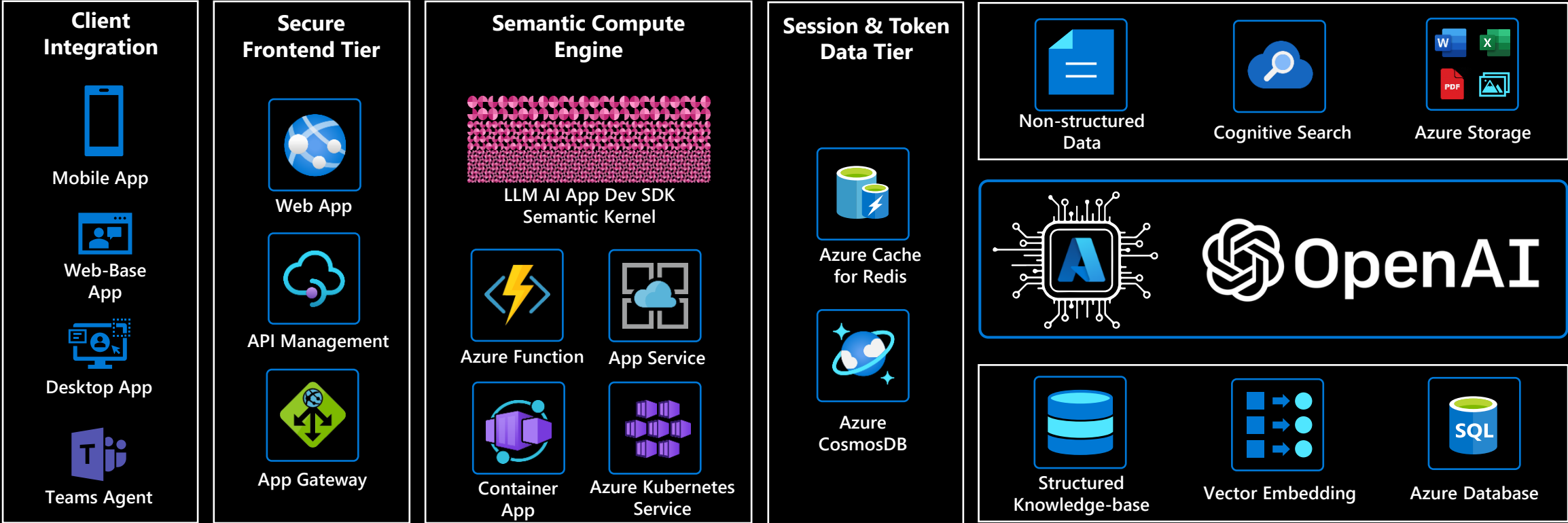
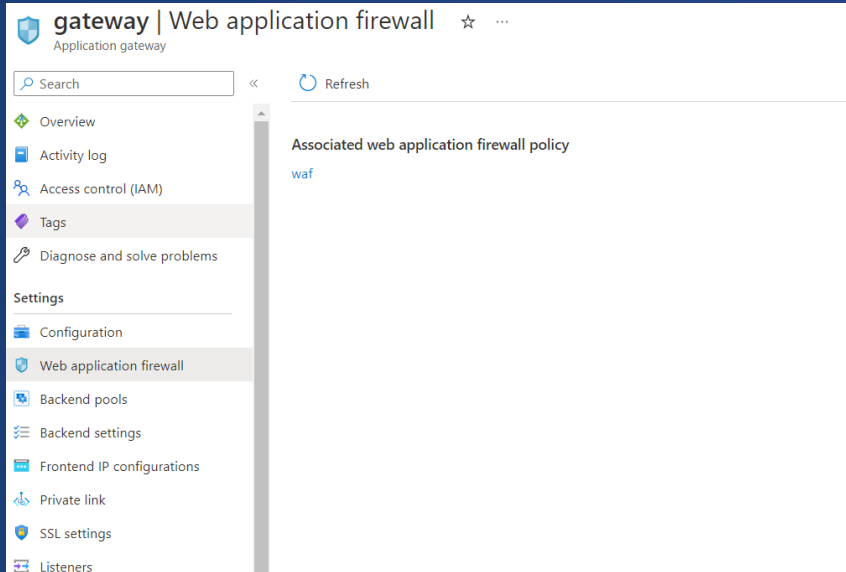


FY23 Q3 Azure In A Day Workshop



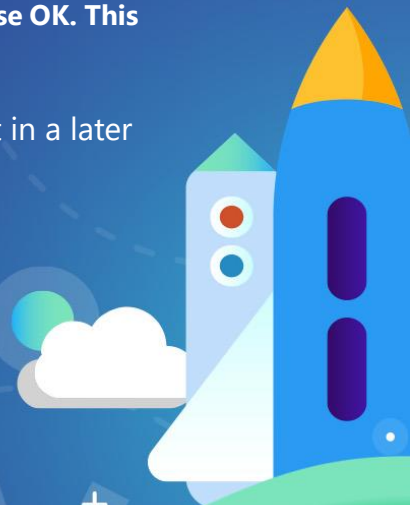
Enterprise “GPT-ize” Intelligent App Architecture

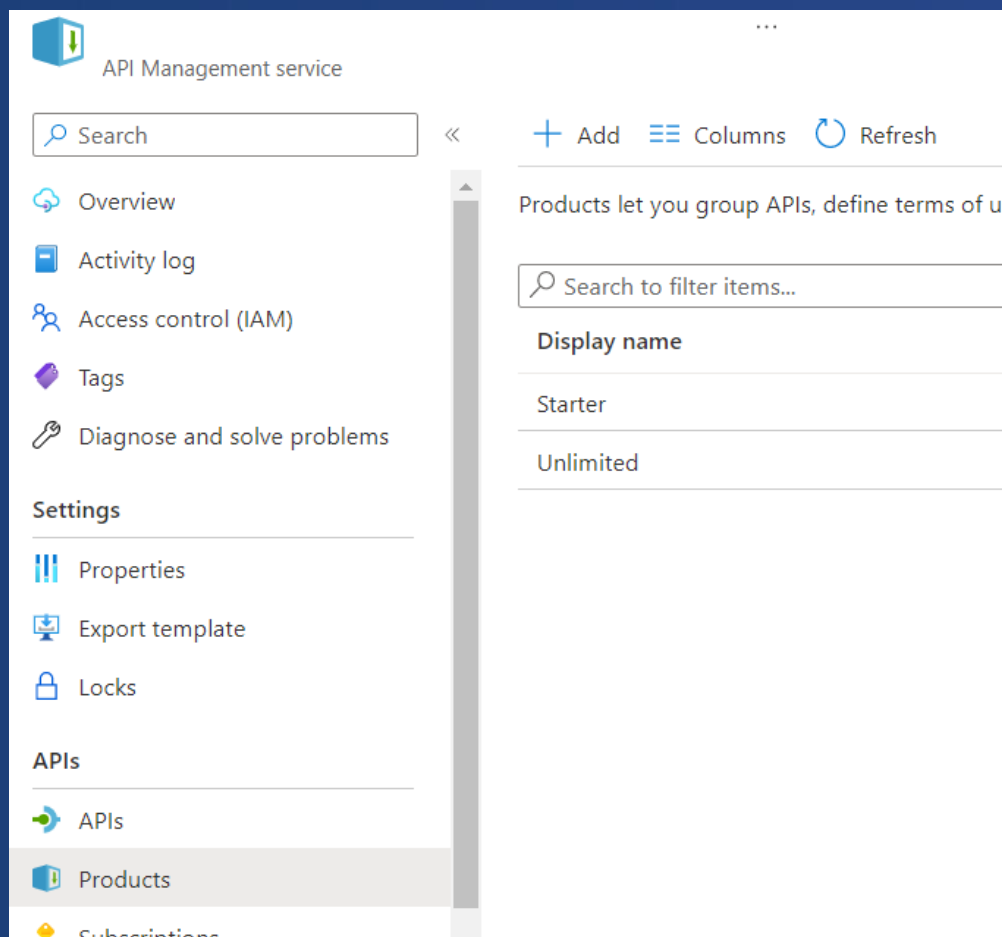




Create an application gateway (Application Gateway)

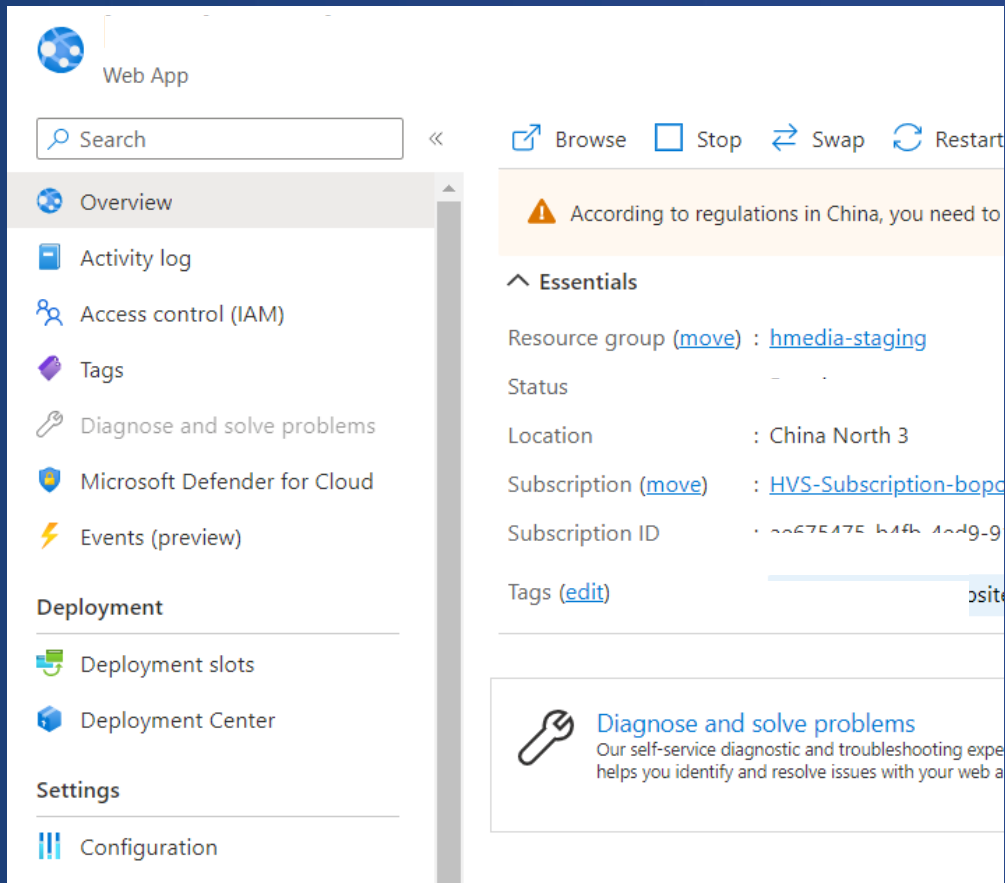
1. Select Create a resource on the left menu of the Azure portal. The New window appears.
2. Select Networking, and then select Application Gateway in the Featured list.
3. On the Basics tab, enter these values as the following Application Gateway settings:
 - **Resource group:** Select **Create new to create a new one**.
 - **Application gateway name:** Enter **myAppGateway** as the name of the application gateway.
 - **Tier:** Select **WAF V2**.
 - **WAF policy:** Choose **New**, type a name for the new policy, and then choose **OK**. This creates a basic WAF policy with a managed core rule set (CRS).
4. Set up the frontend tab: Select **Public**
5. Set backend tab: Select **Backend pool without target** (configure the target in a later step)
6. Set the **Configuration** tab





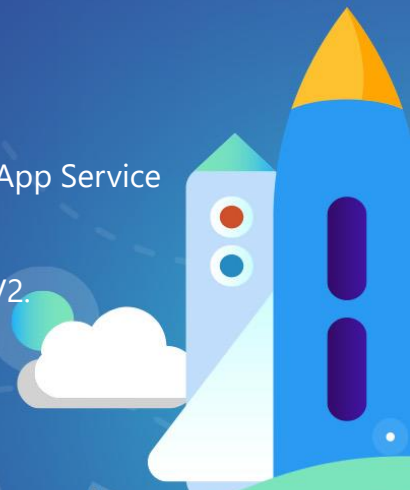
Create Azure API Management (APIM)

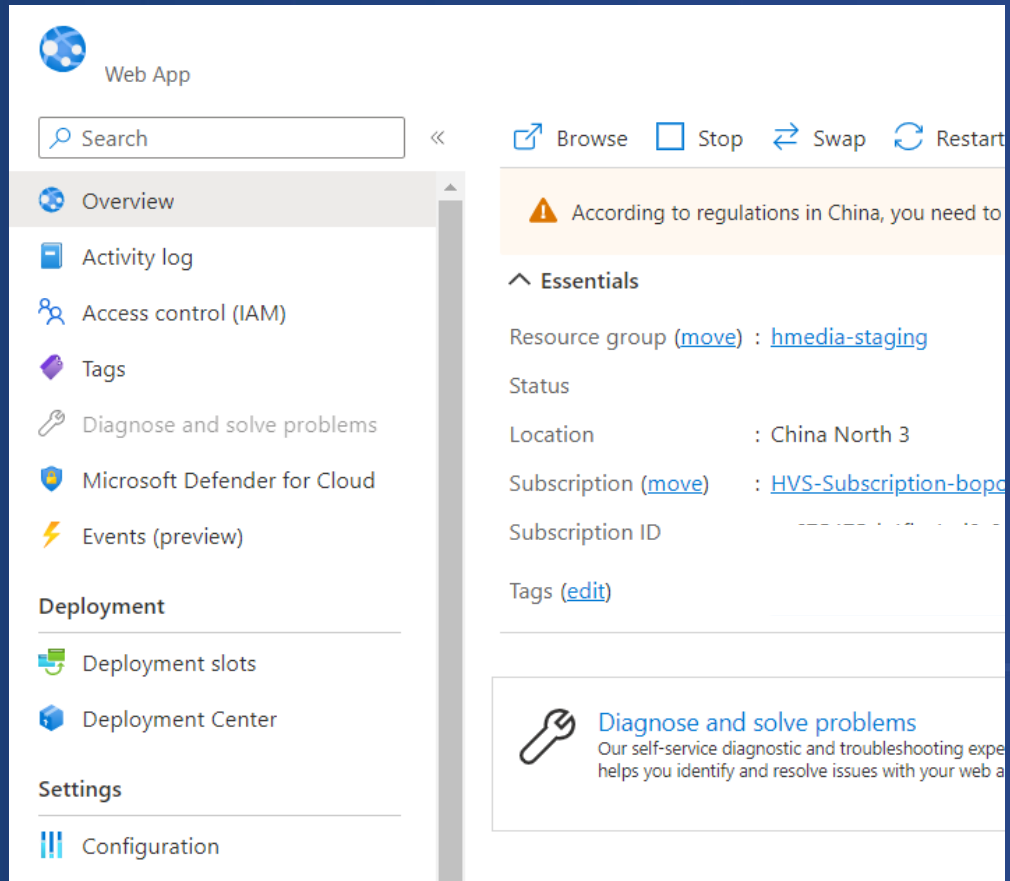
1. In the Azure portal menu, select Create a resource. You can also select Create a resource on the Azure Home page.
2. On the Create a resource page, select Integration > API Management.
3. In the Create API Management page, enter the settings
4. Select Review + create.
5. Import and publish the API (next steps)
6. Include your API (next steps)



Create App Service: A front-end UI app

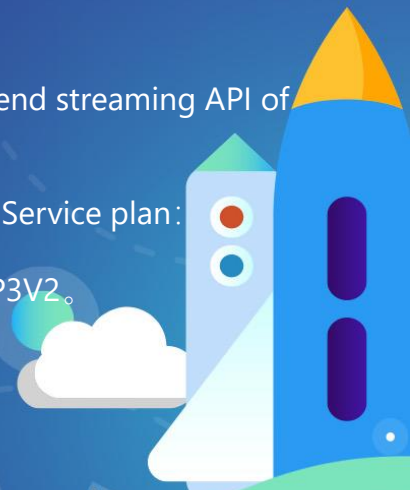
1. In the Azure portal, type App Service in the search box. Under Services, select Application Services
2. In the App Service page, select + Create.
3. In the Basics tab, under Project details, make sure the correct subscription is selected, and then select Create new to create a new resource group.
4. Set the Instance Information tab:
 - Under Name, type a globally unique name for your web app.
 - Under Publish, select Code.
 - Under Runtime stack, select .NET 6 (LTS).
 - Select "Operating System": Windows or Linux.
 - Select the Region where the instance runs: Any.
 - Under App Service plan, select Create new to create a new App Service plan:
 - Type a name.
 - Select Change Size to select a pricing tier, such as S3 or P3V2.
 - Deploy the front-end app (next steps)

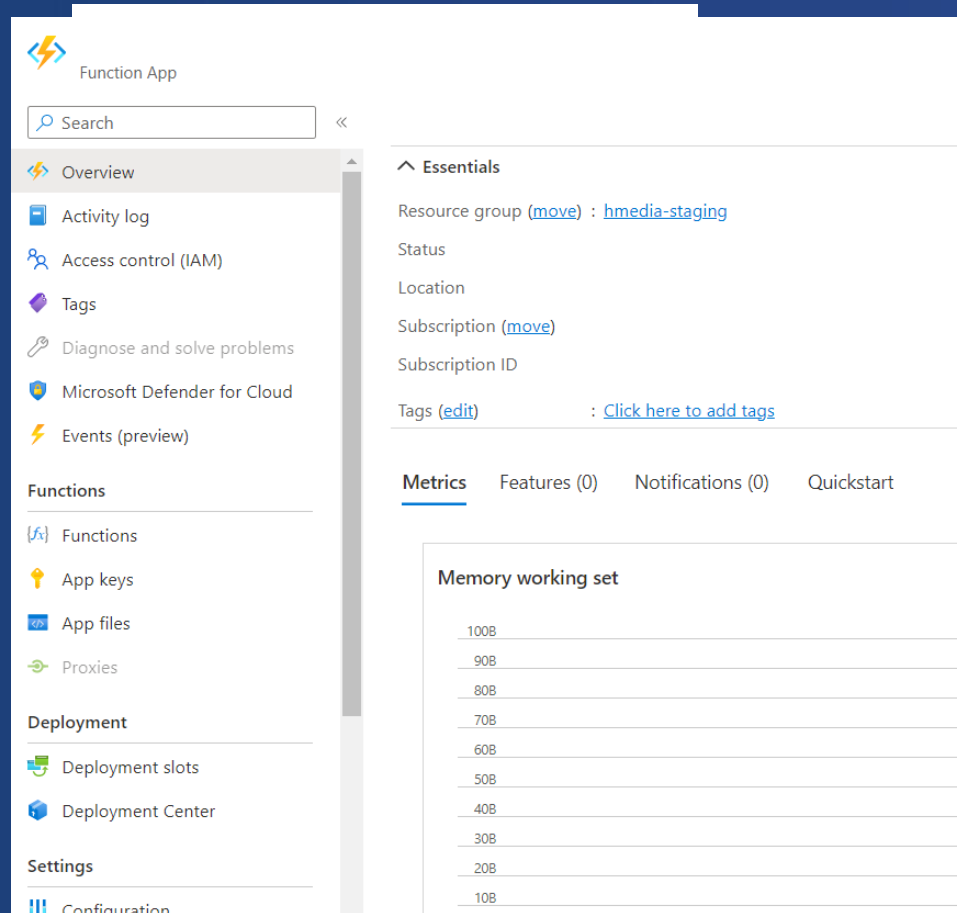




Create an App Service: A back-end API service

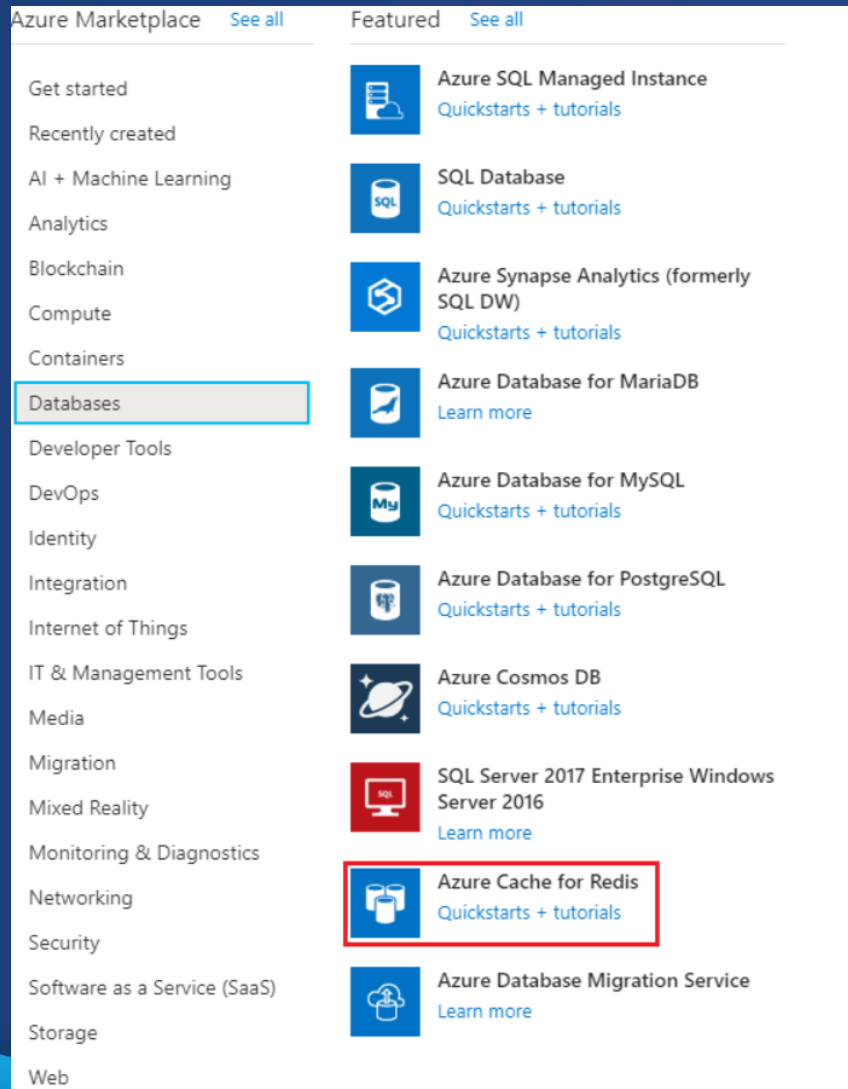
1. In the Azure portal, type App Service in the search box. Under Services, select Application Services
2. In the App Service page, select + Create.
3. In the Basics tab, under Project details, make sure the correct subscription is selected, and then select Create new to create a new resource group.
4. Set the Instance Information tab:
 - Under Name, type a globally unique name for your web app.
 - Under Publish, select Code.
 - Under Runtime stack, select .NET 6 (LTS).
 - Select "Operating System": Select Linux (note that the backend streaming API of this workshop requires Linux OS)
 - Select the Region where the instance runs: Any.
5. Under App Service plan, select Create new to create a new App Service plan:
 - Type a name.
 - Select Change Size to select a pricing tier, such as S3 or P3V2.
6. Deploy the backend API service (next steps)





Create a function app service (Azure Function): A back-end API service

1. On the Azure portal menu or in the portal home page, select Create a resource.
2. On the New page, select Compute > Function App.
3. On the Basics page, set up your function app:
 - subscribe
 - Resource group
 - name
 - Runtime language: C# class library
 - Version: Latest
4. Select Next: Hosting. Set up:
 1. Storage account number
 2. Operating system: Windows
 3. Plan: Select Consumption (serviceless mode.)
5. Next
6. create
7. Deploy API Service (Next Steps)



Create a cache service (Azure Redis)

1. Sign in to the Azure portal and select Create a resource .
2. On the New page, select Database, and then select Azure Cache for Redis”.
3. On the New Redis Cache page, configure the settings for the new cache:
 - subscribe
 - Resource group
 - DNS Name : Unique name
 - location
 - Cache Type: Select a pricing tier, such as "C6 or P3."
4. The next step is to set up the network
5. Next, "Advanced", select the latest Redis version
6. create
7. Get Redis connection string (next steps)



Home > Azure Cosmos DB >

Create Azure Cosmos DB Account

Basics Global Distribution Networking Back

Azure Cosmos DB is a globally distributed, multi-model, fully managed database service.

Project Details

Select the subscription to manage deployed resources and the resource group to create the account in.

Subscription *

Resource Group *

[Create new](#)

Instance Details

Account Name *

API *

Location *

Capacity mode ☒ Provisioned [Learn more about capacity modes](#)

With Azure Cosmos DB free tier, you will get the first 100 RU/s free.

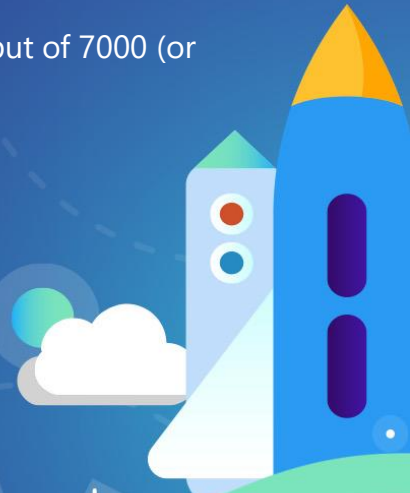
Apply Free Tier Discount ☒ Apply

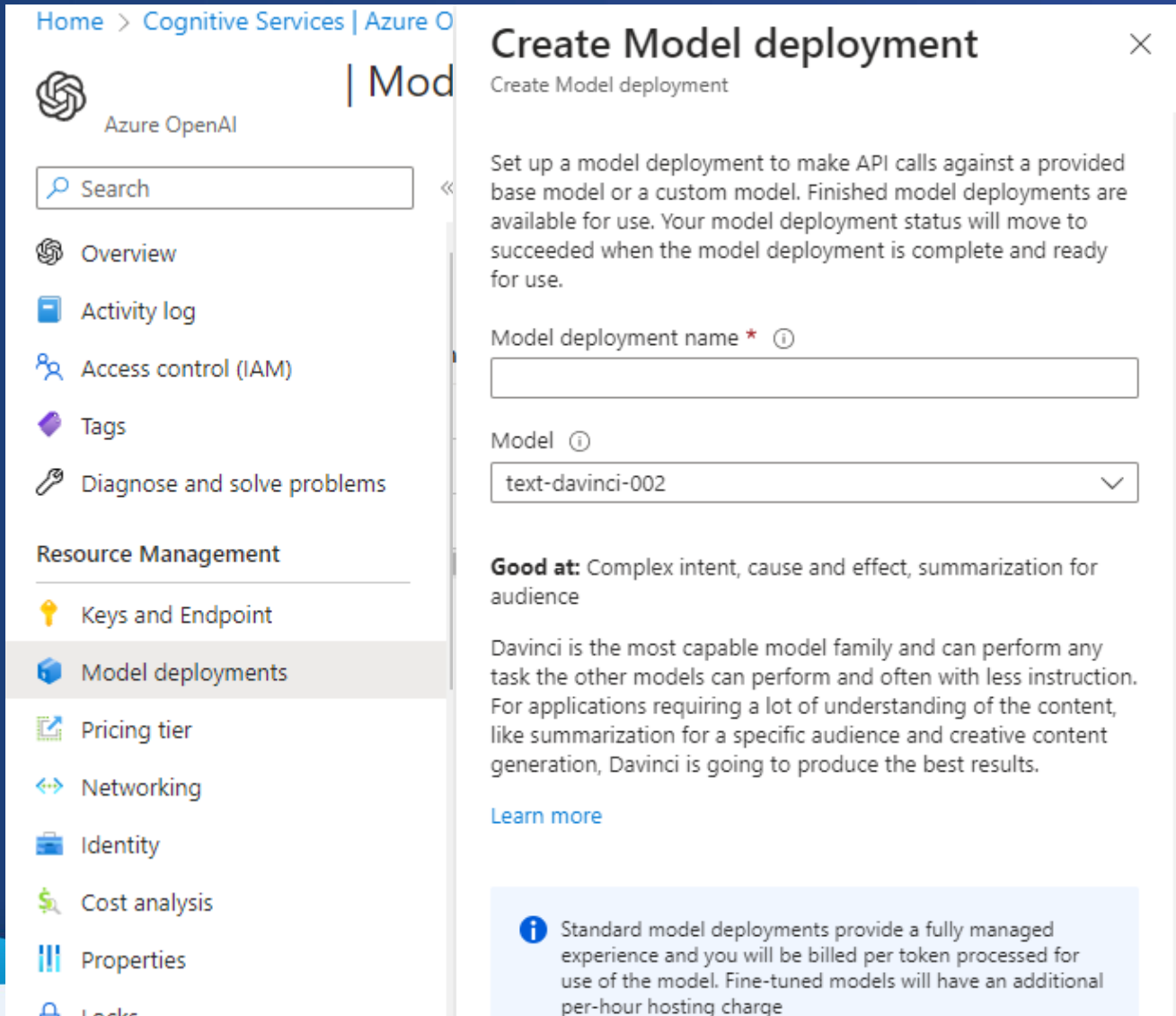
Limit total account throughput ☒ Limit the total throughput of the account

[This limit applies to all accounts in the subscription.](#)

Create CosmosDB

1. In the Azure portal menu or home page, select Create a resource.
2. Search for Azure Cosmos DB. Select Create > Azure Cosmos DB.
3. On the Create Azure Cosmos DB account page, select the Create option in the Azure Cosmos DB for NoSQL section.
4. Select DB API: NoSQL
5. In the Create Azure Cosmos DB account page, enter the basic settings for your new Azure Cosmos DB account:
6. subscription
 - Resource group
 - Account Name: A unique name
 - region
 - Capacity mode: Select Provisioned throughput to set a maximum throughput of 7000 (or select Serverless mode.)"
6. Next
7. create
8. Get the database connection string (next steps)





The screenshot shows the 'Create Model deployment' page in the Azure portal. The left sidebar contains the navigation menu with 'Model deployments' selected. The main content area has a title 'Create Model deployment' and a subtitle 'Create Model deployment'. Below this is a description: 'Set up a model deployment to make API calls against a provided base model or a custom model. Finished model deployments are available for use. Your model deployment status will move to succeeded when the model deployment is complete and ready for use.' There are two input fields: 'Model deployment name' with a red asterisk and an information icon, and 'Model' with a dropdown menu showing 'text-davinci-002'. Below these fields is a section titled 'Good at:' with the text 'Complex intent, cause and effect, summarization for audience'. This is followed by a paragraph: 'Davinci is the most capable model family and can perform any task the other models can perform and often with less instruction. For applications requiring a lot of understanding of the content, like summarization for a specific audience and creative content generation, Davinci is going to produce the best results.' There is a 'Learn more' link. At the bottom, there is a blue information box with a white 'i' icon and the text: 'Standard model deployments provide a fully managed experience and you will be billed per token processed for use of the model. Fine-tuned models will have an additional per-hour hosting charge.'

Home > Cognitive Services | Azure OpenAI | Model deployments

Create Model deployment ×

Create Model deployment

Set up a model deployment to make API calls against a provided base model or a custom model. Finished model deployments are available for use. Your model deployment status will move to succeeded when the model deployment is complete and ready for use.

Model deployment name * ⓘ

Model ⓘ

text-davinci-002

Good at: Complex intent, cause and effect, summarization for audience

Davinci is the most capable model family and can perform any task the other models can perform and often with less instruction. For applications requiring a lot of understanding of the content, like summarization for a specific audience and creative content generation, Davinci is going to produce the best results.

[Learn more](#)

i Standard model deployments provide a fully managed experience and you will be billed per token processed for use of the model. Fine-tuned models will have an additional per-hour hosting charge

Create Azure OpenAI

1. In the Azure portal menu or home page, select Create a resource.
2. Search for Azure OpenAI. Select Create
3. Create a deployment model: Select text-davinci-003 (GPT3 model)
4. Get the instance in Keys and Endpoint
5. API address
6. Key

Code : About the Azure OpenAI restful API

POST <https://{your-resource-name}.openai.azure.com/openai/deployments/{deployment-id}/completions?api-version={api-version}>

Path parameter	type	Whether it is required	illustrate
your-resource-name	string	Yes	Azure OpenAI Resource.
deployment-id	string	Yes	Deployment instance name
api-version	string	yes	YYYY-MM-DD format

Body parameter	type	Default value	illustrate
prompt	string		Prompt words
max_tokens	int	16	Maximum number of tokens

```
var options = new CompletionsOptions
{
    Prompt = { prompt },
    MaxTokens = MaxTokens
};
var completions = await _client.GetCompletionsAsync(_config.DeploymentId, options);
var completion = completions.Value.Choices[0].Text;
return completion;
```

<https://learn.microsoft.com/en-us/azure/cognitive-services/openai/reference#completions>

Code: About using the Azure Redis caching service

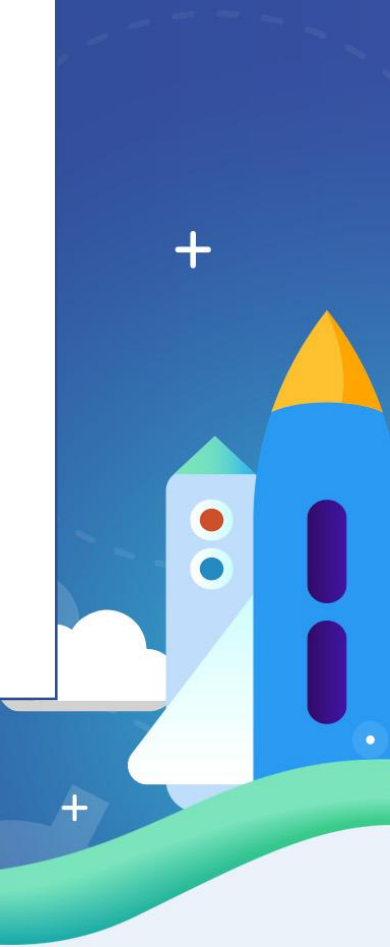
1. Reduce the frequency of calls to Azure OpenAI through application-layer caching
2. The custom context caching mechanism is weak
3. Improve application layer performance with Redis Queue
- 4.

```
public class Engine: IEngine
{
    private readonly EngineConfig _config;
    private readonly OpenAIClient _client;
    private readonly CosmosClient _cosmosClient;
    public Engine(EngineConfig config)
    public async Task<string> GetCompletionAsync(string userId, string prompt)
    {
        var cachedCompletion = await TryGetCachedCompletionAsync(userId, prompt);
        if (cachedCompletion != null)
        {
            return cachedCompletion;
        }
        else
        {
            var options = new CompletionsOptions
            {
                Prompt = { prompt }
            };
            var completions = await _client.GetCompletionsAsync(_config.DeploymentId, options);
            var completion = completions.Value.Choices[0].Text;
            await AddToCacheAsync(userId, prompt, completion);
            await SaveToDatabaseAsync(userId, prompt, completion);
            return completion;
        }
    }
}
```

About Redis Windows native development:

<https://learn.microsoft.com/en-us/azure/azure-cache-for-redis/cache-development-faq>

<https://github.com/microsoftarchive/redis/releases>



代码解析: 关于使用Cosmos DB

1. Session recording
2. Embeddings Store (domain knowledge base)

```
var database = _cosmosClient.GetDatabase("completionsDB");

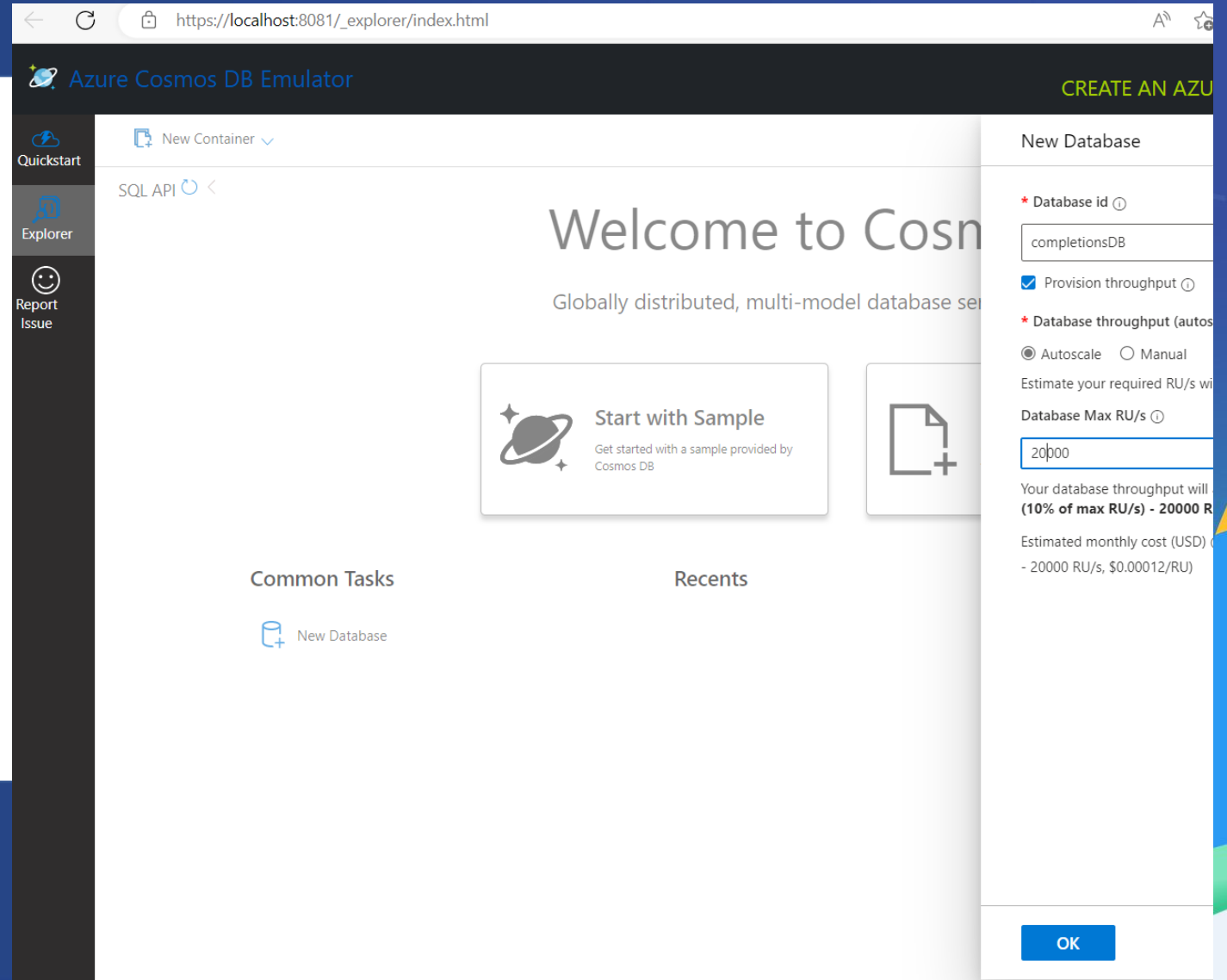
var containerResponse = await database.CreateContainerIfNotExistsAsync(
    id: "completions",
    partitionKeyPath: "/userId",
    throughput: 400
);

await containerResponse.Container.CreateItemAsync<CompletionCacheItem>(new CompletionCacheItem
{
    UserId = userId,
    Prompt = prompt,
    Completion = completion
});
```

About local development of
CosmosDB:

[https://learn.microsoft.com/
en-us/azure/cosmos-
db/local-emulator?tabs=ssl-
netstd21](https://learn.microsoft.com/en-us/azure/cosmos-db/local-emulator?tabs=ssl-netstd21)

https://localhost:8081/_explorer/index.html



The screenshot shows the Azure Cosmos DB Emulator web interface. The browser address bar displays `https://localhost:8081/_explorer/index.html`. The interface has a dark sidebar on the left with navigation links: Quickstart, Explorer (selected), and Report Issue. The main content area features a 'Welcome to Cosmos' message and a 'Start with Sample' button. A 'New Database' modal is open on the right, showing configuration options for a database named 'completionsDB'. The modal includes settings for Provision throughput (checked), Database throughput (Autoscale selected), and a 'Database Max RU/s' field set to 20000. It also displays a warning about database throughput limits and an estimated monthly cost of \$0.00012/RU. An 'OK' button is at the bottom of the modal.

Azure Cosmos DB Emulator

CREATE AN AZURE COSMOS DB

SQL API

Welcome to Cosmos

Globally distributed, multi-model database service

Start with Sample

Get started with a sample provided by Cosmos DB

Common Tasks

Recents

New Database

New Database

* Database id ⓘ

completionsDB

☒ Provision throughput ⓘ

* Database throughput (autoscale) ⓘ

☒ Autoscale ☐ Manual

Estimate your required RU/s with the following settings

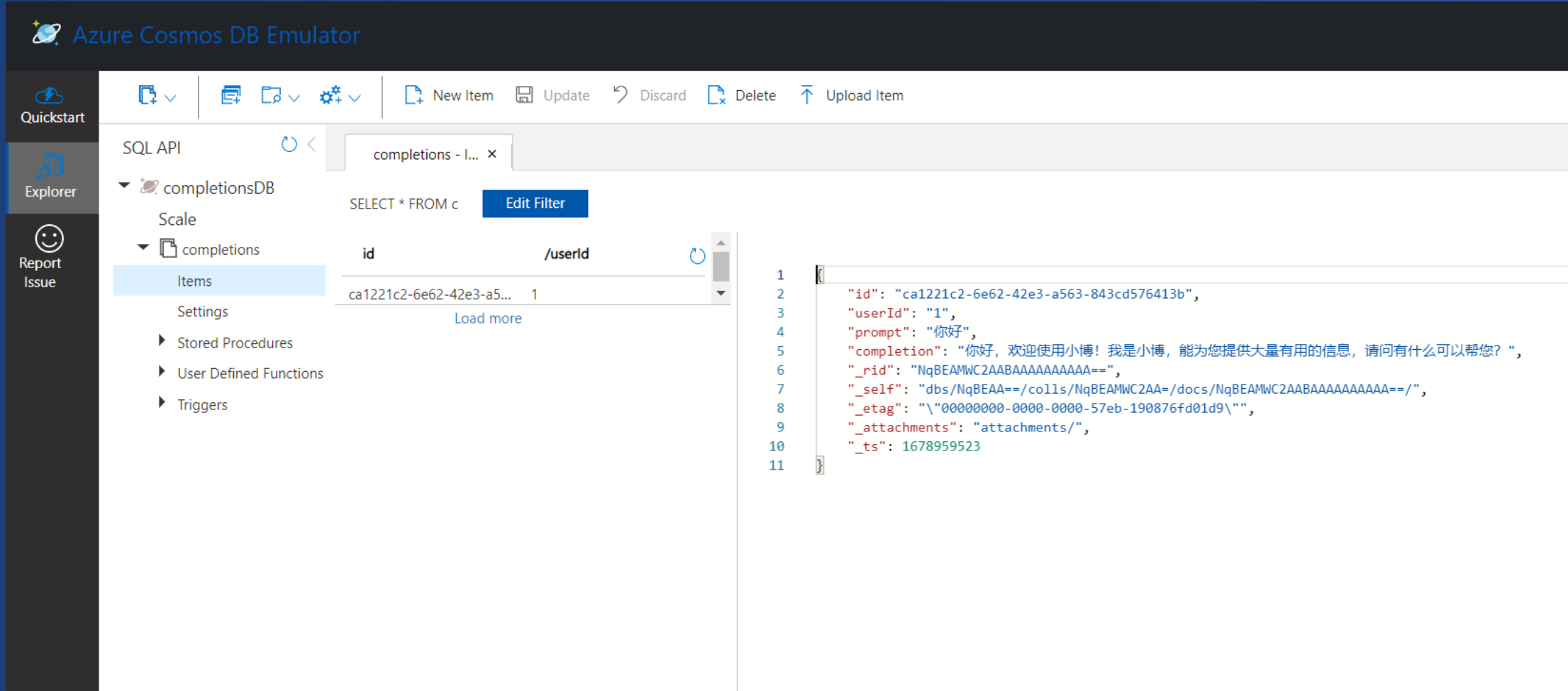
Database Max RU/s ⓘ

20000

Your database throughput will be limited to 10% of the maximum RU/s for your database (10% of max RU/s) - 20000 RU/s

Estimated monthly cost (USD) for 20000 RU/s - \$0.00012/RU

OK



The screenshot displays the Azure Cosmos DB Emulator interface. On the left, a sidebar contains 'Quickstart', 'Explorer', and 'Report Issue' buttons. The 'Explorer' panel shows a tree view with 'completionsDB' expanded, containing 'Scale', 'completions' (with 'Items' selected), 'Settings', 'Stored Procedures', 'User Defined Functions', and 'Triggers'. The main area shows a SQL query 'SELECT * FROM c' with an 'Edit Filter' button. Below the query, a table displays the results of the query:

id	/userId
ca1221c2-6e62-42e3-a5...	1

A 'Load more' link is visible below the table. On the right, a code editor shows the JSON document corresponding to the query result:

```
1 {  
2   "id": "ca1221c2-6e62-42e3-a563-843cd576413b",  
3   "userId": "1",  
4   "prompt": "你好",  
5   "completion": "你好, 欢迎使用小博! 我是小博, 能为您提供大量有用的信息, 请问有什么可以帮您? ",  
6   "_rid": "NqBEAMWC2AABAAAAAAAAA==",  
7   "_self": "dbs/NqBEAA==/colls/NqBEAMWC2AA=/docs/NqBEAMWC2AABAAAAAAAAA==/",  
8   "_etag": "\"00000000-0000-0000-57eb-190876fd01d9\"",  
9   "_attachments": "attachments/",  
10  "_ts": 1678959523  
11 }
```

Code: About configuration information

You must configure the following information:

AzureOpenAIAPIURL

AzureOpenAIKey

AzureOpenAIDeploymentId

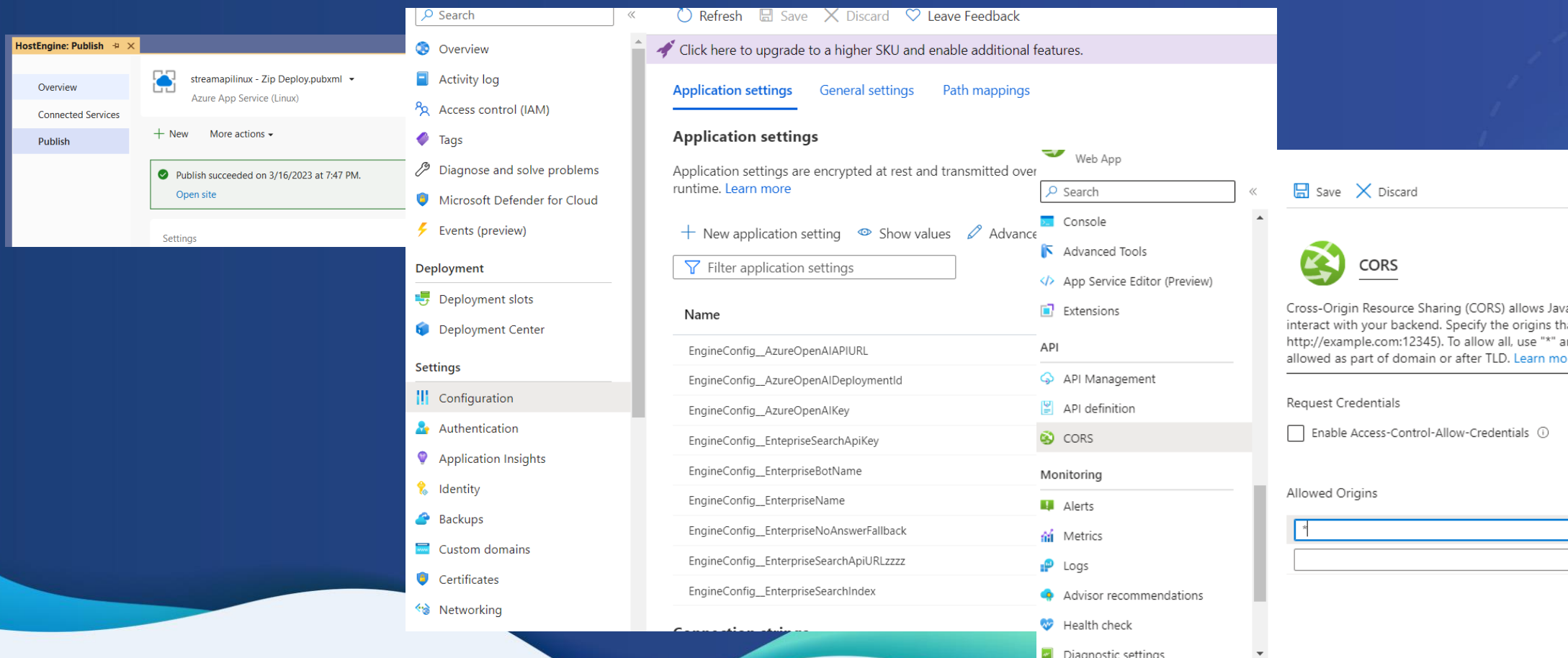
RedisConnectionString

CosmosbDBConnectionString

CosmosDBName

```
"EngineConfig": {  
  "AzureOpenAIAPIURL": "",  
  "AzureOpenAIKey": "",  
  "AzureOpenAIDeploymentId": "",  
  "RedisConnectionString": "",  
  "CosmosDBConnectionString": "",  
  "CosmosDBName": "completionsDB",  
}
```

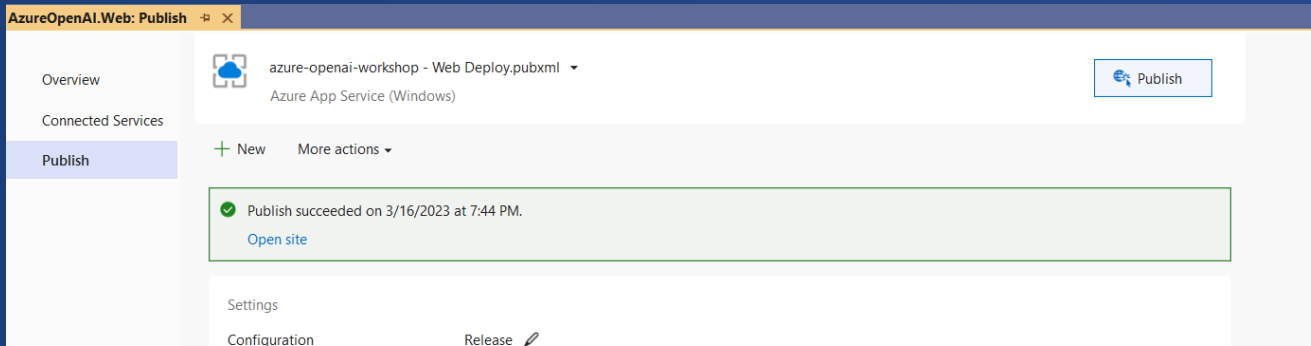
Deployment and Configuration: Backend Services (APIs)



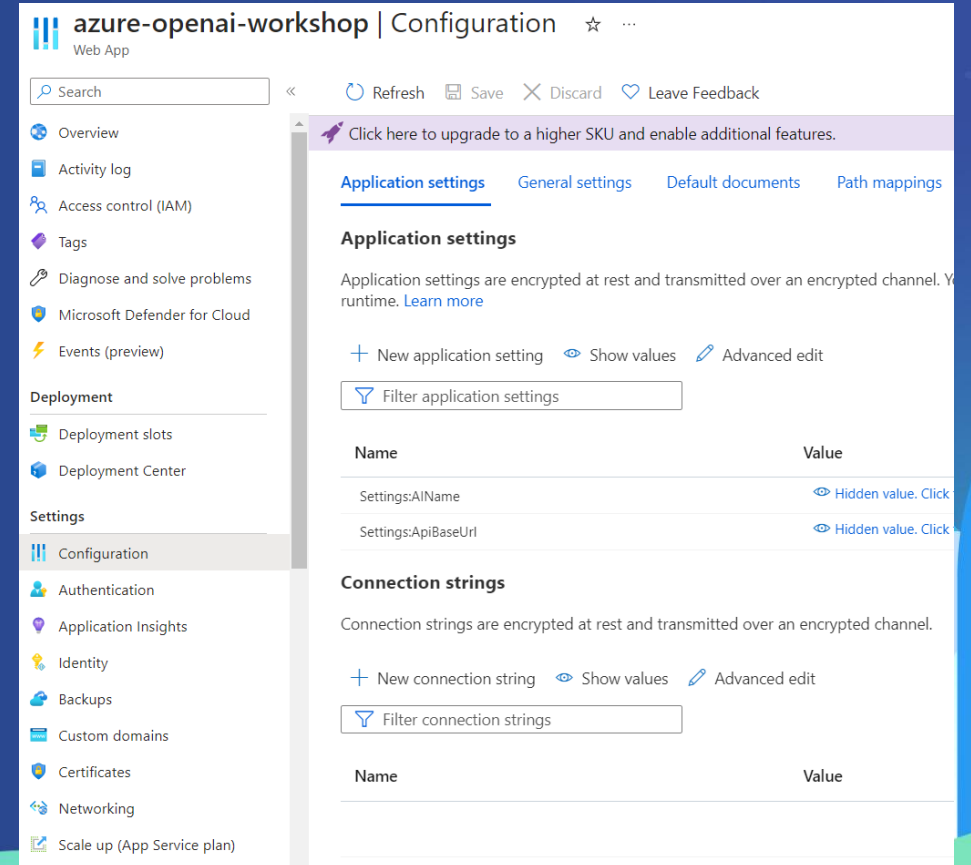
The screenshot displays the Azure Portal interface for configuring an Azure App Service Web App. The left sidebar shows the navigation menu with sections like Overview, Connected Services, Publish, Deployment, and Settings. The main content area is divided into three panes:

- Left Pane (HostEngine: Publish):** Shows the deployment status with a green checkmark indicating "Publish succeeded on 3/16/2023 at 7:47 PM." and a link to "Open site".
- Middle Pane (Application settings):** Displays the "Application settings" tab, which lists various configuration keys. The "Name" column lists keys like "EngineConfig_AzureOpenAIURL", "EngineConfig_AzureOpenAIDeploymentId", "EngineConfig_AzureOpenAIKey", "EngineConfig_EnterpriseSearchApiKey", "EngineConfig_EnterpriseBotName", "EngineConfig_EnterpriseName", "EngineConfig_EnterpriseNoAnswerFallback", "EngineConfig_EnterpriseSearchApiURLzzzz", and "EngineConfig_EnterpriseSearchIndex". The "Value" column shows the corresponding values, such as "API", "API Management", "API definition", "CORS", "Alerts", "Metrics", "Logs", "Advisor recommendations", "Health check", and "Diagnostic settings".
- Right Pane (CORS):** Shows the "CORS" configuration settings. It includes a description of Cross-Origin Resource Sharing (CORS) and a section for "Request Credentials" with a checkbox for "Enable Access-Control-Allow-Credentials". Below this is the "Allowed Origins" section, which contains a text input field.

Deployment & Configuration: Front-End Apps (UI)



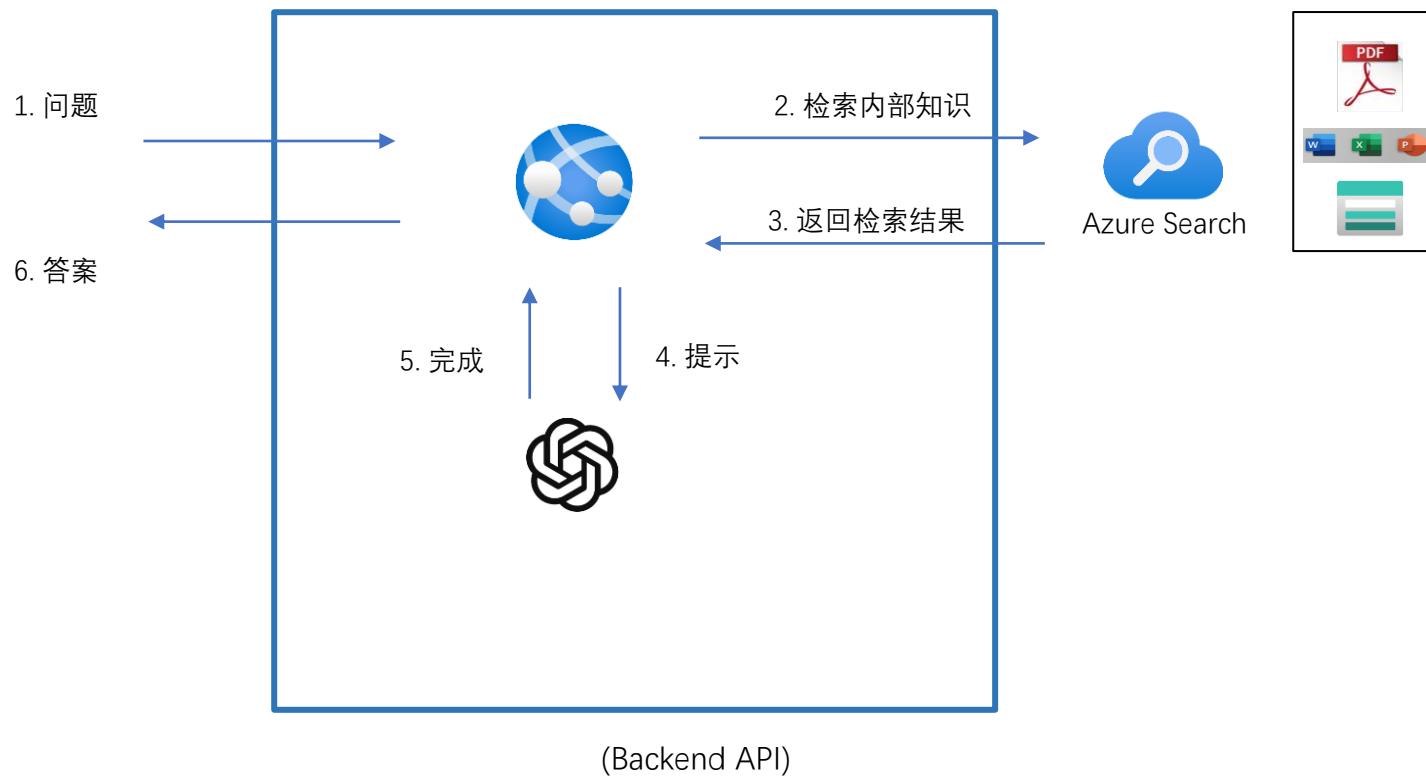
The screenshot shows the 'Publish' tab in the Azure OpenAI Web interface. The main area displays a success message: 'Publish succeeded on 3/16/2023 at 7:44 PM.' with an 'Open site' link. The left sidebar shows the 'Publish' tab selected under 'Connected Services'. The top bar shows the project name 'azure-openai-workshop - Web Deploy.pubxml' and a 'Publish' button.

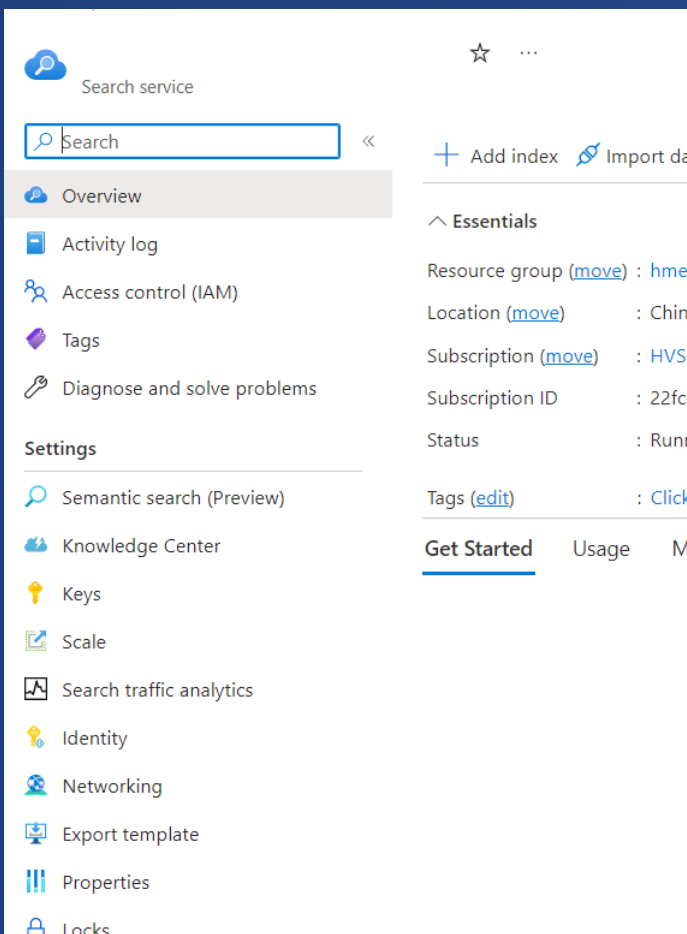


The screenshot shows the 'Configuration' page for the 'azure-openai-workshop' Web App. The page has a sidebar with navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Deployment slots, Deployment Center, Authentication, Application Insights, Identity, Backups, Custom domains, Certificates, Networking, and Scale up (App Service plan). The main content area is titled 'Configuration' and includes a search bar, a 'Refresh' button, and a 'Save' button. Below the search bar, there is a link to 'Click here to upgrade to a higher SKU and enable additional features.' The 'Application settings' section is active, showing a table of application settings. The table has columns for 'Name' and 'Value'. The settings listed are 'Settings:AIName' and 'Settings:ApiBaseUrl', both with 'Hidden value. Click' links. The 'Connection strings' section is also visible, showing a table with columns for 'Name' and 'Value'.

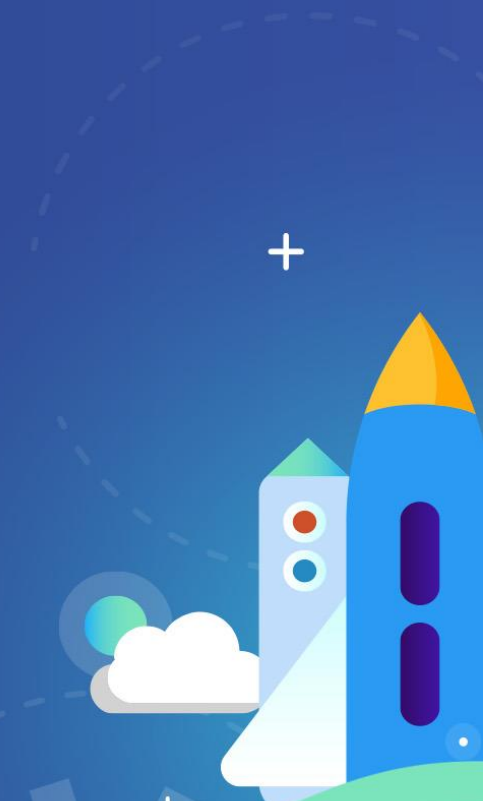
Name	Value
Settings:AIName	Hidden value. Click
Settings:ApiBaseUrl	Hidden value. Click

ChatGPT + enterprise knowledge base/data





Create an Azure Search service



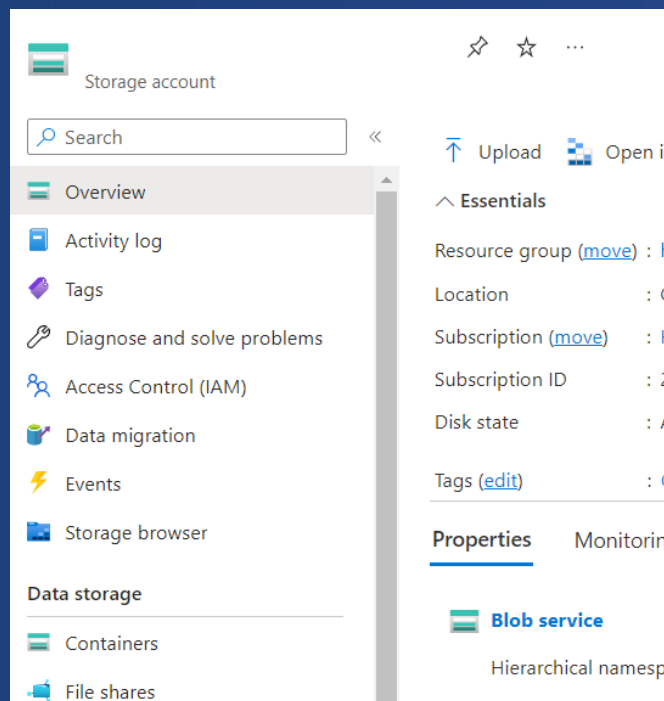
Create an Azure Search index

```
var indexClient = new SearchIndexClient(new Uri(searchServiceEndPoint), new AzureKeyCredential(searchServiceKey));
var indexerClient = new SearchIndexerClient(new Uri(searchServiceEndPoint), new AzureKeyCredential(searchServiceKey));

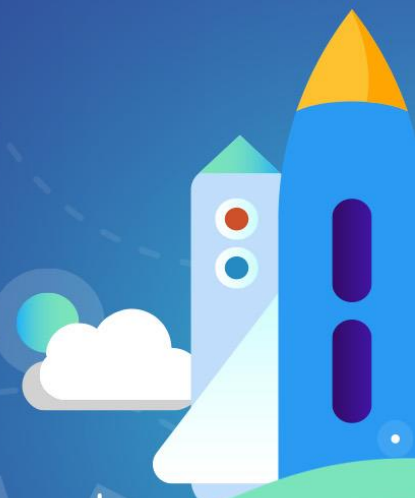
var searchFields = new List<SearchField>
{
    new SimpleField("Id", SearchFieldType.String) { IsKey = true, IsFilterable = true, IsSortable = true },
    new SearchableField("Name") { IsFilterable = true, IsSortable = true },
    new SearchableField("Content"), // large content don't enable filterable, sortable, faceting
};
var index = new SearchIndex(searchIndexName, searchFields);
await indexClient.CreateOrUpdateIndexAsync(index);

var docDataSource = new SearchIndexerDataSourceConnection(
    assetIndexDataSourceName,
    SearchIndexerDataSourceType.AzureBlob,
    assetBlobConnectionString,
    new SearchIndexerDataContainer(assetBlobContainerName)
);
await indexerClient.CreateOrUpdateDataSourceConnectionAsync(docDataSource);
var docIndexerParameters = new IndexingParameters();
docIndexerParameters.IndexingParametersConfiguration = new IndexingParametersConfiguration();
docIndexerParameters.IndexingParametersConfiguration.IndexedFileNameExtensions = ".pdf, .docx, .doc, .docm, .pptx, .ppt, .pptm";
docIndexerParameters.IndexingParametersConfiguration.DataToExtract = BlobIndexerDataToExtract.ContentAndMetadata;
var docIndexer = new SearchIndexer(assetIndexerName, docDataSource.Name, index.Name)
{
    Parameters = docIndexerParameters,
    Schedule = new IndexingSchedule(TimeSpan.FromDays(1)),
    FieldMappings =
    {
        new FieldMapping("Id") { TargetFieldName = "Id" },
        new FieldMapping("Name") { TargetFieldName = "Name", MappingFunction = new FieldMappingFunction("urlDecode") },
        new FieldMapping("content") { TargetFieldName = "Content" }
    }
};
await indexerClient.CreateOrUpdateIndexerAsync(docIndexer);
```

Create a Storage Account (to store enterprise knowledge base files)



1. Create a storage account
2. Create a container in the Blob service "workshop"



Upload a sample document (enterprise knowledge base file)

CONTENT-CONTROL:

CONTENT-TYPE:

CONTENT-MD5:

CONTENT-ENCODING:

CONTENT-LANGUAGE:

CONTENT-DISPOSITION:

LEASE STATUS: Unlocked

LEASE STATE: Available




LEASE DURATION: -

COPY STATUS: -

COPY COMPLETION TIME: -

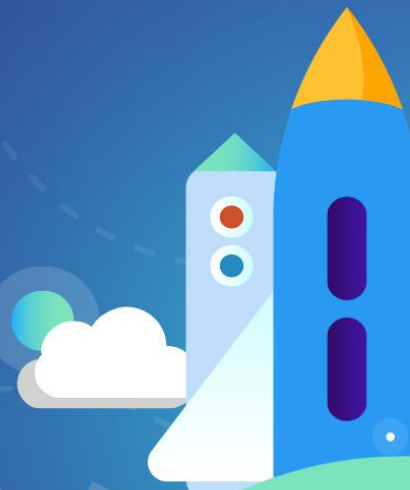
[Undelete](#)

Metadata

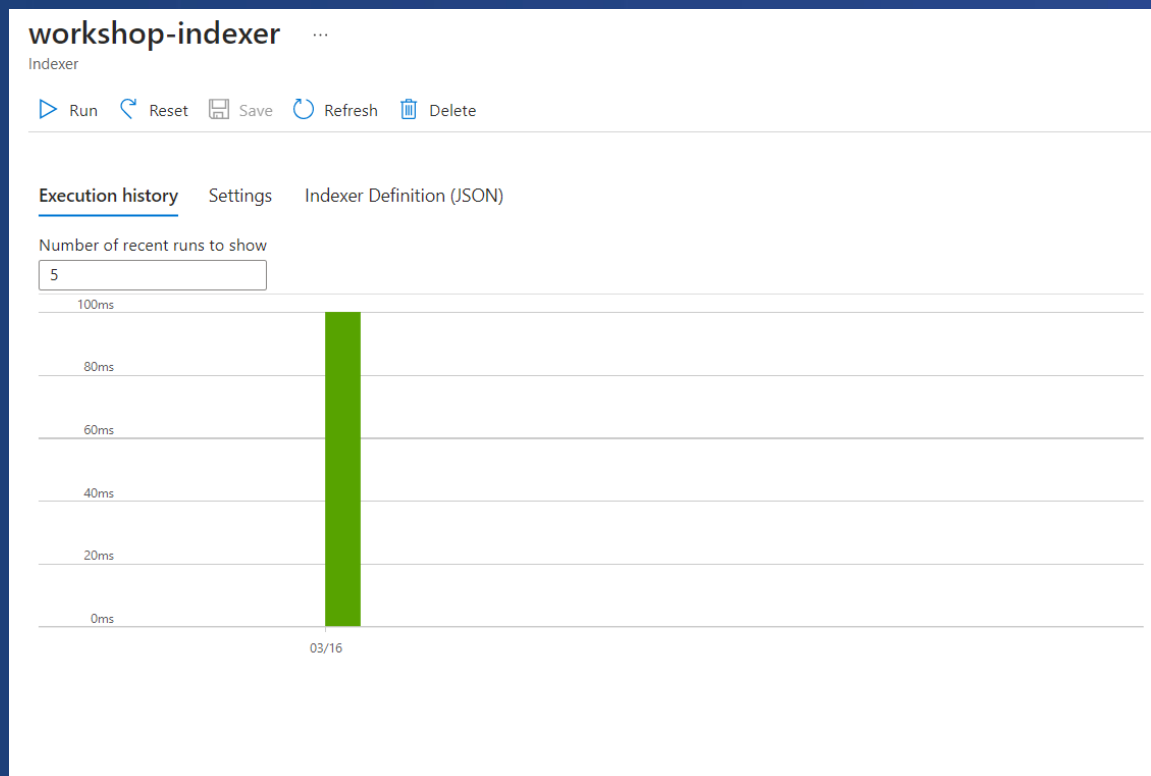
Key	Value
Id	doc1 
Name	产品介绍  
<input type="text"/>	<input type="text"/>

Blob index tags

1. Upload the document
2. Edit the document metadata:
 - Id
 - Name



Run the indexer



Debug indexes

workshop-index ...

Save Discard Refresh Create Demo App Edit JSON Delete

Documents 0 Storage 0 Bytes

Search explorer Fields CORS Scoring profiles

Query string 产品 API version 2021-04-30-Preview Search

Request URL fexes/workshop-index/docs?api-version=2021-04-30-Preview&search=%E4%BA%A7%E5%93%81

Results

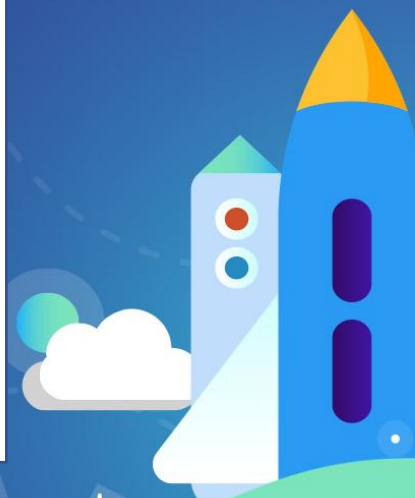
```
1 {
2   "@odata.context": "https://hmedia-search-test12.search.azure.cn/indexes('workshop-index')/$metadata#docs(*)",
3   "value": [
4     {
5       "@search.score": 1.6726563,
6       "Id": "doc1",
7       "Name": "产品介绍",
8       "Content": "\nbopoda\n产品介绍\n\nbopoda 博普达\n\n\nbopoda 博普达\n\n\n产品家族\n\nwww.bopoda.cn\n2\n\nVideo Hub\n视频中"
9     }
10  ]
11 }
```

reference:
<https://learn.microsoft.com/en-us/azure/search/search-explorer>



ChatGPT + enterprise knowledge base/data

```
var internalData = await SearchEnterpriseData(prompt);
var internalResult = internalData?.Content ?? "";
var options = new CompletionsOptions
{
    MaxTokens = MaxTokens,
    Prompt = { BuildPropmtGPT3(prompt, internalResult) }
};
var completions = await _client.GetCompletionsAsync(_config.DeploymentId, options);
var completion = completions.Value.Choices[0].Text;
return completion;
```



Code: About configuration information

You must configure the following information:

EnterpriseBotName

EnterpriseName

EnterpriseNoAnswerFallback

EnterpriseSearchApiURL

EnterpriseSearchApiKey

EnterpriseSearchIndex

```
"EnterpriseBotName": "",  
"EnterpriseName": "",  
"EnterpriseNoAnswerFallback": "",  
"EnterpriseSearchApiURL": "",  
"EnterpriseSearchApiKey": "",  
"EnterpriseSearchIndex": ""  
}
```


Thanks

