Progress Report
hshahin2

1. Current Progress Made

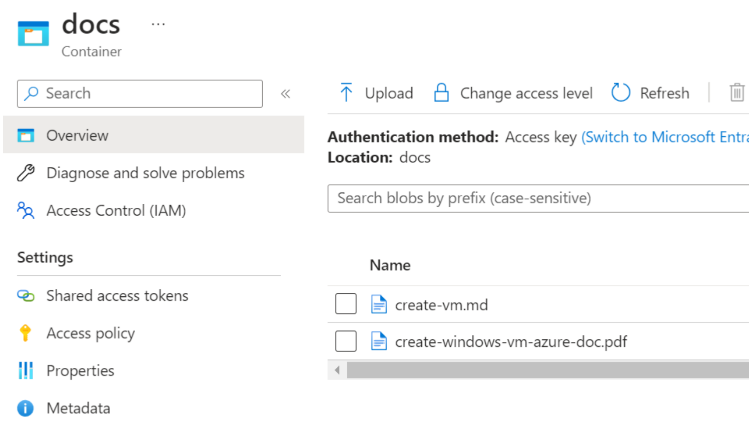
At a high level, the progress made thus far has been focused on building out the necessary Azure resources and confirming the Search Service and OpenAI Service produces accurate results.

The detailed tasks are noted below:

Task: Store a Sample Set of Azure Documents and GitHub README Documents in Azure Storage

Status: Complete

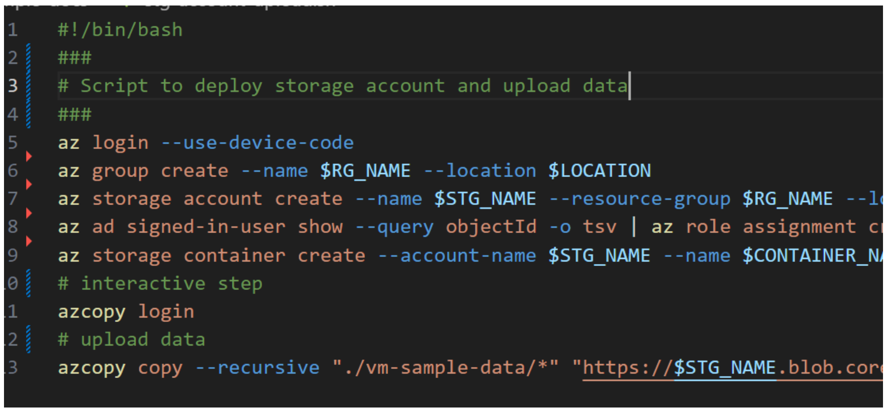
This task is currently complete with a small sample set of documents being used for testing. It is expected that the sample set of documents will grow, but for now there is an Azure Blob Storage Account (with associated deployment commands) created and storing a basic PDF and Markdown file for testing.



Authentication method: Access key (Switch to Microsoft Entra
Location: docs

Search blobs by prefix (case-sensitive)

Name
create-vm.md
create-windows-vm-azure-doc.pdf



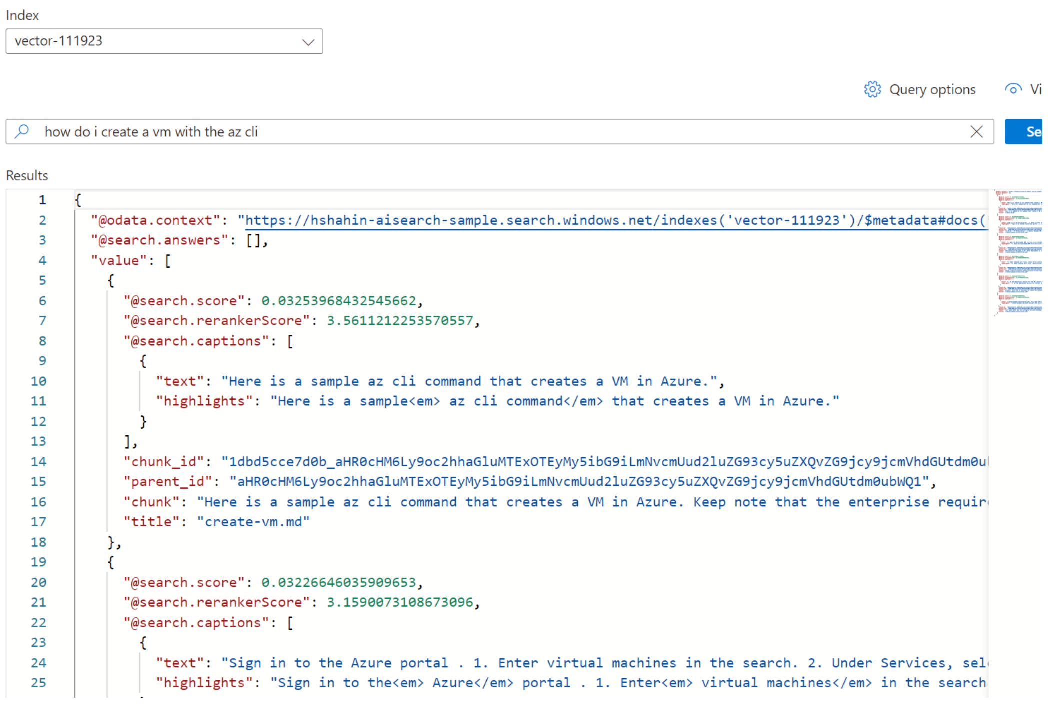
```
#!/bin/bash
###
# Script to deploy storage account and upload data
###
az login --use-device-code
az group create --name $RG_NAME --location $LOCATION
az storage account create --name $STG_NAME --resource-group $RG_NAME --l
az ad signed-in-user show --query objectId -o tsv | az role assignment c
az storage container create --account-name $STG_NAME --name $CONTAINER_N
# interactive step
azcopy login
# upload data
azcopy copy --recursive "./vm-sample-data/*" "https://$STG_NAME.blob.cor
```

Task: Deploy Azure Cognitive Search and Test to Confirm Index Returns Relevant Documents

Status: In-Progress

So far an Azure Cognitive Search (now called Azure AI Search) has been deployed and an index has been created on the data which supports both semantic search and vector search using the newly released preview capability of "integrated vectorization" (https://learn.microsoft.com/en-us/azure/search/vector-search-integrated-vectorization). What remains for this task is completing the documentation and associated scripts and Infrastructure-as-Code to allow others to deploy this. Currently, the Azure Search instance was deployed through the Azure Portal for testing and development.

The screenshot below is a sample query executed against the Search instance deployed in Azure, which returns relevant VM documentation based on the samples uploaded to the blob storage account.



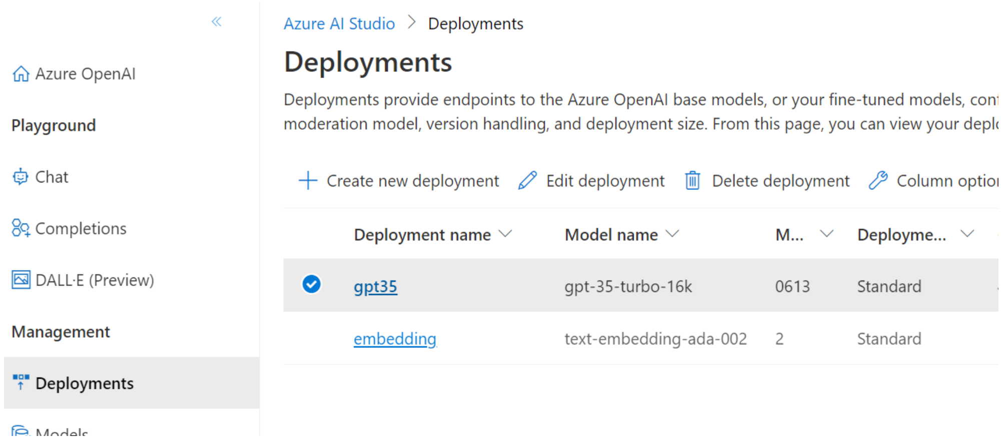
```
{
  "@odata.context": "https://hshahin-aisearch-sample.search.windows.net/indexes('vector-111923')/$metadata#docs(
  "@search.answers": [],
  "value": [
    {
      "@search.score": 0.03253968432545662,
      "@search.rerankerScore": 3.5611212253570557,
      "@search.captions": [
        {
          "text": "Here is a sample az cli command that creates a VM in Azure.",
          "highlights": "Here is a sample<em> az cli command</em> that creates a VM in Azure."
        }
      ],
      "chunk_id": "1dbd5cce7d0b_aHR0cHM6Ly9oc2hhaGluMTExOTEyMy5ibG9iLmNvcmUud2luZG93cy5uZXQvZG9jcy9jcmVhdGUtdm0u
      "parent_id": "aHR0cHM6Ly9oc2hhaGluMTExOTEyMy5ibG9iLmNvcmUud2luZG93cy5uZXQvZG9jcy9jcmVhdGUtdm0ubWQ1",
      "chunk": "Here is a sample az cli command that creates a VM in Azure. Keep note that the enterprise requir
      "title": "create-vm.md"
    },
    {
      "@search.score": 0.03226646035909653,
      "@search.rerankerScore": 3.1590073108673096,
      "@search.captions": [
        {
          "text": "Sign in to the Azure portal . 1. Enter virtual machines in the search. 2. Under Services, sel
          "highlights": "Sign in to the<em> Azure</em> portal . 1. Enter<em> virtual machines</em> in the search
```

Task: Deploy Azure OpenAI Service that Integrates with Azure Cognitive Search

Status: In-Progress

The Azure OpenAI service has been deployed with both a GPT model and embeddings model (shown below in the screenshot). Additionally, using the OpenAI on your data capability (https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/use-your-data?tabs=ai-search) I was able to link the Search Index with the OpenAI service so that chats are grounded using the data in the index. This is demonstrated below in the sample chat where a question brings back a citation referencing the data in the Search index.

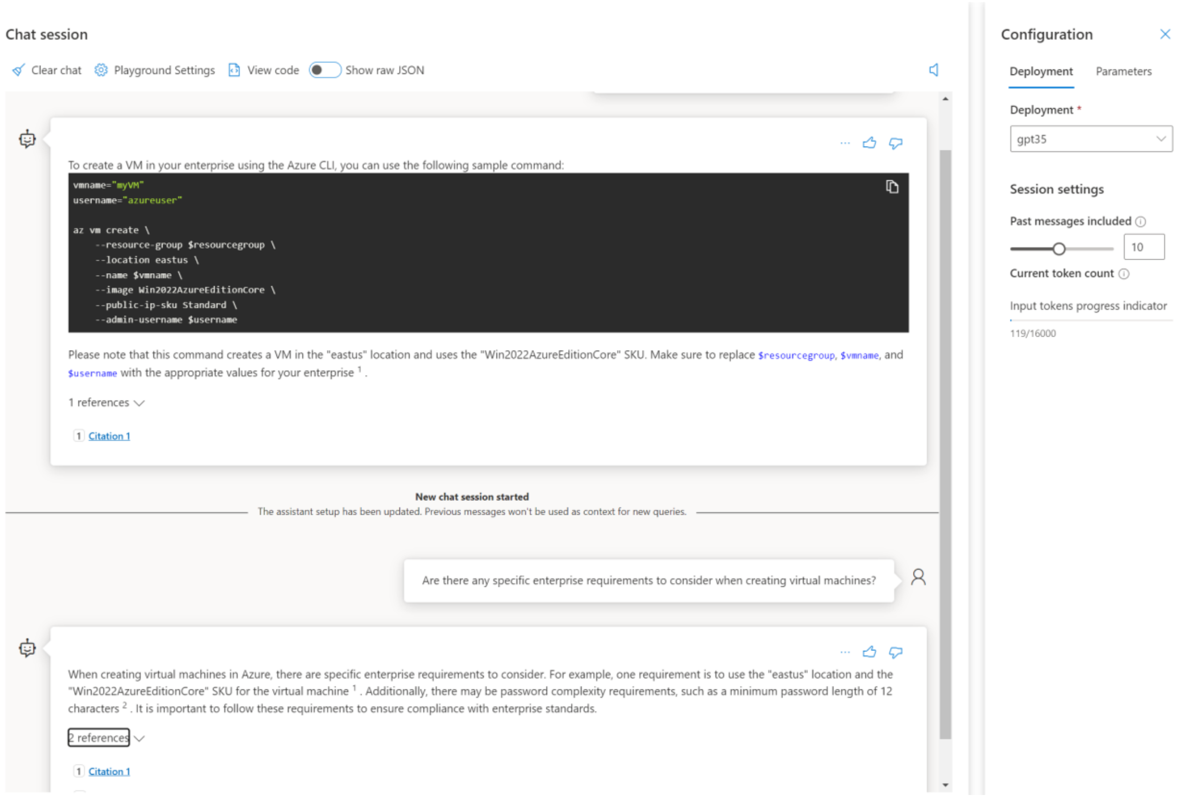
This task is still in progress. It is necessary to continue refining the approach/prompts when using OpenAI, and also to modify the system message to ensure queries are accurate but flexible for end-users. Additionally, further testing is needed to confirm whether the Preview “Bring Your Own Data” capability is the right fit for this project. Also left as a task is documenting the deployment of this resource in Azure and building automation for the deployment if possible.



Deployments

Deployments provide endpoints to the Azure OpenAI base models, or your fine-tuned models, con
moderation model, version handling, and deployment size. From this page, you can view your depl

	Deployment name	Model name	M...	Deployme...
✓	gpt35	gpt-35-turbo-16k	0613	Standard
	embedding	text-embedding-ada-002	2	Standard



Deployment *: gpt35

Past messages included: 10

119/16000

To create a VM in your enterprise using the Azure CLI, you can use the following sample command:

```
vmname="myVM"
username="azureuser"

az vm create \
    --resource-group $resourcegroup \
    --location eastus \
    --name $vmname \
    --image Win2022AzureEditionCore \
    --public-ip-sku Standard \
    --admin-username $username
```

Please note that this command creates a VM in the "eastus" location and uses the "Win2022AzureEditionCore" SKU. Make sure to replace `$resourcegroup`, `$vmname`, and `$username` with the appropriate values for your enterprise [1].

1 references

[1] Citation 1

New chat session started

The assistant setup has been updated. Previous messages won't be used as context for new queries.

Are there any specific enterprise requirements to consider when creating virtual machines?

When creating virtual machines in Azure, there are specific enterprise requirements to consider. For example, one requirement is to use the "eastus" location and the "Win2022AzureEditionCore" SKU for the virtual machine [1]. Additionally, there may be password complexity requirements, such as a minimum password length of 12 characters [2]. It is important to follow these requirements to ensure compliance with enterprise standards.

2 references

[1] Citation 1

2. Remaining Tasks

At a high level, what remains for this project is building out a custom frontend/backend that can be integrated with the deployed Azure resources to produce a fully functioning chat application using the custom data uploaded to the Search Service. I have identified multiple GitHub sample repos that can accelerate the development and give me a base to start from.

Additionally, what remains is containerizing the applications and documenting the full solution. If possible, I would like to develop the automation/Infrastructure-as-Code necessary for others to deploy this solution in their own Azure environment.

The listed tasks are noted below:

1. Storing a sample set of Azure documents and GitHub Readme documents in Azure Storage
2. Creating an index on the documents using Azure Cognitive Search and testing to confirm that the index functions as expected
3. Creating an Azure OpenAI service that integrates with Cognitive Search
4. Building a backend that calls Azure OpenAI with the end users query and returns a response
5. Building a frontend that enables end-users to ask their questions and integrates with the developed backend to return responses.
6. Containerization of frontend and backend
7. Documentation to deploy solution
8. BONUS: Infrastructure as Code to redeploy the solution in Azure with automation

3. Any Challenges/Issues Being Faced

One challenge I have moving forward is deciphering how best to integrate Azure OpenAI with Azure Search. Right now, the solution is using the “Bring Your Own Data” preview capability with Azure OpenAI, but I will need to test other approaches to determine whether the “Bring Your Own Data” capability is limiting results. Another approach instead of using this preview feature would be to manually develop the retrieval augmented generation approach such that when one inputs a prompt to the GPT model, the application first calls the Search Service, runs a query to retrieve relevant documentation, and then augments the prompt with the retrieved content. The benefit of the “Bring Your Own Data” capability is that this is all completed by the service itself, which means less code is necessary to develop to still apply the augmentation to the prompt.

Another challenge will be ensuring that others can build this solution within their own environment. Given that key components of this architecture are deployed in Azure, it will be necessary for others to have an Azure account and potentially incur fees to run the solution as I have it deployed. The goal is that the documentation (and the bonus automation / Infrastructure-as-Code) will help others quickly build the solution in their own tenants in Azure. Additionally, if they are unable to build the solution, I am planning to record the solution being run in my environment so that others can understand the capabilities and see how it works.

Finally, the last challenge I will face is completing the development work, which is the core component of the solution that is left to be completed. This will take a lot of time, and as mentioned there are sample applications I can likely build from to shorten the development time.