with the command below.

```
az cognitiveservices account create \
    --name anomaly-detector-resource \
    --resource-group cognitive-services-resource-group \
    --kind AnomalyDetector \
    --sku F0 \
    --location westus2 \
    --yes
```

TIP

If your subscription doesn't allow you to create a Cognitive Service resource, you may need to enable that ability of the Azure resource provider with the Azure portal, PowerShell command or an Azure CLI command. If you are not the subscription owner, ask the *Subscription Owner* or someone with a role of *admin* to complete the registration for you or ask for the /register/action privileges granted to your account.

Get the keys for your resource

To log into your local installation of the Command-Line Interface(CLI), use the az login command.

```
az login
```

Use the az cognitiveservices account keys list command to get the keys for your Cognitive Service resource.

```
    az cognitiveservices account keys list \
    --name anomaly-detector-resource \
    --resource-group cognitive-services-resource-group
```

Configure an environment variable for authentication

Applications need to authenticate access to the Cognitive Services they use. To authenticate, we recommend creating an environment variable to store the keys for your Azure Resources.

After you have your key, write it to a new environment variable on the local machine running the application. To set the environment variable, open a console window, and follow the instructions for your operating system. Replace `your-key` with one of the keys for your resource.

- Windows
- Linux
- macOS

```
setx COGNITIVE_SERVICE_KEY "your-key"
```

After you add the environment variable, you may need to restart any running programs that will need to read the environment variable, including the console window. For example, if you are using Visual Studio as your editor, restart Visual Studio before running the example.

Pricing tiers and billing

Pricing tiers (and the amount you get billed) are based on the number of transactions you send using your authentication information. Each pricing tier specifies the:

- maximum number of allowed transactions per second (TPS).
- service features enabled within the pricing tier.
- The cost for a predefined amount of transactions. Going above this amount will cause an extra charge as specified in the pricing details for your service.

Get current quota usage for your resource

Use the az cognitiveservices account list-usage command to get the usage for your Cognitive Service resource.

```
az cognitiveservices account list-usage \
    --name anomaly-detector-resource \
    --resource-group cognitive-services-resource-group \
    --subscription subscription-name
```

Clean up resources

If you want to clean up and remove a Cognitive Services resource, you can delete it or the resource group. Deleting the resource group also deletes any other resources contained in the group.

To remove the resource group and its associated resources, use the az group delete command.

```
az group delete --name cognitive-services-resource-group
```

See also

- Authenticate requests to Azure Cognitive Services
- What is Azure Cognitive Services?
- Natural language support
- Docker container support

Quickstart: Create a Cognitive Services resource using the Azure Management client library

10/4/2020 • 24 minutes to read • Edit Online

Use this quickstart to create and manage Azure Cognitive Services resources using the Azure Management client library.

Azure Cognitive Services are cloud-base 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.

Individual AI services are represented by Azure resources that you create under your Azure subscription. After you create a resource, you can use the keys and endpoint generated to authenticate your applications.

Reference documentation | Library source code | Package (NuGet) | Samples

Prerequisites

- A valid Azure subscription - Create one for free.
- The current version of .NET Core.

Create an Azure Service Principal

To have your application interact with your Azure account, you need an Azure service principal to manage permissions. Follow the instructions in Create an Azure service principal.

When you create a service principal, you'll see it has a secret value, an ID, and an application ID. Save the application ID and secret to a temporary location for later steps.

Create a resource group

Before you create a Cognitive Services resource, your account must have an Azure resource group to contain the resource. If you don't already have a resource group, create one in the Azure portal before continuing.

Create a new C# application

Create a new .NET Core application. In a console window (such as cmd, PowerShell, or Bash), use the `dotnet new` command to create a new console app with the name `azure-management-quickstart`. This command creates a simple "Hello World" C# project with a single source file: *program.cs*.

```
dotnet new console -n azure-management-quickstart
```

Change your directory to the newly created app folder. You can build the application with:

```
dotnet build
```

The build output should contain no warnings or errors.

```
...
Build succeeded.
 0 Warning(s)
 0 Error(s)
...
```

Install the client library

Within the application directory, install the Azure Management client library for .NET with the following command:

```
dotnet add package Microsoft.Azure.Management.CognitiveServices
dotnet add package Microsoft.Azure.Management.Fluent
dotnet add package Microsoft.Azure.Management.ResourceManager.Fluent
```

If you're using the Visual Studio IDE, the client library is available as a downloadable NuGet package.

Import libraries

Open *program.cs* and add the following `using` statements to the top of the file:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Azure.Management.Fluent;
using Microsoft.Azure.Management.ResourceManager.Fluent;
using Microsoft.Azure.Management.ResourceManager.Fluent.Authentication;
using Microsoft.Azure.Management.CognitiveServices;
using Microsoft.Azure.Management.CognitiveServices.Models;
```

Authenticate the client

Add the following fields to the root of *program.cs* and populate their values, using the service principal you created and your Azure account information.

```
const string  service_principal_application_id = "TODO_REPLACE";
const string  service_principal_secret = "TODO_REPLACE";

/* The ID of your Azure subscription. You can find this in the Azure Dashboard under Home > Subscriptions. */
const string  subscription_id = "TODO_REPLACE";

/* The Active Directory tenant ID. You can find this in the Azure Dashboard under Home > Azure Active
Directory. */
const string  tenant_id = "TODO_REPLACE";

/* The name of the Azure resource group in which you want to create the resource.
You can find resource groups in the Azure Dashboard under Home > Resource groups. */
const string  resource_group_name = "TODO_REPLACE";
```

Then, in your **Main** method, use these values to construct a **CognitiveServicesManagementClient** object. This object is needed for all of your Azure management operations.

```
var service_principal_credentials = new ServicePrincipalLoginInformation ();
service_principal_credentials.ClientId = service_principal_application_id;
service_principal_credentials.ClientSecret = service_principal_secret;

var credentials = SdkContext.AzureCredentialsFactory.FromServicePrincipal(service_principal_application_id,
service_principal_secret, tenant_id, AzureEnvironment.AzureGlobalCloud);
var client = new CognitiveServicesManagementClient(credentials);
client.SubscriptionId = subscription_id;
```

Call management methods

Add the following code to your **Main** method to list available resources, create a sample resource, list your owned resources, and then delete the sample resource. You'll define these methods in the next steps.

```
    // List all available resource kinds, SKUs, and locations for your Azure account:
    list_available_kinds_skus_locations(client);

    // Create a resource with kind TextTranslation, F0 (free tier), location global.
    create_resource(client, "test_resource", "TextTranslation", "F0", "Global");

    // List all resources for your Azure account:
    list_resources(client);

    // Delete the resource.
    delete_resource(client, "test_resource");

    Console.WriteLine("Press any key to exit.");
    Console.ReadKey();
```

Create a Cognitive Services resource

Choose a service and pricing tier

When you create a new resource, you'll need to know the "kind" of service you want to use, along with the pricing tier (or SKU) you want. You'll use this and other information as parameters when creating the resource. You can find a list of available Cognitive Service "kinds" by calling the following method in your script:

```
static void list_available_kinds_skus_locations(CognitiveServicesManagementClient client)
{

    Console.WriteLine("Available SKUs:");
    var result = client.ResourceSkus.List();
    Console.WriteLine("Kind\tSKU Name\tSKU Tier\tLocations");
    foreach (var x in result) {
        var locations = "";
        foreach (var region in x.Locations)
        {
            locations += region;
        }
        Console.WriteLine(x.Kind + "\t" + x.Name + "\t" + x.Tier + "\t" + locations);
    };
}
```

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.

See the list of SKUs and pricing information below.

Multi-service

SERVICE	KIND
Multiple services. For more information, see the pricing page.	`CognitiveServices`

Vision

SERVICE	KIND
Computer Vision	`ComputerVision`
Custom Vision - Prediction	`CustomVision.Prediction`
Custom Vision - Training	`CustomVision.Training`
Face	`Face`
Form Recognizer	`FormRecognizer`
Ink Recognizer	`InkRecognizer`

Search

SERVICE	KIND
Bing Autosuggest	`Bing.Autosuggest.v7`
Bing Custom Search	`Bing.CustomSearch`
Bing Entity Search	`Bing.EntitySearch`
Bing Search	`Bing.Search.v7`
Bing Spell Check	`Bing.SpellCheck.v7`

Speech

SERVICE	KIND
Speech Services	`SpeechServices`
Speech Recognition	`SpeakerRecognition`

Language

SERVICE	KIND

SERVICE	KIND
Form Understanding	`FormUnderstanding`
LUIS	`LUIS`
QnA Maker	`QnAMaker`
Text Analytics	`TextAnalytics`
Text Translation	`TextTranslation`

Decision

SERVICE	KIND
Anomaly Detector	`AnomalyDetector`
Content Moderator	`ContentModerator`
Personalizer	`Personalizer`

Pricing tiers and billing

Pricing tiers (and the amount you get billed) are based on the number of transactions you send using your authentication information. Each pricing tier specifies the:

- maximum number of allowed transactions per second (TPS).
- service features enabled within the pricing tier.
- cost for a predefined number of transactions. Going above this number will cause an extra charge as specified in the pricing details for your service.

NOTE

Many of the Cognitive Services have a free tier you can use to try the service. To use the free tier, use `F0` as the SKU for your resource.

Create a Cognitive Services resource

To create and subscribe to a new Cognitive Services resource, use the **Create** method. This method adds a new billable resource to the resource group you pass in. When creating your new resource, you'll need to know the "kind" of service you want to use, along with its pricing tier (or SKU) and an Azure location. The following method takes all of these as arguments and creates a resource.

```
static void create_resource(CognitiveServicesManagementClient client, string resource_name, string kind,
string account_tier, string location)
{
    Console.WriteLine("Creating resource: " + resource_name + "...");
    // The parameter "properties" must be an empty object.
    CognitiveServicesAccount parameters =
        new CognitiveServicesAccount(null, null, kind, location, resource_name, new
CognitiveServicesAccountProperties(), new Sku(account_tier));
    var result = client.Accounts.Create(resource_group_name, account_tier, parameters);
    Console.WriteLine("Resource created.");
    Console.WriteLine("ID: " + result.Id);
    Console.WriteLine("Kind: " + result.Kind);
    Console.WriteLine();
}
```

View your resources

To view all of the resources under your Azure account (across all resource groups), use the following method:

```
static void list_resources(CognitiveServicesManagementClient client)
{
    Console.WriteLine("Resources in resource group: " + resource_group_name);
    var result = client.Accounts.ListByResourceGroup(resource_group_name);
    foreach (var x in result)
    {
        Console.WriteLine("ID: " + x.Id);
        Console.WriteLine("Name: " + x.Name);
        Console.WriteLine("Type: " + x.Type);
        Console.WriteLine("Kind: " + x.Kind);
        Console.WriteLine();
    }
}
```

Delete a resource

The following method deletes the specified resource from the given resource group.

```
static void delete_resource(CognitiveServicesManagementClient client, string resource_name)
{
    Console.WriteLine("Deleting resource: " + resource_name + "...");
    client.Accounts.Delete (resource_group_name, resource_name);

    Console.WriteLine("Resource deleted.");
    Console.WriteLine();
}
```

Run the application

Run the application from your application directory with the `dotnet run` command.

```
dotnet run
```

See also

- Azure Management SDK reference documentation
- What are Azure Cognitive Services?

- Authenticate requests to Azure Cognitive Services
- Create a new resource using the Azure portal

Reference documentation | Library source code | Package (Maven)

Prerequisites

- A valid Azure subscription - Create one for free.
- The current version of the Java Development Kit(JDK)
- The Gradle build tool, or another dependency manager.

Create an Azure Service Principal

To have your application interact with your Azure account, you need an Azure service principal to manage permissions. Follow the instructions in Create an Azure service principal.

When you create a service principal, you'll see it has a secret value, an ID, and an application ID. Save the application ID and secret to a temporary location for later steps.

Create a resource group

Before you create a Cognitive Services resource, your account must have an Azure resource group to contain the resource. If you don't already have a resource group, create one in the Azure portal before continuing.

Create a new Java application

In a console window (such as cmd, PowerShell, or Bash), create a new directory for your app, and navigate to it.

```
mkdir myapp && cd myapp
```

Run the `gradle init` command from your working directory. This command will create essential build files for Gradle, including *build.gradle.kts* which is used at runtime to create and configure your application.

```
gradle init --type basic
```

When prompted to choose a **DSL**, select **Kotlin**.

From your working directory, run the following command:

```
mkdir -p src/main/java
```

Install the client library

This quickstart uses the Gradle dependency manager. You can find the client library and information for other dependency managers on the Maven Central Repository.

In your project's *build.gradle.kts* file, include the client library as an `implementation` statement, along with the required plugins and settings.

```
plugins {
    java
    application
}
application {
    mainClass.set("FormRecognizer")
}
repositories {
    mavenCentral()
}
dependencies {
    implementation(group = "com.microsoft.azure", name = "azure-mgmt-cognitiveservices", version = "1.10.0-
beta")
}
```

Import libraries

Navigate to the new **src/main/java** folder and create a file called *Management.java*. Open it in your preferred editor or IDE and add the following `import` statements:

```
import com.microsoft.azure.*;
import com.microsoft.azure.arm.resources.Region;
import com.microsoft.azure.credentials.*;
import com.microsoft.azure.management.cognitiveservices.v2017_04_18.*;
import com.microsoft.azure.management.cognitiveservices.v2017_04_18.implementation.*;

import java.io.*;
import java.lang.Object.*;
import java.util.*;
import java.net.*;
```

Authenticate the client

Add a class in *Management.java*, and then add the following fields and their values inside of it. Populate their values, using the service principal you created and your other Azure account information.

```
/*
Be sure to use the service pricipal application ID, not simply the ID.
*/
private static String applicationId = "INSERT APPLICATION ID HERE";
private static String applicationSecret = "INSERT APPLICATION SECRET HERE";

/* The ID of your Azure subscription. You can find this in the Azure Dashboard under Home > Subscriptions. */
private static String subscriptionId = "INSERT SUBSCRIPTION ID HERE";

/* The Active Directory tenant ID. You can find this in the Azure Dashboard under Home > Azure Active
Directory. */
private static String tenantId = "INSERT TENANT ID HERE";

/* The name of the Azure resource group in which you want to create the resource.
You can find resource groups in the Azure Dashboard under Home > Resource groups. */
private static String resourceGroupName = "INSERT RESOURCE GROUP NAME HERE";
```

Then, in your **main** method, use these values to construct a **CognitiveServicesManager** object. This object is needed for all of your Azure management operations.

```
// auth
private static ApplicationTokenCredentials credentials = new ApplicationTokenCredentials(applicationId,
tenantId, applicationSecret, AzureEnvironment.AZURE);

CognitiveServicesManager client = CognitiveServicesManager.authenticate(credentials, subscriptionId);
```

Call management methods

Add the following code to your **Main** method to list available resources, create a sample resource, list your owned resources, and then delete the sample resource. You'll define these methods in the next steps.

```
    // list all available resource kinds, SKUs, and locations for your Azure account.
    list_available_kinds_skus_locations (client);

    // list all resources for your Azure account.
    list_resources (client);

    // Create a resource with kind Text Translation, SKU F0 (free tier), location global.
    String resourceId = create_resource (client, "test_resource", resourceGroupName, "TextAnalytics", "S0",
Region.US_WEST);

    // Delete the resource.
    delete_resource (client, resourceId);
}
```

Create a Cognitive Services resource

Choose a service and pricing tier

When you create a new resource, you'll need to know the "kind" of service you want to use, along with the pricing tier (or SKU) you want. You'll use this and other information as parameters when creating the resource. You can find a list of available Cognitive Service "kinds" by calling the following method:

```
public void list_available_kinds_skus_locations (CognitiveServicesManager client) {
    System.out.println ("Available SKUs:");
    System.out.println("Kind\tSKU Name\tSKU Tier\tLocations");
    ResourceSkus skus = client.resourceSkus();
    // See https://github.com/ReactiveX/RxJava/wiki/Blocking-Observable-Operators
    for (ResourceSku sku : skus.listAsync().toBlocking().toIterable()) {
        String locations = String.join (",", sku.locations());
        System.out.println (sku.kind() + "\t" + sku.name() + "\t" + sku.tier() + "\t" + locations);
    }
}
```

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.

See the list of SKUs and pricing information below.

Multi-service

SERVICE	KIND
Multiple services. For more information, see the pricing page.	`CognitiveServices`

Vision

SERVICE	KIND
Computer Vision	`ComputerVision`
Custom Vision - Prediction	`CustomVision.Prediction`
Custom Vision - Training	`CustomVision.Training`
Face	`Face`
Form Recognizer	`FormRecognizer`
Ink Recognizer	`InkRecognizer`

Search

SERVICE	KIND
Bing Autosuggest	`Bing.Autosuggest.v7`
Bing Custom Search	`Bing.CustomSearch`
Bing Entity Search	`Bing.EntitySearch`
Bing Search	`Bing.Search.v7`
Bing Spell Check	`Bing.SpellCheck.v7`

Speech

SERVICE	KIND
Speech Services	`SpeechServices`
Speech Recognition	`SpeakerRecognition`

Language

SERVICE	KIND
Form Understanding	`FormUnderstanding`
LUIS	`LUIS`
QnA Maker	`QnAMaker`
Text Analytics	`TextAnalytics`

SERVICE	KIND
Text Translation	`TextTranslation`

Decision

SERVICE	KIND
Anomaly Detector	`AnomalyDetector`
Content Moderator	`ContentModerator`
Personalizer	`Personalizer`

Pricing tiers and billing

Pricing tiers (and the amount you get billed) are based on the number of transactions you send using your authentication information. Each pricing tier specifies the:

- maximum number of allowed transactions per second (TPS).
- service features enabled within the pricing tier.
- cost for a predefined number of transactions. Going above this number will cause an extra charge as specified in the pricing details for your service.

NOTE

Many of the Cognitive Services have a free tier you can use to try the service. To use the free tier, use `F0` as the SKU for your resource.

Create a Cognitive Services resource

To create and subscribe to a new Cognitive Services resource, use the **create** method. This method adds a new billable resource to the resource group you pass in. When creating your new resource, you'll need to know the "kind" of service you want to use, along with its pricing tier (or SKU) and an Azure location. The following method takes all of these as arguments and creates a resource.

```
public String create_resource (CognitiveServicesManager client, String resourceName, String resourceGroupName,
String kind, String skuName, Region region) {
    System.out.println ("Creating resource: " + resourceName + "...");

    CognitiveServicesAccount result = client.accounts().define(resourceName)
        .withRegion(region)
        .withExistingResourceGroup(resourceGroupName)
        .withKind(kind)
        .withSku(new Sku().withName(skuName))
        .create();

    System.out.println ("Resource created.");
    System.out.println ("ID: " + result.id());
    System.out.println ("Provisioning state: " + result.properties().provisioningState().toString());
    System.out.println ();

    return result.id();
}
```

View your resources

To view all of the resources under your Azure account (across all resource groups), use the following method:

```
public void list_resources (CognitiveServicesManager client) {
    System.out.println ("Resources in resource group: " + resourceGroupName);
    // Note Azure resources are also sometimes referred to as accounts.
    Accounts accounts = client.accounts();
    for (CognitiveServicesAccount account :
accounts.listByResourceGroupAsync(resourceGroupName).toBlocking().toIterable()) {
        System.out.println ("Kind: " + account.kind ());
        System.out.println ("SKU Name: " + account.sku().name());
        System.out.println ();
    }
}
```

Delete a resource

The following method deletes the specified resource from the given resource group.

```
public void delete_resource (CognitiveServicesManager client, String resourceId) {
    System.out.println ("Deleting resource: " + resourceId + "...");
    client.accounts().deleteByIds (resourceId);
    System.out.println ("Resource deleted.");
    System.out.println ();
}
```

See also

- Azure Management SDK reference documentation
- What are Azure Cognitive Services?
- Authenticate requests to Azure Cognitive Services
- Create a new resource using the Azure portal

Reference documentation | Library source code | Package (NPM) | Samples

Prerequisites

- A valid Azure subscription - Create one for free.
- The current version of Node.js

Create an Azure Service Principal

To have your application interact with your Azure account, you need an Azure service principal to manage permissions. Follow the instructions in Create an Azure service principal.

When you create a service principal, you'll see it has a secret value, an ID, and an application ID. Save the application ID and secret to a temporary location for later steps.

Create a resource group

Before you create a Cognitive Services resource, your account must have an Azure resource group to contain the resource. If you don't already have a resource group, create one in the Azure portal before continuing.

Create a new Node.js application

In a console window (such as cmd, PowerShell, or Bash), create a new directory for your app, and navigate to it.

```
mkdir myapp && cd myapp
```

Run the `npm init` command to create a node application with a `package.json` file.

```
npm init
```

Create a file named *index.js* before going on.

Install the client library

Install the following NPM packages:

```
npm install @azure/arm-cognitiveservices
npm install @azure/ms-rest-js
npm install @azure/ms-rest-nodeauth
```

Your app's `package.json` file will be updated with the dependencies.

Import libraries

Open your *index.js* script and import the following libraries.

```
"use strict";
/* To run this sample, install the following modules.
 * npm install @azure/arm-cognitiveservices
 * npm install @azure/ms-rest-js
 * npm install @azure/ms-rest-nodeauth
 */
var Arm = require("@azure/arm-cognitiveservices");
var msRestNodeAuth = require("@azure/ms-rest-nodeauth");
```

Authenticate the client

Add the following fields to the root of your script and fill in their values, using the service principal you created and your Azure account information.

```
const service_principal_application_id = "TODO_REPLACE";
const service_principal_secret = "TODO_REPLACE";

/* The ID of your Azure subscription. You can find this in the Azure Dashboard under Home > Subscriptions. */
const subscription_id = "TODO_REPLACE";

/* The Active Directory tenant ID. You can find this in the Azure Dashboard under Home > Azure Active
Directory. */
const tenant_id = "TODO_REPLACE";

/* The name of the Azure resource group in which you want to create the resource.
You can find resource groups in the Azure Dashboard under Home > Resource groups. */
const resource_group_name = "TODO_REPLACE";
```

Next, add the following `quickstart` function to handle the main work of your program. The first block of code constructs a **CognitiveServicesManagementClient** object using the credential variables you entered above. This object is needed for all of your Azure management operations.

```
async function quickstart() {
    const credentials = await msRestNodeAuth.loginWithServicePrincipalSecret
(service_principal_application_id, service_principal_secret, tenant_id);
    const client = new Arm.CognitiveServicesManagementClient (credentials, subscription_id);
    // Note Azure resources are also sometimes referred to as accounts.
    const accounts_client = new Arm.Accounts (client);
    const resource_skus_client = new Arm.ResourceSkus (client);
```

Call management functions

Add the following code to the end of your `quickstart` function to list available resources, create a sample resource, list your owned resources, and then delete the sample resource. You'll define these functions in the next steps.

Create a Cognitive Services resource

Choose a service and pricing tier

When you create a new resource, you'll need to know the "kind" of service you want to use, along with the pricing tier (or SKU) you want. You'll use this and other information as parameters when creating the resource. The following function lists the available Cognitive Service "kinds."

```
async function list_available_kinds_skus_locations (client) {
    console.log ("Available SKUs:");
    var result = await client.list ();
    console.log("Kind\tSKU Name\tSKU Tier\tLocations");
    result.forEach (function (x) {
        var locations = x.locations.join(",");
        console.log(x.kind + "\t" + x.name + "\t" + x.tier + "\t" + locations);
    });
}
```

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.

See the list of SKUs and pricing information below.

Multi-service

SERVICE	KIND
Multiple services. For more information, see the pricing page.	`CognitiveServices`

Vision

SERVICE	KIND
Computer Vision	`ComputerVision`
Custom Vision - Prediction	`CustomVision.Prediction`

SERVICE	KIND
Custom Vision - Training	`CustomVision.Training`
Face	`Face`
Form Recognizer	`FormRecognizer`
Ink Recognizer	`InkRecognizer`

Search

SERVICE	KIND
Bing Autosuggest	`Bing.Autosuggest.v7`
Bing Custom Search	`Bing.CustomSearch`
Bing Entity Search	`Bing.EntitySearch`
Bing Search	`Bing.Search.v7`
Bing Spell Check	`Bing.SpellCheck.v7`

Speech

SERVICE	KIND
Speech Services	`SpeechServices`
Speech Recognition	`SpeakerRecognition`

Language

SERVICE	KIND
Form Understanding	`FormUnderstanding`
LUIS	`LUIS`
QnA Maker	`QnAMaker`
Text Analytics	`TextAnalytics`
Text Translation	`TextTranslation`

Decision

SERVICE	KIND
Anomaly Detector	`AnomalyDetector`
Content Moderator	`ContentModerator`

SERVICE	KIND
Personalizer	`Personalizer`

Pricing tiers and billing

Pricing tiers (and the amount you get billed) are based on the number of transactions you send using your authentication information. Each pricing tier specifies the:

- maximum number of allowed transactions per second (TPS).
- service features enabled within the pricing tier.
- cost for a predefined number of transactions. Going above this number will cause an extra charge as specified in the pricing details for your service.

NOTE

Many of the Cognitive Services have a free tier you can use to try the service. To use the free tier, use `F0` as the SKU for your resource.

Create a Cognitive Services resource

To create and subscribe to a new Cognitive Services resource, use the **Create** function. This function adds a new billable resource to the resource group you pass in. When you create your new resource, you'll need to know the "kind" of service you want to use, along with its pricing tier (or SKU) and an Azure location. The following function takes all of these arguments and creates a resource.

```
function create_resource (client, resource_name, kind, sku_name, location) {
    console.log ("Creating resource: " + resource_name + "...");
    // The parameter "properties" must be an empty object.
    var parameters = { sku : { name: sku_name }, kind : kind, location : location, properties : {} };

    return client.create(resource_group_name, resource_name, parameters)
        .then((result) => {
        console.log("Resource created.");
        print();
        console.log("ID: " + result.id);
        console.log("Kind: " + result.kind);
        console.log();
        })
        .catch((err) =>{
                console.log(err)
        })
}
```

View your resources

To view all of the resources under your Azure account (across all resource groups), use the following function:

```
async function list_resources (client) {
    console.log ("Resources in resource group: " + resource_group_name);
    var result = await client.listByResourceGroup (resource_group_name);
    result.forEach (function (x) {
        console.log(x);
        console.log();
    });
}
```

Delete a resource

The following function deletes the specified resource from the given resource group.

```
async function delete_resource (client, resource_name) {
    console.log ("Deleting resource: " + resource_name + "...");
    await client.deleteMethod (resource_group_name, resource_name)
    console.log ("Resource deleted.");
    console.log ();
}
```

Run the application

Add the following code to the bottom of your script to call your main `quickstart` function with error handling.

```
try {
    quickstart();
}
catch (error) {
    console.log(error);
}
```

Then, in your console window, run the application with the `node` command.

```
node index.js
```

See also

- Azure Management SDK reference documentation
- What are Azure Cognitive Services?
- Authenticate requests to Azure Cognitive Services
- Create a new resource using the Azure portal

Reference documentation | Library source code | Package (PyPi) | Samples

Prerequisites

- A valid Azure subscription - Create one for free.
- Python 3.x

Create an Azure Service Principal

To have your application interact with your Azure account, you need an Azure service principal to manage permissions. Follow the instructions in Create an Azure service principal.

When you create a service principal, you'll see it has a secret value, an ID, and an application ID. Save the application ID and secret to a temporary location for later steps.

Create a resource group

Before you create a Cognitive Services resource, your account must have an Azure resource group to contain the resource. If you don't already have a resource group, create one in the Azure portal before continuing.

Create a new Python application

Create a new Python application in your preferred editor or IDE and navigate to your project in a console window.

Install the client library

You can install the client library with:

```
pip install azure-mgmt-cognitiveservices
```

If you're using the Visual Studio IDE, the client library is available as a downloadable NuGet package.

Import libraries

Open your Python script and import the following libraries.

```
from msrestazure.azure_active_directory import ServicePrincipalCredentials
from azure.mgmt.cognitiveservices import CognitiveServicesManagementClient
from azure.mgmt.cognitiveservices.models import CognitiveServicesAccount, Sku
```

Authenticate the client

Add the following fields to the root of your script and fill in their values, using the service principal you created and your Azure account information.

```
# Be sure to use the service pricipal application ID, not simply the ID.
service_principal_application_id = "MY-SERVICE-PRINCIPAL-APPLICATION-ID"
service_principal_secret = "MY-SERVICE-PRINCIPAL-SECRET"

# The ID of your Azure subscription. You can find this in the Azure Dashboard under Home > Subscriptions.
subscription_id = "MY-SUBSCRIPTION-ID"

# The Active Directory tenant ID. You can find this in the Azure Dashboard under Home > Azure Active
Directory.
tenant_id = "MY-TENANT-ID"

# The name of the Azure resource group in which you want to create the resource.
# You can find resource groups in the Azure Dashboard under Home > Resource groups.
resource_group_name = "MY-RESOURCE-GROUP"
```

Then add the following code to construct a `CognitiveServicesManagementClient` object. This object is needed for all of your Azure management operations.

```
credentials = ServicePrincipalCredentials(service_principal_application_id, service_principal_secret,
tenant=tenant_id)
client = CognitiveServicesManagementClient(credentials, subscription_id)
```

Create a Cognitive Services resource

Choose a service and pricing tier

When you create a new resource, you'll need to know the "kind" of service you want to use, along with the pricing tier (or SKU) you want. You'll use this and other information as parameters when creating the resource. The following function lists the available Cognitive Service "kinds."

```
def list_available_kinds_skus_locations():
    print("Available SKUs:")
    result = client.resource_skus.list()
    print("Kind\tSKU Name\tSKU Tier\tLocations")
    for x in result:
        locations = ",".join(x.locations)
        print(x.kind + "\t" + x.name + "\t" + x.tier + "\t" + locations)
```

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.

See the list of SKUs and pricing information below.

Multi-service

SERVICE	KIND
Multiple services. For more information, see the pricing page.	`CognitiveServices`

Vision

SERVICE	KIND
Computer Vision	`ComputerVision`
Custom Vision - Prediction	`CustomVision.Prediction`
Custom Vision - Training	`CustomVision.Training`
Face	`Face`
Form Recognizer	`FormRecognizer`
Ink Recognizer	`InkRecognizer`

Search

SERVICE	KIND
Bing Autosuggest	`Bing.Autosuggest.v7`
Bing Custom Search	`Bing.CustomSearch`
Bing Entity Search	`Bing.EntitySearch`
Bing Search	`Bing.Search.v7`

SERVICE	KIND
Bing Spell Check	`Bing.SpellCheck.v7`

Speech

SERVICE	KIND
Speech Services	`SpeechServices`
Speech Recognition	`SpeakerRecognition`

Language

SERVICE	KIND
Form Understanding	`FormUnderstanding`
LUIS	`LUIS`
QnA Maker	`QnAMaker`
Text Analytics	`TextAnalytics`
Text Translation	`TextTranslation`

Decision

SERVICE	KIND
Anomaly Detector	`AnomalyDetector`
Content Moderator	`ContentModerator`
Personalizer	`Personalizer`

Pricing tiers and billing

Pricing tiers (and the amount you get billed) are based on the number of transactions you send using your authentication information. Each pricing tier specifies the:

- maximum number of allowed transactions per second (TPS).
- service features enabled within the pricing tier.
- cost for a predefined number of transactions. Going above this number will cause an extra charge as specified in the pricing details for your service.

NOTE

Many of the Cognitive Services have a free tier you can use to try the service. To use the free tier, use `F0` as the SKU for your resource.

Create a Cognitive Services resource

To create and subscribe to a new Cognitive Services resource, use the **Create** function. This function adds a new billable resource to the resource group you pass in. When you create your new resource, you'll need to know the "kind" of service you want to use, along with its pricing tier (or SKU) and an Azure location. The following function takes all of these arguments and creates a resource.

```
def create_resource (resource_name, kind, sku_name, location):
    print("Creating resource: " + resource_name + "...")
# The parameter "properties" must be an empty object.
    parameters = CognitiveServicesAccount(sku=Sku(name=sku_name), kind=kind, location=location, properties={})
    result = client.accounts.create(resource_group_name, resource_name, parameters)
    print("Resource created.")
    print()
    print("ID: " + result.id)
    print("Name: " + result.name)
    print("Type: " + result.type)
    print()
```

View your resources

To view all of the resources under your Azure account (across all resource groups), use the following function:

```
def list_resources():
    print("Resources in resource group: " + resource_group_name)
    result = client.accounts.list_by_resource_group(resource_group_name)
    for x in result:
        print(x)
        print()
```

Delete a resource

The following function deletes the specified resource from the given resource group.

```
def delete_resource(resource_name) :
    print("Deleting resource: " + resource_name + "...")
    client.accounts.delete(resource_group_name, resource_name)
    print("Resource deleted.")
```

Call management functions

Add the following code to the bottom of your script to call the above functions. This code lists available resources, creates a sample resource, lists your owned resources, and then deletes the sample resource.

```
# Uncomment this to list all available resource kinds, SKUs, and locations for your Azure account.
list_available_kinds_skus_locations()

# Create a resource with kind Text Translation, SKU F0 (free tier), location global.
create_resource("test_resource", "TextTranslation", "F0", "Global")

# Uncomment this to list all resources for your Azure account.
list_resources()

# Delete the resource.
delete_resource("test_resource")
```

Run the application

Run your application from the command line with the `python` command.

```
python <your-script-name>.py
```

See also

- Azure Management SDK reference documentation
- What are Azure Cognitive Services?
- Authenticate requests to Azure Cognitive Services
- Create a new resource using the Azure portal

Quickstart: Create a Cognitive Services resource using an ARM template

10/4/2020 • 4 minutes to read • Edit Online

This quickstart describes how to use an Azure Resource Manager template (ARM template) to create Cognitive Services.

Azure Cognitive Services are cloud-base 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.

Create a resource using an Azure Resource Manager template (ARM template). This multi-service resource lets you:

- Access multiple Azure Cognitive Services with a single key and endpoint.
- Consolidate billing from the services you use.

An ARM template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax, which lets you state what you intend to deploy without having to write the sequence of programming commands to create it.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Deploy to Azure

Prerequisites

- If you don't have an Azure subscription, create one for free.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "cognitiveServiceName": {
      "type": "string",
      "defaultValue": "[concat('CognitiveService-', uniqueString(resourceGroup().id))]",
      "metadata": {
        "description": "That name is the name of our application. It has to be unique.Type a name followed by
your resource group name. (<name>-<resourceGroupName>)"
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    },
    "sku": {
      "type": "string",
      "defaultValue": "S0",
      "allowedValues": [
        "S0"
      ]
    }
  },
  "resources": [
    {
      "type": "Microsoft.CognitiveServices/accounts",
      "apiVersion": "2017-04-18",
      "name": "[parameters('cognitiveServiceName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('sku')]"
      },
      "kind": "CognitiveServices",
      "properties": {
        "statisticsEnabled": false
      }
    }
  ]
}
```

One Azure resource is defined in the template:

- Microsoft.CognitiveServices/accounts: creates a Cognitive Services resource.

Deploy the template

- Azure portal
- Azure CLI

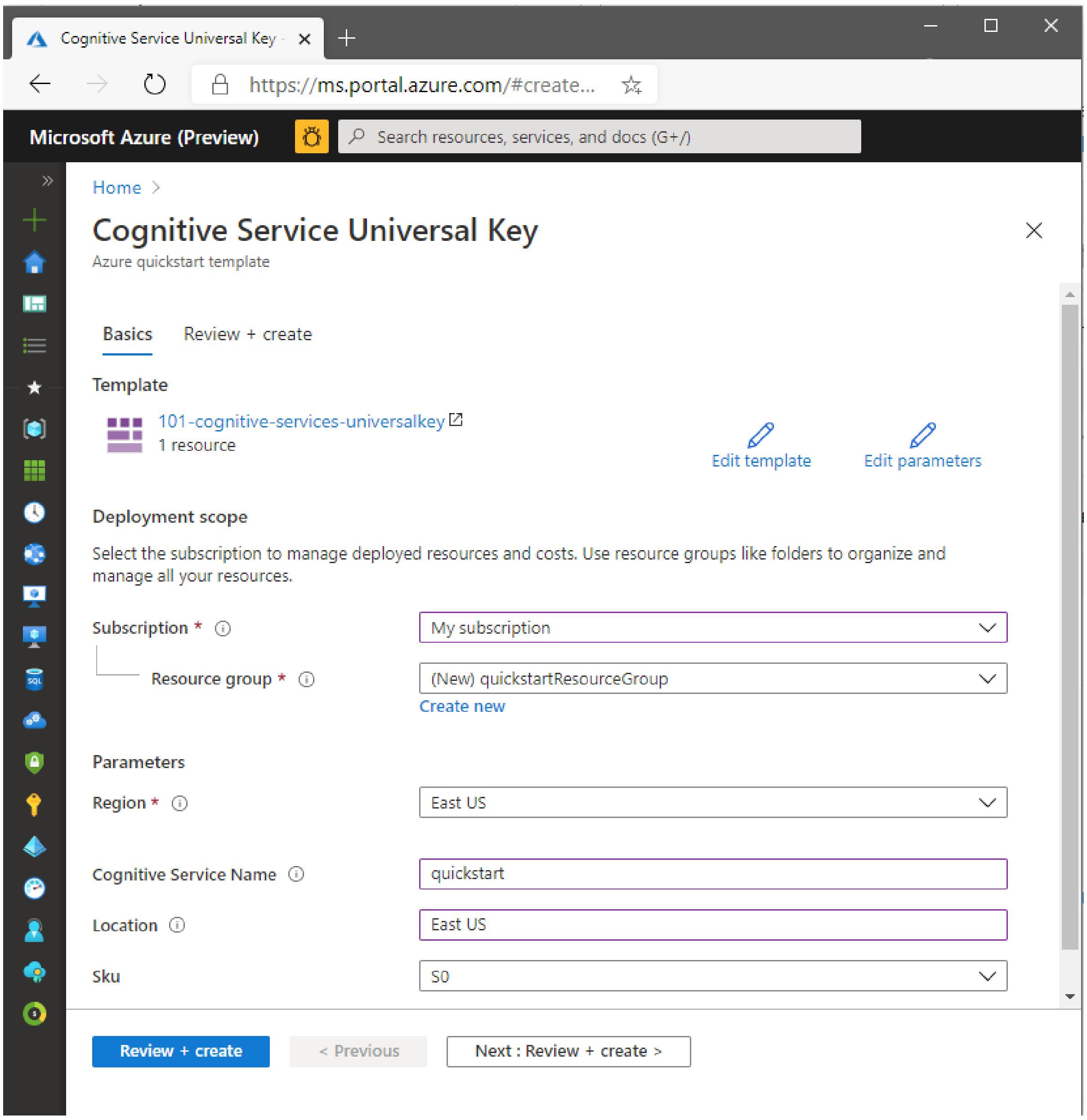
1. Click the **Deploy to Azure** button.

   Deploy to Azure

2. Enter the following values.

VALUE	DESCRIPTION
Subscription	Select an Azure subscription.

VALUE	DESCRIPTION
Resource group	Select **Create new**, enter a unique name for the resource group, and then click **OK**.
Region	Select a region. For example, **East US**
Cognitive Service Name	Replace with a unique name for your resource. You will need the name in the next section when you validate the deployment.
Location	Replace with the region used above.
Sku	The pricing tier for your resource.


Cognitive Service Universal Key
https://ms.portal.azure.com/#create...
Microsoft Azure (Preview)
Search resources, services, and docs (G+/)
Home

Cognitive Service Universal Key

Azure quickstart template

Template

101-cognitive-services-universalkey

1 resource

Edit template
Edit parameters

Deployment scope

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	My subscription
Resource group *	(New) quickstartResourceGroup

Parameters

Region *	East US
Cognitive Service Name	quickstart
Location	East US
Sku	S0

3. Select **Review + Create**, then **Create**. After the resource has successfully finished deploying, the **Go to resource** button will be highlighted.

TIP

If your subscription doesn't allow you to create a Cognitive Service resource, you may need to enable that ability of the Azure resource provider with the Azure portal, PowerShell command or an Azure CLI command. If you are not the subscription owner, ask the *Subscription Owner* or someone with a role of *admin* to complete the registration for you or ask for the /register/action privileges granted to your account.

Review deployed resources

- Portal
- Azure CLI

When your deployment finishes, you will be able to click the **Go to resource** button to see your new resource. You can also find the resource group by:

1. Selecting **Resource groups** from the left navigation menu.
2. Selecting the resource group name.

Clean up resources

If you want to clean up and remove a Cognitive Services subscription, you can delete the resource or resource group. Deleting the resource group also deletes any other resources contained in the group.

- Azure portal
- Azure CLI

1. In the Azure portal, expand the menu on the left side to open the menu of services, and choose **Resource Groups** to display the list of your resource groups.
2. Locate the resource group containing the resource to be deleted
3. Right-click on the resource group listing. Select **Delete resource group**, and confirm.

Next steps

- Authenticate requests to Azure Cognitive Services
- What is Azure Cognitive Services?
- Natural language support
- Docker container support

Enable diagnostic logging for Azure Cognitive Services

10/2/2019 • 4 minutes to read • Edit Online

This guide provides step-by-step instructions to enable diagnostic logging for an Azure Cognitive Service. These logs provide rich, frequent data about the operation of a resource that are used for issue identification and debugging. Before you continue, you must have an Azure account with a subscription to at least one Cognitive Service, such as Bing Web Search, Speech Services, or LUIS.

Prerequisites

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 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.

NOTE

Additional configuration options are available. To learn more, see Collect and consume log data from your Azure resources.

Enable diagnostic log collection

Let's start by enabling diagnostic logging using the Azure portal.

NOTE

To enable this feature using PowerShell or the Azure CLI, use the instructions provided in Collect and consume log data from your Azure resources.

1. Navigate to the Azure portal. Then locate and select a Cognitive Services resource. For example, your subscription to Bing Web Search.
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 you 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**.

It can take up to two hours before logging data is available to query and analyze. So don't worry if you don't see anything right away.

View and export diagnostic data from Azure Storage

Azure Storage is a robust object storage solution that is optimized for storing large amounts of unstructured data. In this section, you'll learn to query your storage account for total transactions over a 30-day timeframe and export the data to excel.

1. From the Azure portal, locate the Azure Storage resource that you created in the last section.
2. From the left-hand navigation menu, locate **Monitoring** and select **Metrics**.
3. Use the available drop-downs to configure your query. For this example, let's set the time range to **Last 30 days** and the metric to **Transaction**.
4. When the query is complete, you'll see a visualization of transaction over the last 30 days. To export this data, use the **Export to Excel** button located at the top of the page.

Learn more about what you can do with diagnostic data in Azure Storage.

View logs in Log Analytics

Follow these instructions to explore log analytics data for your resource.

1. From the Azure portal, locate and select **Log Analytics** from the left-hand navigation menu.
2. Locate and select the resource you created when enabling diagnostics.
3. Under **General**, locate and select **Logs**. From this page, you can run queries against your logs.

Sample queries

Here are a few basic Kusto queries you can use to explore your log data.

Run this query for all diagnostic logs from Azure Cognitive Services for a specified time period:

```
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.COGNITIVESERVICES"
```

Run this query to see the 10 most recent logs:

```
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.COGNITIVESERVICES"
| take 10
```

Run this query to group operations by **Resource**:

```
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.COGNITIVESERVICES" |
summarize count() by Resource
```

Run this query to find the average time it takes to perform an operation:

```
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.COGNITIVESERVICES"
| summarize avg(DurationMs)
by OperationName
```

Run this query to view the volume of operations over time split by OperationName with counts binned for every 10s.

```
AzureDiagnostics
| where ResourceProvider == "MICROSOFT.COGNITIVESERVICES"
| summarize count()
by bin(TimeGenerated, 10s), OperationName
| render areachart kind=unstacked
```

Next steps

- To understand how to enable logging, and also the metrics and log categories that are supported by the various Azure services, read both the Overview of metrics in Microsoft Azure and Overview of Azure Diagnostic Logs articles.
- Read these articles to learn about event hubs:
  - What is Azure Event Hubs?
  - Get started with Event Hubs
- Read Download metrics and diagnostic logs from Azure Storage.
- Read Understand log searches in Azure Monitor logs.

Azure Cognitive Services security

10/4/2020 • 5 minutes to read • Edit Online

Security should be considered a top priority when developing any and all applications. With the onset of artificial intelligence enabled applications, security is even more important. In this article various aspects of Azure Cognitive Services security are outlined, such as the use of transport layer security, authentication, securely configuring sensitive data, and Customer Lockbox for customer data access.

Transport Layer Security (TLS)

All of the Cognitive Services endpoints exposed over HTTP enforce TLS 1.2. With an enforced security protocol, consumers attempting to call a Cognitive Services endpoint should adhere to these guidelines:

- The client Operating System (OS) needs to support TLS 1.2
- The language (and platform) used to make the HTTP call need to specify TLS 1.2 as part of the request
  - Depending on the language and platform, specifying TLS is done either implicitly or explicitly

For .NET users, consider the Transport Layer Security best practices .

Authentication

When discussing authentication, there are several common misconceptions. Authentication and authorization are often confused for one another. Identity is also a major component in security. An identity is a collection of information about a principal . Identity providers (IdP) provide identities to authentication services. Authentication is the act of verifying a user's identity. Authorization is the specification of access rights and privileges to resources for a given identity. Several of the Cognitive Services offerings, include role-based access control (RBAC). RBAC could be used to simplify some of the ceremony involved with manually managing principals. For more details, see role-based access control for Azure resources.

For more information on authentication with subscription keys, access tokens and Azure Active Directory (AAD), see authenticate requests to Azure Cognitive Services.

Environment variables and application configuration

Environment variables are name-value pairs, stored within a specific environment. A more secure alternative to using hardcoded values for sensitive data, is to use environment variables. Hardcoded values are insecure and should be avoided.

Caution

Do **not** use hardcoded values for sensitive data, doing so is a major security vulnerability.

> **NOTE**
>
> While environment variables are stored in plain text, they are isolated to an environment. If an environment is compromised, so too are the variables with the environment.

Set environment variable

To set environment variables, use one the following commands - where the `ENVIRONMENT_VARIABLE_KEY` is the named key and `value` is the value stored in the environment variable.

- Command Line

- PowerShell
- Bash

Create and assign persisted environment variable, given the value.

```
:: Assigns the env var to the value
setx ENVIRONMENT_VARIABLE_KEY="value"
```

In a new instance of the **Command Prompt**, read the environment variable.

```
:: Prints the env var value
echo %ENVIRONMENT_VARIABLE_KEY%
```

TIP

After setting an environment variable, restart your integrated development environment (IDE) to ensure that newly added environment variables are available.

Get environment variable

To get an environment variable, it must be read into memory. Depending on the language you're using, consider the following code snippets. These code snippets demonstrate how to get environment variable given the `ENVIRONMENT_VARIABLE_KEY` and assign to a variable named `value`.

- C#
- C++
- Java
- Node.js
- Python
- Objective-C

For more information, see `Environment.GetEnvironmentVariable`.

```
using static System.Environment;

class Program
{
    static void Main()
    {
        // Get the named env var, and assign it to the value variable
        var value =
            GetEnvironmentVariable(
                "ENVIRONMENT_VARIABLE_KEY");
    }
}
```

Customer Lockbox

Customer Lockbox for Microsoft Azure provides an interface for customers to review, and approve or reject customer data access requests. It is used in cases where a Microsoft engineer needs to access customer data during a support request. For information on how Customer Lockbox requests are initiated, tracked, and stored for later reviews and audits, see Customer Lockbox.

Customer Lockbox is available for this Cognitive Service:

- Translator

For the following services, Microsoft engineers will not access any customer data in the E0 tier:

- Language Understanding
- Face
- Content Moderator
- Personalizer

> **IMPORTANT**
>
> For **Form Recognizer**, Microsoft engineers will not access any customer data in resources created after July 10, 2020.

To request the ability to use the E0 SKU, fill out and submit thisrequest Form. It will take approximately 3-5 business days to hear back on the status of your request. Depending on demand, you may be placed in a queue and approved as space becomes available. Once approved for using the E0 SKU with LUIS, you'll need to create a new resource from the Azure portal and select E0 as the Pricing Tier. Users won't be able to upgrade from the F0 to the new E0 SKU.

The Speech service doesn't currently support Customer Lockbox. However, customer data can be stored using bring your own storage (BYOS), allowing you to achieve similar data controls to Customer Lockbox. Keep in mind that Speech service data stays and is processed in the region where the Speech resource was created. This applies to any data at rest and data in transit. When using customization features, like Custom Speech and Custom Voice, all customer data is transferred, stored, and processed in the same region where your BYOS (if used) and Speech service resource reside.

> **IMPORTANT**
>
> Microsoft **does not** use customer data to improve its Speech models. Additionally, if endpoint logging is disabled and no customizations are used, then no customer data is stored.

Next steps

- Explore the various Cognitive Services
- Learn more about Cognitive Services Virtual Networks

Configure Azure Cognitive Services virtual networks

10/4/2020 • 16 minutes to read • Edit Online

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.

IMPORTANT

Turning on firewall rules for your Cognitive Services account blocks incoming requests for data by default. In order to allow requests through, one of the following conditions needs to be met:

- The request should originate from a service operating within an Azure Virtual Network (VNet) on the allowed subnet list of the target Cognitive Services account. The endpoint in requests originated from VNet needs to be set as the custom subdomain of your Cognitive Services account.
- Or the request should originate from an allowed list of IP addresses.

Requests that are blocked include those from other Azure services, from the Azure portal, from logging and metrics services, and so on.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Scenarios

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.

Supported regions and service offerings

Virtual networks (VNETs) are supported in regions where Cognitive Services are available. If the Cognitive Service isn't listed, it doesn't currently support virtual networks.

- Anomaly Detector
- Computer Vision
- Content Moderator
- Custom Vision
- Face
- Form Recognizer
- Language Understanding
- Personalizer
- Text Analytics
- QnA Maker
- Translator Text
- Immersive Reader

Service Tags

Cognitive Services supports service tags for network rules configuration. The services listed below are included in the **CognitiveServicesManagement** service tag.

- Anomaly Detector
- Computer Vision
- Content Moderator
- Custom Vision
- Face
- Form Recognizer
- Language Understanding (LUIS)
- Personalizer
- Text Analytics
- QnA Maker
- Translator
- Speech Service
- Immersive Reader

Change the default network access rule

By default, Cognitive Services resources accept connections from clients on any network. To limit access to selected networks, you must first change the default action.

WARNING

Making changes to network rules can impact your applications' ability to connect to Azure Cognitive Services. Setting the default network rule to **deny** blocks all access to the data unless specific network rules that **grant** access are also applied. Be sure to grant access to any allowed networks using network rules before you change the default rule to deny access. If you are allow listing IP addresses for your on-premises network, be sure to add all possible outgoing public IP addresses from your on-premises network.

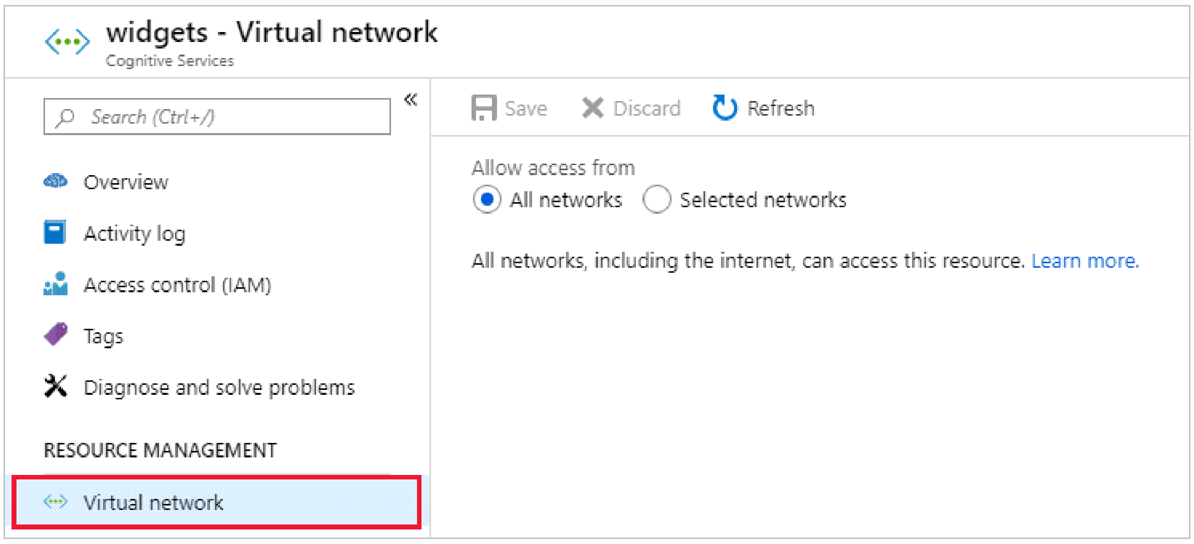
Managing default network access rules

You can manage default network access rules for Cognitive Services resources through the Azure portal, PowerShell, or the Azure CLI.

- Azure portal

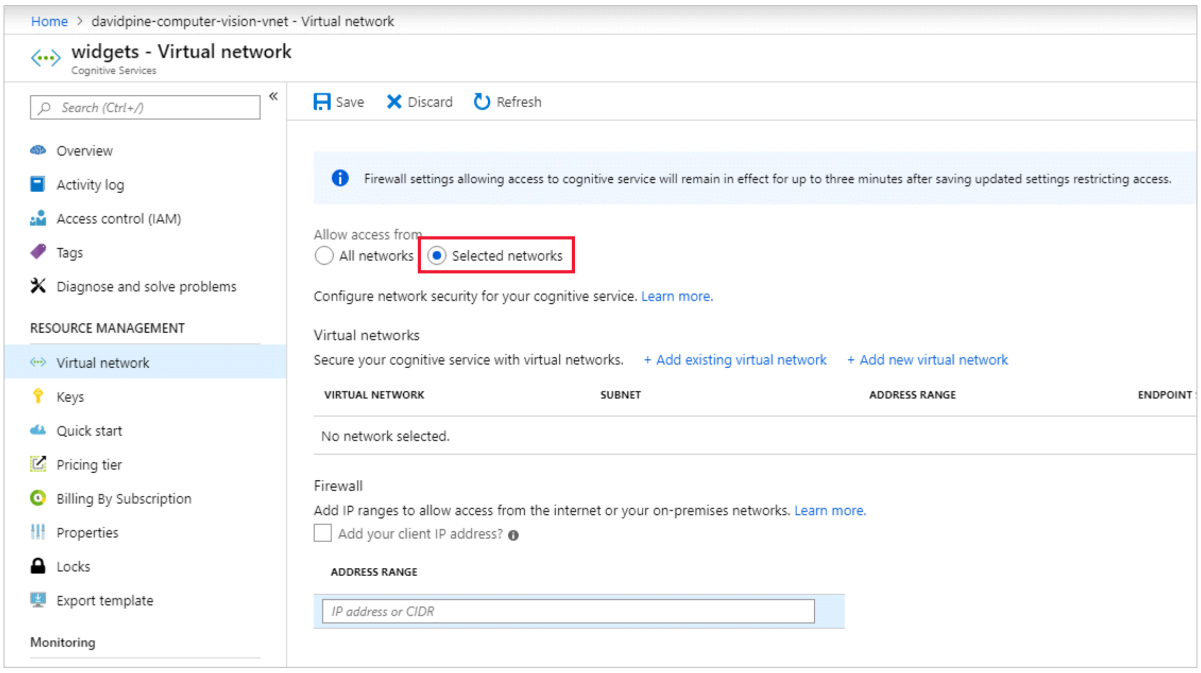
- PowerShell
- Azure CLI

1. Go to the Cognitive Services resource you want to secure.
2. Select the **RESOURCE MANAGEMENT** menu called **Virtual network**.



All networks, including the internet, can access this resource.

3. To deny access by default, choose to allow access from **Selected networks**. With the **Selected networks** setting alone, unaccompanied by configured **Virtual networks** or **Address ranges** - all access is effectively denied. When all access is denied, requests attempting to consume the Cognitive Services resource aren't permitted. The Azure portal, Azure PowerShell or, Azure CLI can still be used to configure the Cognitive Services resource.
4. To allow traffic from all networks, choose to allow access from **All networks**.



Firewall settings allowing access to cognitive service will remain in effect for up to three minutes after saving updated settings restricting access.

Configure network security for your cognitive service.

Secure your cognitive service with virtual networks.

No network selected.

Add IP ranges to allow access from the internet or your on-premises networks.

5. Select **Save** to apply your changes.

Grant access from a virtual network

You can configure Cognitive Services resources to allow access only from specific subnets. The allowed subnets may belong to a VNet in the same subscription, or in a different subscription, including subscriptions belonging to a different Azure Active Directory tenant.

Enable a service endpoint for Azure Cognitive Services within the VNet. The service endpoint routes traffic from the VNet through an optimal path to the Azure Cognitive Services service. The identities of the subnet and the virtual network are also transmitted with each request. Administrators can then configure network rules for the Cognitive Services resource that allow requests to be received from specific subnets in a VNet. Clients granted access via these network rules must continue to meet the authorization requirements of the Cognitive Services resource to access the data.

Each Cognitive Services resource supports up to 100 virtual network rules, which may be combined with IP network rules.

Required permissions

To apply a virtual network rule to a Cognitive Services resource, the user must have the appropriate permissions for the subnets being added. The required permission is the default *Contributor* role, or the *Cognitive Services Contributor* role. Required permissions can also be added to custom role definitions.

Cognitive Services resource and the virtual networks granted access may be in different subscriptions, including subscriptions that are a part of a different Azure AD tenant.

NOTE

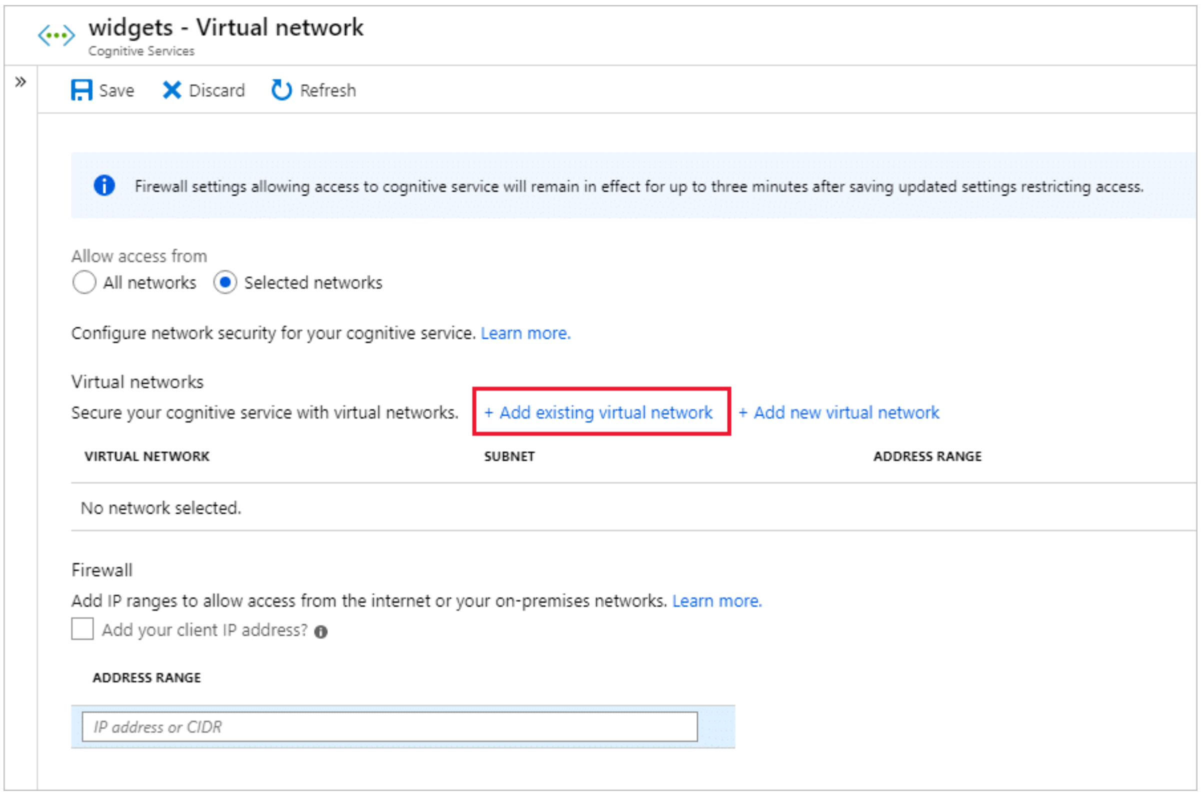
Configuration of rules that grant access to subnets in virtual networks that are a part of a different Azure Active Directory tenant are currently only supported through Powershell, CLI and REST APIs. Such rules cannot be configured through the Azure portal, though they may be viewed in the portal.

Managing virtual network rules

You can manage virtual network rules for Cognitive Services resources through the Azure portal, PowerShell, or the Azure CLI.

- Azure portal
- PowerShell
- Azure CLI

1. Go to the Cognitive Services resource you want to secure.
2. Select the **RESOURCE MANAGEMENT** menu called **Virtual network**.
3. Check that you've selected to allow access from **Selected networks**.
4. To grant access to a virtual network with an existing network rule, under **Virtual networks**, select **Add existing virtual network**.



widgets - Virtual network
Cognitive Services

Firewall settings allowing access to cognitive service will remain in effect for up to three minutes after saving updated settings restricting access.

Allow access from
All networks [✓ Selected networks]

Configure network security for your cognitive service. Learn more.

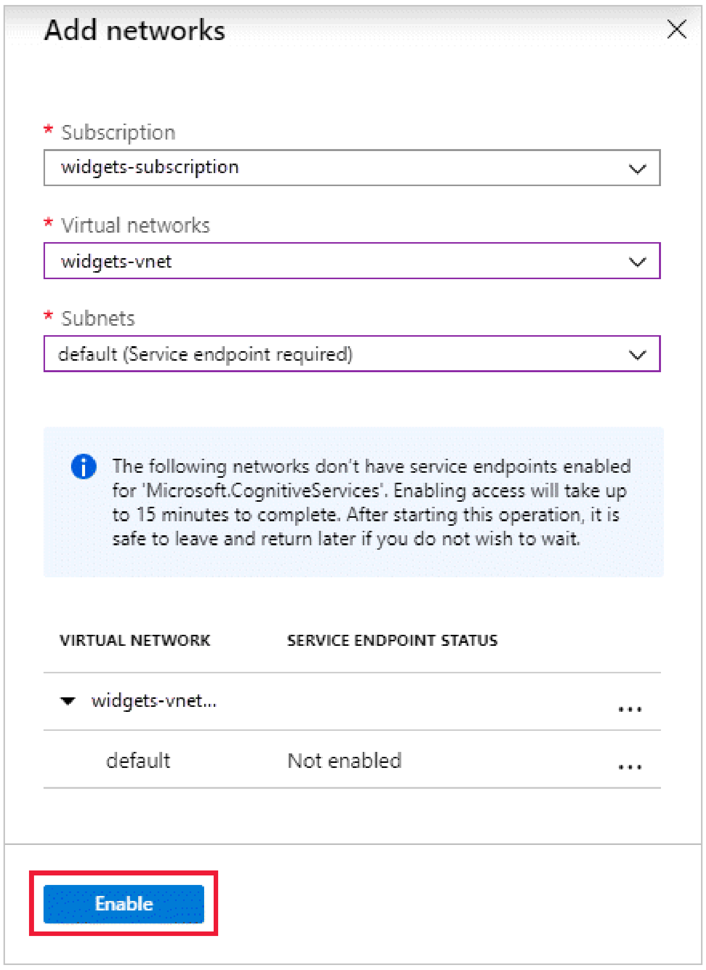
Virtual networks
Secure your cognitive service with virtual networks. + Add existing virtual network + Add new virtual network

VIRTUAL NETWORK	SUBNET	ADDRESS RANGE
No network selected.		

Firewall
Add IP ranges to allow access from the internet or your on-premises networks. Learn more.
Add your client IP address?

ADDRESS RANGE
IP address or CIDR

5. Select the **Virtual networks** and **Subnets** options, and then select **Enable**.



Add networks

* Subscription
widgets-subscription

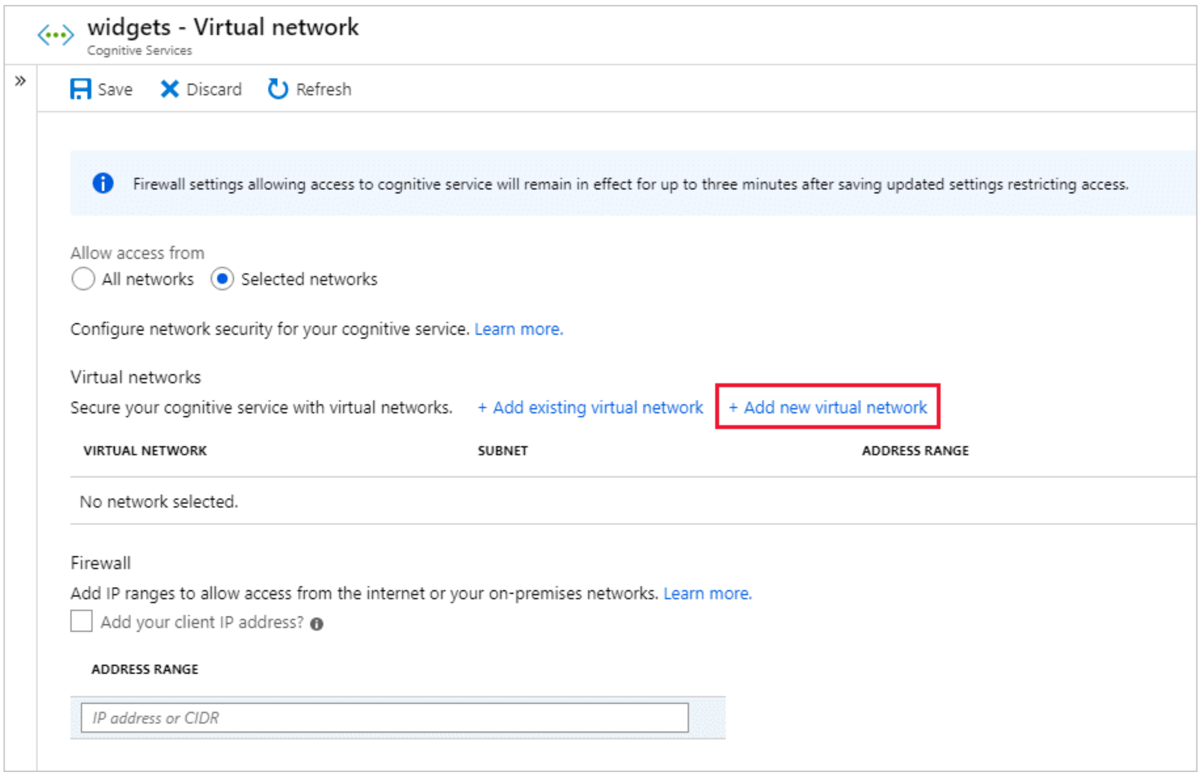
* Virtual networks
widgets-vnet

* Subnets
default (Service endpoint required)

The following networks don't have service endpoints enabled for 'Microsoft.CognitiveServices'. Enabling access will take up to 15 minutes to complete. After starting this operation, it is safe to leave and return later if you do not wish to wait.

VIRTUAL NETWORK	SERVICE ENDPOINT STATUS
widgets-vnet...	
default	Not enabled

6. To create a new virtual network and grant it access, select **Add new virtual network**.



widgets - Virtual network
Cognitive Services

Save Discard Refresh

Firewall settings allowing access to cognitive service will remain in effect for up to three minutes after saving updated settings restricting access.

Allow access from
All networks [✓ Selected networks]

Configure network security for your cognitive service. Learn more.

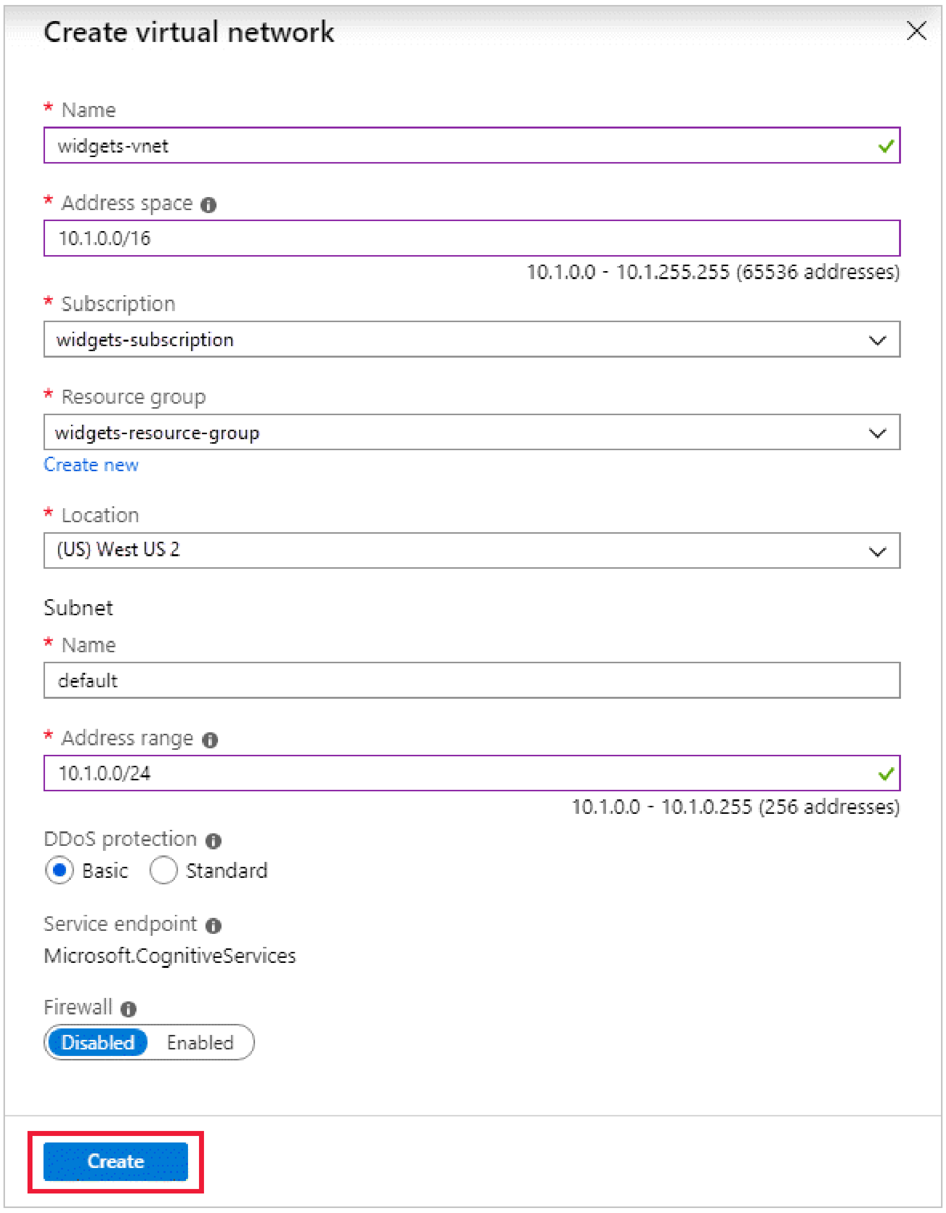
Virtual networks
Secure your cognitive service with virtual networks. + Add existing virtual network + Add new virtual network

VIRTUAL NETWORK	SUBNET	ADDRESS RANGE
No network selected.		

Firewall
Add IP ranges to allow access from the internet or your on-premises networks. Learn more.
☐ Add your client IP address?

ADDRESS RANGE
IP address or CIDR

7. Provide the information necessary to create the new virtual network, and then select **Create**.



Create virtual network

* Name
widgets-vnet

* Address space
10.1.0.0/16
10.1.0.0 - 10.1.255.255 (65536 addresses)

* Subscription
widgets-subscription

* Resource group
widgets-resource-group
Create new

* Location
(US) West US 2

Subnet

* Name
default

* Address range
10.1.0.0/24
10.1.0.0 - 10.1.0.255 (256 addresses)

DDoS protection
[✓ Basic] Standard

Service endpoint
Microsoft.CognitiveServices

Firewall
[✓ Disabled] Enabled

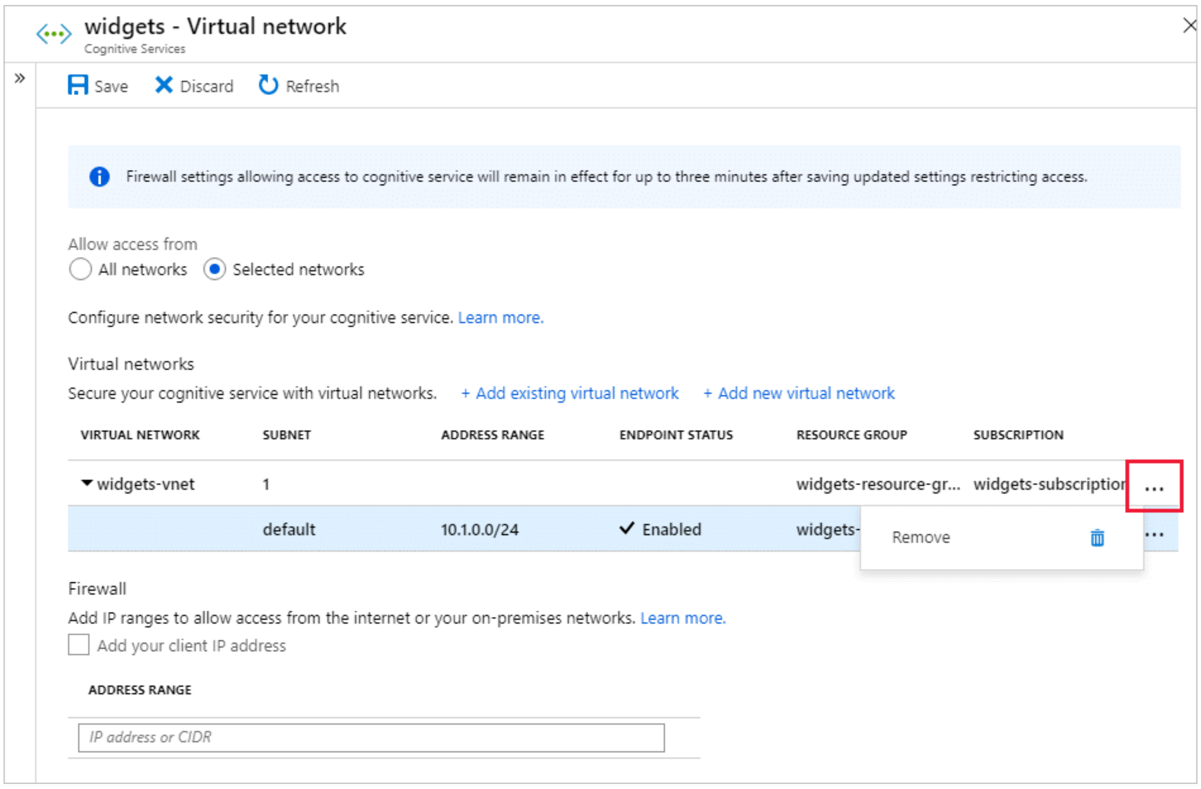
Create

NOTE

If a service endpoint for Azure Cognitive Services wasn't previously configured for the selected virtual network and subnets, you can configure it as part of this operation.

Presently, only virtual networks belonging to the same Azure Active Directory tenant are shown for selection during rule creation. To grant access to a subnet in a virtual network belonging to another tenant, please use Powershell, CLI or REST APIs.

8. To remove a virtual network or subnet rule, select **...** to open the context menu for the virtual network or subnet, and select **Remove**.



widgets - Virtual network
Cognitive Services

Save | Discard | Refresh

Firewall settings allowing access to cognitive service will remain in effect for up to three minutes after saving updated settings restricting access.

Allow access from
All networks [✓ Selected networks]

Configure network security for your cognitive service. Learn more.

Virtual networks
Secure your cognitive service with virtual networks. + Add existing virtual network + Add new virtual network

VIRTUAL NETWORK	SUBNET	ADDRESS RANGE	ENDPOINT STATUS	RESOURCE GROUP	SUBSCRIPTION
widgets-vnet	1			widgets-resource-gr...	widgets-subscription
	default	10.1.0.0/24	✓ Enabled	widgets-	

Firewall
Add IP ranges to allow access from the internet or your on-premises networks. Learn more.
Add your client IP address

ADDRESS RANGE
IP address or CIDR

9. Select **Save** to apply your changes.

IMPORTANT

Be sure to set the default rule to **deny**, or network rules have no effect.

Grant access from an internet IP range

You can configure Cognitive Services resources to allow access from specific public internet IP address ranges. This configuration grants access to specific services and on-premises networks, effectively blocking general internet traffic.

Provide allowed internet address ranges using CIDR notation in the form `16.17.18.0/24` or as individual IP addresses like `16.17.18.19`.

TIP

Small address ranges using "/31" or "/32" prefix sizes are not supported. These ranges should be configured using individual IP address rules.

IP network rules are only allowed for **public internet** IP addresses. IP address ranges reserved for private networks (as defined in RFC 1918) aren't allowed in IP rules. Private networks include addresses that start with `10.*`, `172.16.*` - `172.31.*`, and `192.168.*`.

> **NOTE**
>
> IP network rules have no effect on requests originating from the same Azure region as the Cognitive Services resource. Use Virtual network rules to allow same-region requests.

Only IPV4 addresses are supported at this time. Each Cognitive Services resource supports up to 100 IP network rules, which may be combined with Virtual network rules.

Configuring access from on-premises networks

To grant access from your on-premises networks to your Cognitive Services resource with an IP network rule, you must identify the internet facing IP addresses used by your network. Contact your network administrator for help.

If you're using ExpressRoute on-premises for public peering or Microsoft peering, you'll need to identify the NAT IP addresses. For public peering, each ExpressRoute circuit by default uses two NAT IP addresses. Each is applied to Azure service traffic when the traffic enters the Microsoft Azure network backbone. For Microsoft peering, the NAT IP addresses that are used are either customer provided or are provided by the service provider. To allow access to your service resources, you must allow these public IP addresses in the resource IP firewall setting. To find your public peering ExpressRoute circuit IP addresses, open a support ticket with ExpressRoute via the Azure portal. Learn more about NAT for ExpressRoute public and Microsoft peering.

Managing IP network rules

You can manage IP network rules for Cognitive Services resources through the Azure portal, PowerShell, or the Azure CLI.

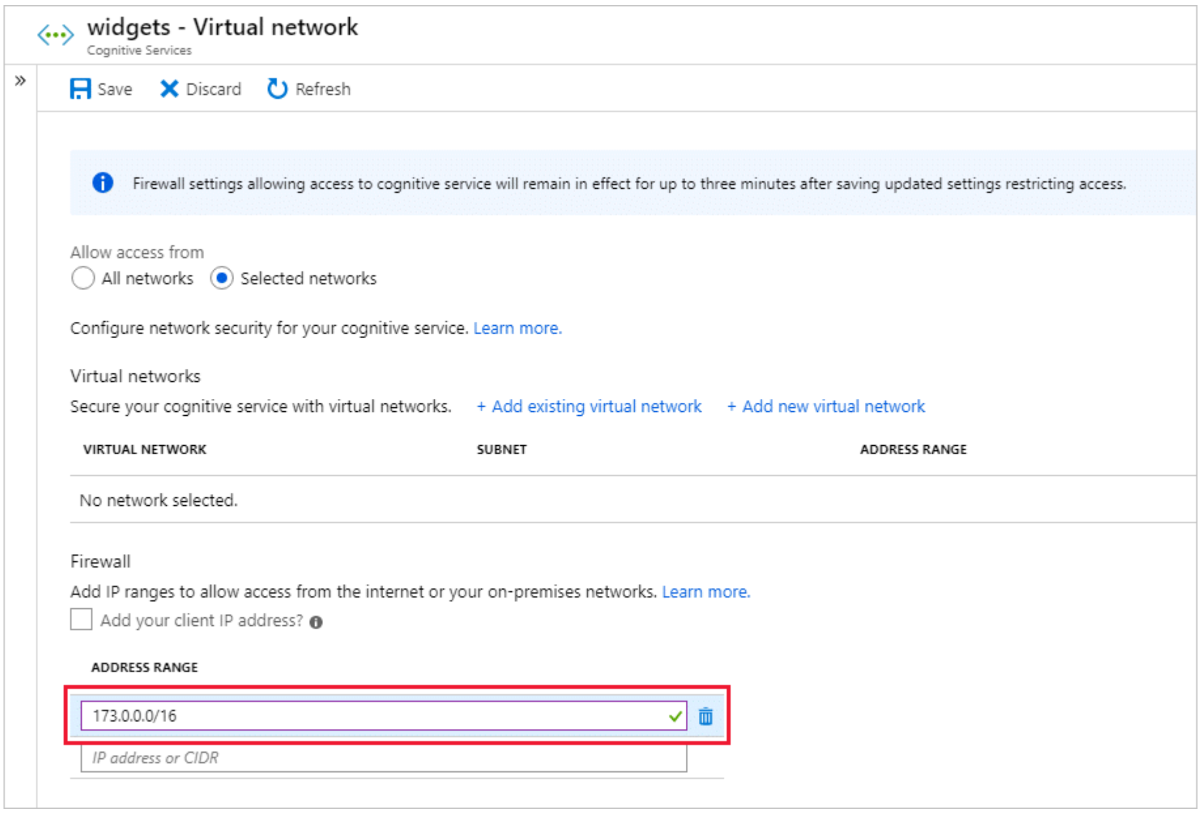
- Azure portal
- PowerShell
- Azure CLI

1. Go to the Cognitive Services resource you want to secure.

2. Select the **RESOURCE MANAGEMENT** menu called **Virtual network**.

3. Check that you've selected to allow access from **Selected networks**.

4. To grant access to an internet IP range, enter the IP address or address range (in CIDR format) under **Firewall** > **Address Range**. Only valid public IP (non-reserved) addresses are accepted.


widgets - Virtual network
Cognitive Services

Save | Discard | Refresh

Firewall settings allowing access to cognitive service will remain in effect for up to three minutes after saving updated settings restricting access.

Allow access from
All networks [✓ Selected networks]

Configure network security for your cognitive service. Learn more.

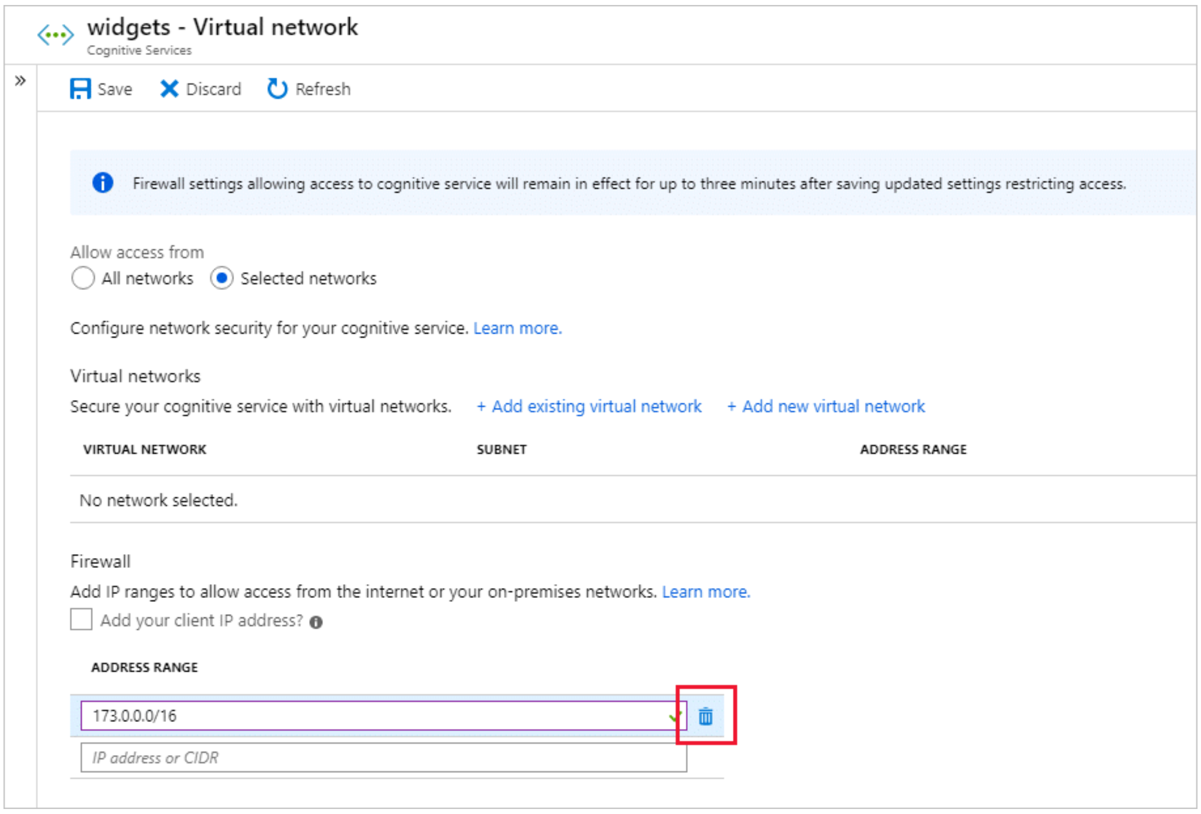
Virtual networks
Secure your cognitive service with virtual networks. + Add existing virtual network + Add new virtual network

VIRTUAL NETWORK	SUBNET	ADDRESS RANGE
No network selected.		

Firewall
Add IP ranges to allow access from the internet or your on-premises networks. Learn more.
Add your client IP address?

ADDRESS RANGE
173.0.0.0/16
IP address or CIDR

5. To remove an IP network rule, select the trash can icon next to the address range.


widgets - Virtual network
Cognitive Services

Save | Discard | Refresh

Firewall settings allowing access to cognitive service will remain in effect for up to three minutes after saving updated settings restricting access.

Allow access from
All networks [✓ Selected networks]

Configure network security for your cognitive service. Learn more.

Virtual networks
Secure your cognitive service with virtual networks. + Add existing virtual network + Add new virtual network

VIRTUAL NETWORK	SUBNET	ADDRESS RANGE
No network selected.		

Firewall
Add IP ranges to allow access from the internet or your on-premises networks. Learn more.
Add your client IP address?

ADDRESS RANGE
173.0.0.0/16
IP address or CIDR

6. Select **Save** to apply your changes.

> **IMPORTANT**
>
> Be sure to set the default rule to **deny**, or network rules have no effect.

Use private endpoints

You can use private endpoints for your Cognitive Services resources to allow clients on a virtual network (VNet) to securely access data over a Private Link. The private endpoint uses an IP address from the VNet address space for your Cognitive Services resource. Network traffic between the clients on the VNet and the resource traverses the VNet and a private link on the Microsoft backbone network, eliminating exposure from the public internet.

Private endpoints for Cognitive Services resources let you:

- Secure your Cognitive Services resource by configuring the firewall to block all connections on the public endpoint for the Cognitive Services service.
- Increase security for the VNet, by enabling you to block exfiltration of data from the VNet.
- Securely connect to Cognitive Services resources from on-premises networks that connect to the VNet using VPN or ExpressRoutes with private-peering.

Conceptual overview

A private endpoint is a special network interface for an Azure resource in your VNet. Creating a private endpoint for your Cognitive Services resource provides secure connectivity between clients in your VNet and your resource. The private endpoint is assigned an IP address from the IP address range of your VNet. The connection between the private endpoint and the Cognitive Services service uses a secure private link.

Applications in the VNet can connect to the service over the private endpoint seamlessly, using the same connection strings and authorization mechanisms that they would use otherwise. The exception is the Speech Service, which requires a separate endpoint. See the section on Private endpoints with the Speech Service. Private endpoints can be used with all protocols supported by the Cognitive Services resource, including REST.

Private endpoints can be created in subnets that use Service Endpoints. Clients in a subnet can connect to one Cognitive Services resource using private endpoint, while using service endpoints to access others.

When you create a private endpoint for a Cognitive Services resource in your VNet, a consent request is sent for approval to the Cognitive Services resource owner. If the user requesting the creation of the private endpoint is also an owner of the resource, this consent request is automatically approved.

Cognitive Services resource owners can manage consent requests and the private endpoints, through the '*Private endpoints*' tab for the Cognitive Services resource in the Azure portal.

Private endpoints

When creating the private endpoint, you must specify the Cognitive Services resource it connects to. For more information on creating a private endpoint, see:

- Create a private endpoint using the Private Link Center in the Azure portal
- Create a private endpoint using Azure CLI
- Create a private endpoint using Azure PowerShell

Connecting to private endpoints

Clients on a VNet using the private endpoint should use the same connection string for the Cognitive Services resource as clients connecting to the public endpoint. The exception is the Speech Service, which requires a separate endpoint. See the section on Private endpoints with the Speech Service. We rely upon DNS resolution to automatically route the connections from the VNet to the Cognitive Services resource over a private link. The Speech Service

We create a private DNS zone attached to the VNet with the necessary updates for the private endpoints, by default. However, if you're using your own DNS server, you may need to make additional changes to your DNS configuration. The section on DNS changes below describes the updates required for private endpoints.

Private endpoints with the Speech Service

When using private endpoints with the Speech Service, you must use a custom endpoint to call the Speech Service. You cannot use the global endpoint. The endpoint must follow this pattern:

`{account}.{stt|tts|voice|dls}.speech.microsoft.com`.

DNS changes for private endpoints

When you create a private endpoint, the DNS CNAME resource record for the Cognitive Services resource is updated to an alias in a subdomain with the prefix '*privatelink*'. By default, we also create a private DNS zone, corresponding to the '*privatelink*' subdomain, with the DNS A resource records for the private endpoints.

When you resolve the endpoint URL from outside the VNet with the private endpoint, it resolves to the public endpoint of the Cognitive Services resource. When resolved from the VNet hosting the private endpoint, the endpoint URL resolves to the private endpoint's IP address.

This approach enables access to the Cognitive Services resource using the same connection string for clients in the VNet hosting the private endpoints and clients outside the VNet.

If you are using a custom DNS server on your network, clients must be able to resolve the fully qualified domain name (FQDN) for the Cognitive Services resource endpoint to the private endpoint IP address. Configure your DNS server to delegate your private link subdomain to the private DNS zone for the VNet.

TIP

When using a custom or on-premises DNS server, you should configure your DNS server to resolve the Cognitive Services resource name in the 'privatelink' subdomain to the private endpoint IP address. You can do this by delegating the 'privatelink' subdomain to the private DNS zone of the VNet, or configuring the DNS zone on your DNS server and adding the DNS A records.

For more information on configuring your own DNS server to support private endpoints, refer to the following articles:

- Name resolution for resources in Azure virtual networks
- DNS configuration for private endpoints

Pricing

For pricing details, see Azure Private Link pricing.

Next steps

- Explore the various Azure Cognitive Services
- Learn more about Azure Virtual Network Service Endpoints

Configure customer-managed keys with Azure Key Vault for Cognitive Services

10/4/2020 • 2 minutes to read • Edit Online

The process to enable Customer-Managed Keys with Azure Key Vault for Cognitive Services varies by product. Use these links for service-specific instructions:

Vision

- Custom Vision encryption of data at rest
- Face Services encryption of data at rest
- Form Recognizer encryption of data at rest

Language

- Language Understanding service encryption of data at rest
- QnA Maker encryption of data at rest
- Translator encryption of data at rest

Decision

- Content Moderator encryption of data at rest
- Personalizer encryption of data at rest

Next steps

- What is Azure Key Vault?
- Cognitive Services Customer-Managed Key Request Form

Authenticate requests to Azure Cognitive Services

10/4/2020 • 8 minutes to read • Edit Online

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)

Prerequisites

Before you make a request, you need an Azure account and an Azure Cognitive Services subscription. If you already have an account, go ahead and skip to the next section. If you don't have an account, we have a guide to get you set up in minutes: Create a Cognitive Services account for Azure.

You can get your subscription key from the Azure portal after creating your account.

Authentication headers

Let's quickly review the authentication headers available for use with Azure Cognitive Services.

HEADER	DESCRIPTION
Ocp-Apim-Subscription-Key	Use this header to authenticate with a subscription key for a specific service or a multi-service subscription key.
Ocp-Apim-Subscription-Region	This header is only required when using a multi-service subscription key with the Translator service. Use this header to specify the subscription region.
Authorization	Use this header if you are using an authentication token. The steps to perform a token exchange are detailed in the following sections. The value provided follows this format: `Bearer <TOKEN>`.

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.

These sample requests demonstrates how to use the `Ocp-Apim-Subscription-Key` header. Keep in mind, when using this sample you'll need to include a valid subscription key.

This is a sample call to the Bing Web Search API:

```
curl -X GET 'https://api.cognitive.microsoft.com/bing/v7.0/search?q=Welsch%20Pembroke%20Corgis' \
-H 'Ocp-Apim-Subscription-Key: YOUR_SUBSCRIPTION_KEY' | json_pp
```

This is a sample call to the Translator service:

```
curl -X POST 'https://api.cognitive.microsofttranslator.com/translate?api-version=3.0&from=en&to=de' \
-H 'Ocp-Apim-Subscription-Key: YOUR_SUBSCRIPTION_KEY' \
-H 'Content-Type: application/json' \
--data-raw '[{ "text": "How much for the cup of coffee?" }]' | json_pp
```

The following video demonstrates using a Cognitive Services key.

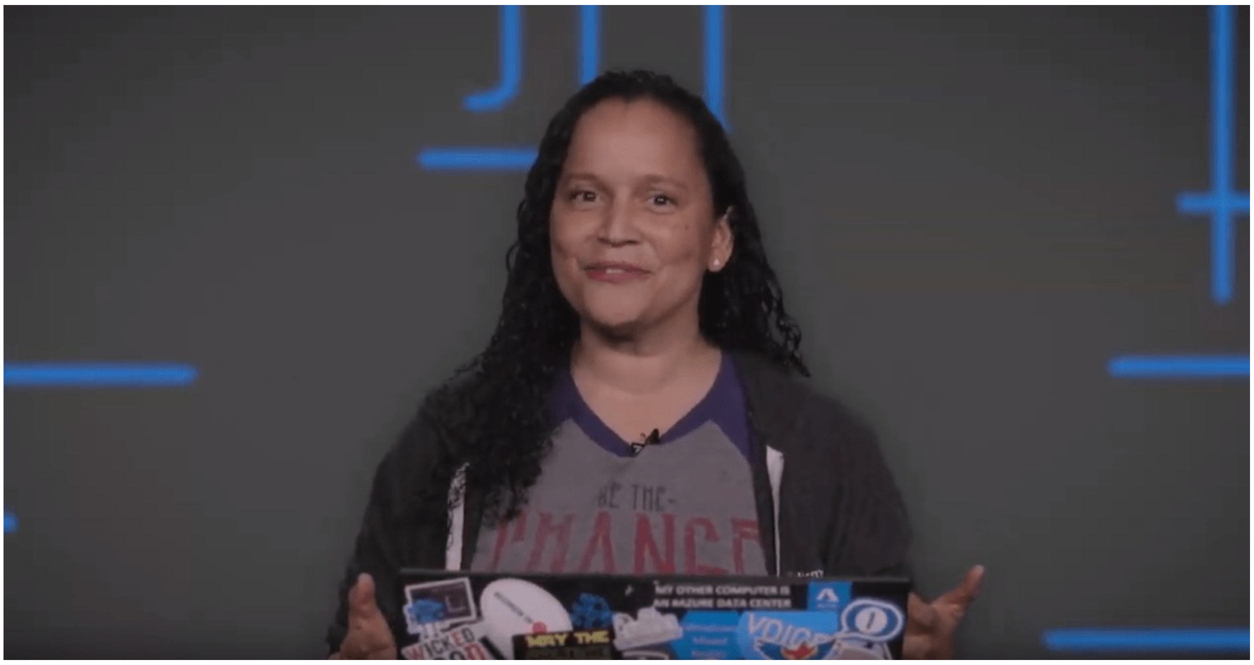
Authenticate with a multi-service subscription key

WARNING

At this time, these services **don't** support multi-service keys: QnA Maker, Speech Services, Custom Vision, and Anomaly Detector.

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. See Cognitive Services pricing for information about regional availability, supported features, and pricing.

The subscription key is provided in each request as the `Ocp-Apim-Subscription-Key` header.



Supported regions

When using the multi-service subscription key to make a request to `api.cognitive.microsoft.com`, you must include the region in the URL. For example: `westus.api.cognitive.microsoft.com`.

When using multi-service subscription key with the Translator service, you must specify the subscription region with the `Ocp-Apim-Subscription-Region` header.

Multi-service authentication is supported in these regions:

- `australiaeast`
- `brazilsouth`
- `canadacentral`
- `centralindia`

- `eastasia`
- `eastus`
- `japaneast`
- `northeurope`
- `southcentralus`
- `southeastasia`
- `uksouth`
- `westcentralus`
- `westeurope`
- `westus`
- `westus2`

Sample requests

This is a sample call to the Bing Web Search API:

```
curl -X GET 'https://YOUR-REGION.api.cognitive.microsoft.com/bing/v7.0/search?q=Welsch%20Pembroke%20Corgis'
\
-H 'Ocp-Apim-Subscription-Key: YOUR_SUBSCRIPTION_KEY' | json_pp
```

This is a sample call to the Translator service:

```
curl -X POST 'https://api.cognitive.microsofttranslator.com/translate?api-version=3.0&from=en&to=de' \
-H 'Ocp-Apim-Subscription-Key: YOUR_SUBSCRIPTION_KEY' \
-H 'Ocp-Apim-Subscription-Region: YOUR_SUBSCRIPTION_REGION' \
-H 'Content-Type: application/json' \
--data-raw '[{ "text": "How much for the cup of coffee?" }]' | json_pp
```

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

NOTE

QnA Maker also uses the Authorization header, but requires an endpoint key. For more information, see QnA Maker: Get answer from knowledge base.

WARNING

The services that support authentication tokens may change over time, please check the API reference for a service before using this authentication method.

Both single service and multi-service subscription keys can be exchanged for authentication tokens. Authentication tokens are valid for 10 minutes.

Authentication tokens are included in a request as the `Authorization` header. The token value provided must be preceded by `Bearer`, for example: `Bearer YOUR_AUTH_TOKEN`.

Sample requests

Use this URL to exchange a subscription key for an authentication token: `https://YOUR-REGION.api.cognitive.microsoft.com/sts/v1.0/issueToken`.

```
curl -v -X POST \
"https://YOUR-REGION.api.cognitive.microsoft.com/sts/v1.0/issueToken" \
-H "Content-type: application/x-www-form-urlencoded" \
-H "Content-length: 0" \
-H "Ocp-Apim-Subscription-Key: YOUR_SUBSCRIPTION_KEY"
```

These multi-service regions support token exchange:

- `australiaeast`
- `brazilsouth`
- `canadacentral`
- `centralindia`
- `eastasia`
- `eastus`
- `japaneast`
- `northeurope`
- `southcentralus`
- `southeastasia`
- `uksouth`
- `westcentralus`
- `westeurope`
- `westus`
- `westus2`

After you get an authentication token, you'll need to pass it in each request as the `Authorization` header. This is a sample call to the Translator service:

```
curl -X POST 'https://api.cognitive.microsofttranslator.com/translate?api-version=3.0&from=en&to=de' \
-H 'Authorization: Bearer YOUR_AUTH_TOKEN' \
-H 'Content-Type: application/json' \
--data-raw '[{ "text": "How much for the cup of coffee?" }]' | json_pp
```

Authenticate with Azure Active Directory

IMPORTANT

1. Currently, **only** the Computer Vision API, Face API, Text Analytics API, Immersive Reader, Form Recognizer, Anomaly Detector, and all Bing services except Bing Custom Search support authentication using Azure Active Directory (AAD).
2. AAD authentication needs to be always used together with custom subdomain name of your Azure resource. Regional endpoints does not support AAD authentication.

In the previous sections, we showed you how to authenticate against Azure Cognitive Services using either 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). Let's take a look at what's required to authenticate using Azure Active Directory (AAD).

In the following sections, you'll use either the Azure Cloud Shell environment or the Azure CLI to create a subdomain, assign roles, and obtain a bearer token to call the Azure Cognitive Services. If you get stuck, links are provided in each section with all available options for each command in Azure Cloud Shell/Azure CLI.

Create a resource with a custom subdomain

The first step is to create a custom subdomain. If you want to use an existing Cognitive Services resource which does not have custom subdomain name, follow the instructions in Cognitive Services Custom Subdomains to enable custom subdomain for your resource.

1. Start by opening the Azure Cloud Shell. Then select a subscription:

```
Set-AzContext -SubscriptionName <SubscriptionName>
```

2. Next, create a Cognitive Services resource with a custom subdomain. The subdomain name needs to be globally unique and cannot include special characters, such as: ".", "!", ",".

```
$account = New-AzCognitiveServicesAccount -ResourceGroupName <RESOURCE_GROUP_NAME> -name
<ACCOUNT_NAME> -Type <ACCOUNT_TYPE> -SkuName <SUBSCRIPTION_TYPE> -Location <REGION> -
CustomSubdomainName <UNIQUE_SUBDOMAIN>
```

3. If successful, the **Endpoint** should show the subdomain name unique to your resource.

Assign a role to a service principal

Now that you have a custom subdomain associated with your resource, you're going to need to assign a role to a service principal.

NOTE

Keep in mind that Azure role assignments may take up to five minutes to propagate.

1. First, let's register an AAD application.

```
$SecureStringPassword = ConvertTo-SecureString -String <YOUR_PASSWORD> -AsPlainText -Force

$app = New-AzADApplication -DisplayName <APP_DISPLAY_NAME> -IdentifierUris <APP_URIS> -Password
$SecureStringPassword
```

   You're going to need the **ApplicationId** in the next step.

2. Next, you need to create a service principal for the AAD application.

```
New-AzADServicePrincipal -ApplicationId <APPLICATION_ID>
```

   NOTE

   If you register an application in the Azure portal, this step is completed for you.

3. The last step is to assign the "Cognitive Services User" role to the service principal (scoped to the resource). By assigning a role, you're granting service principal access to this resource. You can grant the same service principal access to multiple resources in your subscription.

NOTE

The ObjectId of the service principal is used, not the ObjectId for the application. The ACCOUNT_ID will be the Azure resource Id of the Cognitive Services account you created. You can find Azure resource Id from "properties" of the resource in Azure portal.

```
New-AzRoleAssignment -ObjectId <SERVICE_PRINCIPAL_OBJECTID> -Scope <ACCOUNT_ID> -RoleDefinitionName
"Cognitive Services User"
```

Sample request

In this sample, a password is used to authenticate the service principal. The token provided is then used to call the Computer Vision API.

1. Get your **TenantId**:

```
$context=Get-AzContext
$context.Tenant.Id
```

2. Get a token:

NOTE

If you're using Azure Cloud Shell, the `SecureClientSecret` class isn't available.

- PowerShell
- Azure Cloud Shell

```
$authContext = New-Object "Microsoft.IdentityModel.Clients.ActiveDirectory.AuthenticationContext" -
ArgumentList "https://login.windows.net/<TENANT_ID>"
$secureSecretObject = New-Object "Microsoft.IdentityModel.Clients.ActiveDirectory.SecureClientSecret"
-ArgumentList $SecureStringPassword
$clientCredential = New-Object "Microsoft.IdentityModel.Clients.ActiveDirectory.ClientCredential" -
ArgumentList $app.ApplicationId, $secureSecretObject
$token=$authContext.AcquireTokenAsync("https://cognitiveservices.azure.com/",
$clientCredential).Result
$token
```

3. Call the Computer Vision API:

```
$url = $account.Endpoint+"vision/v1.0/models"
$result = Invoke-RestMethod -Uri $url  -Method Get -Headers
@{"Authorization"=$token.CreateAuthorizationHeader()} -Verbose
$result | ConvertTo-Json
```

Alternatively, the service principal can be authenticated with a certificate. Besides service principal, user principal is also supported by having permissions delegated through another AAD application. In this case, instead of passwords or certificates, users would be prompted for two-factor authentication when acquiring token.

Authorize access to managed identities

Cognitive Services support Azure Active Directory (Azure AD) authentication with managed identities for Azure resources. Managed identities for Azure resources can authorize access to Cognitive Services resources using

Azure AD credentials from applications running in Azure virtual machines (VMs), function apps, virtual machine scale sets, and other services. By using managed identities for Azure resources together with Azure AD authentication, you can avoid storing credentials with your applications that run in the cloud.

Enable managed identities on a VM

Before you can use managed identities for Azure resources to authorize access to Cognitive Services resources from your VM, you must enable managed identities for Azure resources on the VM. To learn how to enable managed identities for Azure Resources, see:

- Azure portal
- Azure PowerShell
- Azure CLI
- Azure Resource Manager template
- Azure Resource Manager client libraries

For more information about managed identities, see Managed identities for Azure resources.

See also

- What is Cognitive Services?
- Cognitive Services pricing
- Custom subdomains

Azure security baseline for Cognitive Services

10/4/2020 • 33 minutes to read • Edit Online

The Azure Security Baseline for Cognitive Services contains recommendations that will help you improve the security posture of your deployment.

The baseline for this service is drawn from the Azure Security Benchmark version 1.0, which provides recommendations on how you can secure your cloud solutions on Azure with our best practices guidance.

For more information, see the Azure security baselines overview.

Network security

For more information, see Security control: Network security.

1.1: Protect Azure resources within virtual networks

Guidance: 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 that originate from specified IP addresses, IP ranges, or from a list of subnets in Azure Virtual Networks.

Virtual network and service endpoint support for Cognitive Services is limited to a specific set of regions.

- How to configure Azure Cognitive Services virtual networks
- Overview of Azure Virtual Networks

Azure Security Center monitoring: Yes

Responsibility: Customer

1.2: Monitor and log the configuration and traffic of virtual networks, subnets, and NICs

Guidance: When Virtual Machines are deployed in the same virtual network as your Azure Cognitive Services container, you can use network security groups (NSG) to reduce the risk of data exfiltration. Enable NSG flow logs and send logs into an Azure Storage Account for traffic audit. You may also send NSG flow logs to a Log Analytics workspace and use Traffic Analytics to provide insights into traffic flow in your Azure cloud. Some advantages of Traffic Analytics are the ability to visualize network activity and identify hot spots, identify security threats, understand traffic flow patterns, and pinpoint network misconfigurations.

- How to Enable NSG Flow Logs
- How to Enable and use Traffic Analytics

Azure Security Center monitoring: Yes

Responsibility: Customer

1.3: Protect critical web applications

Guidance: If you are using Cognitive Services within a container, you may augment your container deployment with a front-facing web-application firewall solution that filters malicious traffic and supports end-to-end TLS encryption, keeping the container endpoint private and secure.

Bear in mind that Cognitive Services containers are required to submit metering information for billing purposes.

The only exception, is Offline containers as they follow a different billing methodology. Failure to allow list various network channels that the Cognitive Services containers rely on will prevent the container from working. The host should allow list port 443 and the following domains:

- *.cognitive.microsoft.com
- *.cognitiveservices.azure.com

Also note that you must disable deep packet inspection for your firewall solution on the secure channels that the Cognitive Services containers create to Microsoft servers. Failure to do so will prevent the container from functioning correctly.

- Understand Azure Cognitive Services container security

Azure Security Center monitoring: Yes

Responsibility: Customer

1.4: Deny communications with known malicious IP addresses

Guidance: When virtual machines are deployed in the same virtual network as your Azure Cognitive Services container, define and implement standard security configurations for related network resources with Azure Policy. Use Azure Policy aliases in the "Microsoft.CognitiveServices" and "Microsoft.Network" namespaces to create custom policies to audit or enforce the network configuration of your Azure Cache for Redis instances. You may also make use of built-in policy definitions such as:

- DDoS Protection Standard should be enabled

You may also use Azure Blueprints to simplify large-scale Azure deployments by packaging key environment artifacts, such as Azure Resource Manager templates, Azure role-based access control (Azure RBAC), and policies, in a single blueprint definition. Easily apply the blueprint to new subscriptions and environments, and fine-tune control and management through versioning.

If you are using Cognitive Services within a container, you may augment your container deployment with a front-facing web-application firewall solution that filters malicious traffic and supports end-to-end TLS encryption, keeping the container endpoint private and secure.

- How to configure and manage Azure Policy
- How to create an Azure Blueprint
- Understand Azure Cognitive Services container security

Azure Security Center monitoring: Yes

Responsibility: Customer

1.5: Record network packets

Guidance: When virtual machines are deployed in the same virtual network as your Azure Cognitive Services container, you can use network security groups (NSG) to reduce the risk of data exfiltration. Enable NSG flow logs and send logs into an Azure Storage Account for traffic audit. You may also send NSG flow logs to a Log Analytics workspace and use Traffic Analytics to provide insights into traffic flow in your Azure cloud. Some advantages of Traffic Analytics are the ability to visualize network activity and identify hot spots, identify security threats, understand traffic flow patterns, and pinpoint network misconfigurations.

- How to Enable NSG Flow Logs
- How to Enable and use Traffic Analytics

Azure Security Center monitoring: Yes

Responsibility: Customer

1.6: Deploy network-based intrusion detection/intrusion prevention systems (IDS/IPS)

Guidance: If using Cognitive Services within a container, you may augment your container deployment with a front-facing web-application firewall solution that filters malicious traffic and supports end-to-end TLS encryption, keeping the container endpoint private and secure. You can select an offer from the Azure Marketplace that supports IDS/IPS functionality with the ability to disable payload inspection.

Bear in mind that Cognitive Services containers are required to submit metering information for billing purposes. The only exception is Offline containers as they follow a different billing methodology. Failure to allow list various network channels that the Cognitive Services containers rely on will prevent the container from working. The host should allow list port 443 and the following domains:

- *.cognitive.microsoft.com
- *.cognitiveservices.azure.com

Also note that you must disable deep packet inspection for your firewall solution on the secure channels that the Cognitive Services containers create to Microsoft servers. Failure to do so will prevent the container from functioning correctly.

- Understand Azure Cognitive Services container security
- Azure Marketplace

Azure Security Center monitoring: Yes

Responsibility: Customer

1.7: Manage traffic to web applications

Guidance: If using Cognitive Services within a container, you may augment your container deployment with a front-facing web-application firewall solution that filters malicious traffic and supports end-to-end TLS encryption, keeping the container endpoint private and secure.

Bear in mind that Cognitive Services containers are required to submit metering information for billing purposes. The only exception, is Offline containers as they follow a different billing methodology. Failure to allow list various network channels that the Cognitive Services containers rely on will prevent the container from working. The host should allow list port 443 and the following domains:

- *.cognitive.microsoft.com
- *.cognitiveservices.azure.com

Also note that you must disable deep packet inspection for your firewall solution on the secure channels that the Cognitive Services containers create to Microsoft servers. Failure to do so will prevent the container from functioning correctly.

- Understand Azure Cognitive Services container security

Azure Security Center monitoring: Yes

Responsibility: Customer

1.8: Minimize complexity and administrative overhead of network security rules

Guidance: Use virtual network service tags to define network access controls on network security groups (NSG) or Azure Firewall. You can use service tags in place of specific IP addresses when creating security rules. By specifying the service tag name (e.g., ApiManagement) in the appropriate source or destination field of a rule, you can allow or deny the traffic for the corresponding service. Microsoft manages the address prefixes encompassed by the service tag and automatically updates the service tag as addresses change.

You may also use application security groups (ASG) to help simplify complex security configuration. ASGs enable you to configure network security as a natural extension of an application's structure, allowing you to group virtual machines and define network security policies based on those groups.

- Virtual network service tags
- Application Security Groups

Azure Security Center monitoring: Not applicable

Responsibility: Customer

1.9: Maintain standard security configurations for network devices

Guidance: Define and implement standard security configurations for network resources related to your Azure Cognitive Services container with Azure Policy. Use Azure Policy aliases in the "Microsoft.CognitiveServices" and "Microsoft.Network" namespaces to create custom policies to audit or enforce the network configuration of your Azure Cache for Redis instances.

You can also use Azure Blueprints to simplify large-scale Azure deployments by packaging key environment artifacts, such as Azure Resource Manager templates, Azure role-based access control (Azure RBAC), and policies, in a single blueprint definition. Easily apply the blueprint to new subscriptions and environments, and fine-tune control and management through versioning.

- How to configure and manage Azure Policy
- How to create an Azure Blueprint

Azure Security Center monitoring: Currently not available

Responsibility: Customer

1.10: Document traffic configuration rules

Guidance: Use tags for network resources associated with your Azure Cognitive Services container in order to logically organize them into a taxonomy.

- How to create and use tags

Azure Security Center monitoring: Not applicable

Responsibility: Customer

1.11: Use automated tools to monitor network resource configurations and detect changes

Guidance: Use the Azure Activity log to monitor network resource configurations and detect changes for network resources related to your Azure Cognitive Services container. Create alerts within Azure Monitor that will trigger when changes to critical network resources take place.

- How to view and retrieve Azure Activity Log events
- How to create alerts in Azure Monitor

Azure Security Center monitoring: Currently not available

Responsibility: Customer

Logging and monitoring

For more information, see Security control: Logging and monitoring.

2.1: Use approved time synchronization sources

Guidance: Microsoft maintains the time source used for Azure resources such as Azure Cognitive Services for timestamps in the logs.

Azure Security Center monitoring: Not applicable

Responsibility: Microsoft

2.2: Configure central security log management

Guidance: Enable Azure Activity Log diagnostic settings and send the logs to a Log Analytics workspace, Azure event hub, or Azure storage account for archive. Activity logs provide insight into the operations that were performed on your Azure Cognitive Services container at the control plane level. Using Azure Activity Log data, you can determine the "what, who, and when" for any write operations (PUT, POST, DELETE) performed at the control plane level for your Azure Cache for Redis instances.

- How to enable Diagnostic Settings for Azure Activity Log

Azure Security Center monitoring: Yes

Responsibility: Customer

2.3: Enable audit logging for Azure resources

Guidance: For control plane audit logging, enable Azure Activity Log diagnostic settings and send the logs to a Log Analytics workspace, Azure event hub, or Azure storage account for archive. Using Azure Activity Log data, you can determine the "what, who, and when" for any write operations (PUT, POST, DELETE) performed at the control plane level for your Azure resources.

Additionally, Azure Cognitive Services sends diagnostics events that can be collected and used for the purposes of analysis, alerting and reporting. You can configure diagnostics settings for a Cognitive Services container via the Azure portal. You can send one or more diagnostics events to a Storage Account, Event Hub, or a Log Analytics workspace.

- How to enable Diagnostic Settings for Azure Activity Log
- Using diagnostic settings to for Azure Cognitive Services

Azure Security Center monitoring: Yes

Responsibility: Customer

2.4: Collect security logs from operating systems

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

2.5: Configure security log storage retention

Guidance: Within Azure Monitor, set your Log Analytics Workspace retention period according to your organization's compliance regulations. Use Azure Storage accounts for long-term/archival storage.

- How to set log retention parameters for Log Analytics Workspaces

Azure Security Center monitoring: Not applicable

Responsibility: Customer

2.6: Monitor and review Logs

Guidance: Enable Azure Activity Log diagnostic settings and send the logs to a Log Analytics workspace. These logs provide rich, frequent data about the operation of a resource that are used for issue identification and debugging. Perform queries in Log Analytics to search terms, identify trends, analyze patterns, and provide many other insights based on the Activity Log Data that may have been collected for Azure Cognitive Services.

- How to enable Diagnostic Settings for Azure Activity Log
- How to collect and analyze Azure activity logs in Log Analytics workspace in Azure Monitor

Azure Security Center monitoring: Not applicable

Responsibility: Customer

2.7: Enable alerts for anomalous activities

Guidance: You can raise alerts on supported metrics in Azure Cognitive Services by going to the Alerts & Metrics section in Azure Monitor.

Configure diagnostic settings for your Cognitive Services container and send logs to a Log Analytics workspace. Within your Log Analytics workspace, configure alerts to take place for when a pre-defined set of conditions takes place. Alternatively, you may enable and on-board data to Azure Sentinel or a third-party SIEM.

- How to onboard Azure Sentinel
- Create, view, and manage log alerts using Azure Monitor

Azure Security Center monitoring: Yes

Responsibility: Customer

2.8: Centralize anti-malware logging

Guidance: Not applicable; Azure Cognitive Services does not process or produce anti-malware related logs.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

2.9: Enable DNS query logging

Guidance: Not applicable; Azure Cognitive Services does not process or produce DNS related logs.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

2.10: Enable command-line audit logging

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

Identity and access control

For more information, see Security control: Identity and access control.

3.1: Maintain an inventory of administrative accounts

Guidance: Azure Active Directory (AD) has built-in roles that must be explicitly assigned and are queryable. Use the Azure AD PowerShell module to perform ad hoc queries to discover accounts that are members of administrative groups.

- How to get a directory role in Azure AD with PowerShell
- How to get members of a directory role in Azure AD with PowerShell

Azure Security Center monitoring: Yes

Responsibility: Customer

3.2: Change default passwords where applicable

Guidance: Control plane access to Azure Cognitive Services is controlled through Azure Active Directory (AD). Azure AD does not have the concept of default passwords.

Data plane access to Azure Cognitive Services is controlled through access keys. These keys are used by the clients connecting to your cache and can be regenerated at any time.

It is not recommended that you build default passwords into your application. Instead, you can store your passwords in Azure Key Vault and then use Azure Active Directory to retrieve them.

- How to regenerate Azure Cache for Redis access keys

Azure Security Center monitoring: Not applicable

Responsibility: Customer

3.3: Use dedicated administrative accounts

Guidance: Create standard operating procedures around the use of dedicated administrative accounts. Use Azure Security Center Identity and Access Management to monitor the number of administrative accounts.

Additionally, to help you keep track of dedicated administrative accounts, you may use recommendations from Azure Security Center or built-in Azure Policies, such as:

- There should be more than one owner assigned to your subscription
- Deprecated accounts with owner permissions should be removed from your subscription
- External accounts with owner permissions should be removed from your subscription

- How to use Azure Security Center to monitor identity and access (Preview)

- How to use Azure Policy

Azure Security Center monitoring: Yes

Responsibility: Customer

3.4: Use single sign-on (SSO) with Azure Active Directory

Guidance: Azure Cognitive Services uses access keys to authenticate users and does not support single sign-on (SSO) at the data plane level. Access to the control plane for Azure Cognitive Services is available via REST API and supports SSO. To authenticate, set the Authorization header for your requests to a JSON Web Token that you obtain from Azure Active Directory.

- Understand Azure Cognitive Services REST API

- Understand SSO with Azure AD

Azure Security Center monitoring: Not applicable

Responsibility: Customer

3.5: Use multi-factor authentication for all Azure Active Directory based access

Guidance: Enable Azure Active Directory (AD) Multi-Factor Authentication (MFA) and follow Azure Security Center Identity and Access Management recommendations.

- How to enable MFA in Azure

- How to monitor identity and access within Azure Security Center

Azure Security Center monitoring: Yes

Responsibility: Customer

3.6: Use dedicated machines (Privileged Access Workstations) for all administrative tasks

Guidance: Use privileged access workstations (PAW) with Multi-Factor Authentication (MFA) configured to log into and configure Azure resources.

- Learn about Privileged Access Workstations
- How to enable MFA in Azure

Azure Security Center monitoring: Not applicable

Responsibility: Customer

3.7: Log and alert on suspicious activities from administrative accounts

Guidance: Use Azure Active Directory (AD) Privileged Identity Management (PIM) for generation of logs and alerts when suspicious or unsafe activity occurs in the environment.

In addition, use Azure AD risk detections to view alerts and reports on risky user behavior.

- How to deploy Privileged Identity Management (PIM)
- Understand Azure AD risk detections

Azure Security Center monitoring: Yes

Responsibility: Customer

3.8: Manage Azure resources from only approved locations

Guidance: Configure named locations in Azure Active Directory (AD) Conditional Access to allow access from only specific logical groupings of IP address ranges or countries/regions.

- How to configure Named Locations in Azure

Azure Security Center monitoring: Not applicable

Responsibility: Customer

3.9: Use Azure Active Directory

Guidance: Use Azure Active Directory (AD) as the central authentication and authorization system. Azure AD protects data by using strong encryption for data at rest and in transit. Azure AD also salts, hashes, and securely stores user credentials. If your use case supports AD authentication, use Azure AD to authenticate requests to your Cognitive Services API.

Currently, only the Computer Vision API, Face API, Text Analytics API, Immersive Reader, Form Recognizer, Anomaly Detector, and all Bing services except Bing Custom Search support authentication using Azure AD.

- How to authenticate requests to Cognitive Services

Azure Security Center monitoring: Not applicable

Responsibility: Customer

3.10: Regularly review and reconcile user access

Guidance: Azure AD provides logs to help discover stale accounts. In addition, customer to utilize Azure Identity Access Reviews to efficiently manage group memberships, access to enterprise applications, and role assignments. User's access can be reviewed on a regular basis to make sure only the right Users have continued access.

Customer to maintain inventory of API Management user accounts, reconcile access as needed. In API Management, developers are the users of the APIs that you expose using API Management. By default, newly created developer accounts are Active, and associated with the Developers group. Developer accounts that are in an active state can be used to access all of the APIs for which they have subscriptions.

- How to manage user accounts in Azure API Management
- How to get list of API Management users
- How to use Azure Identity Access Reviews

Azure Security Center monitoring: Not applicable

Responsibility: Customer

3.11: Monitor attempts to access deactivated credentials

Guidance: You have access to Azure Active Directory (AD) sign-in activity, audit and risk event log sources, which allow you to integrate with Azure Sentinel or a third-party SIEM.

You can streamline this process by creating diagnostic settings for Azure AD user accounts and sending the audit logs and sign-in logs to a Log Analytics workspace. You can configure desired log alerts within Log Analytics.

- How to integrate Azure Activity Logs into Azure Monitor
- How to on-board Azure Sentinel

Azure Security Center monitoring: Not applicable

Responsibility: Customer

3.12: Alert on account login behavior deviation

Guidance: For account login behavior deviation on the control plane, use Azure Active Directory (AD) Identity Protection and risk detection features to configure automated responses to detected suspicious actions related to user identities. You can also ingest data into Azure Sentinel for further investigation.

- How to view Azure AD risky sign-ins
- How to configure and enable Identity Protection risk policies
- How to onboard Azure Sentinel

Azure Security Center monitoring: Currently not available

Responsibility: Customer

3.13: Provide Microsoft with access to relevant customer data during support scenarios

Guidance: Not yet available; Customer Lockbox is not yet supported for Azure Cognitive Services.

- List of Customer Lockbox-supported services

Azure Security Center monitoring: Currently not available

Responsibility: Currently not available

Data protection

For more information, see Security control: Data protection.

4.1: Maintain an inventory of sensitive Information

Guidance: Use tags to assist in tracking Azure resources that store or process sensitive information.

- How to create and use tags

Azure Security Center monitoring: Not applicable

Responsibility: Customer

4.2: Isolate systems storing or processing sensitive information

Guidance: Implement separate subscriptions and/or management groups for development, test, and production. Resources should be separated by VNet/Subnet, tagged appropriately, and secured by an NSG or Azure Firewall. Resources storing or processing sensitive data should be sufficiently isolated. For Virtual Machines storing or processing sensitive data, implement policy and procedure(s) to turn them off when not in use.

- How to create additional Azure subscriptions
- How to create Management Groups
- How to create and use Tags
- How to create a Virtual Network
- How to create an NSG with a Security Config
- How to deploy Azure Firewall
- How to configure alert or alert and deny with Azure Firewall

Azure Security Center monitoring: Not applicable

Responsibility: Customer

4.3: Monitor and block unauthorized transfer of sensitive information

Guidance: Not yet available; data identification, classification, and loss prevention features are not yet available for Azure Cognitive Services.

Microsoft manages the underlying infrastructure for Azure Cognitive Services and has implemented strict controls to prevent the loss or exposure of customer data.

- Understand customer data protection in Azure

Azure Security Center monitoring: Currently not available

Responsibility: Currently not available

4.4: Encrypt all sensitive information in transit

Guidance: All of the Cognitive Services endpoints exposed over HTTP enforce TLS 1.2. With an enforced security protocol, consumers attempting to call a Cognitive Services endpoint should adhere to these guidelines:

- The client Operating System (OS) needs to support TLS 1.2.
- The language (and platform) used to make the HTTP call need to specify TLS 1.2 as part of the request. (Depending on the language and platform, specifying TLS is done either implicitly or explicitly.)

- Understand Transport Layer Security for Azure Cognitive Services

Azure Security Center monitoring: Currently not available

Responsibility: Shared

4.5: Use an active discovery tool to identify sensitive data

Guidance: Data identification, classification, and loss prevention features are not yet available for Azure Cognitive Services. Tag instances containing sensitive information as such and implement third-party solution if required for compliance purposes.

For the underlying platform which is managed by Microsoft, Microsoft treats all customer content as sensitive and goes to great lengths to guard against customer data loss and exposure. To ensure customer data within Azure remains secure, Microsoft has implemented and maintains a suite of robust data protection controls and capabilities.

- Understand customer data protection in Azure

Azure Security Center monitoring: Not applicable

Responsibility: Shared

4.6: Use Azure RBAC to control access to resources

Guidance: Use Azure role-based access control (Azure RBAC) to control access to the Azure Cognitive Services control plane (i.e. Azure portal).

- How to configure Azure RBAC

Azure Security Center monitoring: Not applicable

Responsibility: Customer

4.7: Use host-based data loss prevention to enforce access control

Guidance: Not applicable; this recommendation is intended for compute resources.

Microsoft manages the underlying infrastructure for Azure Cognitive Services and has implemented strict controls to prevent the loss or exposure of customer data.

- Understand customer data protection in Azure

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

4.8: Encrypt sensitive information at rest

Guidance: Encryption at rest for Cognitive Services is dependent on the specific service being used. In most cases, data is encrypted and decrypted using FIPS 140-2 compliant 256-bit AES encryption. Encryption and decryption are transparent, meaning encryption and access are managed for you. Your data is secure by default and you don’t need to modify your code or applications to take advantage of encryption.

You may also use Azure Key Vault to store your customer-managed keys. You can either create your own keys and store them in a key vault, or you can use the Azure Key Vault APIs to generate keys.

- List of services that encrypt information at rest

Azure Security Center monitoring: Currently not available

Responsibility: Customer

4.9: Log and alert on changes to critical Azure resources

Guidance: Use Azure Monitor with the Azure Activity log to create alerts for when changes take place to production instances of Azure Cognitive Services and other critical or related resources.

- How to create alerts for Azure Activity Log events

Azure Security Center monitoring: Yes

Responsibility: Customer

Vulnerability management

For more information, see Security control: Vulnerability management.

5.1: Run automated vulnerability scanning tools

Guidance: Microsoft performs vulnerability management on the underlying systems that support Azure Cognitive Services.

Azure Security Center monitoring: Currently not available

Responsibility: Microsoft

5.2: Deploy automated operating system patch management solution

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

5.3: Deploy automated patch management solution for third-party software titles

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

5.4: Compare back-to-back vulnerability scans

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

5.5: Use a risk-rating process to prioritize the remediation of discovered vulnerabilities

Guidance: Microsoft performs vulnerability management on the underlying systems that support Azure Cognitive Services.

Azure Security Center monitoring: Not applicable

Responsibility: Microsoft

Inventory and asset management

For more information, see Security control: Inventory and asset management.

6.1: Use automated Asset Discovery solution

Guidance: Use Azure Resource Graph to query/discover all resources (such as compute, storage, network, ports, and protocols etc.) within your subscription(s). Ensure appropriate (read) permissions in your tenant and enumerate all Azure subscriptions as well as resources within your subscriptions.

Although classic Azure resources may be discovered via Resource Graph, it is highly recommended to create and use Azure Resource Manager resources going forward.

- How to create queries with Azure Resource Graph
- How to view your Azure Subscriptions
- Understand Azure RBAC

Azure Security Center monitoring: Not applicable

Responsibility: Customer

6.2: Maintain asset metadata

Guidance: Apply tags to Azure resources giving metadata to logically organize them into a taxonomy.

- How to create and use tags

Azure Security Center monitoring: Not applicable

Responsibility: Customer

6.3: Delete unauthorized Azure resources

Guidance: Use tagging, management groups, and separate subscriptions, where appropriate, to organize and track Azure Cache for Redis instances and related resources. Reconcile inventory on a regular basis and ensure unauthorized resources are deleted from the subscription in a timely manner.

In addition, use Azure Policy to put restrictions on the type of resources that can be created in customer subscription(s) using the following built-in policy definitions:

- Not allowed resource types
- Allowed resource types

- How to create additional Azure subscriptions
- How to create management groups
- How to create and use tags

Azure Security Center monitoring: Not applicable

Responsibility: Customer

6.4: Define and Maintain an inventory of approved Azure resources

Guidance: Not applicable; this recommendation is intended for compute resources and Azure as a whole.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

6.5: Monitor for unapproved Azure resources

Guidance: Use Azure Policy to put restrictions on the type of resources that can be created in customer subscription(s) using the following built-in policy definitions:

- Not allowed resource types
- Allowed resource types

In addition, use Azure Resource Graph to query/discover resources within the subscription(s).

- How to configure and manage Azure Policy
- How to create queries with Azure Resource Graph

Azure Security Center monitoring: Not applicable

Responsibility: Customer

6.6: Monitor for unapproved software applications within compute resources

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

6.7: Remove unapproved Azure resources and software applications

Guidance: Not applicable; this recommendation is intended for compute resources and Azure as a whole.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

6.8: Use only approved applications

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

6.9: Use only approved Azure services

Guidance: Use Azure Policy to put restrictions on the type of resources that can be created in customer subscription(s) using the following built-in policy definitions:

- Not allowed resource types
- Allowed resource types

- How to configure and manage Azure Policy

- How to deny a specific resource type with Azure Policy

Azure Security Center monitoring: Not applicable

Responsibility: Customer

6.10: Maintain an inventory of approved software titles

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

6.11: Limit users' ability to interact with Azure Resource Manager

Guidance: Configure Azure Conditional Access to limit users' ability to interact with Azure Resource Manager by configuring "Block access" for the "Microsoft Azure Management" App.

- How to configure Conditional Access to block access to Azure Resource Manager

Azure Security Center monitoring: Not applicable

Responsibility: Customer

6.12: Limit users' ability to execute scripts within compute resources

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

6.13: Physically or logically segregate high risk applications

Guidance: Not applicable; this recommendation is intended for web applications running on Azure App Service or compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

Secure configuration

For more information, see Security control: Secure configuration.

7.1: Establish secure configurations for all Azure resources

Guidance: Define and implement standard security configurations for your Azure Cognitive Services container with Azure Policy. Use Azure Policy aliases in the "Microsoft.CognitiveServices" namespace to create custom policies to audit or enforce the configuration of your Azure Cache for Redis instances.

- How to view available Azure Policy Aliases
- How to configure and manage Azure Policy

Azure Security Center monitoring: Not applicable

Responsibility: Customer

7.2: Establish secure operating system configurations

Guidance: Not applicable; this guideline is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

7.3: Maintain secure Azure resource configurations

Guidance: Use Azure Policy [deny] and [deploy if not exist] to enforce secure settings across your Azure resources.

- How to configure and manage Azure Policy
- Understand Azure Policy Effects

Azure Security Center monitoring: Not applicable

Responsibility: Customer

7.4: Maintain secure operating system configurations

Guidance: Not applicable; this guideline is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

7.5: Securely store configuration of Azure resources

Guidance: If you are using custom Azure Policy definitions or Azure Resource Manager templates for your Azure Cognitive Services containers and related resources, use Azure Repos to securely store and manage your code.

- How to store code in Azure DevOps
- Azure Repos Documentation

Azure Security Center monitoring: Not applicable

Responsibility: Customer

7.6: Securely store custom operating system images

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

7.7: Deploy configuration management tools for Azure resources

Guidance: Use Azure Policy aliases in the "Microsoft.Cache" namespace to create custom policies to alert, audit, and enforce system configurations. Additionally, develop a process and pipeline for managing policy exceptions.

- How to configure and manage Azure Policy

Azure Security Center monitoring: Not applicable

Responsibility: Customer

7.8: Deploy configuration management tools for operating systems

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

7.9: Implement automated configuration monitoring for Azure resources

Guidance: Use Azure Policy aliases in the "Microsoft.CognitiveServices" namespace to create custom Azure Policy definitions to alert, audit, and enforce system configurations. Use Azure Policy [audit], [deny], and [deploy if not exist] to automatically enforce configurations for your Azure Cache for Redis instances and related resources.

- How to configure and manage Azure Policy

Azure Security Center monitoring: Not applicable

Responsibility: Customer

7.10: Implement automated configuration monitoring for operating systems

Guidance: Not applicable; this recommendation is intended for compute resources.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

7.11: Manage Azure secrets securely

Guidance: For Azure virtual machines or web applications running on Azure App Service being used to access your Azure Cognitive Services API, use Managed Service Identity in conjunction with Azure Key Vault to simplify and secure Azure Cognitive Services key management. Ensure Key Vault soft delete is enabled.

- How to integrate with Azure Managed Identities
- How to create a Key Vault
- How to authenticate to Key Vault
- How to assign a Key Vault access policy

Azure Security Center monitoring: Yes

Responsibility: Customer

7.12: Manage identities securely and automatically

Guidance: For Azure virtual machines or web applications running on Azure App Service being used to access your Azure Cognitive Services API, use Managed Service Identity in conjunction with Azure Key Vault to simplify and secure Azure Cognitive Services key management. Ensure Key Vault Soft Delete is enabled.

Use Managed Identities to provide Azure services with an automatically managed identity in Azure Active Directory. Managed Identities allows you to authenticate to any service that supports Azure AD authentication, including Azure Key Vault, without any credentials in your code.

- How to configure Managed Identities
- How to integrate with Azure Managed Identities

Azure Security Center monitoring: Yes

Responsibility: Customer

7.13: Eliminate unintended credential exposure

Guidance: Implement Credential Scanner to identify credentials within code. Credential Scanner will also encourage moving discovered credentials to more secure locations such as Azure Key Vault.

- How to setup Credential Scanner

Azure Security Center monitoring: Not applicable

Responsibility: Customer

Malware defense

For more information, see Security control: Malware defense.

8.1: Use centrally managed anti-malware software

Guidance: Not applicable; this recommendation is intended for compute resources.

Microsoft anti-malware is enabled on the underlying host that supports Azure services (for example, Azure Cognitive Services), however it does not run on customer content.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

8.2: Pre-scan files to be uploaded to non-compute Azure resources

Guidance: Microsoft anti-malware is enabled on the underlying host that supports Azure services (for example, Azure Cache for Redis), however it does not run on customer content.

Pre-scan any content being uploaded to non-compute Azure resources, such as App Service, Data Lake Storage, Blob Storage, Azure Database for PostgreSQL, etc. Microsoft cannot access your data in these instances.

Azure Security Center monitoring: Not applicable

Responsibility: Customer

8.3: Ensure anti-malware software and signatures are updated

Guidance: Not applicable; this recommendation is intended for compute resources.

Microsoft anti-malware is enabled on the underlying host that supports Azure services (for example, Azure Cognitive Services), however it does not run on customer content.

Azure Security Center monitoring: Not applicable

Responsibility: Not applicable

Data recovery

For more information, see Security control: Data recovery.

9.1: Ensure regular automated back ups

Guidance: The data in your Microsoft Azure storage account is always automatically replicated to ensure durability and high availability. Azure Storage copies your data so that it is protected from planned and unplanned events, including transient hardware failures, network or power outages, and massive natural disasters. You can choose to replicate your data within the same data center, across zonal data centers within the same region, or across geographically separated regions.

You can also use lifecycle management feature to backup data to the Archive tier. Additionally, enable soft delete for your backups stored in Storage account.

- Understanding Azure Storage redundancy and Service-Level Agreements
- Manage the Azure Blob storage lifecycle
- Soft delete for Azure Storage blobs

Azure Security Center monitoring: Yes

Responsibility: Customer

9.2: Perform complete system backups and backup any customer-managed keys

Guidance: Use Azure Resource Manager to deploy Cognitive Services and related resources. Azure Resource Manager provides the ability to export templates, which allows you to redeploy your solution throughout the development lifecycle and have confidence your resources are deployed in a consistent state. Use Azure Automation to call the Azure Resource Manager template export API on a regular basis. Backup pre-shared keys within Azure Key Vault.

- Overview of Azure Resource Manager
- How to create a Cognitive Services resource using an Azure Resource Manager template
- Single and multi-resource export to a template in Azure portal
- Resource Groups - Export Template
- Introduction to Azure Automation
- How to backup key vault keys in Azure

Azure Security Center monitoring: Yes

Responsibility: Customer

9.3: Validate all backups including customer-managed keys

Guidance: Ensure ability to periodically perform deployment of Azure Resource Manager templates on a regular basis to an isolated subscription if required. Test restoration of backed up pre-shared keys.

- Deploy resources with ARM templates and Azure portal
- How to restore key vault keys in Azure

Azure Security Center monitoring: Not applicable

Responsibility: Customer

9.4: Ensure protection of backups and customer-managed keys

Guidance: Use Azure DevOps to securely store and manage your Azure Resource Manager templates. To protect resources you manage in Azure DevOps, you can grant or deny permissions to specific users, built-in security groups, or groups defined in Azure Active Directory (Azure AD) if integrated with Azure DevOps, or Active Directory if integrated with TFS. Use role-based access control to protect customer managed keys. Enable Soft-Delete and purge protection in Key Vault to protect keys against accidental or malicious deletion.

- How to store code in Azure DevOps
- About permissions and groups in Azure DevOps
- How to enable Soft-Delete and Purge protection in Key Vault

Azure Security Center monitoring: Yes

Responsibility: Customer

Incident response

For more information, see Security control: Incident response.

10.1: Create an incident response guide

Guidance: Build out an incident response guide for your organization. Ensure that there are written incident response plans that define all roles of personnel as well as phases of incident handling/management from detection to post-incident review.

- How to configure Workflow Automations within Azure Security Center
- Guidance on building your own security incident response process
- Microsoft Security Response Center's Anatomy of an Incident
- You may also leverage NIST's Computer Security Incident Handling Guide to aid in the creation of your own incident response plan

Azure Security Center monitoring: Not applicable

Responsibility: Customer

10.2: Create an incident scoring and prioritization procedure

Guidance: Security Center assigns a severity to each alert to help you prioritize which alerts should be investigated first. The severity is based on how confident Security Center is in the finding or the analytic used to issue the alert as well as the confidence level that there was malicious intent behind the activity that led to the alert.

Additionally, clearly mark subscriptions (for ex. production, non-prod) and create a naming system to clearly identify and categorize Azure resources.

Azure Security Center monitoring: Yes

Responsibility: Customer

10.3: Test security response procedures

Guidance: Conduct exercises to test your systems' incident response capabilities on a regular cadence. Identify weak points and gaps and revise plan as needed.

- Refer to NIST's publication: Guide to Test, Training, and Exercise Programs for IT Plans and Capabilities

Azure Security Center monitoring: Not applicable

Responsibility: Customer

10.4: Provide security incident contact details and configure alert notifications for security incidents

Guidance: Security incident contact information will be used by Microsoft to contact you if the Microsoft Security Response Center (MSRC) discovers that the customer's data has been accessed by an unlawful or unauthorized party. Review incidents after the fact to ensure that issues are resolved.

- How to set the Azure Security Center Security Contact

Azure Security Center monitoring: Yes

Responsibility: Customer

10.5: Incorporate security alerts into your incident response system

Guidance: Export your Azure Security Center alerts and recommendations using the Continuous Export feature.

Continuous Export allows you to export alerts and recommendations either manually or in an ongoing, continuous fashion. You may use the Azure Security Center data connector to stream the alerts Sentinel.

- How to configure continuous export
- How to stream alerts into Azure Sentinel

Azure Security Center monitoring: Not applicable

Responsibility: Customer

10.6: Automate the response to security alerts

Guidance: Use the Workflow Automation feature in Azure Security Center to automatically trigger responses via "Logic Apps" on security alerts and recommendations.

- How to configure Workflow Automation and Logic Apps

Azure Security Center monitoring: Not applicable

Responsibility: Customer

Penetration tests and red team exercises

For more information, see Security control: Penetration tests and red team exercises.

11.1: Conduct regular penetration testing of your Azure resources and ensure remediation of all critical security findings

Guidance: * Follow the Microsoft Rules of Engagement to ensure your Penetration Tests are not in violation of Microsoft policies

- You can find more information on Microsoft’s strategy and execution of Red Teaming and live site penetration testing against Microsoft-managed cloud infrastructure, services, and applications, here

Azure Security Center monitoring: Not applicable

Responsibility: Shared

Next steps

- See the Azure security benchmark
- Learn more about Azure security baselines

Azure Cognitive Services containers

10/4/2020 • 10 minutes to read • Edit Online

WARNING

On June 11, 2020, Microsoft announced that it will not sell facial recognition technology to police departments in the United States until strong regulation, grounded in human rights, has been enacted. As such, customers may not use facial recognition features or functionality included in Azure Services, such as Face or Video Indexer, if a customer is, or is allowing use of such services by or for, a police department in the United States.

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, including parts of:

- Anomaly Detector
- Read OCR (Optical Character Recognition)
- Spatial analysis
- Face
- Form Recognizer
- Language Understanding (LUIS)
- Speech Service API
- Text Analytics

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.

Cognitive Services resources are available on Microsoft Azure. Sign into the Azure portal to create and explore Azure resources for these services.

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 do not 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.

Containers in Azure Cognitive Services

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:

SERVICE	SUPPORTED PRICING TIER	CONTAINER	DESCRIPTION
Anomaly detector	F0, S0	**Anomaly-Detector** (image)	The Anomaly Detector API enables you to monitor and detect abnormalities in your time series data with machine learning. Request access
Computer Vision	F0, S1	**Read** OCR (image)	The Read OCR container allows you to extract printed and handwritten text from images and documents with support for JPEG, PNG, BMP, PDF, and TIFF file formats. For more information, see the Read API documentation. Request access
Face	F0, S0	**Face**	Detects human faces in images, and identifies attributes, including face landmarks (such as noses and eyes), gender, age, and other machine-predicted facial features. In addition to detection, Face can check if two faces in the same image or different images are the same by using a confidence score, or compare faces against a database to see if a similar-looking or identical face already exists. It can also organize similar faces into groups, using shared visual traits.
Form recognizer	F0, S0	**Form Recognizer**	Form Understanding applies machine learning technology to identify and extract key-value pairs and tables from forms.
LUIS	F0, S0	**LUIS** (image)	Loads a trained or published Language Understanding model, also known as a LUIS app, into a docker container and provides access to the query predictions from the container's API endpoints. You can collect query logs from the container and upload these back to the LUIS portal to improve the app's prediction accuracy.

SERVICE	SUPPORTED PRICING TIER	CONTAINER	DESCRIPTION
Speech Service API	F0, S0	**Speech-to-text** (image)	Transcribes continuous real-time speech into text.
Speech Service API	F0, S0	**Custom Speech-to-text** (image)	Transcribes continuous real-time speech into text using a custom model.
Speech Service API	F0, S0	**Text-to-speech** (image)	Converts text to natural-sounding speech.
Speech Service API	F0, S0	**Custom Text-to-speech** (image)	Converts text to natural-sounding speech using a custom model.
Speech Service API	F0, S0	**Neural Text-to-speech** (image)	Converts text to natural-sounding speech using deep neural network technology, allowing for more natural synthesized speech.
Text Analytics	F0, S	**Key Phrase Extraction** (image)	Extracts key phrases to identify the main points. For example, for the input text "The food was delicious and there were wonderful staff", the API returns the main talking points: "food" and "wonderful staff".
Text Analytics	F0, S	**Language Detection** (image)	For up to 120 languages, detects which language the input text is written in and report a single language code for every document submitted on the request. The language code is paired with a score indicating the strength of the score.
Text Analytics	F0, S	**Sentiment Analysis v3** (image)	Analyzes raw text for clues about positive or negative sentiment. This version of sentiment analysis returns sentiment labels (for example *positive* or *negative*) for each document and sentence within it.
Text Analytics	F0, S	**Text Analytics for health**	Extract and label medical information from unstructured clinical text.
Spatial Analysis	S0	**Spatial analysis**	Extract and label medical information from unstructured clinical text.

In addition, some containers are supported in Cognitive Services **All-In-One offering** resource keys. You can create one single Cognitive Services All-In-One resource and use the same billing key across supported services for the following services:

- Computer Vision

- Face
- LUIS
- Text Analytics

Container availability in Azure Cognitive Services

Azure Cognitive Services containers are publicly available through your Azure subscription, and Docker container images can be pulled from either the Microsoft Container Registry or Docker Hub. You can use the docker pull command to download a container image from the appropriate registry.

Container repositories and images

The tables below are a listing of the available container images offered by Azure Cognitive Services. For a complete list of all the available container image names and their available tags, see Cognitive Services container image tags.

Generally available

The Microsoft Container Registry (MCR) syndicates all of the generally available containers for Cognitive Services. The containers are also available directly from the Docker hub.

LUIS

CONTAINER	CONTAINER REGISTRY / REPOSITORY / IMAGE NAME
LUIS	`mcr.microsoft.com/azure-cognitive-services/language/luis`

See How to run and install LUIS containers for more information.

Text Analytics

CONTAINER	CONTAINER REGISTRY / REPOSITORY / IMAGE NAME
Sentiment Analysis v3 (English)	`mcr.microsoft.com/azure-cognitive-services/textanalytics/sentiment:3.0-en`
Sentiment Analysis v3 (Spanish)	`mcr.microsoft.com/azure-cognitive-services/textanalytics/sentiment:3.0-es`
Sentiment Analysis v3 (French)	`mcr.microsoft.com/azure-cognitive-services/textanalytics/sentiment:3.0-fr`
Sentiment Analysis v3 (Italian)	`mcr.microsoft.com/azure-cognitive-services/textanalytics/sentiment:3.0-it`
Sentiment Analysis v3 (German)	`mcr.microsoft.com/azure-cognitive-services/textanalytics/sentiment:3.0-de`
Sentiment Analysis v3 (Chinese - simplified)	`mcr.microsoft.com/azure-cognitive-services/textanalytics/sentiment:3.0-zh`
Sentiment Analysis v3 (Chinese - traditional)	`mcr.microsoft.com/azure-cognitive-services/textanalytics/sentiment:3.0-zht`
Sentiment Analysis v3 (Japanese)	`mcr.microsoft.com/azure-cognitive-services/textanalytics/sentiment:3.0-ja`
Sentiment Analysis v3 (Portuguese)	`mcr.microsoft.com/azure-cognitive-services/textanalytics/sentiment:3.0-pt`
Sentiment Analysis v3 (Dutch)	`mcr.microsoft.com/azure-cognitive-services/textanalytics/sentiment:3.0-nl`

See How to run and install Text Analytics containers for more information.

Anomaly Detector

CONTAINER	CONTAINER REGISTRY / REPOSITORY / IMAGE NAME
Anomaly detector	`mcr.microsoft.com/azure-cognitive-services/decision/anomaly-detector`

See How to run and install Anomaly detector containers for more information.

Speech Service

NOTE

To use Speech containers, you will need to complete an online request form.

CONTAINER	CONTAINER REGISTRY / REPOSITORY / IMAGE NAME
Speech-to-text	`mcr.microsoft.com/azure-cognitive-services/speechservices/speech-to-text`
Custom Speech-to-text	`mcr.microsoft.com/azure-cognitive-services/speechservices/custom-speech-to-text`
Text-to-speech	`mcr.microsoft.com/azure-cognitive-services/speechservices/text-to-speech`

"Ungated" preview

The following preview containers are available publicly. The Microsoft Container Registry (MCR) syndicates all of the publicly available ungated containers for Cognitive Services. The containers are also available directly from the Docker hub.

SERVICE	CONTAINER	CONTAINER REGISTRY / REPOSITORY / IMAGE NAME
Text Analytics	Key Phrase Extraction	`mcr.microsoft.com/azure-cognitive-services/textanalytics/keyphrase`
Text Analytics	Language Detection	`mcr.microsoft.com/azure-cognitive-services/textanalytics/language`

"Gated" preview

Previously, gated preview containers were hosted on the `containerpreview.azurecr.io` repository. Starting September 22nd 2020, these containers (except Text Analytics for health) are hosted on the Microsoft Container Registry (MCR), and downloading them doesn't require using the docker login command. To use the container you will need to:

1. Complete a request form with your Azure Subscription ID and user scenario.
2. Upon approval, download the container from the MCR.
3. Use the key and endpoint from an appropriate Azure resource to authenticate the container at runtime.

SERVICE	CONTAINER	CONTAINER REGISTRY / REPOSITORY / IMAGE NAME
Computer Vision	Read v3.0	`mcr.microsoft.com/azure-cognitive-services/vision/read:3.0-preview`
Computer Vision	Read v3.1	`mcr.microsoft.com/azure-cognitive-services/vision/read:3.1-preview`

SERVICE	CONTAINER	CONTAINER REGISTRY / REPOSITORY / IMAGE NAME
Computer Vision	Spatial Analysis	`mcr.microsoft.com/azure-cognitive-services/vision/spatial-analysis`
Speech Service API	Custom Text-to-speech	`mcr.microsoft.com/azure-cognitive-services/speechservices/custom-text-to-speech`
Speech Service API	Language Detection	`mcr.microsoft.com/azure-cognitive-services/speechservices/language-detection`
Speech Service API	Neural Text-to-speech	`mcr.microsoft.com/azure-cognitive-services/speechservices/neural-text-to-speech`
Text Analytics for health	Text Analytics for health	`containerpreview.azurecr.io/microsoft/cognit services-healthcare`

Prerequisites

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.

For a primer on Docker and container basics, see the Docker overview.

Individual containers can have their own requirements, as well, including server and memory allocation requirements.

Azure Cognitive Services container security

Security should be a primary focus whenever you're developing applications. The importance of security is a metric for success. When you're architecting a software solution that includes Cognitive Services containers, it's vital to understand the limitations and capabilities available to you. For more information about network security, see Configure Azure Cognitive Services virtual networks.

IMPORTANT

By default there is *no security* on the Cognitive Services container API. The reason for this is that most often the container will run as part of a pod which is protected from the outside by a network bridge. However, it is possible to enable authentication which works identically to the authentication used when accessing the cloud-based Cognitive Services.

The diagram below illustrates the default and **non-secure** approach:

As an alternative and *secure* approach, consumers of Cognitive Services containers could augment a container with a front-facing component, keeping the container endpoint private. Let's consider a scenario where we use Istio as an ingress gateway. Istio supports HTTPS/TLS and client-certificate authentication. In this scenario, the Istio frontend exposes the container access, presenting the client certificate that is approved beforehand with Istio.

Nginx is another popular choice in the same category. Both Istio and Nginx act as a service mesh and offer additional features including things like load-balancing, routing, and rate-control.

Container networking

The Cognitive Services containers are required to submit metering information for billing purposes. The only exception, is *Offline containers* as they follow a different billing methodology. Failure to allow list various network channels that the Cognitive Services containers rely on will prevent the container from working.

Allow list Cognitive Services domains and ports

The host should allow list `port 443` and the following domains:

- `*.cognitive.microsoft.com`
- `*.cognitiveservices.azure.com`

Disable deep packet inspection

> Deep packet inspection (DPI) is a type of data processing that inspects in detail the data being sent over a computer network, and usually takes action by blocking, re-routing, or logging it accordingly.

Disable DPI on the secure channels that the Cognitive Services containers create to Microsoft servers. Failure to do so will prevent the container from functioning correctly.

Blog posts

- Running Cognitive Services Containers
- Azure Cognitive Services

Developer samples

Developer samples are available at our GitHub repository.

View webinar

Join the webinar to learn about:

- How to deploy Cognitive Services to any machine using Docker
- How to deploy Cognitive Services to AKS

Next steps

Learn about container recipes you can use with the Cognitive Services.

Install and explore the functionality provided by containers in Azure Cognitive Services:

- Anomaly Detector containers
- Computer Vision containers
- Face containers
- Form Recognizer containers
- Language Understanding (LUIS) containers
- Speech Service API containers
- Text Analytics containers

Azure Cognitive Services container image tags

10/4/2020 • 46 minutes to read • Edit Online

Azure Cognitive Services offers many container images. The container registries and corresponding repositories vary between container images. Each container image name offers multiple tags. A container image tag is a mechanism of versioning the container image. This article is intended to be used as a comprehensive reference for listing all the Cognitive Services container images and their available tags.

> **TIP**
> When using `docker pull`, pay close attention to the casing of the container registry, repository, container image name and corresponding tag - as they are **case sensitive**.

Anomaly Detector

The Anomaly Detector container image can be found on the `mcr.microsoft.com` container registry syndicate. It resides within the `azure-cognitive-services` repository and is named `anomaly-detector`. The fully qualified container image name is, `mcr.microsoft.com/azure-cognitive-services/anomaly-detector`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	

Computer Vision

The Computer Vision Read OCR container image can be found on the `containerpreview.azurecr.io` container registry. It resides within the `microsoft` repository and is named `cognitive-services-read`. The fully qualified container image name is, `containerpreview.azurecr.io/microsoft/cognitive-services-read`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest ( (2.0.013250001-amd64-preview)`	• Further decrease memory usage for container.
	• External cache is required for multi-pods setup. For example, set-up Redis for caching.
	• Fix results missing issue when Redis cache is set-up and ResultExpirationPeriod=0.
	• Remove request body size limitation of 26MB. Container can now accept >26MB files.
	• Add time stamp and build version to console logging.

IMAGE TAGS	NOTES
`1.1.013050001-amd64-preview`	* Added ReadEngineConfig:ResultExpirationPeriod container initialization configuration to specify when the system should clean up recognition results.
	The setting is in hours, and default value is 48hr.
	The setting can reduce memory usage for result storing, especially when container in-memory storage is used.
	* Example 1. ReadEngineConfig:ResultExpirationPeriod=1, the system will clear the recognition result 1hr after the process.
	* Example 2. ReadEngineConfig:ResultExpirationPeriod=0, the system will clear the recognition result after result retrieval.
	Fixed an 500 Internal Server Error when invalid image format is passed into the system. It will now return a 400 error:
	`{`
	`"error": {`
	`"code": "InvalidImageSize",`
	`"message": "Image must be between 1024 and 209715200 bytes."`
	`}`
	`}`
`1.1.011580001-amd64-preview`	
`1.1.009920003-amd64-preview`	
`1.1.009910003-amd64-preview`	

Face

The Face container image can be found on the `containerpreview.azurecr.io` container registry. It resides within the `microsoft` repository and is named `cognitive-services-face`. The fully qualified container image name is, `containerpreview.azurecr.io/microsoft/cognitive-services-face`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	
`1.1.009301-amd64-preview`	

IMAGE TAGS	NOTES
`1.1.008710001-amd64-preview`	
`1.1.007750002-amd64-preview`	
`1.1.007360001-amd64-preview`	
`1.1.006770001-amd64-preview`	
`1.1.006490002-amd64-preview`	
`1.0.005940002-amd64-preview`	
`1.0.005550001-amd64-preview`	

Form Recognizer

The Form Recognizer container image can be found on the `containerpreview.azurecr.io` container registry. It resides within the `microsoft` repository and is named `cognitive-services-form-recognizer`. The fully qualified container image name is, `containerpreview.azurecr.io/microsoft/cognitive-services-form-recognizer`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	
`1.1.009301-amd64-preview`	
`1.1.008640001-amd64-preview`	
`1.1.008510001-amd64-preview`	

Language Understanding (LUIS)

The LUIS container image can be found on the `mcr.microsoft.com` container registry syndicate. It resides within the `azure-cognitive-services` repository and is named `luis`. The fully qualified container image name is, `mcr.microsoft.com/azure-cognitive-services/luis`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	
`1.1.010330004-amd64-preview`	
`1.1.009301-amd64-preview`	
`1.1.008710001-amd64-preview`	

IMAGE TAGS	NOTES
`1.1.008510001-amd64-preview`	
`1.1.008010002-amd64-preview`	
`1.1.007750002-amd64-preview`	
`1.1.007360001-amd64-preview`	
`1.1.007020001-amd64-preview`	

Custom Speech-to-text

The Custom Speech-to-text container image can be found on the `containerpreview.azurecr.io` container registry. It resides within the `microsoft` repository and is named `cognitive-services-custom-speech-to-text`. The fully qualified container image name is, `containerpreview.azurecr.io/microsoft/cognitive-services-custom-speech-to-text`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	
`2.5.0-amd64`	
`2.4.0-amd64-preview`	
`2.3.1-amd64-preview`	
`2.3.0-amd64-preview`	
`2.2.0-amd64-preview`	
`2.1.1-amd64-preview`	
`2.1.0-amd64-preview`	
`2.0.2-amd64-preview`	
`2.0.0-amd64-preview`	

Custom Text-to-speech

The Custom Text-to-speech container image can be found on the `containerpreview.azurecr.io` container registry. It resides within the `microsoft` repository and is named `cognitive-services-custom-text-to-speech`. The fully qualified container image name is, `containerpreview.azurecr.io/microsoft/cognitive-services-custom-text-to-speech`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	
`1.7.0-amd64`	
`1.6.0-amd64-preview`	
`1.6.0-amd64-preview`	
`1.5.0-amd64-preview`	
`1.4.0-amd64-preview`	
`1.3.0-amd64-preview`	

Speech-to-text

The Speech-to-text container image can be found on the `containerpreview.azurecr.io` container registry. It resides within the `microsoft` repository and is named `cognitive-services-speech-to-text`. The fully qualified container image name is, `containerpreview.azurecr.io/microsoft/cognitive-services-speech-to-text`. Speech-to-text v2.5.0 images are supported in *US Government Virginia*. Please use *US Government Virginia* billing endpoint and api keys to try.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	Container image with the `en-US` locale.
`2.5.0-amd64-ar-ae`	Container image with the `ar-AE` locale.
`2.5.0-amd64-ar-eg`	Container image with the `ar-EG` locale.
`2.5.0-amd64-ar-kw`	Container image with the `ar-KW` locale.
`2.5.0-amd64-ar-qa`	Container image with the `ar-QA` locale.
`2.5.0-amd64-ar-sa`	Container image with the `ar-SA` locale.
`2.5.0-amd64-ca-es`	Container image with the `ca-ES` locale.
`2.5.0-amd64-da-dk`	Container image with the `da-DK` locale.
`2.5.0-amd64-de-de`	Container image with the `de-DE` locale.
`2.5.0-amd64-en-au`	Container image with the `en-AU` locale.
`2.5.0-amd64-en-ca`	Container image with the `en-CA` locale.
`2.5.0-amd64-en-gb`	Container image with the `en-GB` locale.

IMAGE TAGS	NOTES
`2.5.0-amd64-en-in`	Container image with the `en-IN` locale.
`2.5.0-amd64-en-nz`	Container image with the `en-NZ` locale.
`2.5.0-amd64-en-us`	Container image with the `en-US` locale.
`2.5.0-amd64-es-es`	Container image with the `es-ES` locale.
`2.5.0-amd64-es-mx`	Container image with the `es-MX` locale.
`2.5.0-amd64-fi-fi`	Container image with the `fi-FI` locale.
`2.5.0-amd64-fr-ca`	Container image with the `fr-CA` locale.
`2.5.0-amd64-fr-fr`	Container image with the `fr-FR` locale.
`2.5.0-amd64-gu-in`	Container image with the `gu-IN` locale.
`2.5.0-amd64-hi-in`	Container image with the `hi-IN` locale.
`2.5.0-amd64-it-it`	Container image with the `it-IT` locale.
`2.5.0-amd64-ja-jp`	Container image with the `ja-JP` locale.
`2.5.0-amd64-ko-kr`	Container image with the `ko-KR` locale.
`2.5.0-amd64-mr-in`	Container image with the `mr-IN` locale.
`2.5.0-amd64-nb-no`	Container image with the `nb-NO` locale.
`2.5.0-amd64-nl-nl`	Container image with the `nl-NL` locale.
`2.5.0-amd64-pl-pl`	Container image with the `pl-PL` locale.
`2.5.0-amd64-pt-br`	Container image with the `pt-BR` locale.
`2.5.0-amd64-pt-pt`	Container image with the `pt-PT` locale.
`2.5.0-amd64-ru-ru`	Container image with the `ru-RU` locale.
`2.5.0-amd64-sv-se`	Container image with the `sv-SE` locale.
`2.5.0-amd64-ta-in`	Container image with the `ta-IN` locale.
`2.5.0-amd64-te-in`	Container image with the `te-IN` locale.

IMAGE TAGS	NOTES
`2.5.0-amd64-th-th`	Container image with the `th-TH` locale.
`2.5.0-amd64-tr-tr`	Container image with the `tr-TR` locale.
`2.5.0-amd64-zh-cn`	Container image with the `zh-CN` locale.
`2.5.0-amd64-zh-hk`	Container image with the `zh-HK` locale.
`2.5.0-amd64-zh-tw`	Container image with the `zh-TW` locale.
`2.4.0-amd64-ar-ae-preview`	Container image with the `ar-AE` locale.
`2.4.0-amd64-ar-eg-preview`	Container image with the `ar-EG` locale.
`2.4.0-amd64-ar-kw-preview`	Container image with the `ar-KW` locale.
`2.4.0-amd64-ar-qa-preview`	Container image with the `ar-QA` locale.
`2.4.0-amd64-ar-sa-preview`	Container image with the `ar-SA` locale.
`2.4.0-amd64-ca-es-preview`	Container image with the `ca-ES` locale.
`2.4.0-amd64-da-dk-preview`	Container image with the `da-DK` locale.
`2.4.0-amd64-de-de-preview`	Container image with the `de-DE` locale.
`2.4.0-amd64-en-au-preview`	Container image with the `en-AU` locale.
`2.4.0-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`2.4.0-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`2.4.0-amd64-en-in-preview`	Container image with the `en-IN` locale.
`2.4.0-amd64-en-nz-preview`	Container image with the `en-NZ` locale.
`2.4.0-amd64-en-us-preview`	Container image with the `en-US` locale.
`2.4.0-amd64-es-es-preview`	Container image with the `es-ES` locale.
`2.4.0-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`2.4.0-amd64-fi-fi-preview`	Container image with the `fi-FI` locale.
`2.4.0-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`2.4.0-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.

IMAGE TAGS	NOTES
`2.4.0-amd64-gu-in-preview`	Container image with the `gu-IN` locale.
`2.4.0-amd64-hi-in-preview`	Container image with the `hi-IN` locale.
`2.4.0-amd64-it-it-preview`	Container image with the `it-IT` locale.
`2.4.0-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`2.4.0-amd64-ko-kr-preview`	Container image with the `ko-KR` locale.
`2.4.0-amd64-mr-in-preview`	Container image with the `mr-IN` locale.
`2.4.0-amd64-nb-no-preview`	Container image with the `nb-NO` locale.
`2.4.0-amd64-nl-nl-preview`	Container image with the `nl-NL` locale.
`2.4.0-amd64-pl-pl-preview`	Container image with the `pl-PL` locale.
`2.4.0-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`2.4.0-amd64-pt-pt-preview`	Container image with the `pt-PT` locale.
`2.4.0-amd64-ru-ru-preview`	Container image with the `ru-RU` locale.
`2.4.0-amd64-sv-se-preview`	Container image with the `sv-SE` locale.
`2.4.0-amd64-ta-in-preview`	Container image with the `ta-IN` locale.
`2.4.0-amd64-te-in-preview`	Container image with the `te-IN` locale.
`2.4.0-amd64-th-th-preview`	Container image with the `th-TH` locale.
`2.4.0-amd64-tr-tr-preview`	Container image with the `tr-TR` locale.
`2.4.0-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`2.4.0-amd64-zh-hk-preview`	Container image with the `zh-HK` locale.
`2.4.0-amd64-zh-tw-preview`	Container image with the `zh-TW` locale.
`2.3.1-amd64-ar-ae-preview`	Container image with the `ar-AE` locale.
`2.3.1-amd64-ar-eg-preview`	Container image with the `ar-EG` locale.
`2.3.1-amd64-ar-kw-preview`	Container image with the `ar-KW` locale.
`2.3.1-amd64-ar-qa-preview`	Container image with the `ar-QA` locale.

IMAGE TAGS	NOTES
`2.3.1-amd64-ar-sa-preview`	Container image with the `ar-SA` locale.
`2.3.1-amd64-ca-es-preview`	Container image with the `ca-ES` locale.
`2.3.1-amd64-da-dk-preview`	Container image with the `da-DK` locale.
`2.3.1-amd64-de-de-preview`	Container image with the `de-DE` locale.
`2.3.1-amd64-en-au-preview`	Container image with the `en-AU` locale.
`2.3.1-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`2.3.1-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`2.3.1-amd64-en-in-preview`	Container image with the `en-IN` locale.
`2.3.1-amd64-en-nz-preview`	Container image with the `en-NZ` locale.
`2.3.1-amd64-en-us-preview`	Container image with the `en-US` locale.
`2.3.1-amd64-es-es-preview`	Container image with the `es-ES` locale.
`2.3.1-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`2.3.1-amd64-fi-fi-preview`	Container image with the `fi-FI` locale.
`2.3.1-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`2.3.1-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`2.3.1-amd64-gu-in-preview`	Container image with the `gu-IN` locale.
`2.3.1-amd64-hi-in-preview`	Container image with the `hi-IN` locale.
`2.3.1-amd64-it-it-preview`	Container image with the `it-IT` locale.
`2.3.1-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`2.3.1-amd64-ko-kr-preview`	Container image with the `ko-KR` locale.
`2.3.1-amd64-mr-in-preview`	Container image with the `mr-IN` locale.
`2.3.1-amd64-nb-no-preview`	Container image with the `nb-NO` locale.
`2.3.1-amd64-nl-nl-preview`	Container image with the `nl-NL` locale.
`2.3.1-amd64-pl-pl-preview`	Container image with the `pl-PL` locale.

IMAGE TAGS	NOTES
`2.3.1-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`2.3.1-amd64-pt-pt-preview`	Container image with the `pt-PT` locale.
`2.3.1-amd64-ru-ru-preview`	Container image with the `ru-RU` locale.
`2.3.1-amd64-sv-se-preview`	Container image with the `sv-SE` locale.
`2.3.1-amd64-ta-in-preview`	Container image with the `ta-IN` locale.
`2.3.1-amd64-te-in-preview`	Container image with the `te-IN` locale.
`2.3.1-amd64-th-th-preview`	Container image with the `th-TH` locale.
`2.3.1-amd64-tr-tr-preview`	Container image with the `tr-TR` locale.
`2.3.1-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`2.3.1-amd64-zh-hk-preview`	Container image with the `zh-HK` locale.
`2.3.1-amd64-zh-tw-preview`	Container image with the `zh-TW` locale.
`2.3.0-amd64-ar-ae-preview`	Container image with the `ar-AE` locale.
`2.3.0-amd64-ar-eg-preview`	Container image with the `ar-EG` locale.
`2.3.0-amd64-ar-kw-preview`	Container image with the `ar-KW` locale.
`2.3.0-amd64-ar-qa-preview`	Container image with the `ar-QA` locale.
`2.3.0-amd64-ar-sa-preview`	Container image with the `ar-SA` locale.
`2.3.0-amd64-ca-es-preview`	Container image with the `ca-ES` locale.
`2.3.0-amd64-da-dk-preview`	Container image with the `da-DK` locale.
`2.3.0-amd64-de-de-preview`	Container image with the `de-DE` locale.
`2.3.0-amd64-en-au-preview`	Container image with the `en-AU` locale.
`2.3.0-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`2.3.0-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`2.3.0-amd64-en-in-preview`	Container image with the `en-IN` locale.
`2.3.0-amd64-en-nz-preview`	Container image with the `en-NZ` locale.

IMAGE TAGS	NOTES
`2.3.0-amd64-en-us-preview`	Container image with the `en-US` locale.
`2.3.0-amd64-es-es-preview`	Container image with the `es-ES` locale.
`2.3.0-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`2.3.0-amd64-fi-fi-preview`	Container image with the `fi-FI` locale.
`2.3.0-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`2.3.0-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`2.3.0-amd64-gu-in-preview`	Container image with the `gu-IN` locale.
`2.3.0-amd64-hi-in-preview`	Container image with the `hi-IN` locale.
`2.3.0-amd64-it-it-preview`	Container image with the `it-IT` locale.
`2.3.0-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`2.3.0-amd64-ko-kr-preview`	Container image with the `ko-KR` locale.
`2.3.0-amd64-mr-in-preview`	Container image with the `mr-IN` locale.
`2.3.0-amd64-nb-no-preview`	Container image with the `nb-NO` locale.
`2.3.0-amd64-nl-nl-preview`	Container image with the `nl-NL` locale.
`2.3.0-amd64-pl-pl-preview`	Container image with the `pl-PL` locale.
`2.3.0-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`2.3.0-amd64-pt-pt-preview`	Container image with the `pt-PT` locale.
`2.3.0-amd64-ru-ru-preview`	Container image with the `ru-RU` locale.
`2.3.0-amd64-sv-se-preview`	Container image with the `sv-SE` locale.
`2.3.0-amd64-ta-in-preview`	Container image with the `ta-IN` locale.
`2.3.0-amd64-te-in-preview`	Container image with the `te-IN` locale.
`2.3.0-amd64-th-th-preview`	Container image with the `th-TH` locale.
`2.3.0-amd64-tr-tr-preview`	Container image with the `tr-TR` locale.
`2.3.0-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.

IMAGE TAGS	NOTES
`2.3.0-amd64-zh-hk-preview`	Container image with the `zh-HK` locale.
`2.3.0-amd64-zh-tw-preview`	Container image with the `zh-TW` locale.
`2.2.0-amd64-ar-ae-preview`	Container image with the `ar-AE` locale.
`2.2.0-amd64-ar-eg-preview`	Container image with the `ar-EG` locale.
`2.2.0-amd64-ar-kw-preview`	Container image with the `ar-KW` locale.
`2.2.0-amd64-ar-qa-preview`	Container image with the `ar-QA` locale.
`2.2.0-amd64-ar-sa-preview`	Container image with the `ar-SA` locale.
`2.2.0-amd64-ca-es-preview`	Container image with the `ca-ES` locale.
`2.2.0-amd64-da-dk-preview`	Container image with the `da-DK` locale.
`2.2.0-amd64-de-de-preview`	Container image with the `de-DE` locale.
`2.2.0-amd64-en-au-preview`	Container image with the `en-AU` locale.
`2.2.0-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`2.2.0-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`2.2.0-amd64-en-in-preview`	Container image with the `en-IN` locale.
`2.2.0-amd64-en-nz-preview`	Container image with the `en-NZ` locale.
`2.2.0-amd64-en-us-preview`	Container image with the `en-US` locale.
`2.2.0-amd64-es-es-preview`	Container image with the `es-ES` locale.
`2.2.0-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`2.2.0-amd64-fi-fi-preview`	Container image with the `fi-FI` locale.
`2.2.0-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`2.2.0-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`2.2.0-amd64-gu-in-preview`	Container image with the `gu-IN` locale.
`2.2.0-amd64-hi-in-preview`	Container image with the `hi-IN` locale.
`2.2.0-amd64-it-it-preview`	Container image with the `it-IT` locale.

IMAGE TAGS	NOTES
`2.2.0-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`2.2.0-amd64-ko-kr-preview`	Container image with the `ko-KR` locale.
`2.2.0-amd64-mr-in-preview`	Container image with the `mr-IN` locale.
`2.2.0-amd64-nb-no-preview`	Container image with the `nb-NO` locale.
`2.2.0-amd64-nl-nl-preview`	Container image with the `nl-NL` locale.
`2.2.0-amd64-pl-pl-preview`	Container image with the `pl-PL` locale.
`2.2.0-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`2.2.0-amd64-pt-pt-preview`	Container image with the `pt-PT` locale.
`2.2.0-amd64-ru-ru-preview`	Container image with the `ru-RU` locale.
`2.2.0-amd64-sv-se-preview`	Container image with the `sv-SE` locale.
`2.2.0-amd64-ta-in-preview`	Container image with the `ta-IN` locale.
`2.2.0-amd64-te-in-preview`	Container image with the `te-IN` locale.
`2.2.0-amd64-th-th-preview`	Container image with the `th-TH` locale.
`2.2.0-amd64-tr-tr-preview`	Container image with the `tr-TR` locale.
`2.2.0-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`2.2.0-amd64-zh-hk-preview`	Container image with the `zh-HK` locale.
`2.2.0-amd64-zh-tw-preview`	Container image with the `zh-TW` locale.
`2.1.1-amd64-en-us-preview`	Container image with the `en-US` locale.
`2.1.1-amd64-ar-ae-preview`	Container image with the `ar-AE` locale.
`2.1.1-amd64-ar-eg-preview`	Container image with the `ar-EG` locale.
`2.1.1-amd64-ar-kw-preview`	Container image with the `ar-KW` locale.
`2.1.1-amd64-ar-qa-preview`	Container image with the `ar-QA` locale.
`2.1.1-amd64-ar-sa-preview`	Container image with the `ar-SA` locale.
`2.1.1-amd64-ca-es-preview`	Container image with the `ca-ES` locale.

IMAGE TAGS	NOTES
`2.1.1-amd64-da-dk-preview`	Container image with the `da-DK` locale.
`2.1.1-amd64-de-de-preview`	Container image with the `de-DE` locale.
`2.1.1-amd64-en-au-preview`	Container image with the `en-AU` locale.
`2.1.1-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`2.1.1-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`2.1.1-amd64-en-in-preview`	Container image with the `en-IN` locale.
`2.1.1-amd64-en-nz-preview`	Container image with the `en-NZ` locale.
`2.1.1-amd64-en-us-preview`	Container image with the `en-US` locale.
`2.1.1-amd64-es-es-preview`	Container image with the `es-ES` locale.
`2.1.1-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`2.1.1-amd64-fi-fi-preview`	Container image with the `fi-FI` locale.
`2.1.1-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`2.1.1-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`2.1.1-amd64-gu-in-preview`	Container image with the `gu-IN` locale.
`2.1.1-amd64-hi-in-preview`	Container image with the `hi-IN` locale.
`2.1.1-amd64-it-it-preview`	Container image with the `it-IT` locale.
`2.1.1-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`2.1.1-amd64-ko-kr-preview`	Container image with the `ko-KR` locale.
`2.1.1-amd64-mr-in-preview`	Container image with the `mr-IN` locale.
`2.1.1-amd64-nb-no-preview`	Container image with the `nb-NO` locale.
`2.1.1-amd64-nl-nl-preview`	Container image with the `nl-NL` locale.
`2.1.1-amd64-pl-pl-preview`	Container image with the `pl-PL` locale.
`2.1.1-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`2.1.1-amd64-pt-pt-preview`	Container image with the `pt-PT` locale.

IMAGE TAGS	NOTES
`2.1.1-amd64-ru-ru-preview`	Container image with the `ru-RU` locale.
`2.1.1-amd64-sv-se-preview`	Container image with the `sv-SE` locale.
`2.1.1-amd64-ta-in-preview`	Container image with the `ta-IN` locale.
`2.1.1-amd64-te-in-preview`	Container image with the `te-IN` locale.
`2.1.1-amd64-th-th-preview`	Container image with the `th-TH` locale.
`2.1.1-amd64-tr-tr-preview`	Container image with the `tr-TR` locale.
`2.1.1-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`2.1.1-amd64-zh-hk-preview`	Container image with the `zh-HK` locale.
`2.1.1-amd64-zh-tw-preview`	Container image with the `zh-TW` locale.
`2.1.0-amd64-ar-ae-preview`	Container image with the `ar-AE` locale.
`2.1.0-amd64-ar-eg-preview`	Container image with the `ar-EG` locale.
`2.1.0-amd64-ar-kw-preview`	Container image with the `ar-KW` locale.
`2.1.0-amd64-ar-qa-preview`	Container image with the `ar-QA` locale.
`2.1.0-amd64-ar-sa-preview`	Container image with the `ar-SA` locale.
`2.1.0-amd64-ca-es-preview`	Container image with the `ca-ES` locale.
`2.1.0-amd64-da-dk-preview`	Container image with the `da-DK` locale.
`2.1.0-amd64-de-de-preview`	Container image with the `de-DE` locale.
`2.1.0-amd64-en-au-preview`	Container image with the `en-AU` locale.
`2.1.0-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`2.1.0-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`2.1.0-amd64-en-in-preview`	Container image with the `en-IN` locale.
`2.1.0-amd64-en-nz-preview`	Container image with the `en-NZ` locale.
`2.1.0-amd64-en-us-preview`	Container image with the `en-US` locale.
`2.1.0-amd64-es-es-preview`	Container image with the `es-ES` locale.

IMAGE TAGS	NOTES
`2.1.0-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`2.1.0-amd64-fi-fi-preview`	Container image with the `fi-FI` locale.
`2.1.0-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`2.1.0-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`2.1.0-amd64-gu-in-preview`	Container image with the `gu-IN` locale.
`2.1.0-amd64-hi-in-preview`	Container image with the `hi-IN` locale.
`2.1.0-amd64-it-it-preview`	Container image with the `it-IT` locale.
`2.1.0-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`2.1.0-amd64-ko-kr-preview`	Container image with the `ko-KR` locale.
`2.1.0-amd64-mr-in-preview`	Container image with the `mr-IN` locale.
`2.1.0-amd64-nb-no-preview`	Container image with the `nb-NO` locale.
`2.1.0-amd64-nl-nl-preview`	Container image with the `nl-NL` locale.
`2.1.0-amd64-pl-pl-preview`	Container image with the `pl-PL` locale.
`2.1.0-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`2.1.0-amd64-pt-pt-preview`	Container image with the `pt-PT` locale.
`2.1.0-amd64-ru-ru-preview`	Container image with the `ru-RU` locale.
`2.1.0-amd64-sv-se-preview`	Container image with the `sv-SE` locale.
`2.1.0-amd64-ta-in-preview`	Container image with the `ta-IN` locale.
`2.1.0-amd64-te-in-preview`	Container image with the `te-IN` locale.
`2.1.0-amd64-th-th-preview`	Container image with the `th-TH` locale.
`2.1.0-amd64-tr-tr-preview`	Container image with the `tr-TR` locale.
`2.1.0-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`2.1.0-amd64-zh-hk-preview`	Container image with the `zh-HK` locale.
`2.1.0-amd64-zh-tw-preview`	Container image with the `zh-TW` locale.

IMAGE TAGS	NOTES
`2.0.3-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`2.0.2-amd64-ar-ae-preview`	Container image with the `ar-AE` locale.
`2.0.2-amd64-ar-eg-preview`	Container image with the `ar-EG` locale.
`2.0.2-amd64-ar-kw-preview`	Container image with the `ar-KW` locale.
`2.0.2-amd64-ar-qa-preview`	Container image with the `ar-QA` locale.
`2.0.2-amd64-ar-sa-preview`	Container image with the `ar-SA` locale.
`2.0.2-amd64-ca-es-preview`	Container image with the `ca-ES` locale.
`2.0.2-amd64-da-dk-preview`	Container image with the `da-DK` locale.
`2.0.2-amd64-de-de-preview`	Container image with the `de-DE` locale.
`2.0.2-amd64-en-au-preview`	Container image with the `en-AU` locale.
`2.0.2-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`2.0.2-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`2.0.2-amd64-en-in-preview`	Container image with the `en-IN` locale.
`2.0.2-amd64-en-nz-preview`	Container image with the `en-NZ` locale.
`2.0.2-amd64-en-us-preview`	Container image with the `en-US` locale.
`2.0.2-amd64-es-es-preview`	Container image with the `es-ES` locale.
`2.0.2-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`2.0.2-amd64-fi-fi-preview`	Container image with the `fi-FI` locale.
`2.0.2-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`2.0.2-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`2.0.2-amd64-gu-in-preview`	Container image with the `gu-IN` locale.
`2.0.2-amd64-hi-in-preview`	Container image with the `hi-IN` locale.
`2.0.2-amd64-it-it-preview`	Container image with the `it-IT` locale.
`2.0.2-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.

IMAGE TAGS	NOTES
`2.0.2-amd64-ko-kr-preview`	Container image with the `ko-KR` locale.
`2.0.2-amd64-mr-in-preview`	Container image with the `mr-IN` locale.
`2.0.2-amd64-nb-no-preview`	Container image with the `nb-NO` locale.
`2.0.2-amd64-nl-nl-preview`	Container image with the `nl-NL` locale.
`2.0.2-amd64-pl-pl-preview`	Container image with the `pl-PL` locale.
`2.0.2-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`2.0.2-amd64-pt-pt-preview`	Container image with the `pt-PT` locale.
`2.0.2-amd64-ru-ru-preview`	Container image with the `ru-RU` locale.
`2.0.2-amd64-sv-se-preview`	Container image with the `sv-SE` locale.
`2.0.2-amd64-ta-in-preview`	Container image with the `ta-IN` locale.
`2.0.2-amd64-te-in-preview`	Container image with the `te-IN` locale.
`2.0.2-amd64-th-th-preview`	Container image with the `th-TH` locale.
`2.0.2-amd64-tr-tr-preview`	Container image with the `tr-TR` locale.
`2.0.2-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`2.0.2-amd64-zh-hk-preview`	Container image with the `zh-HK` locale.
`2.0.2-amd64-zh-tw-preview`	Container image with the `zh-TW` locale.
`2.0.1-amd64-ar-ae-preview`	Container image with the `ar-AE` locale.
`2.0.1-amd64-ar-eg-preview`	Container image with the `ar-EG` locale.
`2.0.1-amd64-ar-kw-preview`	Container image with the `ar-KW` locale.
`2.0.1-amd64-ar-qa-preview`	Container image with the `ar-QA` locale.
`2.0.1-amd64-ar-sa-preview`	Container image with the `ar-SA` locale.
`2.0.1-amd64-ca-es-preview`	Container image with the `ca-ES` locale.
`2.0.1-amd64-da-dk-preview`	Container image with the `da-DK` locale.
`2.0.1-amd64-de-de-preview`	Container image with the `de-DE` locale.

IMAGE TAGS	NOTES
`2.0.1-amd64-en-au-preview`	Container image with the `en-AU` locale.
`2.0.1-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`2.0.1-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`2.0.1-amd64-en-in-preview`	Container image with the `en-IN` locale.
`2.0.1-amd64-en-nz-preview`	Container image with the `en-NZ` locale.
`2.0.1-amd64-en-us-preview`	Container image with the `en-US` locale.
`2.0.1-amd64-es-es-preview`	Container image with the `es-ES` locale.
`2.0.1-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`2.0.1-amd64-fi-fi-preview`	Container image with the `fi-FI` locale.
`2.0.1-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`2.0.1-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`2.0.1-amd64-gu-in-preview`	Container image with the `gu-IN` locale.
`2.0.1-amd64-hi-in-preview`	Container image with the `hi-IN` locale.
`2.0.1-amd64-it-it-preview`	Container image with the `it-IT` locale.
`2.0.1-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`2.0.1-amd64-ko-kr-preview`	Container image with the `ko-KR` locale.
`2.0.1-amd64-mr-in-preview`	Container image with the `mr-IN` locale.
`2.0.1-amd64-nb-no-preview`	Container image with the `nb-NO` locale.
`2.0.1-amd64-nl-nl-preview`	Container image with the `nl-NL` locale.
`2.0.1-amd64-pl-pl-preview`	Container image with the `pl-PL` locale.
`2.0.1-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`2.0.1-amd64-pt-pt-preview`	Container image with the `pt-PT` locale.
`2.0.1-amd64-ru-ru-preview`	Container image with the `ru-RU` locale.
`2.0.1-amd64-sv-se-preview`	Container image with the `sv-SE` locale.

IMAGE TAGS	NOTES
`2.0.1-amd64-ta-in-preview`	Container image with the `ta-IN` locale.
`2.0.1-amd64-te-in-preview`	Container image with the `te-IN` locale.
`2.0.1-amd64-th-th-preview`	Container image with the `th-TH` locale.
`2.0.1-amd64-tr-tr-preview`	Container image with the `tr-TR` locale.
`2.0.1-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`2.0.1-amd64-zh-hk-preview`	Container image with the `zh-HK` locale.
`2.0.1-amd64-zh-tw-preview`	Container image with the `zh-TW` locale.
`2.0.0-amd64-ar-eg-preview`	Container image with the `ar-EG` locale.
`2.0.0-amd64-ca-es-preview`	Container image with the `ca-ES` locale.
`2.0.0-amd64-da-dk-preview`	Container image with the `da-DK` locale.
`2.0.0-amd64-de-de-preview`	Container image with the `de-DE` locale.
`2.0.0-amd64-en-au-preview`	Container image with the `en-AU` locale.
`2.0.0-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`2.0.0-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`2.0.0-amd64-en-in-preview`	Container image with the `en-IN` locale.
`2.0.0-amd64-en-nz-preview`	Container image with the `en-NZ` locale.
`2.0.0-amd64-en-us-preview`	Container image with the `en-US` locale.
`2.0.0-amd64-es-es-preview`	Container image with the `es-ES` locale.
`2.0.0-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`2.0.0-amd64-fi-fi-preview`	Container image with the `fi-FI` locale.
`2.0.0-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`2.0.0-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`2.0.0-amd64-hi-in-preview`	Container image with the `hi-IN` locale.
`2.0.0-amd64-it-it-preview`	Container image with the `it-IT` locale.

IMAGE TAGS	NOTES
`2.0.0-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`2.0.0-amd64-ko-kr-preview`	Container image with the `ko-KR` locale.
`2.0.0-amd64-nb-no-preview`	Container image with the `nb-NO` locale.
`2.0.0-amd64-nl-nl-preview`	Container image with the `nl-NL` locale.
`2.0.0-amd64-pl-pl-preview`	Container image with the `pl-PL` locale.
`2.0.0-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`2.0.0-amd64-pt-pt-preview`	Container image with the `pt-PT` locale.
`2.0.0-amd64-ru-ru-preview`	Container image with the `ru-RU` locale.
`2.0.0-amd64-sv-se-preview`	Container image with the `sv-SE` locale.
`2.0.0-amd64-th-th-preview`	Container image with the `th-TH` locale.
`2.0.0-amd64-tr-tr-preview`	Container image with the `tr-TR` locale.
`2.0.0-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`2.0.0-amd64-zh-hk-preview`	Container image with the `zh-HK` locale.
`2.0.0-amd64-zh-tw-preview`	Container image with the `zh-TW` locale.
`1.2.0-amd64-de-de-preview`	Container image with the `de-DE` locale.
`1.2.0-amd64-en-au-preview`	Container image with the `en-AU` locale.
`1.2.0-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`1.2.0-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`1.2.0-amd64-en-in-preview`	Container image with the `en-IN` locale.
`1.2.0-amd64-en-us-preview`	Container image with the `en-US` locale.
`1.2.0-amd64-es-es-preview`	Container image with the `es-ES` locale.
`1.2.0-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`1.2.0-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`1.2.0-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.

IMAGE TAGS	NOTES
`1.2.0-amd64-it-it-preview`	Container image with the `it-IT` locale.
`1.2.0-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`1.2.0-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`1.2.0-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`1.1.3-amd64-de-de-preview`	Container image with the `de-DE` locale.
`1.1.3-amd64-en-au-preview`	Container image with the `en-AU` locale.
`1.1.3-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`1.1.3-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`1.1.3-amd64-en-in-preview`	Container image with the `en-IN` locale.
`1.1.3-amd64-en-us-preview`	Container image with the `en-US` locale.
`1.1.3-amd64-es-es-preview`	Container image with the `es-ES` locale.
`1.1.3-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`1.1.3-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`1.1.3-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`1.1.3-amd64-it-it-preview`	Container image with the `it-IT` locale.
`1.1.3-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`1.1.3-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`1.1.3-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`1.1.2-amd64-de-de-preview`	Container image with the `de-DE` locale.
`1.1.2-amd64-en-au-preview`	Container image with the `en-AU` locale.
`1.1.2-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`1.1.2-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`1.1.2-amd64-en-in-preview`	Container image with the `en-IN` locale.
`1.1.2-amd64-en-us-preview`	Container image with the `en-US` locale.

IMAGE TAGS	NOTES
`1.1.2-amd64-es-es-preview`	Container image with the `es-ES` locale.
`1.1.2-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`1.1.2-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`1.1.2-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`1.1.2-amd64-it-it-preview`	Container image with the `it-IT` locale.
`1.1.2-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`1.1.2-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`1.1.2-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`1.1.1-amd64-de-de-preview`	Container image with the `de-DE` locale.
`1.1.1-amd64-en-au-preview`	Container image with the `en-AU` locale.
`1.1.1-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`1.1.1-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`1.1.1-amd64-en-in-preview`	Container image with the `en-IN` locale.
`1.1.1-amd64-en-us-preview`	Container image with the `en-US` locale.
`1.1.1-amd64-es-es-preview`	Container image with the `es-ES` locale.
`1.1.1-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`1.1.1-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`1.1.1-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`1.1.1-amd64-it-it-preview`	Container image with the `it-IT` locale.
`1.1.1-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`1.1.1-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`1.1.1-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`1.1.0-amd64-de-de-preview`	Container image with the `de-DE` locale.
`1.1.0-amd64-en-au-preview`	Container image with the `en-AU` locale.

IMAGE TAGS	NOTES
`1.1.0-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`1.1.0-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`1.1.0-amd64-en-in-preview`	Container image with the `en-IN` locale.
`1.1.0-amd64-en-us-preview`	Container image with the `en-US` locale.
`1.1.0-amd64-es-es-preview`	Container image with the `es-ES` locale.
`1.1.0-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`1.1.0-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`1.1.0-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`1.1.0-amd64-it-it-preview`	Container image with the `it-IT` locale.
`1.1.0-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.
`1.1.0-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`1.1.0-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.
`1.0.0-amd64-de-de-preview`	Container image with the `de-DE` locale.
`1.0.0-amd64-en-au-preview`	Container image with the `en-AU` locale.
`1.0.0-amd64-en-ca-preview`	Container image with the `en-CA` locale.
`1.0.0-amd64-en-gb-preview`	Container image with the `en-GB` locale.
`1.0.0-amd64-en-in-preview`	Container image with the `en-IN` locale.
`1.0.0-amd64-en-us-preview`	Container image with the `en-US` locale.
`1.0.0-amd64-es-es-preview`	Container image with the `es-ES` locale.
`1.0.0-amd64-es-mx-preview`	Container image with the `es-MX` locale.
`1.0.0-amd64-fr-ca-preview`	Container image with the `fr-CA` locale.
`1.0.0-amd64-fr-fr-preview`	Container image with the `fr-FR` locale.
`1.0.0-amd64-it-it-preview`	Container image with the `it-IT` locale.
`1.0.0-amd64-ja-jp-preview`	Container image with the `ja-JP` locale.

IMAGE TAGS	NOTES
`1.0.0-amd64-pt-br-preview`	Container image with the `pt-BR` locale.
`1.0.0-amd64-zh-cn-preview`	Container image with the `zh-CN` locale.

Text-to-speech

The Text-to-speech container image can be found on the `containerpreview.azurecr.io` container registry. It resides within the `microsoft` repository and is named `cognitive-services-text-to-speech`. The fully qualified container image name is, `containerpreview.azurecr.io/microsoft/cognitive-services-text-to-speech`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	Container image with the `en-US` locale and `en-US-AriaRUS` voice.
`1.7.0-amd64-ar-eg-hoda`	Container image with the `ar-EG` locale and `ar-EG-Hoda` voice.
`1.7.0-amd64-ar-sa-naayf`	Container image with the `ar-SA` locale and `ar-SA-Naayf` voice.
`1.7.0-amd64-bg-bg-ivan`	Container image with the `bg-BG` locale and `bg-BG-Ivan` voice.
`1.7.0-amd64-ca-es-herenarus`	Container image with the `ca-ES` locale and `ca-ES-HerenaRUS` voice.
`1.7.0-amd64-cs-cz-jakub`	Container image with the `cs-CZ` locale and `cs-CZ-Jakub` voice.
`1.7.0-amd64-da-dk-hellerus`	Container image with the `da-DK` locale and `da-DK-HelleRUS` voice.
`1.7.0-amd64-de-at-michael`	Container image with the `de-AT` locale and `de-AT-Michael` voice.
`1.7.0-amd64-de-ch-karsten`	Container image with the `de-CH` locale and `de-CH-Karsten` voice.
`1.7.0-amd64-de-de-hedda`	Container image with the `de-DE` locale and `de-DE-Hedda` voice.
`1.7.0-amd64-de-de-heddarus`	Container image with the `de-DE` locale and `de-DE-Hedda` voice.
`1.7.0-amd64-de-de-stefan-apollo`	Container image with the `de-DE` locale and `de-DE-Stefan-Apollo` voice.

IMAGE TAGS	NOTES
`1.7.0-amd64-el-gr-stefanos`	Container image with the `el-GR` locale and `el-GR-Stefanos` voice.
`1.7.0-amd64-en-au-catherine`	Container image with the `en-AU` locale and `en-AU-Catherine` voice.
`1.7.0-amd64-en-au-hayleyrus`	Container image with the `en-AU` locale and `en-AU-HayleyRUS` voice.
`1.7.0-amd64-en-ca-heatherrus`	Container image with the `en-CA` locale and `en-CA-HeatherRUS` voice.
`1.7.0-amd64-en-ca-linda`	Container image with the `en-CA` locale and `en-CA-Linda` voice.
`1.7.0-amd64-en-gb-george-apollo`	Container image with the `en-GB` locale and `en-GB-George-Apollo` voice.
`1.7.0-amd64-en-gb-hazelrus`	Container image with the `en-GB` locale and `en-GB-HazelRUS` voice.
`1.7.0-amd64-en-gb-susan-apollo`	Container image with the `en-GB` locale and `en-GB-Susan-Apollo` voice.
`1.7.0-amd64-en-ie-sean`	Container image with the `en-IE` locale and `en-IE-Sean` voice.
`1.7.0-amd64-en-in-heera-apollo`	Container image with the `en-IN` locale and `en-IN-Heera-Apollo` voice.
`1.7.0-amd64-en-in-priyarus`	Container image with the `en-IN` locale and `en-IN-PriyaRUS` voice.
`1.7.0-amd64-en-in-ravi-apollo`	Container image with the `en-IN` locale and `en-IN-Ravi-Apollo` voice.
`1.7.0-amd64-en-us-benjaminrus`	Container image with the `en-US` locale and `en-US-BenjaminRUS` voice.
`1.7.0-amd64-en-us-guy24krus`	Container image with the `en-US` locale and `en-US-Guy24kRUS` voice.
`1.7.0-amd64-en-us-aria24krus`	Container image with the `en-US` locale and `en-US-Aria24kRUS` voice.
`1.7.0-amd64-en-us-ariarus`	Container image with the `en-US` locale and `en-US-AriaRUS` voice.
`1.7.0-amd64-en-us-zirarus`	Container image with the `en-US` locale and `en-US-ZiraRUS` voice.

IMAGE TAGS	NOTES
`1.7.0-amd64-es-es-helenarus`	Container image with the `es-ES` locale and `es-ES-HelenaRUS` voice.
`1.7.0-amd64-es-es-laura-apollo`	Container image with the `es-ES` locale and `es-ES-Laura-Apollo` voice.
`1.7.0-amd64-es-es-pablo-apollo`	Container image with the `es-ES` locale and `es-ES-Pablo-Apollo` voice.
`1.7.0-amd64-es-mx-hildarus`	Container image with the `es-MX` locale and `es-MX-HildaRUS` voice.
`1.7.0-amd64-es-mx-raul-apollo`	Container image with the `es-MX` locale and `es-MX-Raul-Apollo` voice.
`1.7.0-amd64-fi-fi-heidirus`	Container image with the `fi-FI` locale and `fi-FI-HeidiRUS` voice.
`1.7.0-amd64-fr-ca-caroline`	Container image with the `fr-CA` locale and `fr-CA-Caroline` voice.
`1.7.0-amd64-fr-ca-harmonierus`	Container image with the `fr-CA` locale and `fr-CA-HarmonieRUS` voice.
`1.7.0-amd64-fr-ch-guillaume`	Container image with the `fr-CH` locale and `fr-CH-Guillaume` voice.
`1.7.0-amd64-fr-fr-hortenserus`	Container image with the `fr-FR` locale and `fr-FR-HortenseRUS` voice.
`1.7.0-amd64-fr-fr-julie-apollo`	Container image with the `fr-FR` locale and `fr-FR-Julie-Apollo` voice.
`1.7.0-amd64-fr-fr-paul-apollo`	Container image with the `fr-FR` locale and `fr-FR-Paul-Apollo` voice.
`1.7.0-amd64-he-il-asaf`	Container image with the `he-IL` locale and `he-IL-Asaf` voice.
`1.7.0-amd64-hi-in-hemant`	Container image with the `hi-IN` locale and `hi-IN-Hemant` voice.
`1.7.0-amd64-hi-in-kalpana-apollo`	Container image with the `hi-IN` locale and `hi-IN-Kalpana-Apollo` voice.
`1.7.0-amd64-hi-in-kalpana`	Container image with the `hi-IN` locale and `hi-IN-Kalpana` voice.
`1.7.0-amd64-hr-hr-matej`	Container image with the `hr-HR` locale and `hr-HR-Matej` voice.

IMAGE TAGS	NOTES
`1.7.0-amd64-hu-hu-szabolcs`	Container image with the `hu-HU` locale and `hu-HU-Szabolcs` voice.
`1.7.0-amd64-id-id-andika`	Container image with the `id-ID` locale and `id-ID-Andika` voice.
`1.7.0-amd64-it-it-cosimo-apollo`	Container image with the `it-IT` locale and `it-IT-Cosimo-Apollo` voice.
`1.7.0-amd64-it-it-luciarus`	Container image with the `it-IT` locale and `it-IT-LuciaRUS` voice.
`1.7.0-amd64-ja-jp-ayumi-apollo`	Container image with the `ja-JP` locale and `ja-JP-Ayumi-Apollo` voice.
`1.7.0-amd64-ja-jp-harukarus`	Container image with the `ja-JP` locale and `ja-JP-HarukaRUS` voice.
`1.7.0-amd64-ja-jp-ichiro-apollo`	Container image with the `ja-JP` locale and `ja-JP-Ichiro-Apollo` voice.
`1.7.0-amd64-ko-kr-heamirus`	Container image with the `ko-KR` locale and `ko-KR-HeamiRUS` voice.
`1.7.0-amd64-ms-my-rizwan`	Container image with the `ms-MY` locale and `ms-MY-Rizwan` voice.
`1.7.0-amd64-nb-no-huldarus`	Container image with the `nb-NO` locale and `nb-NO-HuldaRUS` voice.
`1.7.0-amd64-nl-nl-hannarus`	Container image with the `nl-NL` locale and `nl-NL-HannaRUS` voice.
`1.7.0-amd64-pl-pl-paulinarus`	Container image with the `pl-PL` locale and `pl-PL-PaulinaRUS` voice.
`1.7.0-amd64-pt-br-daniel-apollo`	Container image with the `pt-BR` locale and `pt-BR-Daniel-Apollo` voice.
`1.7.0-amd64-pt-br-heloisarus`	Container image with the `pt-BR` locale and `pt-BR-HeloisaRUS` voice.
`1.7.0-amd64-pt-pt-heliarus`	Container image with the `pt-PT` locale and `pt-PT-HeliaRUS` voice.
`1.7.0-amd64-ro-ro-andrei`	Container image with the `ro-RO` locale and `ro-RO-Andrei` voice.
`1.7.0-amd64-ru-ru-ekaterinarus`	Container image with the `ru-RU` locale and `ru-RU-EkaterinaRUS` voice.

IMAGE TAGS	NOTES
`1.7.0-amd64-ru-ru-irina-apollo`	Container image with the `ru-RU` locale and `ru-RU-Irina-Apollo` voice.
`1.7.0-amd64-ru-ru-pavel-apollo`	Container image with the `ru-RU` locale and `ru-RU-Pavel-Apollo` voice.
`1.7.0-amd64-sk-sk-filip`	Container image with the `sk-SK` locale and `sk-SK-Filip` voice.
`1.7.0-amd64-sl-si-lado`	Container image with the `sl-SI` locale and `sl-SI-Lado` voice.
`1.7.0-amd64-sv-se-hedvigrus`	Container image with the `sv-SE` locale and `sv-SE-HedvigRUS` voice.
`1.7.0-amd64-ta-in-valluvar`	Container image with the `ta-IN` locale and `ta-IN-Valluvar` voice.
`1.7.0-amd64-te-in-chitra`	Container image with the `te-IN` locale and `te-IN-Chitra` voice.
`1.7.0-amd64-th-th-pattara`	Container image with the `th-TH` locale and `th-TH-Pattara` voice.
`1.7.0-amd64-tr-tr-sedarus`	Container image with the `tr-TR` locale and `tr-TR-SedaRUS` voice.
`1.7.0-amd64-vi-vn-an`	Container image with the `vi-VN` locale and `vi-VN-An` voice.
`1.7.0-amd64-zh-cn-huihuirus`	Container image with the `zh-CN` locale and `zh-CN-HuihuiRUS` voice.
`1.7.0-amd64-zh-cn-kangkang-apollo`	Container image with the `zh-CN` locale and `zh-CN-Kangkang-Apollo` voice.
`1.7.0-amd64-zh-cn-yaoyao-apollo`	Container image with the `zh-CN` locale and `zh-CN-Yaoyao-Apollo` voice.
`1.7.0-amd64-zh-hk-danny-apollo`	Container image with the `zh-HK` locale and `zh-HK-Danny-Apollo` voice.
`1.7.0-amd64-zh-hk-tracy-apollo`	Container image with the `zh-HK` locale and `zh-HK-Tracy-Apollo` voice.
`1.7.0-amd64-zh-hk-tracyrus`	Container image with the `zh-HK` locale and `zh-HK-TracyRUS` voice.
`1.7.0-amd64-zh-tw-hanhanrus`	Container image with the `zh-TW` locale and `zh-TW-HanHanRUS` voice.

IMAGE TAGS	NOTES
`1.7.0-amd64-zh-tw-yating-apollo`	Container image with the `zh-TW` locale and `zh-TW-Yating-Apollo` voice.
`1.7.0-amd64-zh-tw-zhiwei-apollo`	Container image with the `zh-TW` locale and `zh-TW-Zhiwei-Apollo` voice.
`1.6.0-amd64-ar-eg-hoda-preview`	Container image with the `ar-EG` locale and `ar-EG-Hoda` voice.
`1.6.0-amd64-ar-sa-naayf-preview`	Container image with the `ar-SA` locale and `ar-SA-Naayf` voice.
`1.6.0-amd64-bg-bg-ivan-preview`	Container image with the `bg-BG` locale and `bg-BG-Ivan` voice.
`1.6.0-amd64-ca-es-herenarus-preview`	Container image with the `ca-ES` locale and `ca-ES-HerenaRUS` voice.
`1.6.0-amd64-cs-cz-jakub-preview`	Container image with the `cs-CZ` locale and `cs-CZ-Jakub` voice.
`1.6.0-amd64-da-dk-hellerus-preview`	Container image with the `da-DK` locale and `da-DK-HelleRUS` voice.
`1.6.0-amd64-de-at-michael-preview`	Container image with the `de-AT` locale and `de-AT-Michael` voice.
`1.6.0-amd64-de-ch-karsten-preview`	Container image with the `de-CH` locale and `de-CH-Karsten` voice.
`1.6.0-amd64-de-de-hedda-preview`	Container image with the `de-DE` locale and `de-DE-Hedda` voice.
`1.6.0-amd64-de-de-heddarus-preview`	Container image with the `de-DE` locale and `de-DE-Hedda` voice.
`1.6.0-amd64-de-de-stefan-apollo-preview`	Container image with the `de-DE` locale and `de-DE-Stefan-Apollo` voice.
`1.6.0-amd64-el-gr-stefanos-preview`	Container image with the `el-GR` locale and `el-GR-Stefanos` voice.
`1.6.0-amd64-en-au-catherine-preview`	Container image with the `en-AU` locale and `en-AU-Catherine` voice.
`1.6.0-amd64-en-au-hayleyrus-preview`	Container image with the `en-AU` locale and `en-AU-HayleyRUS` voice.
`1.6.0-amd64-en-ca-heatherrus-preview`	Container image with the `en-CA` locale and `en-CA-HeatherRUS` voice.

IMAGE TAGS	NOTES
`1.6.0-amd64-en-ca-linda-preview`	Container image with the `en-CA` locale and `en-CA-Linda` voice.
`1.6.0-amd64-en-gb-george-apollo-preview`	Container image with the `en-GB` locale and `en-GB-George-Apollo` voice.
`1.6.0-amd64-en-gb-hazelrus-preview`	Container image with the `en-GB` locale and `en-GB-HazelRUS` voice.
`1.6.0-amd64-en-gb-susan-apollo-preview`	Container image with the `en-GB` locale and `en-GB-Susan-Apollo` voice.
`1.6.0-amd64-en-ie-sean-preview`	Container image with the `en-IE` locale and `en-IE-Sean` voice.
`1.6.0-amd64-en-in-heera-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Heera-Apollo` voice.
`1.6.0-amd64-en-in-priyarus-preview`	Container image with the `en-IN` locale and `en-IN-PriyaRUS` voice.
`1.6.0-amd64-en-in-ravi-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Ravi-Apollo` voice.
`1.6.0-amd64-en-us-benjaminrus-preview`	Container image with the `en-US` locale and `en-US-BenjaminRUS` voice.
`1.6.0-amd64-en-us-guy24krus-preview`	Container image with the `en-US` locale and `en-US-Guy24kRUS` voice.
`1.6.0-amd64-en-us-aria24krus-preview`	Container image with the `en-US` locale and `en-US-Aria24kRUS` voice.
`1.6.0-amd64-en-us-ariarus-preview`	Container image with the `en-US` locale and `en-US-AriaRUS` voice.
`1.6.0-amd64-en-us-zirarus-preview`	Container image with the `en-US` locale and `en-US-ZiraRUS` voice.
`1.6.0-amd64-es-es-helenarus-preview`	Container image with the `es-ES` locale and `es-ES-HelenaRUS` voice.
`1.6.0-amd64-es-es-laura-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Laura-Apollo` voice.
`1.6.0-amd64-es-es-pablo-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Pablo-Apollo` voice.
`1.6.0-amd64-es-mx-hildarus-preview`	Container image with the `es-MX` locale and `es-MX-HildaRUS` voice.

IMAGE TAGS	NOTES
`1.6.0-amd64-es-mx-raul-apollo-preview`	Container image with the `es-MX` locale and `es-MX-Raul-Apollo` voice.
`1.6.0-amd64-fi-fi-heidirus-preview`	Container image with the `fi-FI` locale and `fi-FI-HeidiRUS` voice.
`1.6.0-amd64-fr-ca-caroline-preview`	Container image with the `fr-CA` locale and `fr-CA-Caroline` voice.
`1.6.0-amd64-fr-ca-harmonierus-preview`	Container image with the `fr-CA` locale and `fr-CA-HarmonieRUS` voice.
`1.6.0-amd64-fr-ch-guillaume-preview`	Container image with the `fr-CH` locale and `fr-CH-Guillaume` voice.
`1.6.0-amd64-fr-fr-hortenserus-preview`	Container image with the `fr-FR` locale and `fr-FR-HortenseRUS` voice.
`1.6.0-amd64-fr-fr-julie-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Julie-Apollo` voice.
`1.6.0-amd64-fr-fr-paul-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Paul-Apollo` voice.
`1.6.0-amd64-he-il-asaf-preview`	Container image with the `he-IL` locale and `he-IL-Asaf` voice.
`1.6.0-amd64-hi-in-hemant-preview`	Container image with the `hi-IN` locale and `hi-IN-Hemant` voice.
`1.6.0-amd64-hi-in-kalpana-apollo-preview`	Container image with the `hi-IN` locale and `hi-IN-Kalpana-Apollo` voice.
`1.6.0-amd64-hi-in-kalpana-preview`	Container image with the `hi-IN` locale and `hi-IN-Kalpana` voice.
`1.6.0-amd64-hr-hr-matej-preview`	Container image with the `hr-HR` locale and `hr-HR-Matej` voice.
`1.6.0-amd64-hu-hu-szabolcs-preview`	Container image with the `hu-HU` locale and `hu-HU-Szabolcs` voice.
`1.6.0-amd64-id-id-andika-preview`	Container image with the `id-ID` locale and `id-ID-Andika` voice.
`1.6.0-amd64-it-it-cosimo-apollo-preview`	Container image with the `it-IT` locale and `it-IT-Cosimo-Apollo` voice.
`1.6.0-amd64-it-it-luciarus-preview`	Container image with the `it-IT` locale and `it-IT-LuciaRUS` voice.

IMAGE TAGS	NOTES
`1.6.0-amd64-ja-jp-ayumi-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ayumi-Apollo` voice.
`1.6.0-amd64-ja-jp-harukarus-preview`	Container image with the `ja-JP` locale and `ja-JP-HarukaRUS` voice.
`1.6.0-amd64-ja-jp-ichiro-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ichiro-Apollo` voice.
`1.6.0-amd64-ko-kr-heamirus-preview`	Container image with the `ko-KR` locale and `ko-KR-HeamiRUS` voice.
`1.6.0-amd64-ms-my-rizwan-preview`	Container image with the `ms-MY` locale and `ms-MY-Rizwan` voice.
`1.6.0-amd64-nb-no-huldarus-preview`	Container image with the `nb-NO` locale and `nb-NO-HuldaRUS` voice.
`1.6.0-amd64-nl-nl-hannarus-preview`	Container image with the `nl-NL` locale and `nl-NL-HannaRUS` voice.
`1.6.0-amd64-pl-pl-paulinarus-preview`	Container image with the `pl-PL` locale and `pl-PL-PaulinaRUS` voice.
`1.6.0-amd64-pt-br-daniel-apollo-preview`	Container image with the `pt-BR` locale and `pt-BR-Daniel-Apollo` voice.
`1.6.0-amd64-pt-br-heloisarus-preview`	Container image with the `pt-BR` locale and `pt-BR-HeloisaRUS` voice.
`1.6.0-amd64-pt-pt-heliarus-preview`	Container image with the `pt-PT` locale and `pt-PT-HeliaRUS` voice.
`1.6.0-amd64-ro-ro-andrei-preview`	Container image with the `ro-RO` locale and `ro-RO-Andrei` voice.
`1.6.0-amd64-ru-ru-ekaterinarus-preview`	Container image with the `ru-RU` locale and `ru-RU-EkaterinaRUS` voice.
`1.6.0-amd64-ru-ru-irina-apollo-preview`	Container image with the `ru-RU` locale and `ru-RU-Irina-Apollo` voice.
`1.6.0-amd64-ru-ru-pavel-apollo-preview`	Container image with the `ru-RU` locale and `ru-RU-Pavel-Apollo` voice.
`1.6.0-amd64-sk-sk-filip-preview`	Container image with the `sk-SK` locale and `sk-SK-Filip` voice.
`1.6.0-amd64-sl-si-lado-preview`	Container image with the `sl-SI` locale and `sl-SI-Lado` voice.

IMAGE TAGS	NOTES
`1.6.0-amd64-sv-se-hedvigrus-preview`	Container image with the `sv-SE` locale and `sv-SE-HedvigRUS` voice.
`1.6.0-amd64-ta-in-valluvar-preview`	Container image with the `ta-IN` locale and `ta-IN-Valluvar` voice.
`1.6.0-amd64-te-in-chitra-preview`	Container image with the `te-IN` locale and `te-IN-Chitra` voice.
`1.6.0-amd64-th-th-pattara-preview`	Container image with the `th-TH` locale and `th-TH-Pattara` voice.
`1.6.0-amd64-tr-tr-sedarus-preview`	Container image with the `tr-TR` locale and `tr-TR-SedaRUS` voice.
`1.6.0-amd64-vi-vn-an-preview`	Container image with the `vi-VN` locale and `vi-VN-An` voice.
`1.6.0-amd64-zh-cn-huihuirus-preview`	Container image with the `zh-CN` locale and `zh-CN-HuihuiRUS` voice.
`1.6.0-amd64-zh-cn-kangkang-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Kangkang-Apollo` voice.
`1.6.0-amd64-zh-cn-yaoyao-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Yaoyao-Apollo` voice.
`1.6.0-amd64-zh-hk-danny-apollo-preview`	Container image with the `zh-HK` locale and `zh-HK-Danny-Apollo` voice.
`1.6.0-amd64-zh-hk-tracy-apollo-preview`	Container image with the `zh-HK` locale and `zh-HK-Tracy-Apollo` voice.
`1.6.0-amd64-zh-hk-tracyrus-preview`	Container image with the `zh-HK` locale and `zh-HK-TracyRUS` voice.
`1.6.0-amd64-zh-tw-hanhanrus-preview`	Container image with the `zh-TW` locale and `zh-TW-HanHanRUS` voice.
`1.6.0-amd64-zh-tw-yating-apollo-preview`	Container image with the `zh-TW` locale and `zh-TW-Yating-Apollo` voice.
`1.6.0-amd64-zh-tw-zhiwei-apollo-preview`	Container image with the `zh-TW` locale and `zh-TW-Zhiwei-Apollo` voice.
`1.5.0-amd64-ar-eg-hoda-preview`	Container image with the `ar-EG` locale and `ar-EG-Hoda` voice.
`1.5.0-amd64-ar-sa-naayf-preview`	Container image with the `ar-SA` locale and `ar-SA-Naayf` voice.

IMAGE TAGS	NOTES
`1.5.0-amd64-bg-bg-ivan-preview`	Container image with the `bg-BG` locale and `bg-BG-Ivan` voice.
`1.5.0-amd64-ca-es-herenarus-preview`	Container image with the `ca-ES` locale and `ca-ES-HerenaRUS` voice.
`1.5.0-amd64-cs-cz-jakub-preview`	Container image with the `cs-CZ` locale and `cs-CZ-Jakub` voice.
`1.5.0-amd64-da-dk-hellerus-preview`	Container image with the `da-DK` locale and `da-DK-HelleRUS` voice.
`1.5.0-amd64-de-at-michael-preview`	Container image with the `de-AT` locale and `de-AT-Michael` voice.
`1.5.0-amd64-de-ch-karsten-preview`	Container image with the `de-CH` locale and `de-CH-Karsten` voice.
`1.5.0-amd64-de-de-hedda-preview`	Container image with the `de-DE` locale and `de-DE-Hedda` voice.
`1.5.0-amd64-de-de-heddarus-preview`	Container image with the `de-DE` locale and `de-DE-Hedda` voice.
`1.5.0-amd64-de-de-stefan-apollo-preview`	Container image with the `de-DE` locale and `de-DE-Stefan-Apollo` voice.
`1.5.0-amd64-el-gr-stefanos-preview`	Container image with the `el-GR` locale and `el-GR-Stefanos` voice.
`1.5.0-amd64-en-au-catherine-preview`	Container image with the `en-AU` locale and `en-AU-Catherine` voice.
`1.5.0-amd64-en-au-hayleyrus-preview`	Container image with the `en-AU` locale and `en-AU-HayleyRUS` voice.
`1.5.0-amd64-en-ca-heatherrus-preview`	Container image with the `en-CA` locale and `en-CA-HeatherRUS` voice.
`1.5.0-amd64-en-ca-linda-preview`	Container image with the `en-CA` locale and `en-CA-Linda` voice.
`1.5.0-amd64-en-gb-george-apollo-preview`	Container image with the `en-GB` locale and `en-GB-George-Apollo` voice.
`1.5.0-amd64-en-gb-hazelrus-preview`	Container image with the `en-GB` locale and `en-GB-HazelRUS` voice.
`1.5.0-amd64-en-gb-susan-apollo-preview`	Container image with the `en-GB` locale and `en-GB-Susan-Apollo` voice.

IMAGE TAGS	NOTES
`1.5.0-amd64-en-ie-sean-preview`	Container image with the `en-IE` locale and `en-IE-Sean` voice.
`1.5.0-amd64-en-in-heera-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Heera-Apollo` voice.
`1.5.0-amd64-en-in-priyarus-preview`	Container image with the `en-IN` locale and `en-IN-PriyaRUS` voice.
`1.5.0-amd64-en-in-ravi-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Ravi-Apollo` voice.
`1.5.0-amd64-en-us-benjaminrus-preview`	Container image with the `en-US` locale and `en-US-BenjaminRUS` voice.
`1.5.0-amd64-en-us-guy24krus-preview`	Container image with the `en-US` locale and `en-US-Guy24kRUS` voice.
`1.5.0-amd64-en-us-aria24krus-preview`	Container image with the `en-US` locale and `en-US-Aria24kRUS` voice.
`1.5.0-amd64-en-us-ariarus-preview`	Container image with the `en-US` locale and `en-US-AriaRUS` voice.
`1.5.0-amd64-en-us-zirarus-preview`	Container image with the `en-US` locale and `en-US-ZiraRUS` voice.
`1.5.0-amd64-es-es-helenarus-preview`	Container image with the `es-ES` locale and `es-ES-HelenaRUS` voice.
`1.5.0-amd64-es-es-laura-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Laura-Apollo` voice.
`1.5.0-amd64-es-es-pablo-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Pablo-Apollo` voice.
`1.5.0-amd64-es-mx-hildarus-preview`	Container image with the `es-MX` locale and `es-MX-HildaRUS` voice.
`1.5.0-amd64-es-mx-raul-apollo-preview`	Container image with the `es-MX` locale and `es-MX-Raul-Apollo` voice.
`1.5.0-amd64-fi-fi-heidirus-preview`	Container image with the `fi-FI` locale and `fi-FI-HeidiRUS` voice.
`1.5.0-amd64-fr-ca-caroline-preview`	Container image with the `fr-CA` locale and `fr-CA-Caroline` voice.
`1.5.0-amd64-fr-ca-harmonierus-preview`	Container image with the `fr-CA` locale and `fr-CA-HarmonieRUS` voice.

IMAGE TAGS	NOTES
`1.5.0-amd64-fr-ch-guillaume-preview`	Container image with the `fr-CH` locale and `fr-CH-Guillaume` voice.
`1.5.0-amd64-fr-fr-hortenserus-preview`	Container image with the `fr-FR` locale and `fr-FR-HortenseRUS` voice.
`1.5.0-amd64-fr-fr-julie-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Julie-Apollo` voice.
`1.5.0-amd64-fr-fr-paul-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Paul-Apollo` voice.
`1.5.0-amd64-he-il-asaf-preview`	Container image with the `he-IL` locale and `he-IL-Asaf` voice.
`1.5.0-amd64-hi-in-hemant-preview`	Container image with the `hi-IN` locale and `hi-IN-Hemant` voice.
`1.5.0-amd64-hi-in-kalpana-apollo-preview`	Container image with the `hi-IN` locale and `hi-IN-Kalpana-Apollo` voice.
`1.5.0-amd64-hi-in-kalpana-preview`	Container image with the `hi-IN` locale and `hi-IN-Kalpana` voice.
`1.5.0-amd64-hr-hr-matej-preview`	Container image with the `hr-HR` locale and `hr-HR-Matej` voice.
`1.5.0-amd64-hu-hu-szabolcs-preview`	Container image with the `hu-HU` locale and `hu-HU-Szabolcs` voice.
`1.5.0-amd64-id-id-andika-preview`	Container image with the `id-ID` locale and `id-ID-Andika` voice.
`1.5.0-amd64-it-it-cosimo-apollo-preview`	Container image with the `it-IT` locale and `it-IT-Cosimo-Apollo` voice.
`1.5.0-amd64-it-it-luciarus-preview`	Container image with the `it-IT` locale and `it-IT-LuciaRUS` voice.
`1.5.0-amd64-ja-jp-ayumi-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ayumi-Apollo` voice.
`1.5.0-amd64-ja-jp-harukarus-preview`	Container image with the `ja-JP` locale and `ja-JP-HarukaRUS` voice.
`1.5.0-amd64-ja-jp-ichiro-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ichiro-Apollo` voice.
`1.5.0-amd64-ko-kr-heamirus-preview`	Container image with the `ko-KR` locale and `ko-KR-HeamiRUS` voice.

IMAGE TAGS	NOTES
`1.5.0-amd64-ms-my-rizwan-preview`	Container image with the `ms-MY` locale and `ms-MY-Rizwan` voice.
`1.5.0-amd64-nb-no-huldarus-preview`	Container image with the `nb-NO` locale and `nb-NO-HuldaRUS` voice.
`1.5.0-amd64-nl-nl-hannarus-preview`	Container image with the `nl-NL` locale and `nl-NL-HannaRUS` voice.
`1.5.0-amd64-pl-pl-paulinarus-preview`	Container image with the `pl-PL` locale and `pl-PL-PaulinaRUS` voice.
`1.5.0-amd64-pt-br-daniel-apollo-preview`	Container image with the `pt-BR` locale and `pt-BR-Daniel-Apollo` voice.
`1.5.0-amd64-pt-br-heloisarus-preview`	Container image with the `pt-BR` locale and `pt-BR-HeloisaRUS` voice.
`1.5.0-amd64-pt-pt-heliarus-preview`	Container image with the `pt-PT` locale and `pt-PT-HeliaRUS` voice.
`1.5.0-amd64-ro-ro-andrei-preview`	Container image with the `ro-RO` locale and `ro-RO-Andrei` voice.
`1.5.0-amd64-ru-ru-ekaterinarus-preview`	Container image with the `ru-RU` locale and `ru-RU-EkaterinaRUS` voice.
`1.5.0-amd64-ru-ru-irina-apollo-preview`	Container image with the `ru-RU` locale and `ru-RU-Irina-Apollo` voice.
`1.5.0-amd64-ru-ru-pavel-apollo-preview`	Container image with the `ru-RU` locale and `ru-RU-Pavel-Apollo` voice.
`1.5.0-amd64-sk-sk-filip-preview`	Container image with the `sk-SK` locale and `sk-SK-Filip` voice.
`1.5.0-amd64-sl-si-lado-preview`	Container image with the `sl-SI` locale and `sl-SI-Lado` voice.
`1.5.0-amd64-sv-se-hedvigrus-preview`	Container image with the `sv-SE` locale and `sv-SE-HedvigRUS` voice.
`1.5.0-amd64-ta-in-valluvar-preview`	Container image with the `ta-IN` locale and `ta-IN-Valluvar` voice.
`1.5.0-amd64-te-in-chitra-preview`	Container image with the `te-IN` locale and `te-IN-Chitra` voice.
`1.5.0-amd64-th-th-pattara-preview`	Container image with the `th-TH` locale and `th-TH-Pattara` voice.

IMAGE TAGS	NOTES
`1.5.0-amd64-tr-tr-sedarus-preview`	Container image with the `tr-TR` locale and `tr-TR-SedaRUS` voice.
`1.5.0-amd64-vi-vn-an-preview`	Container image with the `vi-VN` locale and `vi-VN-An` voice.
`1.5.0-amd64-zh-cn-huihuirus-preview`	Container image with the `zh-CN` locale and `zh-CN-HuihuiRUS` voice.
`1.5.0-amd64-zh-cn-kangkang-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Kangkang-Apollo` voice.
`1.5.0-amd64-zh-cn-yaoyao-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Yaoyao-Apollo` voice.
`1.5.0-amd64-zh-hk-danny-apollo-preview`	Container image with the `zh-HK` locale and `zh-HK-Danny-Apollo` voice.
`1.5.0-amd64-zh-hk-tracy-apollo-preview`	Container image with the `zh-HK` locale and `zh-HK-Tracy-Apollo` voice.
`1.5.0-amd64-zh-hk-tracyrus-preview`	Container image with the `zh-HK` locale and `zh-HK-TracyRUS` voice.
`1.5.0-amd64-zh-tw-hanhanrus-preview`	Container image with the `zh-TW` locale and `zh-TW-HanHanRUS` voice.
`1.5.0-amd64-zh-tw-yating-apollo-preview`	Container image with the `zh-TW` locale and `zh-TW-Yating-Apollo` voice.
`1.5.0-amd64-zh-tw-zhiwei-apollo-preview`	Container image with the `zh-TW` locale and `zh-TW-Zhiwei-Apollo` voice.
`1.4.0-amd64-ar-eg-hoda-preview`	Container image with the `ar-EG` locale and `ar-EG-Hoda` voice.
`1.4.0-amd64-ar-sa-naayf-preview`	Container image with the `ar-SA` locale and `ar-SA-Naayf` voice.
`1.4.0-amd64-bg-bg-ivan-preview`	Container image with the `bg-BG` locale and `bg-BG-Ivan` voice.
`1.4.0-amd64-ca-es-herenarus-preview`	Container image with the `ca-ES` locale and `ca-ES-HerenaRUS` voice.
`1.4.0-amd64-cs-cz-jakub-preview`	Container image with the `cs-CZ` locale and `cs-CZ-Jakub` voice.
`1.4.0-amd64-da-dk-hellerus-preview`	Container image with the `da-DK` locale and `da-DK-HelleRUS` voice.

IMAGE TAGS	NOTES
`1.4.0-amd64-de-at-michael-preview`	Container image with the `de-AT` locale and `de-AT-Michael` voice.
`1.4.0-amd64-de-ch-karsten-preview`	Container image with the `de-CH` locale and `de-CH-Karsten` voice.
`1.4.0-amd64-de-de-hedda-preview`	Container image with the `de-DE` locale and `de-DE-Hedda` voice.
`1.4.0-amd64-de-de-heddarus-preview`	Container image with the `de-DE` locale and `de-DE-Hedda` voice.
`1.4.0-amd64-de-de-stefan-apollo-preview`	Container image with the `de-DE` locale and `de-DE-Stefan-Apollo` voice.
`1.4.0-amd64-el-gr-stefanos-preview`	Container image with the `el-GR` locale and `el-GR-Stefanos` voice.
`1.4.0-amd64-en-au-catherine-preview`	Container image with the `en-AU` locale and `en-AU-Catherine` voice.
`1.4.0-amd64-en-au-hayleyrus-preview`	Container image with the `en-AU` locale and `en-AU-HayleyRUS` voice.
`1.4.0-amd64-en-ca-heatherrus-preview`	Container image with the `en-CA` locale and `en-CA-HeatherRUS` voice.
`1.4.0-amd64-en-ca-linda-preview`	Container image with the `en-CA` locale and `en-CA-Linda` voice.
`1.4.0-amd64-en-gb-george-apollo-preview`	Container image with the `en-GB` locale and `en-GB-George-Apollo` voice.
`1.4.0-amd64-en-gb-hazelrus-preview`	Container image with the `en-GB` locale and `en-GB-HazelRUS` voice.
`1.4.0-amd64-en-gb-susan-apollo-preview`	Container image with the `en-GB` locale and `en-GB-Susan-Apollo` voice.
`1.4.0-amd64-en-ie-sean-preview`	Container image with the `en-IE` locale and `en-IE-Sean` voice.
`1.4.0-amd64-en-in-heera-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Heera-Apollo` voice.
`1.4.0-amd64-en-in-priyarus-preview`	Container image with the `en-IN` locale and `en-IN-PriyaRUS` voice.
`1.4.0-amd64-en-in-ravi-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Ravi-Apollo` voice.

IMAGE TAGS	NOTES
`1.4.0-amd64-en-us-benjaminrus-preview`	Container image with the `en-US` locale and `en-US-BenjaminRUS` voice.
`1.4.0-amd64-en-us-guy24krus-preview`	Container image with the `en-US` locale and `en-US-Guy24kRUS` voice.
`1.4.0-amd64-en-us-aria24krus-preview`	Container image with the `en-US` locale and `en-US-Aria24kRUS` voice.
`1.4.0-amd64-en-us-ariarus-preview`	Container image with the `en-US` locale and `en-US-AriaRUS` voice.
`1.4.0-amd64-en-us-zirarus-preview`	Container image with the `en-US` locale and `en-US-ZiraRUS` voice.
`1.4.0-amd64-es-es-helenarus-preview`	Container image with the `es-ES` locale and `es-ES-HelenaRUS` voice.
`1.4.0-amd64-es-es-laura-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Laura-Apollo` voice.
`1.4.0-amd64-es-es-pablo-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Pablo-Apollo` voice.
`1.4.0-amd64-es-mx-hildarus-preview`	Container image with the `es-MX` locale and `es-MX-HildaRUS` voice.
`1.4.0-amd64-es-mx-raul-apollo-preview`	Container image with the `es-MX` locale and `es-MX-Raul-Apollo` voice.
`1.4.0-amd64-fi-fi-heidirus-preview`	Container image with the `fi-FI` locale and `fi-FI-HeidiRUS` voice.
`1.4.0-amd64-fr-ca-caroline-preview`	Container image with the `fr-CA` locale and `fr-CA-Caroline` voice.
`1.4.0-amd64-fr-ca-harmonierus-preview`	Container image with the `fr-CA` locale and `fr-CA-HarmonieRUS` voice.
`1.4.0-amd64-fr-ch-guillaume-preview`	Container image with the `fr-CH` locale and `fr-CH-Guillaume` voice.
`1.4.0-amd64-fr-fr-hortenserus-preview`	Container image with the `fr-FR` locale and `fr-FR-HortenseRUS` voice.
`1.4.0-amd64-fr-fr-julie-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Julie-Apollo` voice.
`1.4.0-amd64-fr-fr-paul-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Paul-Apollo` voice.

IMAGE TAGS	NOTES
`1.4.0-amd64-he-il-asaf-preview`	Container image with the `he-IL` locale and `he-IL-Asaf` voice.
`1.4.0-amd64-hi-in-hemant-preview`	Container image with the `hi-IN` locale and `hi-IN-Hemant` voice.
`1.4.0-amd64-hi-in-kalpana-apollo-preview`	Container image with the `hi-IN` locale and `hi-IN-Kalpana-Apollo` voice.
`1.4.0-amd64-hi-in-kalpana-preview`	Container image with the `hi-IN` locale and `hi-IN-Kalpana` voice.
`1.4.0-amd64-hr-hr-matej-preview`	Container image with the `hr-HR` locale and `hr-HR-Matej` voice.
`1.4.0-amd64-hu-hu-szabolcs-preview`	Container image with the `hu-HU` locale and `hu-HU-Szabolcs` voice.
`1.4.0-amd64-id-id-andika-preview`	Container image with the `id-ID` locale and `id-ID-Andika` voice.
`1.4.0-amd64-it-it-cosimo-apollo-preview`	Container image with the `it-IT` locale and `it-IT-Cosimo-Apollo` voice.
`1.4.0-amd64-it-it-luciarus-preview`	Container image with the `it-IT` locale and `it-IT-LuciaRUS` voice.
`1.4.0-amd64-ja-jp-ayumi-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ayumi-Apollo` voice.
`1.4.0-amd64-ja-jp-harukarus-preview`	Container image with the `ja-JP` locale and `ja-JP-HarukaRUS` voice.
`1.4.0-amd64-ja-jp-ichiro-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ichiro-Apollo` voice.
`1.4.0-amd64-ko-kr-heamirus-preview`	Container image with the `ko-KR` locale and `ko-KR-HeamiRUS` voice.
`1.4.0-amd64-ms-my-rizwan-preview`	Container image with the `ms-MY` locale and `ms-MY-Rizwan` voice.
`1.4.0-amd64-nb-no-huldarus-preview`	Container image with the `nb-NO` locale and `nb-NO-HuldaRUS` voice.
`1.4.0-amd64-nl-nl-hannarus-preview`	Container image with the `nl-NL` locale and `nl-NL-HannaRUS` voice.
`1.4.0-amd64-pl-pl-paulinarus-preview`	Container image with the `pl-PL` locale and `pl-PL-PaulinaRUS` voice.

IMAGE TAGS	NOTES
`1.4.0-amd64-pt-br-daniel-apollo-preview`	Container image with the `pt-BR` locale and `pt-BR-Daniel-Apollo` voice.
`1.4.0-amd64-pt-br-heloisarus-preview`	Container image with the `pt-BR` locale and `pt-BR-HeloisaRUS` voice.
`1.4.0-amd64-pt-pt-heliarus-preview`	Container image with the `pt-PT` locale and `pt-PT-HeliaRUS` voice.
`1.4.0-amd64-ro-ro-andrei-preview`	Container image with the `ro-RO` locale and `ro-RO-Andrei` voice.
`1.4.0-amd64-ru-ru-ekaterinarus-preview`	Container image with the `ru-RU` locale and `ru-RU-EkaterinaRUS` voice.
`1.4.0-amd64-ru-ru-irina-apollo-preview`	Container image with the `ru-RU` locale and `ru-RU-Irina-Apollo` voice.
`1.4.0-amd64-ru-ru-pavel-apollo-preview`	Container image with the `ru-RU` locale and `ru-RU-Pavel-Apollo` voice.
`1.4.0-amd64-sk-sk-filip-preview`	Container image with the `sk-SK` locale and `sk-SK-Filip` voice.
`1.4.0-amd64-sl-si-lado-preview`	Container image with the `sl-SI` locale and `sl-SI-Lado` voice.
`1.4.0-amd64-sv-se-hedvigrus-preview`	Container image with the `sv-SE` locale and `sv-SE-HedvigRUS` voice.
`1.4.0-amd64-ta-in-valluvar-preview`	Container image with the `ta-IN` locale and `ta-IN-Valluvar` voice.
`1.4.0-amd64-te-in-chitra-preview`	Container image with the `te-IN` locale and `te-IN-Chitra` voice.
`1.4.0-amd64-th-th-pattara-preview`	Container image with the `th-TH` locale and `th-TH-Pattara` voice.
`1.4.0-amd64-tr-tr-sedarus-preview`	Container image with the `tr-TR` locale and `tr-TR-SedaRUS` voice.
`1.4.0-amd64-vi-vn-an-preview`	Container image with the `vi-VN` locale and `vi-VN-An` voice.
`1.4.0-amd64-zh-cn-huihuirus-preview`	Container image with the `zh-CN` locale and `zh-CN-HuihuiRUS` voice.
`1.4.0-amd64-zh-cn-kangkang-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Kangkang-Apollo` voice.

IMAGE TAGS	NOTES
`1.4.0-amd64-zh-cn-yaoyao-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Yaoyao-Apollo` voice.
`1.4.0-amd64-zh-hk-danny-apollo-preview`	Container image with the `zh-HK` locale and `zh-HK-Danny-Apollo` voice.
`1.4.0-amd64-zh-hk-tracy-apollo-preview`	Container image with the `zh-HK` locale and `zh-HK-Tracy-Apollo` voice.
`1.4.0-amd64-zh-hk-tracyrus-preview`	Container image with the `zh-HK` locale and `zh-HK-TracyRUS` voice.
`1.4.0-amd64-zh-tw-hanhanrus-preview`	Container image with the `zh-TW` locale and `zh-TW-HanHanRUS` voice.
`1.4.0-amd64-zh-tw-yating-apollo-preview`	Container image with the `zh-TW` locale and `zh-TW-Yating-Apollo` voice.
`1.4.0-amd64-zh-tw-zhiwei-apollo-preview`	Container image with the `zh-TW` locale and `zh-TW-Zhiwei-Apollo` voice.
`1.3.0-amd64-ar-eg-hoda-preview`	Container image with the `ar-EG` locale and `ar-EG-Hoda` voice.
`1.3.0-amd64-ar-sa-naayf-preview`	Container image with the `ar-SA` locale and `ar-SA-Naayf` voice.
`1.3.0-amd64-bg-bg-ivan-preview`	Container image with the `bg-BG` locale and `bg-BG-Ivan` voice.
`1.3.0-amd64-ca-es-herenarus-preview`	Container image with the `ca-ES` locale and `ca-ES-HerenaRUS` voice.
`1.3.0-amd64-cs-cz-jakub-preview`	Container image with the `cs-CZ` locale and `cs-CZ-Jakub` voice.
`1.3.0-amd64-da-dk-hellerus-preview`	Container image with the `da-DK` locale and `da-DK-HelleRUS` voice.
`1.3.0-amd64-de-at-michael-preview`	Container image with the `de-AT` locale and `de-AT-Michael` voice.
`1.3.0-amd64-de-ch-karsten-preview`	Container image with the `de-CH` locale and `de-CH-Karsten` voice.
`1.3.0-amd64-de-de-hedda-preview`	Container image with the `de-DE` locale and `de-DE-Hedda` voice.
`1.3.0-amd64-de-de-heddarus-preview`	Container image with the `de-DE` locale and `de-DE-HeddaRUS` voice.

IMAGE TAGS	NOTES
`1.3.0-amd64-de-de-stefan-apollo-preview`	Container image with the `de-DE` locale and `de-DE-Stefan-Apollo` voice.
`1.3.0-amd64-el-gr-stefanos-preview`	Container image with the `el-GR` locale and `el-GR-Stefanos` voice.
`1.3.0-amd64-en-au-catherine-preview`	Container image with the `en-AU` locale and `en-AU-Catherine` voice.
`1.3.0-amd64-en-au-hayleyrus-preview`	Container image with the `en-AU` locale and `en-AU-HayleyRUS` voice.
`1.3.0-amd64-en-ca-heatherrus-preview`	Container image with the `en-CA` locale and `en-CA-HeatherRUS` voice.
`1.3.0-amd64-en-ca-linda-preview`	Container image with the `en-CA` locale and `en-CA-Linda` voice.
`1.3.0-amd64-en-gb-george-apollo-preview`	Container image with the `en-GB` locale and `en-GB-George-Apollo` voice.
`1.3.0-amd64-en-gb-hazelrus-preview`	Container image with the `en-GB` locale and `en-GB-HazelRUS` voice.
`1.3.0-amd64-en-gb-susan-apollo-preview`	Container image with the `en-GB` locale and `en-GB-Susan-Apollo` voice.
`1.3.0-amd64-en-ie-sean-preview`	Container image with the `en-IE` locale and `en-IE-Sean` voice.
`1.3.0-amd64-en-in-heera-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Heera-Apollo` voice.
`1.3.0-amd64-en-in-priyarus-preview`	Container image with the `en-IN` locale and `en-IN-PriyaRUS` voice.
`1.3.0-amd64-en-in-ravi-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Ravi-Apollo` voice.
`1.3.0-amd64-en-us-benjaminrus-preview`	Container image with the `en-US` locale and `en-US-BenjaminRUS` voice.
`1.3.0-amd64-en-us-guy24krus-preview`	Container image with the `en-US` locale and `en-US-Guy24kRUS` voice.
`1.3.0-amd64-en-us-jessa24krus-preview`	Container image with the `en-US` locale and `en-US-Jessa24kRUS` voice.
`1.3.0-amd64-en-us-jessarus-preview`	Container image with the `en-US` locale and `en-US-JessaRUS` voice.

IMAGE TAGS	NOTES
`1.3.0-amd64-en-us-zirarus-preview`	Container image with the `en-US` locale and `en-US-ZiraRUS` voice.
`1.3.0-amd64-es-es-helenarus-preview`	Container image with the `es-ES` locale and `es-ES-HelenaRUS` voice.
`1.3.0-amd64-es-es-laura-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Laura-Apollo` voice.
`1.3.0-amd64-es-es-pablo-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Pablo-Apollo` voice.
`1.3.0-amd64-es-mx-hildarus-preview`	Container image with the `es-MX` locale and `es-MX-HildaRUS` voice.
`1.3.0-amd64-es-mx-raul-apollo-preview`	Container image with the `es-MX` locale and `es-MX-Raul-Apollo` voice.
`1.3.0-amd64-fi-fi-heidirus-preview`	Container image with the `fi-FI` locale and `fi-FI-HeidiRUS` voice.
`1.3.0-amd64-fr-ca-caroline-preview`	Container image with the `fr-CA` locale and `fr-CA-Caroline` voice.
`1.3.0-amd64-fr-ca-harmonierus-preview`	Container image with the `fr-CA` locale and `fr-CA-HarmonieRUS` voice.
`1.3.0-amd64-fr-ch-guillaume-preview`	Container image with the `fr-CH` locale and `fr-CH-Guillaume` voice.
`1.3.0-amd64-fr-fr-hortenserus-preview`	Container image with the `fr-FR` locale and `fr-FR-HortenseRUS` voice.
`1.3.0-amd64-fr-fr-julie-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Julie-Apollo` voice.
`1.3.0-amd64-fr-fr-paul-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Paul-Apollo` voice.
`1.3.0-amd64-he-il-asaf-preview`	Container image with the `he-IL` locale and `he-IL-Asaf` voice.
`1.3.0-amd64-hi-in-hemant-preview`	Container image with the `hi-IN` locale and `hi-IN-Hemant` voice.
`1.3.0-amd64-hi-in-kalpana-preview`	Container image with the `hi-IN` locale and `hi-IN-Kalpana` voice.
`1.3.0-amd64-hi-in-kalpana-preview`	Container image with the `hi-IN` locale and `hi-IN-Kalpana` voice.

IMAGE TAGS	NOTES
`1.3.0-amd64-hr-hr-matej-preview`	Container image with the `hr-HR` locale and `hr-HR-Matej` voice.
`1.3.0-amd64-hu-hu-szabolcs-preview`	Container image with the `hu-HU` locale and `hu-HU-Szabolcs` voice.
`1.3.0-amd64-id-id-andika-preview`	Container image with the `id-ID` locale and `id-ID-Andika` voice.
`1.3.0-amd64-it-it-cosimo-apollo-preview`	Container image with the `it-IT` locale and `it-IT-Cosimo-Apollo` voice.
`1.3.0-amd64-it-it-luciarus-preview`	Container image with the `it-IT` locale and `it-IT-LuciaRUS` voice.
`1.3.0-amd64-ja-jp-ayumi-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ayumi-Apollo` voice.
`1.3.0-amd64-ja-jp-harukarus-preview`	Container image with the `ja-JP` locale and `ja-JP-HarukaRUS` voice.
`1.3.0-amd64-ja-jp-ichiro-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ichiro-Apollo` voice.
`1.3.0-amd64-ko-kr-heamirus-preview`	Container image with the `ko-KR` locale and `ko-KR-HeamiRUS` voice.
`1.3.0-amd64-ms-my-rizwan-preview`	Container image with the `ms-MY` locale and `ms-MY-Rizwan` voice.
`1.3.0-amd64-nb-no-huldarus-preview`	Container image with the `nb-NO` locale and `nb-NO-HuldaRUS` voice.
`1.3.0-amd64-nl-nl-hannarus-preview`	Container image with the `nl-NL` locale and `nl-NL-HannaRUS` voice.
`1.3.0-amd64-pl-pl-paulinarus-preview`	Container image with the `pl-PL` locale and `pl-PL-PaulinaRUS` voice.
`1.3.0-amd64-pt-br-daniel-apollo-preview`	Container image with the `pt-BR` locale and `pt-BR-Daniel-Apollo` voice.
`1.3.0-amd64-pt-br-heloisarus-preview`	Container image with the `pt-BR` locale and `pt-BR-HeloisaRUS` voice.
`1.3.0-amd64-pt-pt-heliarus-preview`	Container image with the `pt-PT` locale and `pt-PT-HeliaRUS` voice.
`1.3.0-amd64-ro-ro-andrei-preview`	Container image with the `ro-RO` locale and `ro-RO-Andrei` voice.

IMAGE TAGS	NOTES
`1.3.0-amd64-ru-ru-ekaterinarus-preview`	Container image with the `ru-RU` locale and `ru-RU-EkaterinaRUS` voice.
`1.3.0-amd64-ru-ru-irina-apollo-preview`	Container image with the `ru-RU` locale and `ru-RU-Irina-Apollo` voice.
`1.3.0-amd64-ru-ru-pavel-apollo-preview`	Container image with the `ru-RU` locale and `ru-RU-Pavel-Apollo` voice.
`1.3.0-amd64-sk-sk-filip-preview`	Container image with the `sk-SK` locale and `sk-SK-Filip` voice.
`1.3.0-amd64-sl-si-lado-preview`	Container image with the `sl-SI` locale and `sl-SI-Lado` voice.
`1.3.0-amd64-sv-se-hedvigrus-preview`	Container image with the `sv-SE` locale and `sv-SE-HedvigRUS` voice.
`1.3.0-amd64-ta-in-valluvar-preview`	Container image with the `ta-IN` locale and `ta-IN-Valluvar` voice.
`1.3.0-amd64-te-in-chitra-preview`	Container image with the `te-IN` locale and `te-IN-Chitra` voice.
`1.3.0-amd64-th-th-pattara-preview`	Container image with the `th-TH` locale and `th-TH-Pattara` voice.
`1.3.0-amd64-tr-tr-sedarus-preview`	Container image with the `tr-TR` locale and `tr-TR-SedaRUS` voice.
`1.3.0-amd64-vi-vn-an-preview`	Container image with the `vi-VN` locale and `vi-VN-An` voice.
`1.3.0-amd64-zh-cn-huihuirus-preview`	Container image with the `zh-CN` locale and `zh-CN-HuihuiRUS` voice.
`1.3.0-amd64-zh-cn-kangkang-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Kangkang-Apollo` voice.
`1.3.0-amd64-zh-cn-yaoyao-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Yaoyao-Apollo` voice.
`1.3.0-amd64-zh-hk-danny-apollo-preview`	Container image with the `zh-HK` locale and `zh-HK-Danny-Apollo` voice.
`1.3.0-amd64-zh-hk-tracy-apollo-preview`	Container image with the `zh-HK` locale and `zh-HK-Tracy-Apollo` voice.
`1.3.0-amd64-zh-hk-tracyrus-preview`	Container image with the `zh-HK` locale and `zh-HK-TracyRUS` voice.

IMAGE TAGS	NOTES
`1.3.0-amd64-zh-tw-hanhanrus-preview`	Container image with the `zh-TW` locale and `zh-TW-HanHanRUS` voice.
`1.3.0-amd64-zh-tw-yating-apollo-preview`	Container image with the `zh-TW` locale and `zh-TW-Yating-Apollo` voice.
`1.3.0-amd64-zh-tw-zhiwei-apollo-preview`	Container image with the `zh-TW` locale and `zh-TW-Zhiwei-Apollo` voice.
`1.2.0-amd64-de-de-hedda-preview`	Container image with the `de-DE` locale and `de-DE-Hedda` voice.
`1.2.0-amd64-de-de-heddarus-preview`	Container image with the `de-DE` locale and `de-DE-HeddaRUS` voice.
`1.2.0-amd64-de-de-stefan-apollo-preview`	Container image with the `de-DE` locale and `de-DE-Stefan-Apollo` voice.
`1.2.0-amd64-en-au-catherine-preview`	Container image with the `en-AU` locale and `en-AU-Catherine` voice.
`1.2.0-amd64-en-au-hayleyrus-preview`	Container image with the `en-AU` locale and `en-AU-HayleyRUS` voice.
`1.2.0-amd64-en-gb-george-apollo-preview`	Container image with the `en-GB` locale and `en-GB-George-Apollo` voice.
`1.2.0-amd64-en-gb-hazelrus-preview`	Container image with the `en-GB` locale and `en-GB-HazelRUS` voice.
`1.2.0-amd64-en-gb-susan-apollo-preview`	Container image with the `en-GB` locale and `en-GB-Susan-Apollo` voice.
`1.2.0-amd64-en-in-heera-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Heera-Apollo` voice.
`1.2.0-amd64-en-in-priyarus-preview`	Container image with the `en-IN` locale and `en-IN-PriyaRUS` voice.
`1.2.0-amd64-en-in-ravi-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Ravi-Apollo` voice.
`1.2.0-amd64-en-us-benjaminrus-preview`	Container image with the `en-US` locale and `en-US-BenjaminRUS` voice.
`1.2.0-amd64-en-us-guy24krus-preview`	Container image with the `en-US` locale and `en-US-Guy24kRUS` voice.
`1.2.0-amd64-en-us-jessa24krus-preview`	Container image with the `en-US` locale and `en-US-Jessa24kRUS` voice.

IMAGE TAGS	NOTES
`1.2.0-amd64-en-us-jessarus-preview`	Container image with the `en-US` locale and `en-US-JessaRUS` voice.
`1.2.0-amd64-en-us-zirarus-preview`	Container image with the `en-US` locale and `en-US-ZiraRUS` voice.
`1.2.0-amd64-es-es-helenarus-preview`	Container image with the `es-ES` locale and `es-ES-HelenaRUS` voice.
`1.2.0-amd64-es-es-laura-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Laura-Apollo` voice.
`1.2.0-amd64-es-es-pablo-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Pablo-Apollo` voice.
`1.2.0-amd64-es-mx-hildarus-preview`	Container image with the `es-MX` locale and `es-MX-HildaRUS` voice.
`1.2.0-amd64-es-mx-raul-apollo-preview`	Container image with the `es-MX` locale and `es-MX-Raul-Apollo` voice.
`1.2.0-amd64-fr-ca-caroline-preview`	Container image with the `fr-CA` locale and `fr-CA-Caroline` voice.
`1.2.0-amd64-fr-ca-harmonierus-preview`	Container image with the `fr-CA` locale and `fr-CA-HarmonieRUS` voice.
`1.2.0-amd64-fr-fr-hortenserus-preview`	Container image with the `fr-FR` locale and `fr-FR-HortenseRUS` voice.
`1.2.0-amd64-fr-fr-julie-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Julie-Apollo` voice.
`1.2.0-amd64-fr-fr-paul-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Paul-Apollo` voice.
`1.2.0-amd64-it-it-cosimo-apollo-preview`	Container image with the `it-IT` locale and `it-IT-Cosimo-Apollo` voice.
`1.2.0-amd64-it-it-luciarus-preview`	Container image with the `it-IT` locale and `it-IT-LuciaRUS` voice.
`1.2.0-amd64-ja-jp-ayumi-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ayumi-Apollo` voice.
`1.2.0-amd64-ja-jp-harukarus-preview`	Container image with the `ja-JP` locale and `ja-JP-HarukaRUS` voice.
`1.2.0-amd64-ja-jp-ichiro-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ichiro-Apollo` voice.

IMAGE TAGS	NOTES
`1.2.0-amd64-ko-kr-heamirus-preview`	Container image with the `ko-KR` locale and `ko-KR-HeamiRUS` voice.
`1.2.0-amd64-pt-br-daniel-apollo-preview`	Container image with the `pt-BR` locale and `pt-BR-Daniel-Apollo` voice.
`1.2.0-amd64-pt-br-heloisarus-preview`	Container image with the `pt-BR` locale and `pt-BR-HeloisaRUS` voice.
`1.2.0-amd64-zh-cn-huihuirus-preview`	Container image with the `zh-CN` locale and `zh-CN-HuihuiRUS` voice.
`1.2.0-amd64-zh-cn-kangkang-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Kangkang-Apollo` voice.
`1.2.0-amd64-zh-cn-yaoyao-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Yaoyao-Apollo` voice.
`1.1.0-amd64-de-de-hedda-preview`	Container image with the `de-DE` locale and `de-DE-Hedda` voice.
`1.1.0-amd64-de-de-heddarus-preview`	Container image with the `de-DE` locale and `de-DE-HeddaRUS` voice.
`1.1.0-amd64-de-de-stefan-apollo-preview`	Container image with the `de-DE` locale and `de-DE-Stefan-Apollo` voice.
`1.1.0-amd64-en-au-catherine-preview`	Container image with the `en-AU` locale and `en-AU-Catherine` voice.
`1.1.0-amd64-en-au-hayleyrus-preview`	Container image with the `en-AU` locale and `en-AU-HayleyRUS` voice.
`1.1.0-amd64-en-gb-george-apollo-preview`	Container image with the `en-GB` locale and `en-GB-George-Apollo` voice.
`1.1.0-amd64-en-gb-hazelrus-preview`	Container image with the `en-GB` locale and `en-GB-HazelRUS` voice.
`1.1.0-amd64-en-gb-susan-apollo-preview`	Container image with the `en-GB` locale and `en-GB-Susan-Apollo` voice.
`1.1.0-amd64-en-in-heera-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Heera-Apollo` voice.
`1.1.0-amd64-en-in-priyarus-preview`	Container image with the `en-IN` locale and `en-IN-PriyaRUS` voice.
`1.1.0-amd64-en-in-ravi-apollo-preview`	Container image with the `en-IN` locale and `en-IN-Ravi-Apollo` voice.

IMAGE TAGS	NOTES
`1.1.0-amd64-en-us-benjaminrus-preview`	Container image with the `en-US` locale and `en-US-BenjaminRUS` voice.
`1.1.0-amd64-en-us-guy24krus-preview`	Container image with the `en-US` locale and `en-US-Guy24kRUS` voice.
`1.1.0-amd64-en-us-jessa24krus-preview`	Container image with the `en-US` locale and `en-US-Jessa24kRUS` voice.
`1.1.0-amd64-en-us-jessarus-preview`	Container image with the `en-US` locale and `en-US-JessaRUS` voice.
`1.1.0-amd64-en-us-zirarus-preview`	Container image with the `en-US` locale and `en-US-ZiraRUS` voice.
`1.1.0-amd64-es-es-helenarus-preview`	Container image with the `es-ES` locale and `es-ES-HelenaRUS` voice.
`1.1.0-amd64-es-es-laura-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Laura-Apollo` voice.
`1.1.0-amd64-es-es-pablo-apollo-preview`	Container image with the `es-ES` locale and `es-ES-Pablo-Apollo` voice.
`1.1.0-amd64-es-mx-hildarus-preview`	Container image with the `es-MX` locale and `es-MX-HildaRUS` voice.
`1.1.0-amd64-es-mx-raul-apollo-preview`	Container image with the `es-MX` locale and `es-MX-Raul-Apollo` voice.
`1.1.0-amd64-fr-ca-caroline-preview`	Container image with the `fr-CA` locale and `fr-CA-Caroline` voice.
`1.1.0-amd64-fr-ca-harmonierus-preview`	Container image with the `fr-CA` locale and `fr-CA-HarmonieRUS` voice.
`1.1.0-amd64-fr-fr-hortenserus-preview`	Container image with the `fr-FR` locale and `fr-FR-HortenseRUS` voice.
`1.1.0-amd64-fr-fr-julie-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Julie-Apollo` voice.
`1.1.0-amd64-fr-fr-paul-apollo-preview`	Container image with the `fr-FR` locale and `fr-FR-Paul-Apollo` voice.
`1.1.0-amd64-it-it-cosimo-apollo-preview`	Container image with the `it-IT` locale and `it-IT-Cosimo-Apollo` voice.
`1.1.0-amd64-it-it-luciarus-preview`	Container image with the `it-IT` locale and `it-IT-LuciaRUS` voice.

IMAGE TAGS	NOTES
`1.1.0-amd64-ja-jp-ayumi-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ayumi-Apollo` voice.
`1.1.0-amd64-ja-jp-harukarus-preview`	Container image with the `ja-JP` locale and `ja-JP-HarukaRUS` voice.
`1.1.0-amd64-ja-jp-ichiro-apollo-preview`	Container image with the `ja-JP` locale and `ja-JP-Ichiro-Apollo` voice.
`1.1.0-amd64-ko-kr-heamirus-preview`	Container image with the `ko-KR` locale and `ko-KR-HeamiRUS` voice.
`1.1.0-amd64-pt-br-daniel-apollo-preview`	Container image with the `pt-BR` locale and `pt-BR-Daniel-Apollo` voice.
`1.1.0-amd64-pt-br-heloisarus-preview`	Container image with the `pt-BR` locale and `pt-BR-HeloisaRUS` voice.
`1.1.0-amd64-zh-cn-huihuirus-preview`	Container image with the `zh-CN` locale and `zh-CN-HuihuiRUS` voice.
`1.1.0-amd64-zh-cn-kangkang-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Kangkang-Apollo` voice.
`1.1.0-amd64-zh-cn-yaoyao-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Yaoyao-Apollo` voice.
`1.0.0-amd64-en-us-benjaminrus-preview`	Container image with the `en-US` locale and `en-US-BenjaminRUS` voice.
`1.0.0-amd64-en-us-guy24krus-preview`	Container image with the `en-US` locale and `en-US-Guy24kRUS` voice.
`1.0.0-amd64-en-us-jessa24krus-preview`	Container image with the `en-US` locale and `en-US-Jessa24kRUS` voice.
`1.0.0-amd64-en-us-jessarus-preview`	Container image with the `en-US` locale and `en-US-JessaRUS` voice.
`1.0.0-amd64-en-us-zirarus-preview`	Container image with the `en-US` locale and `en-US-ZiraRUS` voice.
`1.0.0-amd64-zh-cn-huihuirus-preview`	Container image with the `zh-CN` locale and `zh-CN-HuihuiRUS` voice.
`1.0.0-amd64-zh-cn-kangkang-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Kangkang-Apollo` voice.
`1.0.0-amd64-zh-cn-yaoyao-apollo-preview`	Container image with the `zh-CN` locale and `zh-CN-Yaoyao-Apollo` voice.

Neural Text-to-speech

The [Neural Text-to-speech][sp-ntts] container image can be found on the `containerpreview.azurecr.io` container registry. It resides within the `microsoft` repository and is named `cognitive-services-neural-text-to-speech`. The fully qualified container image name is, `containerpreview.azurecr.io/microsoft/cognitive-services-neural-text-to-speech`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	Container image with the `en-US` locale and `en-US-AriaNeural` voice.
`1.2.0-amd64-de-de-katjaneural-preview`	Container image with the `de-DE` locale and `de-DE-KatjaNeural` voice.
`1.2.0-amd64-en-au-natashaneural-preview`	Container image with the `en-AU` locale and `en-AU-NatashaNeural` voice.
`1.2.0-amd64-en-ca-claraneural-preview`	Container image with the `en-CA` locale and `en-CA-ClaraNeural` voice.
`1.2.0-amd64-en-gb-libbyneural-preview`	Container image with the `en-GB` locale and `en-GB-LibbyNeural` voice.
`1.2.0-amd64-en-gb-mianeural-preview`	Container image with the `en-GB` locale and `en-GB-MiaNeural` voice.
`1.2.0-amd64-en-us-arianeural-preview`	Container image with the `en-US` locale and `en-US-AriaNeural` voice.
`1.2.0-amd64-en-us-guyneural-preview`	Container image with the `en-US` locale and `en-US-GuyNeural` voice.
`1.2.0-amd64-es-es-elviraneural-preview`	Container image with the `es-ES` locale and `es-ES-ElviraNeural` voice.
`1.2.0-amd64-es-mx-dalianeural-preview`	Container image with the `es-MX` locale and `es-MX-DaliaNeural` voice.
`1.2.0-amd64-fr-ca-sylvieneural-preview`	Container image with the `fr-CA` locale and `fr-CA-SylvieNeural` voice.
`1.2.0-amd64-fr-fr-deniseneural-preview`	Container image with the `fr-FR` locale and `fr-FR-DeniseNeural` voice.
`1.2.0-amd64-it-it-elsaneural-preview`	Container image with the `it-IT` locale and `it-IT-ElsaNeural` voice.
`1.2.0-amd64-ja-jp-nanamineural-preview`	Container image with the `ja-JP` locale and `ja-JP-NanamiNeural` voice.

IMAGE TAGS	NOTES
`1.2.0-amd64-ko-kr-sunhineural-preview`	Container image with the `ko-KR` locale and `ko-KR-SunHiNeural` voice.
`1.2.0-amd64-pt-br-franciscaneural-preview`	Container image with the `pt-BR` locale and `pt-BR-FranciscaNeural` voice.
`1.2.0-amd64-zh-cn-xiaoxiaoneural-preview`	Container image with the `zh-CN` locale and `zh-CN-XiaoxiaoNeural` voice.

Key Phrase Extraction

The Key Phrase Extraction container image can be found on the `mcr.microsoft.com` container registry syndicate. It resides within the `azure-cognitive-services` repository and is named `keyphrase`. The fully qualified container image name is, `mcr.microsoft.com/azure-cognitive-services/keyphrase`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	
`1.1.009301-amd64-preview`	
`1.1.008510001-amd64-preview`	
`1.1.007750002-amd64-preview`	
`1.1.007360001-amd64-preview`	
`1.1.006770001-amd64-preview`	

Language Detection

The Language Detection container image can be found on the `mcr.microsoft.com` container registry syndicate. It resides within the `azure-cognitive-services` repository and is named `language`. The fully qualified container image name is, `mcr.microsoft.com/azure-cognitive-services/language`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	
`1.1.009301-amd64-preview`	
`1.1.008510001-amd64-preview`	
`1.1.007750002-amd64-preview`	

IMAGE TAGS	NOTES
`1.1.007360001-amd64-preview`	
`1.1.006770001-amd64-preview`	

Sentiment Analysis

The Sentiment Analysis container image can be found on the `mcr.microsoft.com` container registry syndicate. It resides within the `azure-cognitive-services` repository and is named `sentiment`. The fully qualified container image name is, `mcr.microsoft.com/azure-cognitive-services/sentiment`.

This container image has the following tags available:

IMAGE TAGS	NOTES
`latest`	
`3.0-en`	Sentiment Analysis v3 (English)
`3.0-es`	Sentiment Analysis v3 (Spanish)
`3.0-fr`	Sentiment Analysis v3 (French)
`3.0-it`	Sentiment Analysis v3 (Italian)
`3.0-de`	Sentiment Analysis v3 (German)
`3.0-zh`	Sentiment Analysis v3 (Chinese - simplified)
`3.0-zht`	Sentiment Analysis v3 (Chinese - traditional)
`3.0-ja`	Sentiment Analysis v3 (Japanese)
`3.0-pt`	Sentiment Analysis v3 (Portuguese)
`3.0-nl`	Sentiment Analysis v3 (Dutch)
`1.1.009301-amd64-preview`	Sentiment Analysis v2
`1.1.008510001-amd64-preview`	
`1.1.007750002-amd64-preview`	
`1.1.007360001-amd64-preview`	
`1.1.006770001-amd64-preview`	

Azure Cognitive Services containers frequently asked questions (FAQ)

10/4/2020 • 8 minutes to read • Edit Online

General questions

Q: What is available?

A: Azure Cognitive Services containers allow developers to use the same intelligent APIs that are available in Azure, but with the benefits of containerization. Some containers are available as a gated preview which may require an application to access. Other containers are publicly available as an ungated preview, or are generally available. You can find a full list of containers and their availability in the Container support in Azure Cognitive Services article. You can also view the containers in the Docker Hub.

Q: Is there any difference between the Cognitive Services cloud and the containers?

A: Cognitive Services containers are an alternative to the Cognitive Services cloud. Containers offer the same capabilities as the corresponding cloud services. Customers can deploy the containers on-premises or in Azure. The core AI technology, pricing tiers, API keys, and API signature are the same between the container and the corresponding cloud services. Here are the features and benefits for choosing containers over their cloud service equivalent.

Q: How do I access and use a gated preview container?

A: Previously, gated preview containers were hosted on the `containerpreview.azurecr.io` repository. Starting September 22nd 2020, these containers are hosted on the Microsoft Container Registry, and downloading them doesn't require you to use the docker login command. You'll be able to run a gated preview container if your Azure resource was created with the approved Azure subscription ID. You won't be able to run the container if your Azure subscription has not been approved after completing the request form.

Q: Will containers be available for all Cognitive Services and what are the next set of containers we should expect?

A: We would like to make more Cognitive Services available as container offerings. Contact to your local Microsoft account manager to get updates on new container releases and other Cognitive Services announcements.

Q: What will the Service-Level Agreement (SLA) be for Cognitive Services containers?

A: Cognitive Services containers do not have an SLA.

Cognitive Services container configurations of resources are controlled by customers, so Microsoft will not offer an SLA for general availability (GA). Customers are free to deploy containers on-premises, thus they define the host environments.

> **IMPORTANT**
> To learn more about Cognitive Services Service-Level Agreements, visit our SLA page.

Q: Are these containers available in sovereign clouds?

A: Not everyone is familiar with the term "sovereign cloud", so let's begin with definition:

> The "sovereign cloud" consists of the Azure Government, Azure Germany, and Azure China 21Vianet clouds.

Unfortunately, the Cognitive Services containers are *not* natively supported in the sovereign clouds. The containers can be run in these clouds, but they will be pulled from the public cloud and need to send usage data to the public endpoint.

Versioning

Q: How are containers updated to the latest version?

A: Customers can choose when to update the containers they have deployed. Containers will be marked with standard Docker tags such as `latest` to indicate the most recent version. We encourage customers to pull the latest version of containers as they are released, checkout Azure Container Registry webhooks for details on how to get notified when an image is updated.

Q: What versions will be supported?

A: The current and last major version of the container will be supported. However, we encourage customers to stay current to get the latest technology.

Q: How are updates versioned?

A: Major version changes indicate that there is a breaking change to the API signature. We anticipate that this will generally coincide with major version changes to the corresponding Cognitive Service cloud offering. Minor version changes indicate bug fixes, model updates, or new features that do not make a breaking change to the API signature.

Technical questions

Q: How should I run the Cognitive Services containers on IoT devices?

A: Whether you don't have a reliable internet connection, or want to save on bandwidth cost. Or if have low-latency requirements, or are dealing with sensitive data that needs to be analyzed on-site, Azure IoT Edge with the Cognitive Services containers gives you consistency with the cloud.

Q: Are these containers compatible with OpenShift?

We don't test containers with OpenShift, but generally, Cognitive Services containers should run on any platform that support Docker images. If you're using OpenShift, we recommend running the containers as `root-user`.

Q: How do I provide product feedback and feature recommendations?

A: Customers are encouraged to voice their concerns publicly, and up-vote others who have done the same where potential issues overlap. The user voice tool can be used for both product feedback and feature recommendations.

Q: What status messages and errors are returned by Cognitive Services containers?

A: See the following table for a list of status messages and errors.

STATUS	DESCRIPTION
`Valid`	Your API key is valid, no action is needed.
`Invalid`	Your API key is invalid. You must provide an valid API key to run the container. Find your API key and service region in the **Keys and Endpoint** section for your Azure Cognitive Services resource, in the Azure portal.

STATUS	DESCRIPTION
`Mismatch`	You have provided an API Key or endpoint for a different kind of cognitive services resource. Find your API key and service region in the **Keys and Endpoint** section for your Azure Cognitive Services resource.
`CouldNotConnect`	The container couldn't connect to the billing endpoint. Check the `Retry-After` value and wait for this period to end before making additional requests.
`OutOfQuota`	The API key is out of quota. You can either upgrade your pricing tier, or wait for additional quota to be made available. Find your tier in the **Pricing Tier** section of your Azure Cognitive Service resource, in the Azure portal.
`BillingEndpointBusy`	The billing endpoint is currently busy. Check the `Retry-After` value and wait for this period to end before making additional requests.
`ContainerUseUnauthorized`	The API key provided is not authorized for use with this container. You are likely using a gated container, so make sure your Azure Subscription ID is approved by submitting an online request.
`Unknown`	The server is currently unable to process billing requests.

Q: Who do I contact for support?

A: Customer support channels are the same as the Cognitive Services cloud offering. All Cognitive Services containers include logging features that will help us and the community support customers. For additional support, see the following options.

Customer support plan

Customers should refer to their Azure support plan to see who to contact for support.

Azure knowledge center

Customer are free to explore the Azure knowledge center to answer questions and support issues.

Stack Overflow

> Stack Overflow is a question and answer site for professional and enthusiast programmers.

Explore the following tags for potential questions and answers that align with your needs.

- Azure Cognitive Services
- Microsoft Cognitive

Q: How does billing work?

A: Customers are charged based on consumption, similar to the Cognitive Services cloud. The containers need to be configured to send metering data to Azure, and transactions will be billed accordingly. Resources used across the hosted and on-premises services will add to single quota with tiered pricing, counting against both usages. For more detail, refer to pricing page of the corresponding offering.

- Anomaly Detector
- Computer Vision

- Face
- Form Recognizer
- Language Understanding (LUIS)
- Speech Service API
- Text Analytics

IMPORTANT

Cognitive Services containers are not licensed to run without being connected to Azure for metering. Customers need to enable the containers to communicate billing information with the metering service at all times. Cognitive Services containers do not send customer data to Microsoft.

Q: What is the current support warranty for containers?

A: There is no warranty for previews. Microsoft's standard warranty for enterprise software will apply when containers are formally announced as general availability (GA).

Q: What happens to Cognitive Services containers when internet connectivity is lost?

A: Cognitive Services containers are *not licensed* to run without being connected to Azure for metering. Customers need to enable the containers to communicate with the metering service at all times.

Q: How long can the container operate without being connected to Azure?

A: Cognitive Services containers are *not licensed* to run without being connected to Azure for metering. Customers need to enable the containers to communicate with the metering service at all times.

Q: What is current hardware required to run these containers?

A: Cognitive Services containers are x64 based containers that can run any compatible Linux node, VM, and edge device that supports x64 Linux Docker Containers. They all require CPU processors. The minimum and recommended configurations for each container offering are available below:

- Anomaly Detector
- Computer Vision
- Face
- Form Recognizer
- Language Understanding (LUIS)
- Speech Service API
- Text Analytics

Q: Are these containers currently supported on Windows?

A: The Cognitive Services containers are Linux containers, however there is some support for Linux containers on Windows. For more information about Linux containers on Windows, see Docker documentation.

Q: How do I discover the containers?

A: Cognitive Services containers are available in various locations, such as the Azure portal, Docker hub, and Azure container registries. For the most recent container locations, refer to container repositories and images.

Q: How does Cognitive Services containers compare to AWS and Google offerings?

A: Microsoft is first cloud provider to move their pre-trained AI models in containers with simple billing per transaction as though customers are using a cloud service. Microsoft believes a hybrid cloud gives customers more choice.

Q: What compliance certifications do containers have?

A: Cognitive services containers do not have any compliance certifications

Q: What regions are Cognitive Services containers available in?

A: Containers can be run anywhere in any region however they need a key and to call back to Azure for metering. All supported regions for the Cloud Service are supported for the containers metering call.

Next steps

Let's continue working with Azure Cognitive Services containers.

Use more Cognitive Services Containers

Create containers for reuse

4/7/2020 • 4 minutes to read • Edit Online

Use these container recipes to create Cognitive Services Containers that can be reused. Containers can be built with some or all configuration settings so that they are *not* needed when the container is started.

Once you have this new layer of container (with settings), and you have tested it locally, you can store the container in a container registry. When the container starts, it will only need those settings that are not currently stored in the container. The private registry container provides configuration space for you to pass those settings in.

Docker run syntax

Any `docker run` examples in this document assume a Windows console with a `^` line continuation character. Consider the following for your own use:

- Do not change the order of the arguments unless you are very familiar with docker containers.
- If you are using an operating system other than Windows, or a console other than Windows console, use the correct console/terminal, folder syntax for mounts, and line continuation character for your console and system. Because the Cognitive Services container is a Linux operating system, the target mount uses a Linux-style folder syntax.
- `docker run` examples use the directory off the `c:` drive to avoid any permission conflicts on Windows. If you need to use a specific directory as the input directory, you may need to grant the docker service permission.

Store no configuration settings in image

The example `docker run` commands for each service do not store any configuration settings in the container. When you start the container from a console or registry service, those configuration settings need to pass in. The private registry container provides configuration space for you to pass those settings in.

Reuse recipe: store all configuration settings with container

In order to store all configuration settings, create a `Dockerfile` with those settings.

Issues with this approach:

- The new container has a separate name and tag from the original container.
- In order to change these settings, you will have to change the values of the Dockerfile, rebuild the image, and republish to your registry.
- If someone gets access to your container registry or your local host, they can run the container and use the Cognitive Services endpoints.
- If your Cognitive Service doesn't require input mounts, don't add the `COPY` lines to your Dockerfile.

Create Dockerfile, pulling from the existing Cognitive Services container you want to use, then use docker commands in the Dockerfile to set or pull in information the container needs.

This example:

- Sets the billing endpoint, `{BILLING_ENDPOINT}` from the host's environment key using `ENV`.
- Sets the billing API-key, `{ENDPOINT_KEY}` from the host's environment key using `ENV.

Reuse recipe: store billing settings with container

This example shows how to build the Text Analytics' sentiment container from a Dockerfile.

```
FROM mcr.microsoft.com/azure-cognitive-services/sentiment:latest
ENV billing={BILLING_ENDPOINT}
ENV apikey={ENDPOINT_KEY}
ENV EULA=accept
```

Build and run the container locally or from your private registry container as needed.

Reuse recipe: store billing and mount settings with container

This example shows how to use Language Understanding, saving billing and models from the Dockerfile.

- Copies the Language Understanding (LUIS) model file from the host's file system using `COPY`.
- The LUIS container supports more than one model. If all models are stored in the same folder, you all need one `COPY` statement.
- Run the docker file from the relative parent of the model input directory. For the following example, run the `docker build` and `docker run` commands from the relative parent of `/input`. The first `/input` on the `COPY` command is the host computer's directory. The second `/input` is the container's directory.

```
FROM <container-registry>/<cognitive-service-container-name>:<tag>
ENV billing={BILLING_ENDPOINT}
ENV apikey={ENDPOINT_KEY}
ENV EULA=accept
COPY /input /input
```

Build and run the container locally or from your private registry container as needed.

How to use container on your local host

To build the Docker file, replace `<your-image-name>` with the new name of the image, then use:

```
docker build -t <your-image-name> .
```

To run the image, and remove it when the container stops (`--rm`):

```
docker run --rm <your-image-name>
```

How to add container to private registry

Follow these steps to use the Dockerfile and place the new image in your private container registry.

1. Create a `Dockerfile` with the text from reuse recipe. A `Dockerfile` doesn't have an extension.

2. Replace any values in the angle brackets with your own values.

3. Build the file into an image at the command line or terminal, using the following command. Replace the values in the angle brackets, `<>`, with your own container name and tag.

   The tag option, `-t`, is a way to add information about what you have changed for the container. For example, a container name of `modified-LUIS` indicates the original container has been layered. A tag name of `with-billing-and-model` indicates how the Language Understanding (LUIS) container has been modified.

   ```
   docker build -t <your-new-container-name>:<your-new-tag-name> .
   ```

4. Sign in to Azure CLI from a console. This command opens a browser and requires authentication. Once authenticated, you can close the browser and continue working in the console.

```
az login
```

5. Sign in to your private registry with Azure CLI from a console.

   Replace the values in the angle brackets, `<my-registry>`, with your own registry name.

```
az acr login --name <my-registry>
```

   You can also sign in with docker login if you are assigned a service principal.

```
docker login <my-registry>.azurecr.io
```

6. Tag the container with the private registry location. Replace the values in the angle brackets, `<my-registry>`, with your own registry name.

```
docker tag <your-new-container-name>:<your-new-tag-name> <my-registry>.azurecr.io/<your-new-container-
name-in-registry>:<your-new-tag-name>
```

   If you don't use a tag name, `latest` is implied.

7. Push the new image to your private container registry. When you view your private container registry, the container name used in the following CLI command will be the name of the repository.

```
docker push <my-registry>.azurecr.io/<your-new-container-name-in-registry>:<your-new-tag-name>
```

Next steps

Create and use Azure Container Instance

Deploy and run container on Azure Container Instance

4/7/2020 • 3 minutes to read • Edit Online

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.

Prerequisites

The recipe works with any Cognitive Services container. The Cognitive Service resource must be created in the Azure portal before using the recipe. Each Cognitive Service that supports containers has a "How to install" document specifically 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.

- A Cognitive Service resource, created in Azure portal.
- 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.

Create an Azure Container Instance resource

1. Go to the Create page for Container Instances.
2. On the **Basics** tab, enter the following details:

SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select the available resource group or create a new one such as `cognitive-services`.
Container name	Enter a name such as `cognitive-container-instance`. The name must be in lower caps.
Location	Select a region for deployment.

SETTING	VALUE
Image type	If your container image is stored in a container registry that doesn't require credentials, choose `Public`. If accessing your container image requires credentials, choose `Private`. Refer to container repositories and images for details on whether or not the container image is `Public` or `Private` ("Public Preview").
Image name	Enter the Cognitive Services container location. The location is what's used as an argument to the `docker pull` command. Refer to the container repositories and images for the available image names and their corresponding repository. The image name must be fully qualified specifying three parts. First, the container registry, then the repository, finally the image name: `<container-registry>/<repository>/<image-name>`. Here is an example, `mcr.microsoft.com/azure-cognitive-services/keyphrase` would represent the Key Phrase Extraction image in the Microsoft Container Registry under the Azure Cognitive Services repository. Another example is, `containerpreview.azurecr.io/microsoft/cognitive-services-speech-to-text` which would represent the Speech to Text image in the Microsoft repository of the Container Preview container registry.
OS type	`Linux`
Size	Change size to the suggested recommendations for your specific Cognitive Service container: 2 CPU cores 4 GB

3. On the **Networking** tab, enter the following details:

SETTING	VALUE
Ports	Set the TCP port to `5000`. Exposes the container on port 5000.

4. On the **Advanced** tab, enter the required **Environment Variables** for the container billing settings of the Azure Container Instance resource:

KEY	VALUE
`apikey`	Copied from the **Keys** page of the resource. It is a 32 alphanumeric-character string with no spaces or dashes, `xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`.
`billing`	Copied from the **Overview** page of the resource.
`eula`	`accept`

5. Click **Review and Create**

6. After validation passes, click **Create** to finish the creation process

7. When the resource is successfully deployed, it's ready

Use the Container Instance

1. Select the **Overview** and copy the IP address. It will be a numeric IP address such as `55.55.55.55`.

2. Open a new browser tab and use the IP address, for example, `http://<IP-address>:5000 (http://55.55.55.55:5000`). You will see the container's home page, letting you know the container is running.

3. Select **Service API Description** to view the swagger page for the container.

4. Select any of the **POST** APIs and select **Try it out**. The parameters are displayed including the input. Fill in the parameters.

5. Select **Execute** to send the request to your Container Instance.

   You have successfully created and used Cognitive Services containers in Azure Container Instance.

Deploy the Text Analytics language detection container to Azure Kubernetes Service

10/4/2020 • 10 minutes to read • Edit Online

Learn how to deploy the language detection container. This procedure shows you how create the local Docker containers, push the containers to your own private container registry, run the container in a Kubernetes cluster, and test it in a web browser.

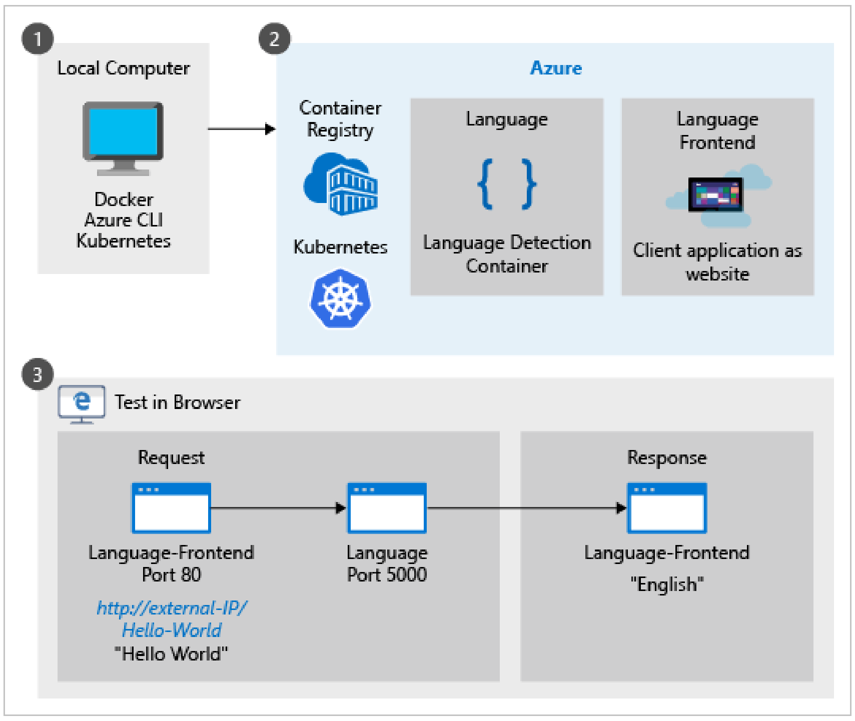
Prerequisites

This procedure requires several tools that must be installed and run locally. Do not use Azure Cloud shell.

- Use an Azure subscription. If you don't have an Azure subscription, create a free account before you begin.
- Git for your operating system so you can clone the sample used in this procedure.
- Azure CLI.
- Docker engine and validate that the Docker CLI works in a console window.
- kubectl.
- An Azure resource with the correct pricing tier. Not all pricing tiers work with this container:
  - **Text Analytics** resource with F0 or Standard pricing tiers only.
  - **Cognitive Services** resource with the S0 pricing tier.

Running the sample

This procedure loads and runs the Cognitive Services Container sample for language detection. The sample has two containers, one for the client application and one for the Cognitive Services container. We'll push both of these images to the Azure Container Registry. Once they are on your own registry, create an Azure Kubernetes Service to access these images and run the containers. When the containers are running, use the **kubectl** CLI to watch the containers performance. Access the client application with an HTTP request and see the results.



The sample containers

The sample has two container images, one for the frontend website. The second image is the language detection container returning the detected language (culture) of text. Both containers are accessible from an external IP when you are done.

The language-frontend container

This website is equivalent to your own client-side application that makes requests of the language detection endpoint. When the procedure is finished, you get the detected language of a string of characters by accessing the website container in a browser with `http://<external-IP>/<text-to-analyze>`. An example of this URL is `http://132.12.23.255/helloworld!`. The result in the browser is `English`.

The language container

The language detection container, in this specific procedure, is accessible to any external request. The container hasn't been changed in any way so the standard Cognitive Services container-specific language detection API is available.

For this container, that API is a POST request for language detection. As with all Cognitive Services containers, you can learn more about the container from its hosted Swagger information, `http://<external-IP>:5000/swagger/index.html`.

Port 5000 is the default port used with the Cognitive Services containers.

Create Azure Container Registry service

To deploy the container to the Azure Kubernetes Service, the container images need to be accessible. Create your own Azure Container Registry service to host the images.

1. Sign in to the Azure CLI

   ```
   az login
   ```

2. Create a resource group named `cogserv-container-rg` to hold every resource created in this procedure.

   ```
   az group create --name cogserv-container-rg --location westus
   ```

3. Create your own Azure Container Registry with the format of your name then `registry`, such as `pattyregistry`. Do not use dashes or underline characters in the name.

   ```
   az acr create --resource-group cogserv-container-rg --name pattyregistry --sku Basic
   ```

   Save the results to get the **loginServer** property. This will be part of the hosted container's address, used later in the `language.yml` file.

   ```
   az acr create --resource-group cogserv-container-rg --name pattyregistry --sku Basic
   ```

```
{
    "adminUserEnabled": false,
    "creationDate": "2019-01-02T23:49:53.783549+00:00",
    "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/cogserv-container-
rg/providers/Microsoft.ContainerRegistry/registries/pattyregistry",
    "location": "westus",
    "loginServer": "pattyregistry.azurecr.io",
    "name": "pattyregistry",
    "provisioningState": "Succeeded",
    "resourceGroup": "cogserv-container-rg",
    "sku": {
        "name": "Basic",
        "tier": "Basic"
    },
    "status": null,
    "storageAccount": null,
    "tags": {},
    "type": "Microsoft.ContainerRegistry/registries"
}
```

4. Sign in to your container registry. You need to login before you can push images to your registry.

```
az acr login --name pattyregistry
```

Get website Docker image

1. The sample code used in this procedure is in the Cognitive Services containers samples repository. Clone the repository to have a local copy of the sample.

```
git clone https://github.com/Azure-Samples/cognitive-services-containers-samples
```

Once the repository is on your local computer, find the website in the \dotnet\Language\FrontendService directory. This website acts as the client application calling the language detection API hosted in the language detection container.

2. Build the Docker image for this website. Make sure the console is in the \FrontendService directory where the Dockerfile is located when you run the following command:

```
docker build -t language-frontend -t pattiyregistry.azurecr.io/language-frontend:v1 .
```

To track the version on your container registry, add the tag with a version format, such as `v1`.

3. Push the image to your container registry. This may take a few minutes.

```
docker push pattyregistry.azurecr.io/language-frontend:v1
```

If you get an `unauthorized: authentication required` error, login with the `az acr login --name <your-container-registry-name>` command.

When the process is done, the results should be similar to:

```
The push refers to repository [pattyregistry.azurecr.io/language-frontend]
82ff52ee6c73: Pushed
07599c047227: Pushed
816caf41a9a1: Pushed
2924be3aed17: Pushed
45b83a23806f: Pushed
ef68f6734aa4: Pushed
v1: digest: sha256:31930445deee181605c0cde53dab5a104528dc1ff57e5b3b34324f0d8a0eb286 size: 1580
```

Get language detection Docker image

1. Pull the latest version of the Docker image to the local machine. This may take a few minutes. If there is a newer version of this container, change the value from `1.1.006770001-amd64-preview` to the newer version.

   ```
   docker pull mcr.microsoft.com/azure-cognitive-services/language:1.1.006770001-amd64-preview
   ```

2. Tag image with your container registry. Find the latest version and replace the version `1.1.006770001-amd64-preview` if you have a more recent version.

   ```
   docker tag mcr.microsoft.com/azure-cognitive-services/language
   pattiyregistry.azurecr.io/language:1.1.006770001-amd64-preview
   ```

3. Push the image to your container registry. This may take a few minutes.

   ```
   docker push pattyregistry.azurecr.io/language:1.1.006770001-amd64-preview
   ```

Get Container Registry credentials

The following steps are needed to get the required information to connect your container registry with the Azure Kubernetes Service you create later in this procedure.

1. Create service principal.

   ```
   az ad sp create-for-rbac --skip-assignment
   ```

   Save the results `appId` value for the assignee parameter in step 3, `<appId>`. Save the `password` for the next section's client-secret parameter `<client-secret>`.

   ```
   {
     "appId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
     "displayName": "azure-cli-2018-12-31-18-39-32",
     "name": "http://azure-cli-2018-12-31-18-39-32",
     "password": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
     "tenant": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
   }
   ```

2. Get your container registry ID.

   ```
   az acr show --resource-group cogserv-container-rg --name pattyregistry --query "id" --o table
   ```

   Save the output for the scope parameter value, `<acrId>`, in the next step. It looks like:

```
/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/cogserv-container-
rg/providers/Microsoft.ContainerRegistry/registries/pattyregistry
```

Save the full value for step 3 in this section.

3. To grant the correct access for the AKS cluster to use images stored in your container registry, create a role assignment. Replace `<appId>` and `<acrId>` with the values gathered in the previous two steps.

```
az role assignment create --assignee <appId> --scope <acrId> --role Reader
```

Create Azure Kubernetes Service

1. Create the Kubernetes cluster. All the parameter values are from previous sections except the name parameter. Choose a name that indicates who created it and its purpose, such as `patty-kube`.

```
az aks create --resource-group cogserv-container-rg --name patty-kube --node-count 2  --service-principal
<appId>  --client-secret <client-secret>  --generate-ssh-keys
```

This step may take a few minutes. The result is:

```
{
  "aadProfile": null,
  "addonProfiles": null,
  "agentPoolProfiles": [
    {
      "count": 2,
      "dnsPrefix": null,
      "fqdn": null,
      "maxPods": 110,
      "name": "nodepool1",
      "osDiskSizeGb": 30,
      "osType": "Linux",
      "ports": null,
      "storageProfile": "ManagedDisks",
      "vmSize": "Standard_DS1_v2",
      "vnetSubnetId": null
    }
  ],
  "dnsPrefix": "patty-kube--65a101",
  "enableRbac": true,
  "fqdn": "patty-kube--65a101-341f1f54.hcp.westus.azmk8s.io",
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourcegroups/cogserv-container-
rg/providers/Microsoft.ContainerService/managedClusters/patty-kube",
  "kubernetesVersion": "1.9.11",
  "linuxProfile": {
    "adminUsername": "azureuser",
    "ssh": {
      "publicKeys": [
        {
          "keyData": "ssh-rsa AAAAB3NzaC...ohR2d81mFC
        }
      ]
    }
  },
  "location": "westus",
  "name": "patty-kube",
  "networkProfile": {
    "dnsServiceIp": "10.0.0.10",
    "dockerBridgeCidr": "172.17.0.1/16",
    "networkPlugin": "kubenet",
    "networkPolicy": null,
    "podCidr": "10.244.0.0/16",
    "serviceCidr": "10.0.0.0/16"
  },
  "nodeResourceGroup": "MC_patty_westus",
  "provisioningState": "Succeeded",
  "resourceGroup": "cogserv-container-rg",
  "servicePrincipalProfile": {
    "clientId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "keyVaultSecretRef": null,
    "secret": null
  },
  "tags": null,
  "type": "Microsoft.ContainerService/ManagedClusters"
}
```

The service is created but it doesn't have the website container or language detection container yet.

2. Get credentials of the Kubernetes cluster.

```
az aks get-credentials --resource-group cogserv-container-rg --name patty-kube
```

Load the orchestration definition into your Kubernetes service

This section uses the **kubectl** CLI to talk with the Azure Kubernetes Service.

1. Before loading the orchestration definition, check **kubectl** has access to the nodes.

   ```
   kubectl get nodes
   ```

   The response looks like:

   ```
   NAME                        STATUS    ROLES     AGE       VERSION
   aks-nodepool1-13756812-0   Ready     agent     6m        v1.9.11
   aks-nodepool1-13756812-1   Ready     agent     6m        v1.9.11
   ```

2. Copy the following file and name it `language.yml`. The file has a `service` section and a `deployment` section each for the two container types, the `language-frontend` website container and the `language` detection container.

   ```
   # A service which exposes the .net frontend app container through a dependable hostname: http://language-
   frontend:5000
   apiVersion: v1
   kind: Service
   metadata:
     name: language-frontend
     labels:
       run: language-frontend
   spec:
     selector:
       app: language-frontend
     type: LoadBalancer
     ports:
     - name: front
       port: 80
       targetPort: 80
       protocol: TCP
   ---
   # A deployment declaratively indicating how many instances of the .net frontend app container we want up
   apiVersion: apps/v1beta1
   kind: Deployment
   metadata:
     name: language-frontend
   spec:
     replicas: 1
     template:
       metadata:
         labels:
           app: language-frontend
       spec:
         containers:
         - name: language-frontend
           image: # < URI of the Frontend App image >
           ports:
           - name: public-port
             containerPort: 80
           livenessProbe:
             httpGet:
               path: /status
               port: public-port
             initialDelaySeconds: 30
             timeoutSeconds: 1
             periodSeconds: 10
         imagePullSecrets:
           - name: # < Name of the registry secret providing access to the frontend image >
         automountServiceAccountToken: false
   ---
   # A service which exposes the cognitive-service containers through a dependable hostname:
   ```

```
http://language:5000
apiVersion: v1
kind: Service
metadata:
  name: language
  labels:
    run: language
spec:
  selector:
    app: language
  type: LoadBalancer
  ports:
  - name: language
    port: 5000
    targetPort: 5000
    protocol: TCP
---
# A deployment declaratively indicating how many instances of the cognitive-service container we want up
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: language
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: language
    spec:
      containers:
      - name: language
        image: # < URI of the Language Image >
        ports:
        - name: public-port
          containerPort: 5000
        livenessProbe:
          httpGet:
            path: /status
            port: public-port
          initialDelaySeconds: 30
          timeoutSeconds: 1
          periodSeconds: 10
        args:
            - "eula=accept"
            - "apikey=" # < API Key for the Language Service >
            - "billing=" # < Language billing endpoint URI >

      imagePullSecrets:
        - name: # < Name of the registry secret providing access to the Language image >

      automountServiceAccountToken: false
```

3. Change the language-frontend deployment lines of `language.yml` based on the following table to add your own container registry image names, client secret, and text analytics settings.

LANGUAGE-FRONTEND DEPLOYMENT SETTINGS	PURPOSE
Line 32 `image` property	Image location for the frontend image in your Container Registry `<container-registry-name>.azurecr.io/language-frontend:v1`
Line 44 `name` property	Container Registry secret for the image, referred to as `<client-secret>` in a previous section.

4. Change the language deployment lines of `language.yml` based on the following table to add your own container registry image names, client secret, and text analytics settings.

LANGUAGE DEPLOYMENT SETTINGS	PURPOSE
Line 78 `image` property	Image location for the language image in your Container Registry `<container-registry-name>.azurecr.io/language:1.1.006770001-amd64-preview`
Line 95 `name` property	Container Registry secret for the image, referred to as `<client-secret>` in a previous section.
Line 91 `apiKey` property	Your text analytics resource key
Line 92 `billing` property	The billing endpoint for your text analytics resource. `https://westus.api.cognitive.microsoft.com/text/analytics/v2.1`

Because the **apiKey** and **billing endpoint** are set as part of the Kubernetes orchestration definition, the website container doesn't need to know about these or pass them as part of the request. The website container refers to the language detection container by its orchestrator name `language`.

5. Load the orchestration definition file for this sample from the folder where you created and saved the `language.yml`.

```
kubectl apply -f language.yml
```

The response is:

```
service "language-frontend" created
deployment.apps "language-frontend" created
service "language" created
deployment.apps "language" created
```

Get external IPs of containers

For the two containers, verify the `language-frontend` and `language` services are running and get the external IP address.

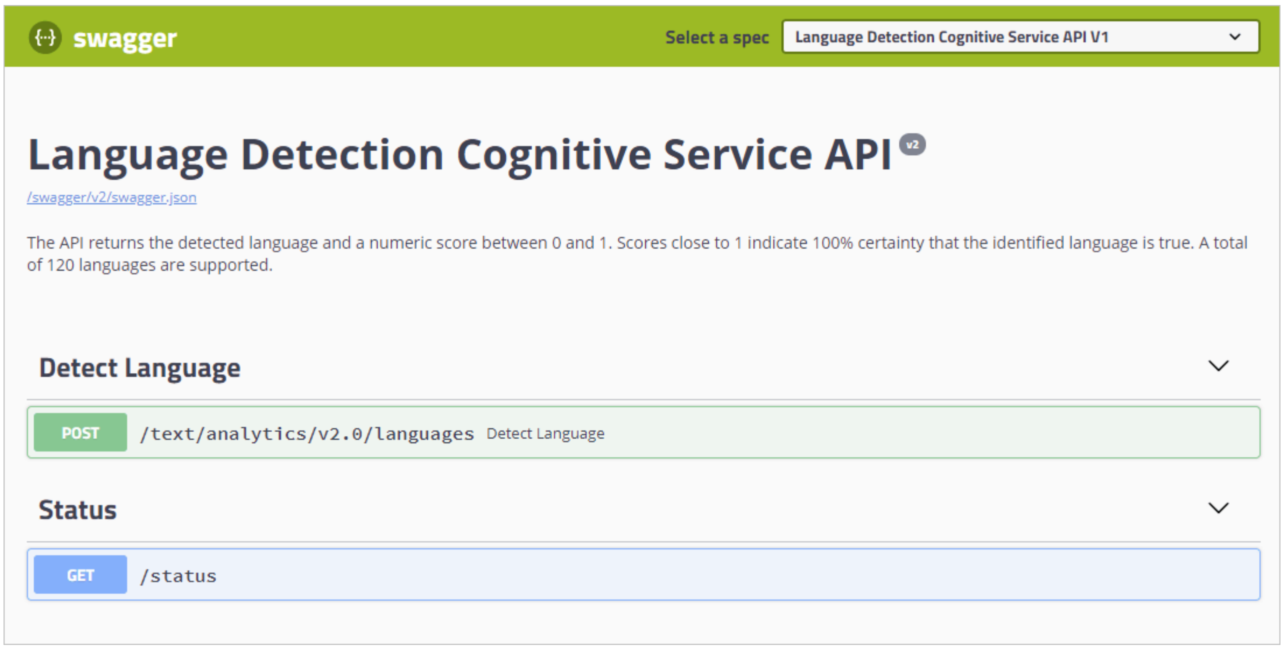
```
kubectl get all
```

```
NAME                                     READY    STATUS    RESTARTS   AGE
pod/language-586849d8dc-7zvz5            1/1      Running   0          13h
pod/language-frontend-68b9969969-bz9bg   1/1      Running   1          13h

NAME                         TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)          AGE
service/kubernetes           ClusterIP      10.0.0.1      <none>          443/TCP          14h
service/language             LoadBalancer   10.0.39.169   104.42.172.68   5000:30161/TCP   13h
service/language-frontend    LoadBalancer   10.0.42.136   104.42.37.219   80:30943/TCP     13h

NAME                                      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/language            1         1         1            1           13h
deployment.extensions/language-frontend   1         1         1            1           13h

NAME                                                 DESIRED   CURRENT   READY   AGE
replicaset.extensions/language-586849d8dc            1         1         1       13h
replicaset.extensions/language-frontend-68b9969969   1         1         1       13h

NAME                                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/language            1         1         1            1           13h
deployment.apps/language-frontend   1         1         1            1           13h

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/language-586849d8dc            1         1         1       13h
replicaset.apps/language-frontend-68b9969969   1         1         1       13h
```

If the `EXTERNAL-IP` for the service is shown as pending, rerun the command until the IP address is shown before moving to the next step.

Test the language detection container

Open a browser and navigate to the external IP of the `language` container from the previous section: `http://<external-ip>:5000/swagger/index.html`. You can use the `Try it` feature of the API to test the language detection endpoint.



Language Detection Cognitive Service API

/swagger/v2/swagger.json

The API returns the detected language and a numeric score between 0 and 1. Scores close to 1 indicate 100% certainty that the identified language is true. A total of 120 languages are supported.

Detect Language

/text/analytics/v2.0/languages Detect Language

Status

/status

Test the client application container

Change the URL in the browser to the external IP of the `language-frontend` container using the following format: `http://<external-ip>/helloworld`. The English culture text of `helloworld` is predicted as `English`.

Clean up resources

When you are done with the cluster, delete the Azure resource group.

```
az group delete --name cogserv-container-rg
```

Related information

- kubectl for Docker Users

Next steps

Cognitive Services Containers

Use Docker Compose to deploy multiple containers

10/4/2020 • 6 minutes to read • Edit Online

This article shows you how to deploy multiple Azure Cognitive Services containers. Specifically, you'll learn how to use Docker Compose to orchestrate multiple Docker container images.

> Docker Compose is a tool for defining and running multi-container Docker applications. In Compose, you use a YAML file to configure your application's services. Then, you create and start all the services from your configuration by running a single command.

It can be useful to orchestrate multiple container images on a single host computer. In this article, we'll pull together the Read and Form Recognizer containers.

Prerequisites

This procedure requires several tools that must be installed and run locally:

- An Azure subscription. If you don't have one, create a free account before you begin.
- Docker Engine. Confirm that the Docker CLI works in a console window.
- An Azure resource with the correct pricing tier. Only the following pricing tiers work with this container:
  - **Computer Vision** resource with F0 or Standard pricing tier only.
  - **Form Recognizer** resource with F0 or Standard pricing tier only.
  - **Cognitive Services** resource with the S0 pricing tier.

Request access to the container registry

Complete and submit the Cognitive Services Speech Containers Request form.

The form requests information about you, your company, and the user scenario for which you'll use the container. After you've submitted the form, the Azure Cognitive Services team reviews it to ensure that you meet the criteria for access to the private container registry.

> **IMPORTANT**
>
> You must use an email address that's associated with either a Microsoft Account (MSA) or Azure Active Directory (Azure AD) account in the form.

If your request is approved, you'll receive an email with instructions that describe how to obtain your credentials and access the private container registry.

Use the Docker CLI to authenticate the private container registry

You can authenticate with the private container registry for Cognitive Services Containers in any of several ways, but the recommended method from the command line is to use the Docker CLI.

Use the `docker login` command, as shown in the following example, to log in to `containerpreview.azurecr.io`, the private container registry for Cognitive Services Containers. Replace *<username>* with the user name and *<password>* with the password that's provided in the credentials you received from the Azure Cognitive Services team.

```
docker login containerpreview.azurecr.io -u <username> -p <password>
```

If you've secured your credentials in a text file, you can concatenate the contents of that text file, by using the `cat` command, to the `docker login` command, as shown in the following example. Replace *<passwordFile>* with the path and name of the text file that contains the password and *<username>* with the user name that's provided in your credentials.

```
cat <passwordFile> | docker login containerpreview.azurecr.io -u <username> --password-stdin
```

Docker Compose file

The YAML file defines all the services to be deployed. These services rely on either a `DockerFile` or an existing container image. In this case, we'll use two preview images. Copy and paste the following YAML file, and save it as *docker-compose.yaml*. Provide the appropriate **apikey**, **billing**, and **EndpointUri** values in the file.

```
version: '3.7'
services:
  forms:
    image: "containerpreview.azurecr.io/microsoft/cognitive-services-form-recognizer"
    environment:
       eula: accept
       billing: # < Your form recognizer billing URL >
       apikey: # < Your form recognizer API key >
       FormRecognizer__ComputerVisionApiKey: # < Your form recognizer API key >
       FormRecognizer__ComputerVisionEndpointUri: # < Your form recognizer URI >
    volumes:
       - type: bind
         source: E:\publicpreview\output
         target: /output
       - type: bind
         source: E:\publicpreview\input
         target: /input
    ports:
      - "5010:5000"

  ocr:
    image: "containerpreview.azurecr.io/microsoft/cognitive-services-read"
    environment:
      eula: accept
      apikey: # < Your computer vision API key >
      billing: # < Your computer vision billing URL >
    ports:
      - "5021:5000"
```

IMPORTANT

Create the directories on the host machine that are specified under the **volumes** node. This approach is required because the directories must exist before you try to mount an image by using volume bindings.

Start the configured Docker Compose services

A Docker Compose file enables the management of all the stages in a defined service's life cycle: starting, stopping, and rebuilding services; viewing the service status; and log streaming. Open a command-line interface from the project directory (where the docker-compose.yaml file is located).

NOTE

To avoid errors, make sure that the host machine correctly shares drives with Docker Engine. For example, if *E:\publicpreview* is used as a directory in the *docker-compose.yaml* file, share drive **E** with Docker.

From the command-line interface, execute the following command to start (or restart) all the services defined in the *docker-compose.yaml* file:

```
docker-compose up
```

The first time Docker executes the `docker-compose up` command by using this configuration, it pulls the images configured under the `services` node and then downloads and mounts them:

```
Pulling forms (containerpreview.azurecr.io/microsoft/cognitive-services-form-recognizer:)...
latest: Pulling from microsoft/cognitive-services-form-recognizer
743f2d6c1f65: Pull complete
72befba99561: Pull complete
2a40b9192d02: Pull complete
c7715c9d5c33: Pull complete
f0b33959f1c4: Pull complete
b8ab86c6ab26: Pull complete
41940c21ed3c: Pull complete
e3d37dd258d4: Pull complete
cdb5eb761109: Pull complete
fd93b5f95865: Pull complete
ef41dcbc5857: Pull complete
4d05c86a4178: Pull complete
34e811d37201: Pull complete
Pulling ocr (containerpreview.azurecr.io/microsoft/cognitive-services-read:)...
latest: Pulling from microsoft/cognitive-services-read
f476d66f5408: Already exists
8882c27f669e: Already exists
d9af21273955: Already exists
f5029279ec12: Already exists
1a578849dcd1: Pull complete
45064b1ab0bf: Download complete
4bb846705268: Downloading [=========================================>         ]  187.1MB/222.8MB
c56511552241: Waiting
e91d2aa0f1ad: Downloading [==============================================>    ]  162.2MB/176.1MB
```

After the images are downloaded, the image services are started:

```
Starting docker_ocr_1   ... done
Starting docker_forms_1 ... doneAttaching to docker_ocr_1, docker_forms_1forms_1  | forms_1  | forms_1  |
Notice: This Preview is made available to you on the condition that you agree to the Supplemental Terms of Use
for Microsoft Azure Previews [https://go.microsoft.com/fwlink/?linkid=2018815], which supplement your agreement
[https://go.microsoft.com/fwlink/?linkid=2018657] governing your use of Azure. If you do not have an existing
agreement governing your use of Azure, you agree that your agreement governing use of Azure is the Microsoft
Online Subscription Agreement [https://go.microsoft.com/fwlink/?linkid=2018755] (which incorporates the Online
Services Terms [https://go.microsoft.com/fwlink/?linkid=2018760]). By using the Preview you agree to these
terms.
forms_1  |
forms_1  |
forms_1  | Using '/input' for reading models and other read-only data.
forms_1  | Using '/output/forms/812d811d1bcc' for writing logs and other output data.
forms_1  | Logging to console.
forms_1  | Submitting metering to 'https://westus2.api.cognitive.microsoft.com/'.
forms_1  | WARNING: No access control enabled!
forms_1  | warn: Microsoft.AspNetCore.Server.Kestrel[0]
forms_1  |       Overriding address(es) 'http://+:80'. Binding to endpoints defined in UseKestrel() instead.
forms_1  | Hosting environment: Production
forms_1  | Content root path: /app/forms
forms_1  | Now listening on: http://0.0.0.0:5000
forms_1  | Application started. Press Ctrl+C to shut down.
ocr_1    |
ocr_1    |
ocr_1    | Notice: This Preview is made available to you on the condition that you agree to the Supplemental
Terms of Use for Microsoft Azure Previews [https://go.microsoft.com/fwlink/?linkid=2018815], which supplement
your agreement [https://go.microsoft.com/fwlink/?linkid=2018657] governing your use of Azure. If you do not
have an existing agreement governing your use of Azure, you agree that your agreement governing use of Azure is
the Microsoft Online Subscription Agreement [https://go.microsoft.com/fwlink/?linkid=2018755] (which
incorporates the Online Services Terms [https://go.microsoft.com/fwlink/?linkid=2018760]). By using the Preview
you agree to these terms.
ocr_1    |
ocr_1    |
ocr_1    | Logging to console.
ocr_1    | Submitting metering to 'https://westcentralus.api.cognitive.microsoft.com/'.
ocr_1    | WARNING: No access control enabled!
ocr_1    | Hosting environment: Production
ocr_1    | Content root path: /
ocr_1    | Now listening on: http://0.0.0.0:5000
ocr_1    | Application started. Press Ctrl+C to shut down.
```

Verify the service availability

TIP

You can use the docker images command to list your downloaded container images. For example, the following command lists the ID, repository, and tag of each downloaded container image, formatted as a table:

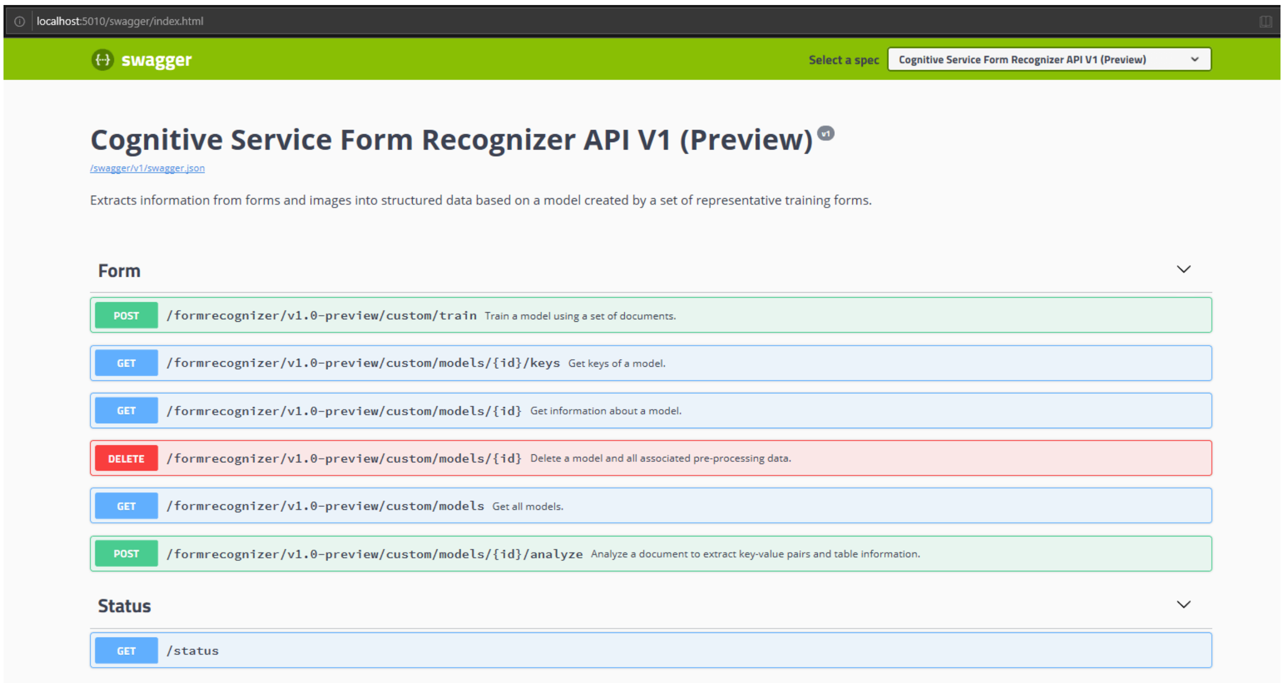
```
docker images --format "table {{.ID}}\t{{.Repository}}\t{{.Tag}}"

IMAGE ID         REPOSITORY                TAG
<image-id>       <repository-path/name>    <tag-name>
```

Here's some example output:

```
IMAGE ID          REPOSITORY                                                            TAG
2ce533f88e80      containerpreview.azurecr.io/microsoft/cognitive-services-form-recognizer   latest
4be104c126c5      containerpreview.azurecr.io/microsoft/cognitive-services-read              latest
```

Test containers

Open a browser on the host machine and go to **localhost** by using the specified port from the *docker-compose.yaml* file, such as http://localhost:5021/swagger/index.html. For example, you could use the **Try It** feature in the API to test the Form Recognizer endpoint. Both containers swagger pages should be available and testable.


localhost:5010/swagger/index.html

Cognitive Service Form Recognizer API V1 (Preview)

/swagger/v1/swagger.json

Extracts information from forms and images into structured data based on a model created by a set of representative training forms.

Form

POST /formrecognizer/v1.0-preview/custom/train Train a model using a set of documents.

GET /formrecognizer/v1.0-preview/custom/models/{id}/keys Get keys of a model.

GET /formrecognizer/v1.0-preview/custom/models/{id} Get information about a model.

DELETE /formrecognizer/v1.0-preview/custom/models/{id} Delete a model and all associated pre-processing data.

GET /formrecognizer/v1.0-preview/custom/models Get all models.

POST /formrecognizer/v1.0-preview/custom/models/{id}/analyze Analyze a document to extract key-value pairs and table information.

Status

GET /status

Next steps

Cognitive Services containers

Tutorial: Create a container image for deployment to Azure Container Instances

10/4/2020 • 3 minutes to read • Edit Online

Azure Container Instances enables deployment of Docker containers onto Azure infrastructure without provisioning any virtual machines or adopting a higher-level service. In this tutorial, you package a small Node.js web application into a container image that can be run using Azure Container Instances.

In this article, part one of the series, you:

- Clone application source code from GitHub
- Create a container image from application source
- Test the image in a local Docker environment

In tutorial parts two and three, you upload your image to Azure Container Registry, and then deploy it to Azure Container Instances.

Before you begin

You must satisfy the following requirements to complete this tutorial:

Azure CLI: You must have Azure CLI version 2.0.29 or later installed on your local computer. Run `az --version` to find the version. If you need to install or upgrade, see Install the Azure CLI.

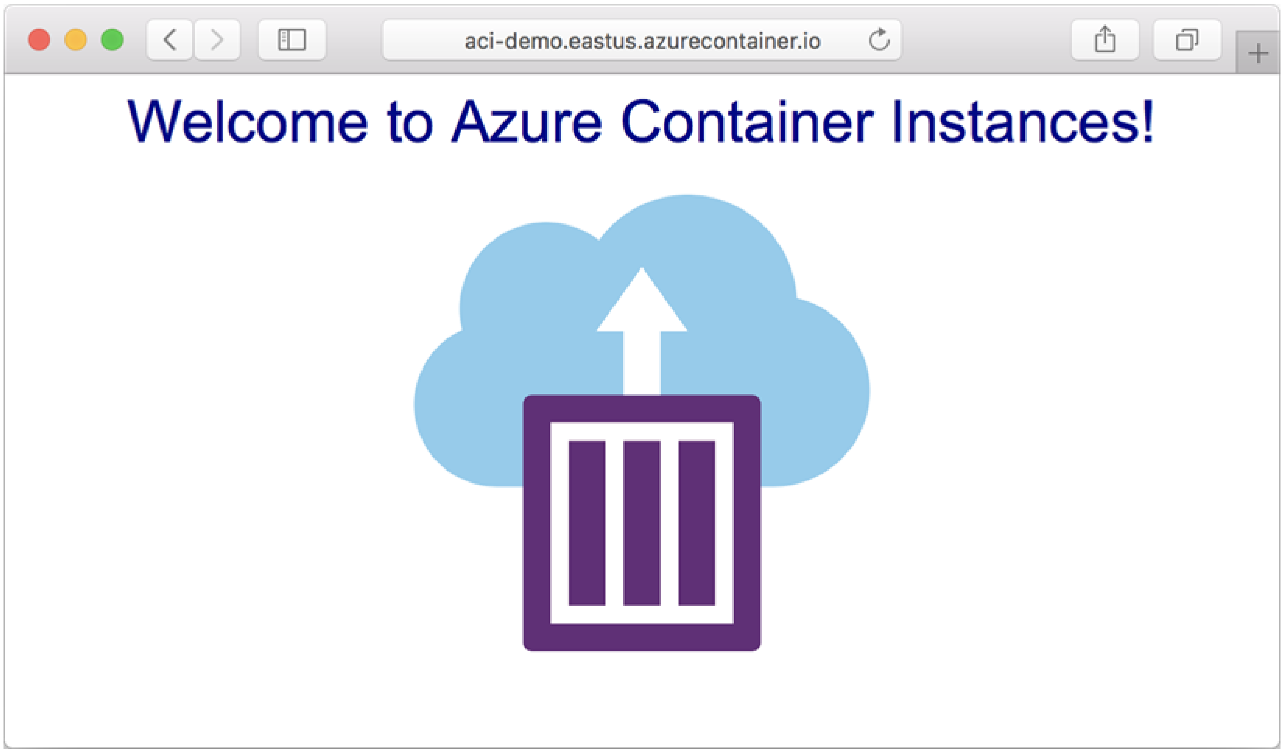
Docker: This tutorial assumes a basic understanding of core Docker concepts like containers, container images, and basic `docker` commands. For a primer on Docker and container basics, see the Docker overview.

Docker: To complete this tutorial, you need Docker installed locally. Docker provides packages that configure the Docker environment on macOS, Windows, and Linux.

> **IMPORTANT**
>
> Because the Azure Cloud shell does not include the Docker daemon, you *must* install both the Azure CLI and Docker Engine on your *local computer* to complete this tutorial. You cannot use the Azure Cloud Shell for this tutorial.

Get application code

The sample application in this tutorial is a simple web app built in Node.js. The application serves a static HTML page, and looks similar to the following screenshot:


aci-demo.eastus.azurecontainer.io
Welcome to Azure Container Instances!

Use Git to clone the sample application's repository:

```
git clone https://github.com/Azure-Samples/aci-helloworld.git
```

You can also download the ZIP archive from GitHub directly.

Build the container image

The Dockerfile in the sample application shows how the container is built. It starts from an official Node.js image based on Alpine Linux, a small distribution that is well suited for use with containers. It then copies the application files into the container, installs dependencies using the Node Package Manager, and finally, starts the application.

```
FROM node:8.9.3-alpine
RUN mkdir -p /usr/src/app
COPY ./app/ /usr/src/app/
WORKDIR /usr/src/app
RUN npm install
CMD node /usr/src/app/index.js
```

Use the docker build command to create the container image and tag it as *aci-tutorial-app*:

```
docker build ./aci-helloworld -t aci-tutorial-app
```

Output from the docker build command is similar to the following (truncated for readability):

```
$ docker build ./aci-helloworld -t aci-tutorial-app
Sending build context to Docker daemon  119.3kB
Step 1/6 : FROM node:8.9.3-alpine
8.9.3-alpine: Pulling from library/node
88286f41530e: Pull complete
84f3a4bf8410: Pull complete
d0d9b2214720: Pull complete
Digest: sha256:c73277ccc763752b42bb2400d1aaecb4e3d32e3a9dbedd0e49885c71bea07354
Status: Downloaded newer image for node:8.9.3-alpine
 ---> 90f5ee24bee2
...
Step 6/6 : CMD node /usr/src/app/index.js
 ---> Running in f4a1ea099eec
 ---> 6edad76d09e9
Removing intermediate container f4a1ea099eec
Successfully built 6edad76d09e9
Successfully tagged aci-tutorial-app:latest
```

Use the docker images command to see the built image:

```
docker images
```

Your newly built image should appear in the list:

```
$ docker images
REPOSITORY          TAG       IMAGE ID        CREATED           SIZE
aci-tutorial-app    latest    5c745774dfa9    39 seconds ago    68.1 MB
```

Run the container locally

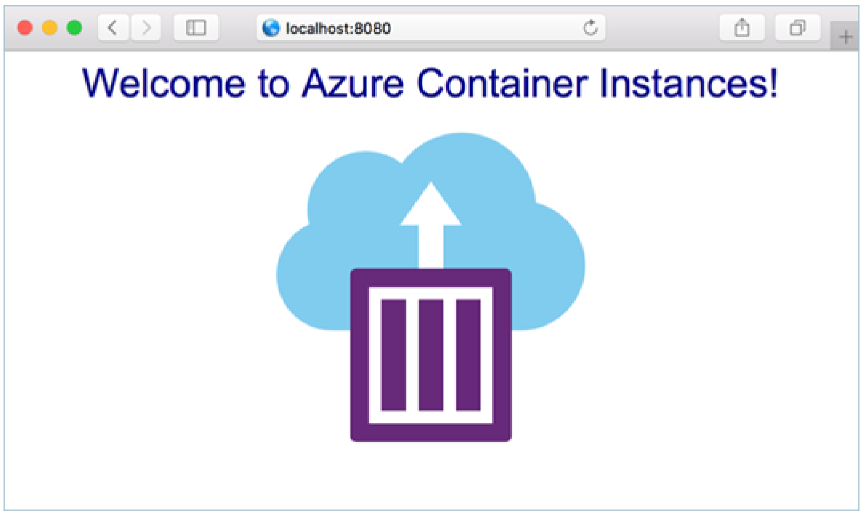
Before you deploy the container to Azure Container Instances, use docker run to run it locally and confirm that it works. The `-d` switch lets the container run in the background, while `-p` allows you to map an arbitrary port on your computer to port 80 in the container.

```
docker run -d -p 8080:80 aci-tutorial-app
```

Output from the `docker run` command displays the running container's ID if the command was successful:

```
$ docker run -d -p 8080:80 aci-tutorial-app
a2e3e4435db58ab0c664ce521854c2e1a1bda88c9cf2fcff46aedf48df86cccf
```

Now, navigate to `http://localhost:8080` in your browser to confirm that the container is running. You should see a web page similar to the following:



Next steps

In this tutorial, you created a container image that can be deployed in Azure Container Instances, and verified that it runs locally. So far, you've done the following:

- Cloned the application source from GitHub
- Created a container image from the application source
- Tested the container locally

Advance to the next tutorial in the series to learn about storing your container image in Azure Container Registry:

Push image to Azure Container Registry

Quickstart: Create a private container registry using the Azure CLI

10/4/2020 • 3 minutes to read • Edit Online

Azure Container Registry is a managed Docker container registry service used for storing private Docker container images. This guide details creating an Azure Container Registry instance using the Azure CLI. Then, use Docker commands to push a container image into the registry, and finally pull and run the image from your registry.

This quickstart requires that you are running the Azure CLI (version 2.0.55 or later recommended). Run `az --version` to find the version. If you need to install or upgrade, see Install Azure CLI.

You must also have Docker installed locally. Docker provides packages that easily configure Docker on any macOS, Windows, or Linux system.

Because the Azure Cloud Shell doesn't include all required Docker components (the `dockerd` daemon), you can't use the Cloud Shell for this quickstart.

Create a resource group

Create a resource group with the az group create command. An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named *myResourceGroup* in the *eastus* location.

```
az group create --name myResourceGroup --location eastus
```

Create a container registry

In this quickstart you create a *Basic* registry, which is a cost-optimized option for developers learning about Azure Container Registry. For details on available service tiers, see Container registry service tiers.

Create an ACR instance using the az acr create command. The registry name must be unique within Azure, and contain 5-50 alphanumeric characters. In the following example, *myContainerRegistry007* is used. Update this to a unique value.

```
az acr create --resource-group myResourceGroup \
  --name myContainerRegistry007 --sku Basic
```

When the registry is created, the output is similar to the following:

```
{
  "adminUserEnabled": false,
  "creationDate": "2019-01-08T22:32:13.175925+00:00",
  "id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.ContainerRegistry/registries/myContainerRegistr
y007",
  "location": "eastus",
  "loginServer": "mycontainerregistry007.azurecr.io",
  "name": "myContainerRegistry007",
  "provisioningState": "Succeeded",
  "resourceGroup": "myResourceGroup",
  "sku": {
    "name": "Basic",
    "tier": "Basic"
  },
  "status": null,
  "storageAccount": null,
  "tags": {},
  "type": "Microsoft.ContainerRegistry/registries"
}
```

Take note of `loginServer` in the output, which is the fully qualified registry name (all lowercase). Throughout the rest of this quickstart `<registry-name>` is a placeholder for the container registry name, and `<login-server>` is a placeholder for the registry's login server name.

Log in to registry

Before pushing and pulling container images, you must log in to the registry. To do so, use the az acr login command.

```
az acr login --name <registry-name>
```

The command returns a `Login Succeeded` message once completed.

Push image to registry

To push an image to an Azure Container registry, you must first have an image. If you don't yet have any local container images, run the following docker pull command to pull an existing image from Docker Hub. For this example, pull the `hello-world` image.

```
docker pull hello-world
```

Before you can push an image to your registry, you must tag it with the fully qualified name of your registry login server. The login server name is in the format *<registry-name>.azurecr.io* (all lowercase), for example, *mycontainerregistry.azurecr.io*.

Tag the image using the docker tag command. Replace `<login-server>` with the login server name of your ACR instance.

```
docker tag hello-world <login-server>/hello-world:v1
```

Example:

```
docker tag hello-world mycontainerregistry.azurecr.io/hello-world:v1
```

Finally, use docker push to push the image to the registry instance. Replace `<login-server>` with the login server name of your registry instance. This example creates the **hello-world** repository, containing the `hello-world:v1` image.

```
docker push <login-server>/hello-world:v1
```

After pushing the image to your container registry, remove the `hello-world:v1` image from your local Docker environment. (Note that this docker rmi command does not remove the image from the **hello-world** repository in your Azure container registry.)

```
docker rmi <login-server>/hello-world:v1
```

List container images

The following example lists the repositories in your registry:

```
az acr repository list --name <registry-name> --output table
```

Output:

```
Result
----------------
hello-world
```

The following example lists the tags on the **hello-world** repository.

```
az acr repository show-tags --name <registry-name> --repository hello-world --output table
```

Output:

```
Result
--------
v1
```

Run image from registry

Now, you can pull and run the `hello-world:v1` container image from your container registry by using docker run:

```
docker run <login-server>/hello-world:v1
```

Example output:

```
Unable to find image 'mycontainerregistry.azurecr.io/hello-world:v1' locally
v1: Pulling from hello-world
Digest: sha256:662dd8e65ef7ccf13f417962c2f77567d3b132f12c95909de6c85ac3c326a345
Status: Downloaded newer image for mycontainerregistry.azurecr.io/hello-world:v1

Hello from Docker!
This message shows that your installation appears to be working correctly.

[...]
```

Clean up resources

When no longer needed, you can use the az group delete command to remove the resource group, the container registry, and the container images stored there.

```
az group delete --name myResourceGroup
```

Next steps

In this quickstart, you created an Azure Container Registry with the Azure CLI, pushed a container image to the registry, and pulled and ran the image from the registry. Continue to the Azure Container Registry tutorials for a deeper look at ACR.

Azure Container Registry tutorials

Azure Container Registry Tasks tutorials

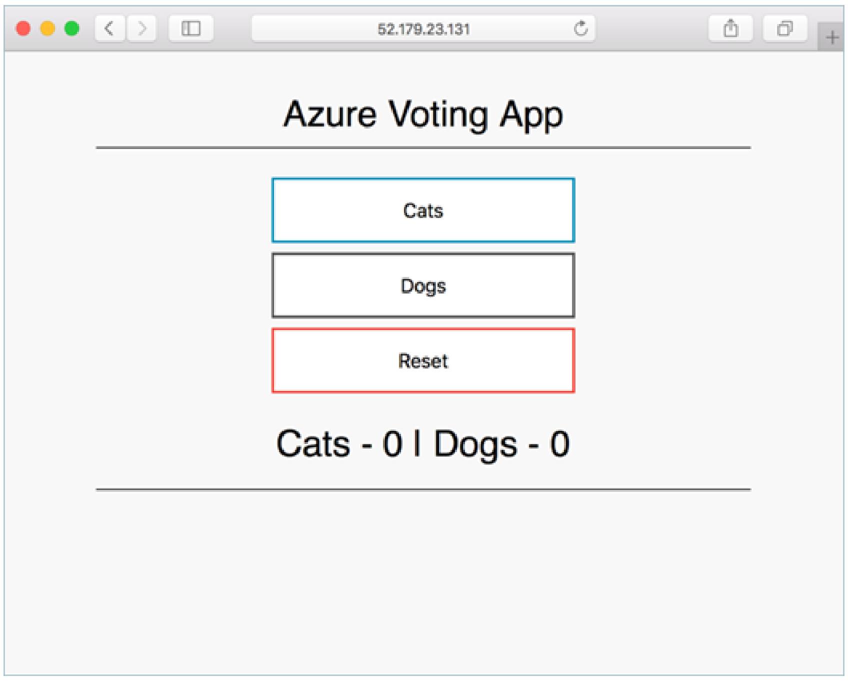
Tutorial: Prepare an application for Azure Kubernetes Service (AKS)

10/4/2020 • 3 minutes to read • Edit Online

In this tutorial, part one of seven, a multi-container application is prepared for use in Kubernetes. Existing development tools such as Docker Compose are used to locally build and test an application. You learn how to:

- Clone a sample application source from GitHub
- Create a container image from the sample application source
- Test the multi-container application in a local Docker environment

Once completed, the following application runs in your local development environment:



In additional tutorials, the container image is uploaded to an Azure Container Registry, and then deployed into an AKS cluster.

Before you begin

This tutorial assumes a basic understanding of core Docker concepts such as containers, container images, and `docker` commands. For a primer on container basics, see Get started with Docker.

To complete this tutorial, you need a local Docker development environment running Linux containers. Docker provides packages that configure Docker on a Mac, Windows, or Linux system.

Azure Cloud Shell does not include the Docker components required to complete every step in these tutorials. Therefore, we recommend using a full Docker development environment.

Get application code

The sample application used in this tutorial is a basic voting app. The application consists of a front-end web component and a back-end Redis instance. The web component is packaged into a custom container image. The Redis instance uses an unmodified image from Docker Hub.

Use git to clone the sample application to your development environment:

```
git clone https://github.com/Azure-Samples/azure-voting-app-redis.git
```

Change into the cloned directory.

```
cd azure-voting-app-redis
```

Inside the directory is the application source code, a pre-created Docker compose file, and a Kubernetes manifest file. These files are used throughout the tutorial set.

Create container images

Docker Compose can be used to automate building container images and the deployment of multi-container applications.

Use the sample `docker-compose.yaml` file to create the container image, download the Redis image, and start the application:

```
docker-compose up -d
```

When completed, use the docker images command to see the created images. Three images have been downloaded or created. The *azure-vote-front* image contains the front-end application and uses the *nginx-flask* image as a base. The *redis* image is used to start a Redis instance.

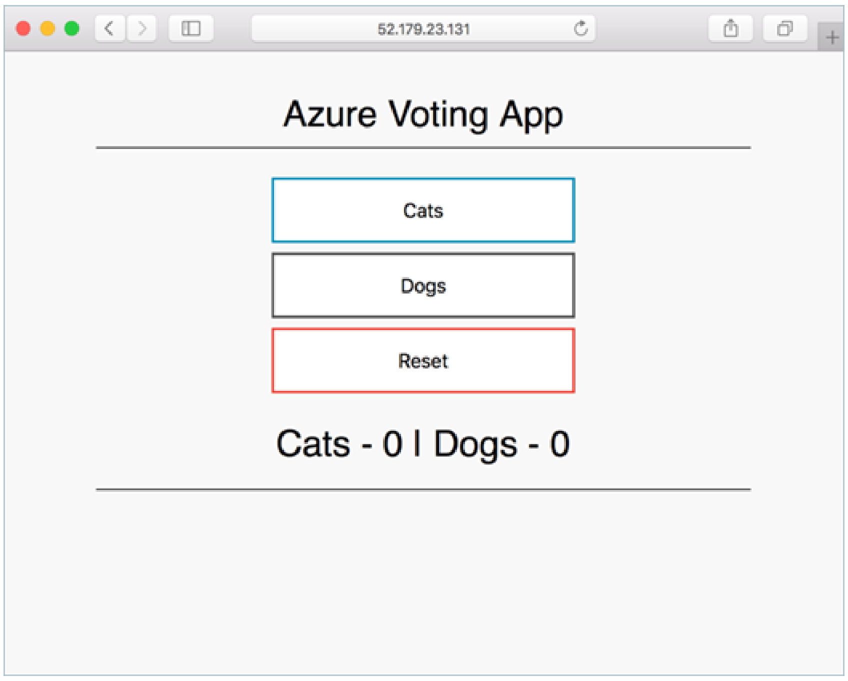
```
$ docker images

REPOSITORY                                   TAG                 IMAGE ID            CREATED             SIZE
mcr.microsoft.com/azuredocs/azure-vote-front v1                  84b41c268ad9        9 seconds ago
944MB
mcr.microsoft.com/oss/bitnami/redis          6.0.8               3a54a920bb6c        2 days ago
103MB
tiangolo/uwsgi-nginx-flask                   python3.6           a16ce562e863        6 weeks ago
944MB
```

Run the docker ps command to see the running containers:

```
$ docker ps

CONTAINER ID        IMAGE                                             COMMAND                  CREATED
STATUS              PORTS                              NAMES
d10e5244f237        mcr.microsoft.com/azuredocs/azure-vote-front:v1   "/entrypoint.sh /sta…"   3 minutes ago
Up 3 minutes        443/tcp, 0.0.0.0:8080->80/tcp      azure-vote-front
21574cb38c1f        mcr.microsoft.com/oss/bitnami/redis:6.0.8         "/opt/bitnami/script…"   3 minutes ago
Up 3 minutes        0.0.0.0:6379->6379/tcp             azure-vote-back
```

Test application locally

To see the running application, enter `http://localhost:8080` in a local web browser. The sample application loads, as shown in the following example:



Clean up resources

Now that the application's functionality has been validated, the running containers can be stopped and removed. Do not delete the container images - in the next tutorial, the *azure-vote-front* image is uploaded to an Azure Container Registry instance.

Stop and remove the container instances and resources with the docker-compose down command:

```
docker-compose down
```

When the local application has been removed, you have a Docker image that contains the Azure Vote application, *azure-vote-front*, for use with the next tutorial.

Next steps

In this tutorial, an application was tested and container images created for the application. You learned how to:

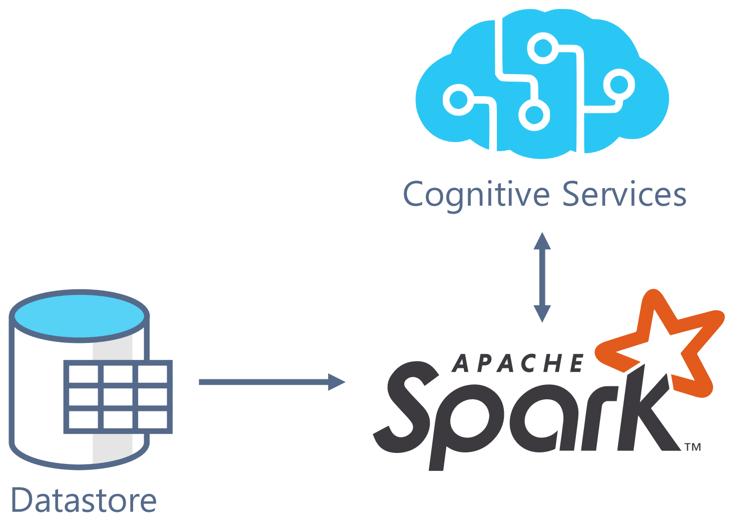
- Clone a sample application source from GitHub
- Create a container image from the sample application source
- Test the multi-container application in a local Docker environment

Advance to the next tutorial to learn how to store container images in Azure Container Registry.

Push images to Azure Container Registry

Azure Cognitive Services for Big Data

10/4/2020 • 5 minutes to read • Edit Online



The Azure Cognitive Services for Big Data lets users channel terabytes of data through Cognitive Services using Apache Spark™. With the Cognitive Services for Big Data, it's easy to create large-scale intelligent applications with any datastore.

With Cognitive Services for Big Data you can embed continuously improving, intelligent models directly into Apache Spark™ and SQL computations. These tools liberate developers from low-level networking details, so that they can focus on creating smart, distributed applications.

Features and benefits

Cognitive Services for Big Data can use services from any region in the world, as well as containerized Cognitive Services. Containers support low or no connectivity deployments with ultra-low latency responses. Containerized Cognitive Services can be run locally, directly on the worker nodes of your Spark cluster, or on an external orchestrator like Kubernetes.

Supported services

Cognitive Services, accessed through APIs and SDKs, help developers build intelligent applications without having AI or data science skills. With Cognitive Services you can make your applications see, hear, speak, understand, and reason. To use the Cognitive Services, your application must send data to the service over the network. Once received, the service sends an intelligent response in return. The following services are available for big data workloads:

Vision

SERVICE NAME	SERVICE DESCRIPTION
Computer Vision	The Computer Vision service provides you with access to advanced algorithms for processing images and returning information.
Face	The Face service provides access to advanced face algorithms, enabling face attribute detection and recognition.

Speech

SERVICE NAME	SERVICE DESCRIPTION
Speech service	The Speech service provides access to features like speech recognition, speech synthesis, speech translation, and speaker verification and identification.

Decision

SERVICE NAME	SERVICE DESCRIPTION
Anomaly Detector	The Anomaly Detector (Preview) service allows you to monitor and detect abnormalities in your time series data.

Language

SERVICE NAME	SERVICE DESCRIPTION
Text Analytics	The Text Analytics service provides natural language processing over raw text for sentiment analysis, key-phrase extraction, and language detection.

Search

SERVICE NAME	SERVICE DESCRIPTION
Bing Image Search	The Bing Image Search service returns a display of images determined to be relevant to the user's query.

Supported programming languages for Cognitive Services for Big Data

The Cognitive Services for Big Data are built on Apache Spark. Apache Spark is a distributed computing library that supports Java, Scala, Python, R, and many other languages. These languages are currently supported.

Python

We provide a PySpark API in the `mmlspark.cognitive` namespace of Microsoft ML for Apache Spark. For more information, see the Python Developer API. For usage examples, see the Python Samples.

Scala and Java

We provide a Scala and Java-based Spark API in the `com.microsoft.ml.spark.cognitive` namespace of Microsoft ML for Apache Spark. For more information, see the Scala Developer API. For usage examples, see the Scala Samples.

Supported platforms and connectors

The Cognitive Services for Big Data requires Apache Spark. There are several Apache Spark platforms that support the Cognitive Services for Big Data.

Azure Databricks

Azure Databricks is an Apache Spark-based analytics platform optimized for the Microsoft Azure cloud services platform. It provides one-click setup, streamlined work-flows, and an interactive workspace that supports collaboration between data scientists, data engineers, and business analysts.

Azure Synapse Analytics

Azure Synapse Analytics (formerly SQL Data Warehouse) is as enterprise data warehouse that uses massive parallel processing. With Synapse Analytics, you can quickly run complex queries across petabytes of data. Azure Synapse Analytics provides managed Spark Pools to run Spark Jobs with an intuitive Jupyter Notebook Interface.

Azure Kubernetes Service

Azure Kubernetes Service (AKS) orchestrates Docker Containers and distributed applications at massive scales. AKS is a managed Kubernetes offering that simplifies using Kubernetes in Azure. Kubernetes can enable fine-grained control of Cognitive Service scale, latency, and networking. However, we recommend using Azure Databricks or Synapse Analytics if you're unfamiliar with Apache Spark.

Data Connectors

Once you have a Spark Cluster, the next step is connecting to your data. Apache Spark has a broad collection of database connectors. These connectors allow applications to work with large datasets no matter where they're stored. For more information about supported databases and connectors, see the list of supported datasources for Azure Databricks.

Concepts

Spark

Apache Spark™ is a unified analytics engine for large-scale data processing. Its parallel processing framework boosts performance of big data and analytic applications. Spark can operate as both a batch and stream processing system, without changing core application code.

The basis of Spark is the DataFrame: a tabular collection of data distributed across the Apache Spark worker nodes. A Spark DataFrame is like a table in a relational database or a data frame in R/Python, but with limitless scale. DataFrames can be constructed from many sources such as: structured data files, tables in Hive, or external databases. Once your data is in a Spark DataFrame, you can:

- Do SQL-style computations such as join and filter tables.
- Apply functions to large datasets using MapReduce style parallelism.
- Apply Distributed Machine Learning using Microsoft Machine Learning for Apache Spark.
- Use the Cognitive Services for Big Data to enrich your data with ready-to-use intelligent services.

Microsoft Machine Learning for Apache Spark (MMLSpark)

Microsoft Machine Learning for Apache Spark (MMLSpark) is an open-source, distributed machine learning library (ML) built on Apache Spark. The Cognitive Services for Big Data is included in this package. Additionally, MMLSpark contains several other ML tools for Apache Spark, such as LightGBM, Vowpal Wabbit, OpenCV, LIME, and more. With MMLSpark, you can build powerful predictive and analytical models from any Spark datasource.

HTTP on Spark

Cognitive Services for Big Data is an example of how we can integrate intelligent web services with big data. Web services power many applications across the globe and most services communicate through the Hypertext Transfer Protocol (HTTP). To work with *arbitrary* web services at large scales, we provide HTTP on Spark. With HTTP on Spark, you can pass terabytes of data through any web service. Under the hood, we use this technology to power Cognitive Services for Big Data.

Developer samples

- Recipe: Predictive Maintenance
- Recipe: Intelligent Art Exploration

Blog posts

- Learn more about how Cognitive Services work on Apache Spark™
- Saving Snow Leopards with Deep Learning and Computer Vision on Spark

- Microsoft Research Podcast: MMLSpark, empowering AI for Good with Mark Hamilton

Webinars and videos

- The Azure Cognitive Services on Spark: Clusters with Embedded Intelligent Services
- Spark Summit Keynote: Scalable AI for Good
- The Cognitive Services for Big Data in Cosmos DB

Next steps

- Getting Started with the Cognitive Services for Big Data
- Simple Python Examples
- Simple Scala Examples

Getting started

10/4/2020 • 4 minutes to read • Edit Online

Setting up your environment is the first step to building a pipeline for your data. After your environment is ready, running a sample is quick and easy.

In this article, we'll perform these steps to get you started:

1. Create a Cognitive Services resource
2. Create an Apache Spark Cluster
3. Try a sample

Create a Cognitive Services resource

To use the Big Data Cognitive Services, we must first create a Cognitive Service for our workflow. There are two main types of Cognitive Services: cloud services hosted in Azure and containerized services managed by users. We recommend starting with the simpler cloud-based Cognitive Services.

Cloud services

Cloud-based Cognitive Services are intelligent algorithms hosted in Azure. These services are ready for use without training, you just need an internet connection. You can create a Cognitive Service in the Azure portal or with the Azure CLI.

Containerized services (optional)

If your application or workload uses extremely large datasets, requires private networking, or can't contact the cloud, communicating with cloud services might be impossible. In this situation, containerized Cognitive Services have these benefits:

- **Low Connectivity**: You can deploy containerized Cognitive Services in any computing environment, both on-cloud and off. If your application can't contact the cloud, consider deploying containerized Cognitive Services on your application.
- **Low Latency**: Because containerized services don't require the round-trip communication to/from the cloud, responses are returned with much lower latencies.
- **Privacy and Data Security**: You can deploy containerized services into private networks, so that sensitive data doesn't leave the network.
- **High Scalability**: Containerized services don't have "rate limits" and run on user-managed computers. So, you can scale Cognitive Services without end to handle much larger workloads.

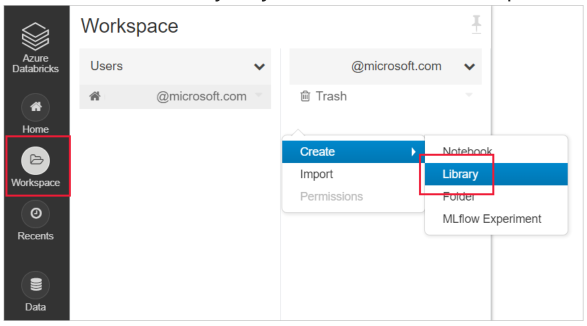
Follow this guide to create a containerized Cognitive Service.

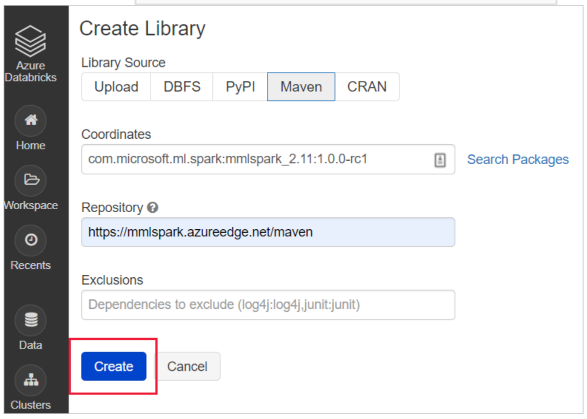
Create an Apache Spark cluster

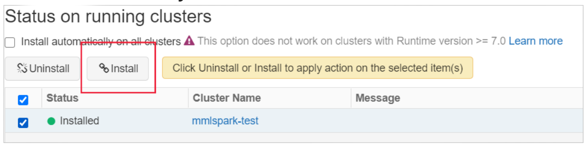
Apache Spark™ is a distributed computing framework designed for big-data data processing. Users can work with Apache Spark in Azure with services like Azure Databricks, Azure Synapse Analytics, HDInsight, and Azure Kubernetes Services. To use the Big Data Cognitive Services, we must first create a cluster. If you already have a Spark cluster, feel free to try an example.

Azure Databricks

Azure Databricks is an Apache Spark-based analytics platform with a one-click setup, streamlined workflows, and an interactive workspace. It's often used to collaborate between data scientists, engineers, and business analysts. To use the Big Data Cognitive Services on Azure Databricks, follow these steps:

1. Create an Azure Databricks workspace
2. Create a Spark cluster in Databricks
3. Install the Big Data Cognitive Services
   - Create a new library in your databricks workspace
     

   - Input the following maven coordinates Coordinates: `com.microsoft.ml.spark:mmlspark_2.11:1.0.0-rc1` Repository: `https://mmlspark.azureedge.net/maven`
     

     Library Source: Maven
     Coordinates: com.microsoft.ml.spark:mmlspark_2.11:1.0.0-rc1
     Repository: https://mmlspark.azureedge.net/maven
     Exclusions: Dependencies to exclude (log4j:log4j,junit:junit)
   - Install the library onto a cluster
     

     Install automatically on all clusters ⚠ This option does not work on clusters with Runtime version >= 7.0

     Click Uninstall or Install to apply action on the selected item(s)

✓	Status	Cluster Name	Message
✓	● Installed	mmlspark-test	

Synapse Analytics (optional)

Optionally, you can use Synapse Analytics to create a spark cluster. Azure Synapse Analytics brings together enterprise data warehousing and big data analytics. It gives you the freedom to query data on your terms, using either serverless on-demand or provisioned resources at scale. To get started using Synapse Analytics, follow these steps:

1. Create a Synapse Workspace (preview).
2. Create a new Apache Spark pool (preview) using the Azure portal.

In Synapse Analytics, Big Data for Cognitive Services is installed by default.

Azure Kubernetes Service

If you're using containerized Cognitive Services, one popular option for deploying Spark alongside containers is the Azure Kubernetes Service.

To get started on Azure Kubernetes Service, follow these steps:

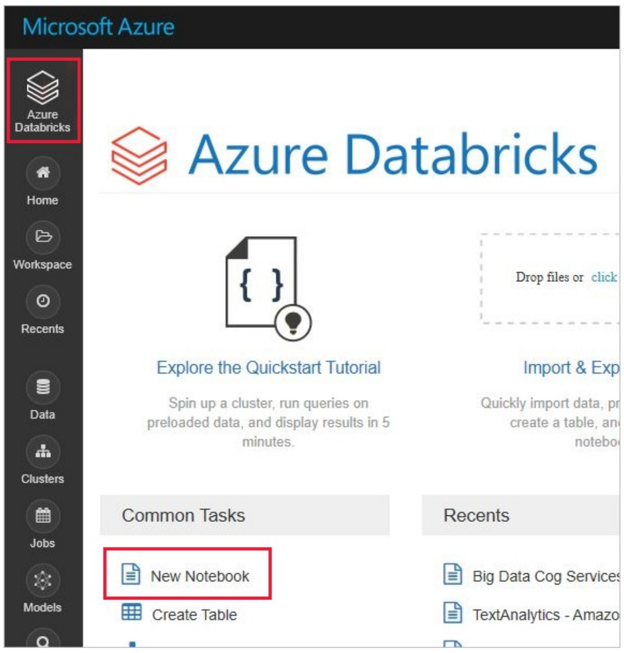
1. Deploy an Azure Kubernetes Service (AKS) cluster using the Azure portal
2. Install the Apache Spark 2.4.0 helm chart
3. Install a cognitive service container using Helm

Try a sample

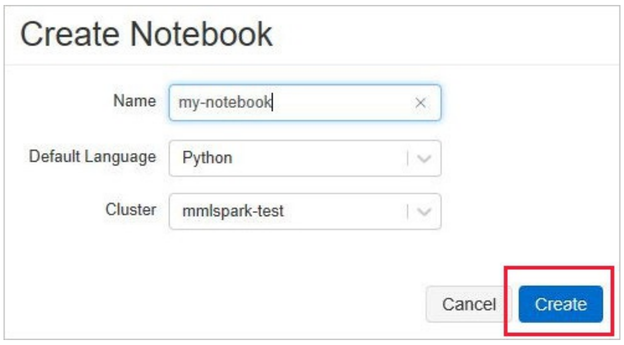
After you set up your Spark cluster and environment, we can run a short sample. This section demonstrates how to use the Big Data for Cognitive Services in Azure Databricks.

First, we can create a notebook in Azure Databricks. For other Spark cluster providers, use their notebooks or Spark Submit.

1. Create a new Databricks notebook, by choosing **New notebook** from the **Azure Databricks** menu.



2. In the **Create Notebook** dialog box, enter a name, select **Python** as the language, and select the Spark cluster that you created earlier.


Create Notebook

Name: my-notebook

Default Language: Python

Cluster: mmlspark-test

Select **Create**.

3. Paste this code snippet below into your new notebook.

```
from mmlspark.cognitive import *
from pyspark.sql.functions import col

# Add your subscription key from Text Analytics (or a general Cognitive Service key)
service_key = "ADD-SUBSCRIPTION-KEY-HERE"

df = spark.createDataFrame([
  ("I am so happy today, its sunny!", "en-US"),
  ("I am frustrated by this rush hour traffic", "en-US"),
  ("The cognitive services on spark aint bad", "en-US"),
], ["text", "language"])

sentiment = (TextSentiment()
    .setTextCol("text")
    .setLocation("eastus")
    .setSubscriptionKey(service_key)
    .setOutputCol("sentiment")
    .setErrorCol("error")
    .setLanguageCol("language"))

results = sentiment.transform(df)

# Show the results in a table
display(results.select("text", col("sentiment")[0].getItem("score").alias("sentiment")))
```

1. Get your subscription key from the **Keys and Endpoint** menu from your Text Analytics dashboard in the Azure portal.
2. Replace the subscription key placeholder in your Databricks notebook code with your subscription key.
3. Select the play, or triangle, symbol in the upper right of your notebook cell to run the sample. Optionally, select **Run All** at the top of your notebook to run all cells. The answers will display below the cell in a table.

Expected results

TEXT	SENTIMENT
I am so happy today, its sunny!	0.978959
I am frustrated by this rush hour traffic	0.0237956
The cognitive services on spark aint bad	0.888896

Next steps

- Short Python Examples
- Short Scala Examples
- Recipe: Predictive Maintenance
- Recipe: Intelligent Art Exploration

Python Samples for Cognitive Services for Big Data

10/4/2020 • 6 minutes to read • Edit Online

The following snippets are ready to run and will help get you started with using Cognitive Services on Spark with Python.

The samples in this article use these Cognitive Services:

- Text Analytics - get the sentiment (or mood) of a set of sentences.
- Computer Vision - get the tags (one-word descriptions) associated with a set of images.
- Bing Image Search - search the web for images related to a natural language query.
- Speech-to-text - transcribe audio files to extract text-based transcripts.
- Anomaly Detector - detect anomalies within a time series data.

Prerequisites

1. Follow the steps in Getting started to set up your Azure Databricks and Cognitive Services environment. This tutorial shows you how to install MMLSpark and how to create your Spark cluster in Databricks.
2. After you create a new notebook in Azure Databricks, copy the **Shared code** below and paste into a new cell in your notebook.
3. Choose a service sample, below, and copy paste it into a second new cell in your notebook.
4. Replace any of the service subscription key placeholders with your own key.
5. Choose the run button (triangle icon) in the upper right corner of the cell, then select **Run Cell**.
6. View results in a table below the cell.

Shared code

To get started, we'll need to add this code to the project:

```
from mmlspark.cognitive import *

# A general Cognitive Services key for Text Analytics and Computer Vision (or use separate keys that belong to
each service)
service_key = "ADD_YOUR_SUBSCRIPION_KEY"
# A Bing Search v7 subscription key
bing_search_key = "ADD_YOUR_SUBSCRIPION_KEY"
# An Anomaly Dectector subscription key
anomaly_key = "ADD_YOUR_SUBSCRIPION_KEY"

# Validate the key
assert service_key != "ADD_YOUR_SUBSCRIPION_KEY"
```

Text Analytics sample

The Text Analytics service provides several algorithms for extracting intelligent insights from text. For example, we can find the sentiment of given input text. The service will return a score between 0.0 and 1.0 where low scores indicate negative sentiment and high score indicates positive sentiment. This sample uses three simple sentences and returns the sentiment for each.

```
from pyspark.sql.functions import col

# Create a dataframe that's tied to it's column names
df = spark.createDataFrame([
  ("I am so happy today, its sunny!", "en-US"),
  ("I am frustrated by this rush hour traffic", "en-US"),
  ("The cognitive services on spark aint bad", "en-US"),
], ["text", "language"])

# Run the Text Analytics service with options
sentiment = (TextSentiment()
    .setTextCol("text")
    .setLocation("eastus")
    .setSubscriptionKey(service_key)
    .setOutputCol("sentiment")
    .setErrorCol("error")
    .setLanguageCol("language"))

# Show the results of your text query in a table format
display(sentiment.transform(df).select("text", col("sentiment")[0].getItem("score").alias("sentiment")))
```

Expected result

TEXT	SENTIMENT
I am so happy today, its sunny!	0.9789592027664185
I am frustrated by this rush hour traffic	0.023795604705810547
The cognitive services on spark aint bad	0.8888956308364868

Computer Vision sample

Computer Vision analyzes images to identify structure such as faces, objects, and natural-language descriptions. In this sample, we tag a list of images. Tags are one-word descriptions of things in the image like recognizable objects, people, scenery, and actions.

```

# Create a dataframe with the image URLs
df = spark.createDataFrame([
        ("https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/objects.jpg", ),
        ("https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/dog.jpg", ),
        ("https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/house.jpg", )
    ], ["image", ])

# Run the Computer Vision service. Analyze Image extracts infortmation from/about the images.
analysis = (AnalyzeImage()
    .setLocation("eastus")
    .setSubscriptionKey(service_key)
    .setVisualFeatures(["Categories","Color","Description","Faces","Objects","Tags"])
    .setOutputCol("analysis_results")
    .setImageUrlCol("image")
    .setErrorCol("error"))

# Show the results of what you wanted to pull out of the images.
display(analysis.transform(df).select("image", "analysis_results.description.tags"))
```

Expected result

IMAGE	TAGS
https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/objects.jpg	['skating' 'person' 'man' 'outdoor' 'riding' 'sport' 'skateboard' 'young' 'board' 'shirt' 'air' 'black' 'park' 'boy' 'side' 'jumping' 'trick' 'ramp' 'doing' 'flying']
https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/dog.jpg	['dog' 'outdoor' 'fence' 'wooden' 'small' 'brown' 'building' 'sitting' 'front' 'bench' 'standing' 'table' 'walking' 'board' 'beach' 'white' 'holding' 'bridge' 'track']
https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/house.jpg	['outdoor' 'grass' 'house' 'building' 'old' 'home' 'front' 'small' 'church' 'stone' 'large' 'grazing' 'yard' 'green' 'sitting' 'leading' 'sheep' 'brick' 'bench' 'street' 'white' 'country' 'clock' 'sign' 'parked' 'field' 'standing' 'garden' 'water' 'red' 'horse' 'man' 'tall' 'fire' 'group']

Bing Image Search sample

Bing Image Search searches the web to retrieve images related to a user's natural language query. In this sample, we use a text query that looks for images with quotes. It returns a list of image URLs that contain photos related to our query.

```
from pyspark.ml import PipelineModel

# Number of images Bing will return per query
imgsPerBatch = 10
# A list of offsets, used to page into the search results
offsets = [(i*imgsPerBatch,) for i in range(100)]
# Since web content is our data, we create a dataframe with options on that data: offsets
bingParameters = spark.createDataFrame(offsets, ["offset"])

# Run the Bing Image Search service with our text query
bingSearch = (BingImageSearch()
    .setSubscriptionKey(bing_search_key)
    .setOffsetCol("offset")
    .setQuery("Martin Luther King Jr. quotes")
    .setCount(imgsPerBatch)
    .setOutputCol("images"))

# Transformer that extracts and flattens the richly structured output of Bing Image Search into a simple URL
column
getUrls = BingImageSearch.getUrlTransformer("images", "url")

# This displays the full results returned, uncomment to use
# display(bingSearch.transform(bingParameters))

# Since we have two services, they are put into a pipeline
pipeline = PipelineModel(stages=[bingSearch, getUrls])

# Show the results of your search: image URLs
display(pipeline.transform(bingParameters))
```

Expected result

URL
https://iheartintelligence.com/wp-content/uploads/2019/01/powerful-quotes-martin-luther-king-jr.jpg

URL
http://everydaypowerblog.com/wp-content/uploads/2014/01/Martin-Luther-King-Jr.-Quotes-16.jpg
http://www.sofreshandsogreen.com/wp-content/uploads/2012/01/martin-luther-king-jr-quote-sofreshandsogreendotcom.jpg
https://everydaypowerblog.com/wp-content/uploads/2014/01/Martin-Luther-King-Jr.-Quotes-18.jpg
https://tsal-eszuskq0bptlfh8awbb.stackpathdns.com/wp-content/uploads/2018/01/MartinLutherKingQuotes.jpg

Speech-to-Text sample

The Speech-to-text service converts streams or files of spoken audio to text. In this sample, we transcribe two audio files. The first file is easy to understand, and the second is more challenging.

```
# Create a dataframe with our audio URLs, tied to the column called "url"
df = spark.createDataFrame([("https://mmlspark.blob.core.windows.net/datasets/Speech/audio2.wav",),
                            ("https://mmlspark.blob.core.windows.net/datasets/Speech/audio3.mp3",)
                           ], ["url"])

# Run the Speech-to-text service to translate the audio into text
speech_to_text = (SpeechToTextSDK()
    .setSubscriptionKey(service_key)
    .setLocation("eastus")
    .setOutputCol("text")
    .setAudioDataCol("url")
    .setLanguage("en-US")
    .setProfanity("Masked"))

# Show the results of the translation
display(speech_to_text.transform(df).select("url", "text.DisplayText"))
```

Expected result

URL	DISPLAYTEXT
https://mmlspark.blob.core.windows.net/datasets/Speech/audio2.wav	Custom speech provides tools that allow you to visually inspect the recognition quality of a model by comparing audio data with the corresponding recognition result from the custom speech portal. You can playback uploaded audio and determine if the provided recognition result is correct. This tool allows you to quickly inspect quality of Microsoft's baseline speech to text model or a trained custom model without having to transcribe any audio data.
https://mmlspark.blob.core.windows.net/datasets/Speech/audio3.mp3	Add a gentleman Sir thinking visual check.
https://mmlspark.blob.core.windows.net/datasets/Speech/audio3.mp3	I hear me.
https://mmlspark.blob.core.windows.net/datasets/Speech/audio3.mp3	I like the reassurance for radio that I can hear it as well.

Anomaly Detector sample

Anomaly Detector is great for detecting irregularities in your time series data. In this sample, we use the service to find anomalies in the entire time series.

```
from pyspark.sql.functions import lit

# Create a dataframe with the point data that Anomaly Detector requires
df = spark.createDataFrame([
    ("1972-01-01T00:00:00Z", 826.0),
    ("1972-02-01T00:00:00Z", 799.0),
    ("1972-03-01T00:00:00Z", 890.0),
    ("1972-04-01T00:00:00Z", 900.0),
    ("1972-05-01T00:00:00Z", 766.0),
    ("1972-06-01T00:00:00Z", 805.0),
    ("1972-07-01T00:00:00Z", 821.0),
    ("1972-08-01T00:00:00Z", 20000.0),
    ("1972-09-01T00:00:00Z", 883.0),
    ("1972-10-01T00:00:00Z", 898.0),
    ("1972-11-01T00:00:00Z", 957.0),
    ("1972-12-01T00:00:00Z", 924.0),
    ("1973-01-01T00:00:00Z", 881.0),
    ("1973-02-01T00:00:00Z", 837.0),
    ("1973-03-01T00:00:00Z", 9000.0)
], ["timestamp", "value"]).withColumn("group", lit("series1"))

# Run the Anomaly Detector service to look for irregular data
anamoly_detector = (SimpleDetectAnomalies()
  .setSubscriptionKey(anomaly_key)
  .setLocation("eastus")
  .setTimestampCol("timestamp")
  .setValueCol("value")
  .setOutputCol("anomalies")
  .setGroupbyCol("group")
  .setGranularity("monthly"))

# Show the full results of the analysis with the anomalies marked as "True"
display(anamoly_detector.transform(df).select("timestamp", "value", "anomalies.isAnomaly"))
```

Expected result

TIMESTAMP	VALUE	ISANOMALY
1972-01-01T00:00:00Z	826	False
1972-02-01T00:00:00Z	799	False
1972-03-01T00:00:00Z	890	False
1972-04-01T00:00:00Z	900	False
1972-05-01T00:00:00Z	766	False
1972-06-01T00:00:00Z	805	False
1972-07-01T00:00:00Z	821	False
1972-08-01T00:00:00Z	20000	True
1972-09-01T00:00:00Z	883	False

TIMESTAMP	VALUE	ISANOMALY
1972-10-01T00:00:00Z	898	False
1972-11-01T00:00:00Z	957	False
1972-12-01T00:00:00Z	924	False
1973-01-01T00:00:00Z	881	False
1973-02-01T00:00:00Z	837	False
1973-03-01T00:00:00Z	9000	True

Arbitrary web APIs

With HTTP on Spark, any web service can be used in your big data pipeline. In this example, we use the World Bank API to get information about various countries around the world.

```
from requests import Request
from mmlspark.io.http import HTTPTransformer, http_udf
from pyspark.sql.functions import udf, col

# Use any requests from the python requests library
def world_bank_request(country):
  return Request("GET", "http://api.worldbank.org/v2/country/{}?format=json".format(country))

# Create a dataframe with spcificies which countries we want data on
df = (spark.createDataFrame([("br",),("usa",)], ["country"])
  .withColumn("request", http_udf(world_bank_request)(col("country"))))

# Much faster for big data because of the concurrency :)
client = (HTTPTransformer()
      .setConcurrency(3)
      .setInputCol("request")
      .setOutputCol("response"))

# Get the body of the response
def get_response_body(resp):
  return resp.entity.content.decode()

# Show the details of the country data returned
display(client.transform(df).select("country", udf(get_response_body)(col("response")).alias("response")))
```

Expected result

COUNTRY	RESPONSE
br	[{"page":1,"pages":1,"per_page":"50","total":1}, [{"id":"BRA","iso2Code":"BR","name":"Brazil","region": {"id":"LCN","iso2code":"ZJ","value":"Latin America & Caribbean "},"adminregion":{"id":"LAC","iso2code":"XJ","value":"Latin America & Caribbean (excluding high income)"},"incomeLevel": {"id":"UMC","iso2code":"XT","value":"Upper middle income"},"lendingType": {"id":"IBD","iso2code":"XF","value":"IBRD"},"capitalCity":"Brasilia", "longitude":"-47.9292","latitude":"-15.7801"}]]

COUNTRY	RESPONSE
usa	[{"page":1,"pages":1,"per_page":"50","total":1},[{"id":"USA","iso2Code":"US","name":"United States","region":{"id":"NAC","iso2code":"XU","value":"North America"},"adminregion":{"id":"","iso2code":"","value":""},"incomeLevel":{"id":"HIC","iso2code":"XD","value":"High income"},"lendingType":{"id":"LNX","iso2code":"XX","value":"Not classified"},"capitalCity":"Washington D.C.","longitude":"-77.032","latitude":"38.8895"}]]

See also

- Recipe: Anomaly Detection
- Recipe: Art Explorer

Quick Examples

10/4/2020 • 5 minutes to read • Edit Online

The following snippets are ready to run and will help get you started with using Cognitive Services on Spark. The samples below are in Scala.

The samples use these Cognitive Services:

- Text Analytics - get the sentiment (or mood) of a set of sentences.
- Computer Vision - get the tags (one-word descriptions) associated with a set of images.
- Bing Image Search - search the web for images related to a natural language query.
- Speech-to-text - transcribe audio files to extract text-based transcripts.
- Anomaly Detector - detect anomalies within a time series data.

Prerequisites

1. Follow the steps in Getting started to set up your Azure Databricks and Cognitive Services environment. This tutorial will include how to install MMLSpark and how to create your Spark cluster in Databricks.
2. After you create a new notebook in Azure Databricks, copy the **Shared code** below and paste into a new cell in your notebook.
3. Choose a service sample, below, and copy paste it into a second new cell in your notebook.
4. Replace any of the service subscription key placeholders with your own key.
5. Choose the run button (triangle icon) in the upper right corner of the cell, then select **Run Cell**.
6. View results in a table below the cell.

Shared code

To get started, add this code to your project:

```
import com.microsoft.ml.spark.cognitive._
import spark.implicits._

val serviceKey = "ADD-YOUR-SUBSCRIPTION-KEY"
val location = "eastus"
```

Text Analytics

The Text Analytics service provides several algorithms for extracting intelligent insights from text. For example, we can find the sentiment of given input text. The service will return a score between `0.0` and `1.0` where low scores indicate negative sentiment and high score indicates positive sentiment. The sample below uses three simple sentences and returns the sentiment score for each.

```
import org.apache.spark.sql.functions.col

val df = Seq(
  ("I am so happy today, its sunny!", "en-US"),
  ("I am frustrated by this rush hour traffic", "en-US"),
  ("The cognitive services on spark aint bad", "en-US")
).toDF("text", "language")

val sentiment = new TextSentiment()
    .setTextCol("text")
    .setLocation(location)
    .setSubscriptionKey(serviceKey)
    .setOutputCol("sentiment")
    .setErrorCol("error")
    .setLanguageCol("language")

display(sentiment.transform(df).select(col("text"), col("sentiment")(0).getItem("score").alias("sentiment")))
```

Expected result

TEXT	SENTIMENT
I am so happy today, its sunny!	0.9789592027664185
I am frustrated by this rush hour traffic	0.023795604705810547
The cognitive services on spark aint bad	0.8888956308364868

Computer Vision

Computer Vision analyzes images to identify structure such as faces, objects, and natural-language descriptions. In this sample, we tag a list of images. Tags are one-word descriptions of things in the image like recognizable objects, people, scenery, and actions.

```
// Create a dataframe with the image URLs
val df = Seq(
    ("https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-
files/master/ComputerVision/Images/objects.jpg"),
    ("https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-
files/master/ComputerVision/Images/dog.jpg"),
    ("https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-
files/master/ComputerVision/Images/house.jpg")
).toDF("image")

// Run the Computer Vision service. Analyze Image extracts infortmation from/about the images.
val analysis = new AnalyzeImage()
    .setLocation(location)
    .setSubscriptionKey(serviceKey)
    .setVisualFeatures(Seq("Categories","Color","Description","Faces","Objects","Tags"))
    .setOutputCol("results")
    .setImageUrlCol("image")
    .setErrorCol("error"))

// Show the results of what you wanted to pull out of the images.
display(analysis.transform(df).select(col("image"),
col("results").getItem("tags").getItem("name")).alias("results")))

// Uncomment for full results with all visual feature requests
//display(analysis.transform(df).select(col("image"), col("results")))
```

Expected result

IMAGE	TAGS
https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/objects.jpg	['skating' 'person' 'man' 'outdoor' 'riding' 'sport' 'skateboard' 'young' 'board' 'shirt' 'air' 'black' 'park' 'boy' 'side' 'jumping' 'trick' 'ramp' 'doing' 'flying']
https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/dog.jpg	['dog' 'outdoor' 'fence' 'wooden' 'small' 'brown' 'building' 'sitting' 'front' 'bench' 'standing' 'table' 'walking' 'board' 'beach' 'white' 'holding' 'bridge' 'track']
https://raw.githubusercontent.com/Azure-Samples/cognitive-services-sample-data-files/master/ComputerVision/Images/house.jpg	['outdoor' 'grass' 'house' 'building' 'old' 'home' 'front' 'small' 'church' 'stone' 'large' 'grazing' 'yard' 'green' 'sitting' 'leading' 'sheep' 'brick' 'bench' 'street' 'white' 'country' 'clock' 'sign' 'parked' 'field' 'standing' 'garden' 'water' 'red' 'horse' 'man' 'tall' 'fire' 'group']

Bing Image Search

Bing Image Search searches the web to retrieve images related to a user's natural language query. In this sample, we use a text query that looks for images with quotes. It returns a list of image URLs that contain photos related to our query.

```
import org.apache.spark.ml.Pipeline

// Number of images Bing will return per query
val imgsPerBatch = 10

// A list of offsets, used to page into the search results
val df = (0 until 100).map(o => Tuple1(o*imgsPerBatch)).toSeq.toDF("offset")

// Run the Bing Image Search service with our text query
val bingSearch = new BingImageSearch()
    .setSubscriptionKey(bingSearchKey)
    .setOffsetCol("offset")
    .setQuery("Martin Luther King Jr. quotes")
    .setCount(imgsPerBatch)
    .setOutputCol("images")

// Transformer that extracts and flattens the richly structured output of Bing Image Search into a simple URL
column
val getUrls = BingImageSearch.getUrlTransformer("images", "url")

// This displays the full results returned, uncomment to use
// display(bingSearch.transform(bingParameters))

// Since we have two services, they are put into a pipeline
val pipeline = new Pipeline().setStages(Array(bingSearch, getUrls))

// Show the results of your search: image URLs
display(pipeline.fit(df).transform(df))
```

Expected result

URL
https://iheartintelligence.com/wp-content/uploads/2019/01/powerful-quotes-martin-luther-king-jr.jpg
http://everydaypowerblog.com/wp-content/uploads/2014/01/Martin-Luther-King-Jr.-Quotes-16.jpg

URL
http://www.sofreshandsogreen.com/wp-content/uploads/2012/01/martin-luther-king-jr-quote-sofreshandsogreendotcom.jpg
https://everydaypowerblog.com/wp-content/uploads/2014/01/Martin-Luther-King-Jr.-Quotes-18.jpg
https://tsal-eszuskq0bptlfh8awbb.stackpathdns.com/wp-content/uploads/2018/01/MartinLutherKingQuotes.jpg

Speech-to-Text

The Speech-to-text service converts streams or files of spoken audio to text. In this sample, we transcribe two audio files. The first file is easy to understand, and the second is more challenging.

```
import org.apache.spark.sql.functions.col

// Create a dataframe with audio URLs, tied to the column called "url"
val df = Seq(("https://mmlspark.blob.core.windows.net/datasets/Speech/audio2.wav"),
             ("https://mmlspark.blob.core.windows.net/datasets/Speech/audio3.mp3")).toDF("url")

// Run the Speech-to-text service to translate the audio into text
val speechToText = new SpeechToTextSDK()
    .setSubscriptionKey(serviceKey)
    .setLocation("eastus")
    .setOutputCol("text")
    .setAudioDataCol("url")
    .setLanguage("en-US")
    .setProfanity("Masked")

// Show the results of the translation
display(speechToText.transform(df).select(col("url"), col("text").getItem("DisplayText")))
```

Expected result

URL	DISPLAYTEXT
https://mmlspark.blob.core.windows.net/datasets/Speech/audio2.wav	Custom speech provides tools that allow you to visually inspect the recognition quality of a model by comparing audio data with the corresponding recognition result from the custom speech portal. You can playback uploaded audio and determine if the provided recognition result is correct. This tool allows you to quickly inspect quality of Microsoft's baseline speech to text model or a trained custom model without having to transcribe any audio data.
https://mmlspark.blob.core.windows.net/datasets/Speech/audio3.mp3	Add a gentleman Sir thinking visual check.
https://mmlspark.blob.core.windows.net/datasets/Speech/audio3.mp3	I hear me.
https://mmlspark.blob.core.windows.net/datasets/Speech/audio3.mp3	I like the reassurance for radio that I can hear it as well.

Anomaly Detector

Anomaly Detector is great for detecting irregularities in your time series data. In this sample, we use the service to find anomalies in the entire time series.

```
import org.apache.spark.sql.functions.{col, lit}

val anomalyKey = "84a2c303cc7e49f6a44d692c27fb9967"

val df = Seq(
    ("1972-01-01T00:00:00Z", 826.0),
    ("1972-02-01T00:00:00Z", 799.0),
    ("1972-03-01T00:00:00Z", 890.0),
    ("1972-04-01T00:00:00Z", 900.0),
    ("1972-05-01T00:00:00Z", 766.0),
    ("1972-06-01T00:00:00Z", 805.0),
    ("1972-07-01T00:00:00Z", 821.0),
    ("1972-08-01T00:00:00Z", 20000.0),
    ("1972-09-01T00:00:00Z", 883.0),
    ("1972-10-01T00:00:00Z", 898.0),
    ("1972-11-01T00:00:00Z", 957.0),
    ("1972-12-01T00:00:00Z", 924.0),
    ("1973-01-01T00:00:00Z", 881.0),
    ("1973-02-01T00:00:00Z", 837.0),
    ("1973-03-01T00:00:00Z", 9000.0)
  ).toDF("timestamp", "value").withColumn("group", lit("series1"))

// Run the Anomaly Detector service to look for irregular data
val anamolyDetector = new SimpleDetectAnomalies()
    .setSubscriptionKey(anomalyKey)
    .setLocation("eastus")
    .setTimestampCol("timestamp")
    .setValueCol("value")
    .setOutputCol("anomalies")
    .setGroupbyCol("group")
    .setGranularity("monthly")

// Show the full results of the analysis with the anomalies marked as "True"
display(anamolyDetector.transform(df).select("timestamp", "value", "anomalies.isAnomaly"))
```

Expected result

TIMESTAMP	VALUE	ISANOMALY
1972-01-01T00:00:00Z	826	False
1972-02-01T00:00:00Z	799	False
1972-03-01T00:00:00Z	890	False
1972-04-01T00:00:00Z	900	False
1972-05-01T00:00:00Z	766	False
1972-06-01T00:00:00Z	805	False
1972-07-01T00:00:00Z	821	False
1972-08-01T00:00:00Z	20000	True
1972-09-01T00:00:00Z	883	False
1972-10-01T00:00:00Z	898	False

TIMESTAMP	VALUE	ISANOMALY
1972-11-01T00:00:00Z	957	False
1972-12-01T00:00:00Z	924	False
1973-01-01T00:00:00Z	881	False
1973-02-01T00:00:00Z	837	False
1973-03-01T00:00:00Z	9000	True

Recipe: Predictive maintenance with the Cognitive Services for Big Data

10/4/2020 • 3 minutes to read • Edit Online

This recipe shows how you can use Azure Synapse Analytics and Cognitive Services on Spark for predictive maintenance of IoT devices. We'll follow along with the CosmosDB and Synapse Link sample. To keep things simple, in this recipe we'll read the data straight from a CSV file rather than getting streamed data through CosmosDB and Synapse Link. We strongly encourage you to look over the Synapse Link sample.

Hypothetical scenario

The hypothetical scenario is a Power Plant, where IoT devices are monitoring steam turbines. The IoTSignals collection has Revolutions per minute (RPM) and Megawatts (MW) data for each turbine. Signals from steam turbines are being analyzed and anomalous signals are detected.

There could be outliers in the data in random frequency. In those situations, RPM values will go up and MW output will go down, for circuit protection. The idea is to see the data varying at the same time, but with different signals.

Prerequisites

- An Azure subscription - Create one for free
- Azure Synapse workspace configured with a Spark pool

Setup

Create an Anomaly Detector resource

Azure Cognitive Services are represented by Azure resources that you subscribe to. Create a resource for Translator using the Azure portal or Azure CLI. You can also:

- View an existing resource in the Azure portal.

Make note of the endpoint and the key for this resource, you'll need it in this guide.

Enter your service keys

Let's start by adding your key and location.

```
service_key = None # Paste your anomaly detector key here
location = None # Paste your anomaly detector location here

assert (service_key is not None)
assert (location is not None)
```

Read data into a DataFrame

Next, let's read the IoTSignals file into a DataFrame. Open a new notebook in your Synapse workspace and create a DataFrame from the file.

```
df_device_info = spark.read.csv("wasbs://publicwasb@mmlspark.blob.core.windows.net/iot/IoTSignals.csv",
header=True, inferSchema=True)
```

Run anomaly detection using Cognitive Services on Spark

The goal is to find instances where the signals from the IoT devices were outputting anomalous values so that we can see when something is going wrong and do predictive maintenance. To do that, let's use Anomaly Detector on Spark:

```
from pyspark.sql.functions import col, struct
from mmlspark.cognitive import SimpleDetectAnomalies
from mmlspark.core.spark import FluentAPI

detector = (SimpleDetectAnomalies()
    .setSubscriptionKey(service_key)
    .setLocation(location)
    .setOutputCol("anomalies")
    .setGroupbyCol("grouping")
    .setSensitivity(95)
    .setGranularity("secondly"))

df_anomaly = (df_signals
    .where(col("unitSymbol") == 'RPM')
    .withColumn("timestamp", col("dateTime").cast("string"))
    .withColumn("value", col("measureValue").cast("double"))
    .withColumn("grouping", struct("deviceId"))
    .mlTransform(detector)).cache()

df_anomaly.createOrReplaceTempView('df_anomaly')
```

Let's take a look at the data:

```
df_anomaly.select("timestamp","value","deviceId","anomalies.isAnomaly").show(3)
```

This cell should yield a result that looks like:

TIMESTAMP	VALUE	DEVICEID	ISANOMALY
2020-05-01 18:33:51	3174	dev-7	False
2020-05-01 18:33:52	2976	dev-7	False
2020-05-01 18:33:53	2714	dev-7	False

Visualize anomalies for one of the devices

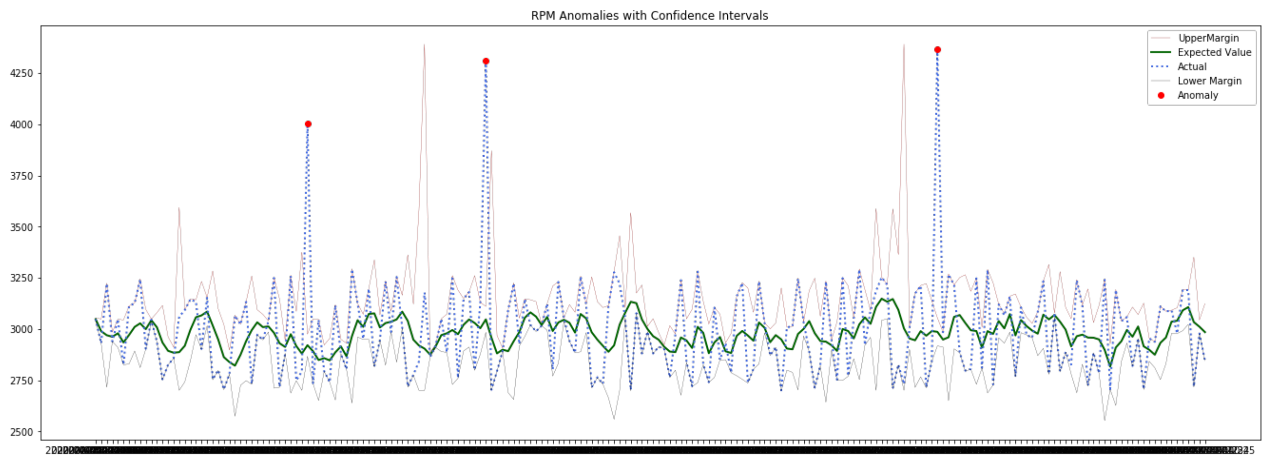
IoTSignals.csv has signals from multiple IoT devices. We'll focus on a specific device and visualize anomalous outputs from the device.

```
df_anomaly_single_device = spark.sql("""
select
  timestamp,
  measureValue,
  anomalies.expectedValue,
  anomalies.expectedValue + anomalies.upperMargin as expectedUpperValue,
  anomalies.expectedValue - anomalies.lowerMargin as expectedLowerValue,
  case when anomalies.isAnomaly=true then 1 else 0 end as isAnomaly
from
  df_anomaly
where deviceid = 'dev-1' and timestamp < '2020-04-29'
order by timestamp
limit 200""")
```

Now that we have created a dataframe that represents the anomalies for a particular device, we can visualize these anomalies:

```
import matplotlib.pyplot as plt
from pyspark.sql.functions import col

adf = df_anomaly_single_device.toPandas()
adf_subset = df_anomaly_single_device.where(col("isAnomaly") == 1).toPandas()

plt.figure(figsize=(23,8))
plt.plot(adf['timestamp'],adf['expectedUpperValue'], color='darkred', linestyle='solid', linewidth=0.25,
label='UpperMargin')
plt.plot(adf['timestamp'],adf['expectedValue'], color='darkgreen', linestyle='solid', linewidth=2,
label='Expected Value')
plt.plot(adf['timestamp'],adf['measureValue'], 'b', color='royalblue', linestyle='dotted', linewidth=2,
label='Actual')
plt.plot(adf['timestamp'],adf['expectedLowerValue'],  color='black', linestyle='solid', linewidth=0.25,
label='Lower Margin')
plt.plot(adf_subset['timestamp'],adf_subset['measureValue'], 'ro', label = 'Anomaly')
plt.legend()
plt.title('RPM Anomalies with Confidence Intervals')
plt.show()
```

If successful, your output will look like this:



Next steps

Learn how to do predictive maintenance at scale with Azure Cognitive Services, Azure Synapse Analytics, and Azure CosmosDB. For more information, see the full sample on GitHub.

Recipe: Intelligent Art Exploration with the Cognitive Services for Big Data

10/4/2020 • 2 minutes to read • Edit Online

In this example, we'll use the Cognitive Services for Big Data to add intelligent annotations to the Open Access collection from the Metropolitan Museum of Art (MET). This will enable us to create an intelligent search engine using Azure Search even without manual annotations.

Prerequisites

- You must have a subscription key for Computer Vision and Cognitive Search. Follow the instructions in Create a Cognitive Services account to subscribe to Computer Vision and get your key.

  NOTE

  For pricing information, see Azure Cognitive Search.

Import Libraries

Run the following command to import libraries for this recipe.

```
import os, sys, time, json, requests
from pyspark.ml import Transformer, Estimator, Pipeline
from pyspark.ml.feature import SQLTransformer
from pyspark.sql.functions import lit, udf, col, split
```

Set up Subscription Keys

Run the following command to set up variables for service keys. Insert your subscription keys for Computer Vision and Azure Cognitive Search.

```
VISION_API_KEY = 'INSERT_COMPUTER_VISION_SUBSCRIPTION_KEY'
AZURE_SEARCH_KEY = 'INSERT_AZURE_COGNITIVE_SEARCH_SUBSCRIPTION_KEY'
search_service = "mmlspark-azure-search"
search_index = "test"
```

Read the Data

Run the following command to load data from the MET's Open Access collection.

```
data = spark.read\
  .format("csv")\
  .option("header", True)\
  .load("wasbs://publicwasb@mmlspark.blob.core.windows.net/metartworks_sample.csv")\
  .withColumn("searchAction", lit("upload"))\
  .withColumn("Neighbors", split(col("Neighbors"), ",").cast("array<string>"))\
  .withColumn("Tags", split(col("Tags"), ",").cast("array<string>"))\
  .limit(25)
```

Analyze the Images

Run the following command to use Computer Vision on the MET's Open Access artworks collection. As a result, you'll get visual features from the artworks.

```
from mmlspark.cognitive import AnalyzeImage
from mmlspark.stages import SelectColumns

#define pipeline
describeImage = (AnalyzeImage()
  .setSubscriptionKey(VISION_API_KEY)
  .setLocation("eastus")
  .setImageUrlCol("PrimaryImageUrl")
  .setOutputCol("RawImageDescription")
  .setErrorCol("Errors")
  .setVisualFeatures(["Categories", "Tags", "Description", "Faces", "ImageType", "Color", "Adult"])
  .setConcurrency(5))

df2 = describeImage.transform(data)\
  .select("*", "RawImageDescription.*").drop("Errors", "RawImageDescription")
```

Create the Search Index

Run the following command to write the results to Azure Search to create a search engine of the artworks with enriched metadata from Computer Vision.

```
from mmlspark.cognitive import *
df2.writeToAzureSearch(
  subscriptionKey=AZURE_SEARCH_KEY,
  actionCol="searchAction",
  serviceName=search_service,
  indexName=search_index,
  keyCol="ObjectID"
)
```

Query the Search Index

Run the following command to query the Azure Search index.

```
url = 'https://{}.search.windows.net/indexes/{}/docs/search?api-version=2019-05-06'.format(search_service,
search_index)
requests.post(url, json={"search": "Glass"}, headers = {"api-key": AZURE_SEARCH_KEY}).json()
```

Next steps

Learn how to use Cognitive Services for Big Data for Anomaly Detection.

Azure Policy built-in policy definitions for Azure Cognitive Services

10/4/2020 • 2 minutes to read • Edit Online

This page is an index of Azure Policy built-in policy definitions for Azure Cognitive Services. For additional Azure Policy built-ins for other services, see Azure Policy built-in definitions.

The name of each built-in policy definition links to the policy definition in the Azure portal. Use the link in the **Version** column to view the source on the Azure Policy GitHub repo.

Azure Cognitive Services

NAME (AZURE PORTAL)	DESCRIPTION	EFFECT(S)	VERSION (GITHUB)
Cognitive Services accounts should enable data encryption	This policy audits any Cognitive Services account not using data encryption. For each Cognitive Services account with storage, should enable data encryption with either customer managed or Microsoft managed key.	Audit, Deny, Disabled	1.0.0
Cognitive Services accounts should enable data encryption with customer-managed key	Customer-managed keys provide enhanced data protection by allowing you to manage your encryption keys for data stored in Cognitive Services. This is often required to meet compliance requirements.	Audit, Deny, Disabled	1.0.1
Cognitive Services accounts should restrict network access	Network access to Cognitive Services accounts should be restricted. Configure network rules so only applications from allowed networks can access the Cognitive Services account. To allow connections from specific internet or on-premises clients, access can be granted to traffic from specific Azure virtual networks or to public internet IP address ranges.	Audit, Deny, Disabled	1.0.0
Cognitive Services accounts should use customer owned storage	This policy audits any Cognitive Services account not using customer owned storage.	Audit, Deny, Disabled	1.0.0

NAME	DESCRIPTION	EFFECT(S)	VERSION
Cognitive Services accounts should use customer owned storage or enable data encryption.	This policy audits any Cognitive Services account not using customer owned storage nor data encryption. For each Cognitive Services account with storage, use either customer owned storage or enable data encryption.	Audit, Deny, Disabled	1.0.0
Public network access should be disabled for Cognitive Services accounts	This policy audits any Cognitive Services account in your environment with public network access enabled. Public network access should be disabled so that only connections from private endpoints are allowed.	Audit, Deny, Disabled	1.0.0

Next steps

- See the built-ins on the Azure Policy GitHub repo.
- Review the Azure Policy definition structure.
- Review Understanding policy effects.

Azure Cognitive Services support and feedback options

5/19/2020 • 4 minutes to read • Edit Online

Are you just starting to explore the functionality of Azure Cognitive Services? Perhaps you are implementing a new feature in your application. Or after using the service, do you have suggestions on how to improve it? Here are options for where you can get support, stay up-to-date, give feedback, and report bugs for Cognitive Services.

Get support

Create an Azure support request

Explore the range of Azure support options and choose the plan that best fits, whether you're a developer just starting your cloud journey or a large organization deploying business-critical, strategic applications. Azure customers can create and manage support requests in the Azure portal.

- Azure portal
- Azure portal for the United States government

Search

For faster results, perform a search to relevant sites such as Stack Overflow, Microsoft docs, or GitHub code samples. Use the `site:` query notation in your favorite search engine, for example:

```
{search keywords} site:stackoverflow.com
```

Where `{search keywords}` is the context of your search. Consider using these scoped searches appropriately:

- Stack Overflow: `site:stackoverflow.com`
- Microsoft Docs: `site:docs.microsoft.com`
- GitHub Samples: `site:github.com/azure-samples`

Post a question on Stack Overflow

If you can't find an answer to your problem with the search box on Stack Overflow, submit a new question. Stack Overflow is the preferred channel for development-related questions. It's where members of the Stack Overflow community and Microsoft team members are directly involved in helping you solve your problems.

Post a question here. Use the appropriate tag for your question, so we are sure to see the question.

TIP

The following posts from Stack Overflow contain tips on how to form questions and add source code. Following these guidelines might help increase the chances that community members assess and respond to your question quickly:

- How do I ask a good question?
- How to create a minimal, reproducible example?

Stay informed

Staying informed about features in a new release or news on the Azure blog can help you find the difference between a programming error, a service bug, or a feature not yet available in Cognitive Services.

Release notes

The Cognitive Services release notes are updated as new releases are made available. The notes contain information about new features, improvements, and bug fixes.

- Custom Vision
- Face
- Language Understanding (LUIS)
- Speech Services
- Speech Services SDK
- Text Analytics
- Video Indexer

Azure blog

News about Cognitive Services is shared in the Azure blog.

Reddit

Reddit is a community-driven discussion website, offering the latest conversations about Azure Cognitive Services news, help, info, tips, and tricks. Registered members can submit content and others can vote on it, which organically elevates the most relevant content to the top of their feeds.

Give feedback

UserVoice forum

To request new features, post them on UserVoice. Share your ideas for making Cognitive Services and its APIs work better for the applications you develop.

NOTE

Although this is a public forum, don't expect support from Microsoft here, but do enjoy discussing new features you'd like to see in upcoming Cognitive Services releases.

SERVICE	COGNITIVE SERVICES USERVOICE URL
Anomaly Detector	https://cognitive.uservoice.com/forums/912196-anomaly-detector
Bing services	https://cognitive.uservoice.com/forums/555907-bing-search
Computer Vision	https://cognitive.uservoice.com/forums/430309-computer-vision

SERVICE	COGNITIVE SERVICES USERVOICE URL
Content Moderator	https://cognitive.uservoice.com/forums/559960-content-moderator
Custom Vision	https://cognitive.uservoice.com/forums/598141-custom-vision-service
Face	https://cognitive.uservoice.com/forums/430315-face
Form Recognizer	https://cognitive.uservoice.com/forums/921556-form-recognizer
Ink Recognizer	https://cognitive.uservoice.com/forums/921559-ink-recognizer
Language Understanding (LUIS)	https://cognitive.uservoice.com/forums/551524-luis
Personalizer	https://cognitive.uservoice.com/forums/921562-personalizer
QnA Maker	https://cognitive.uservoice.com/forums/578689-qna-maker
Speech Services	https://cognitive.uservoice.com/forums/912208-speech-service
Custom Speech	https://cognitive.uservoice.com/forums/555934-custom-speech-service
Text Analytics	https://cognitive.uservoice.com/forums/555922-text-analytics
Translator	https://cognitive.uservoice.com/forums/558796-translator
Video Indexer	https://cognitive.uservoice.com/forums/598144-video-indexer

Report bugs

Create a GitHub issue or pull request

Below are three types of repositories where a developer might post an issue or create a pull request. To create an issue use the *Issues* tab in the respective GitHub repository and select *New issue*. If you want to suggest a correction, submit a pull request by editing a file directly. The pull request will then sit under the *Pull requests* tab until the repository owners can look it over.

Cognitive Services Samples

Samples are often found in public repositories as open source. The quickstart samples have their own repository and are referenced in the Microsoft documentation. If you find errors in the code, create an issue or a pull request.

Here is a list of the Cognitive Services quickstart and sample code:

- Azure Samples - Cognitive Services
- Cognitive Services Quickstarts
- Cognitive Services: Dotnet
- Cognitive Services: Go

- Cognitive Services: Java
- Cognitive Services: Node.js
- Cognitive Services: Python

Cognitive Services SDK source code

The source code of the SDKs show some of the underpinnings of the SDK for each service. If you have found a bug or want to suggest a correction, file an issue or create a pull request.

Here is a list of the Cognitive Services SDK source code by language:

- Azure SDK for Dotnet
- Azure SDK for Go
- Azure SDK for Java
- Azure SDK for JavaScript
- Azure SDK for Python

Azure Cognitive Services documentation

The documentation for Cognitive Services explains how to use the service and provides examples and resources for the developer. If you have found a bug or want to suggest a correction, file an issue or create a pull request.

MicrosoftDocs

Next steps

What are Azure Cognitive Services?