



Microsoft Microhack

Hands-On Lab: RAG AI & Your Data

Contents

Overview	2
Customer Scenario	3
Background.....	4
Business Problem.....	4
Technical Problem	5
Goals.....	6
Challenges	7
Challenge 0: Environment Setup.....	8
Challenge 1: Deploying Search Service	13
Challenge 2: Indexing Data	18
Challenge 3: Retrieving Data.....	24
Challenge 4: Integrating RAG with Application	28
Advanced Challenges.....	33
Challenge 5: Content Safety	34
Challenge 6: Load Testing.....	38
Challenge 7: Security in AI	49
Congratulations!.....	57
Cleaning Up Environment.....	58
Ideas for Enhancements	58

Overview

The Micohack event is designed to engage technical roles through a condensed, half-day hands-on hack experience. Leveraging the latest Microsoft technologies, this event provides participants with the opportunity to work on real-world problems, collaborate with peers, and explore innovative solutions.

The Microhack event is divided into several key challenges, each carefully crafted to test and expand the participants' proficiency with Microsoft's suite of tools. These challenges are not only technical in nature but also reflect real-world scenarios that businesses face, providing a comprehensive understanding of how to apply theoretical knowledge practically.

Hack Duration: 2 hours

The event kicks off with an initial overview of the customer scenario for the business problem the participants will solve by leveraging cutting-edge technology and services.

Following this, the team will complete the setup phase, where participants ensure that their development environments are correctly configured, and all necessary tools are ready for use.

Finally, they will tackle the first challenge, which involves identifying key ideas that underpin the implementation of Microsoft technologies in solving predefined problems.

Customer Scenario

Background

Contoso is a renowned technology company celebrated for its extensive product catalog that spans across various domains of the tech industry. With a history of innovation and excellence, Contoso has firmly established itself as a leader in providing cutting-edge technological solutions. The company offers a diverse range of products, including software applications, cloud services, and hardware devices, catering to both consumer and enterprise markets. Contoso's commitment to quality and performance is evident in its continuous development of state-of-the-art solutions that address the evolving needs of its global customer base.

The company's product portfolio is designed to enhance productivity, streamline operations, and foster collaboration. Contoso's flagship offerings, such as its robust cloud platform, advanced data analytics tools, and innovative artificial intelligence solutions, empower businesses to harness the power of technology for growth and efficiency. Additionally, Contoso's hardware devices, from sleek laptops to powerful servers, are engineered to deliver superior performance and reliability. With a customer-centric approach, Contoso not only provides top-tier products but also ensures exceptional support and services, solidifying its reputation as a trusted partner in the technological landscape.

In addition to its impressive product offerings, Contoso also boasts an extensive and complex employee benefits program designed to attract and retain top talent. This comprehensive package includes competitive salaries, health and wellness programs, retirement savings plans, and professional development opportunities. The company's benefits are meticulously designed to address the diverse needs of its employees, fostering a supportive and engaging work environment. From flexible working arrangements to generous vacation policies, Contoso ensures that its workforce has the resources and support necessary to thrive both professionally and personally.

Business Problem

Despite its many successes, Contoso faces significant business challenges due to its expansive employee benefits catalog and intricate ecosystem. The sheer volume and variety of benefits create complexity in managing and synchronizing information, updates, and support services. This complexity is further compounded by the integration of diverse benefits programs, each requiring meticulous coordination to ensure seamless interoperability and optimal performance.

Another pressing issue is the legacy documentation that accompanies many of Contoso's benefits offerings. As these programs evolve, older documentation often becomes outdated, leading to inefficiencies and potential misunderstandings among both employees and internal teams. Ensuring that all benefits documentation is current, comprehensive, and accessible is crucial for maintaining high standards of employee support and facilitating smooth internal operations.

Contoso aims to streamline its benefits management processes and modernize its documentation practices. By leveraging advanced data analytics and AI-driven solutions to automate the updating and maintenance of benefits documentation, ensuring it remains relevant and user-friendly. Furthermore, consolidating benefits information into a unified, easily navigable platform will enhance both employee experience and operational efficiency, enabling Contoso to maintain its reputation as a leader in employee satisfaction.

These efforts are part of Contoso's broader strategy to optimize its business operations and support its diverse benefits ecosystem. By tackling the complexities associated with its large catalog and legacy documentation, Contoso is poised to deliver even greater value to its employees and continue its legacy of excellence in the tech industry.

Technical Problem

Chatoso, the existing chat application developed by Contoso, encounters significant technical challenges primarily due to its inability to access and utilize Contoso's extensive employee benefits catalog and information. The absence of Retrieval Augmented Generation (RAG) capabilities hinders Chatoso's potential to provide relevant and accurate responses to user inquiries regarding Contoso's diverse range of employee benefits. Without direct access to the latest benefits data, Chatoso struggles with outdated information, leading to inefficiencies in employee support and a diminished user experience.

The lack of RAG capabilities impedes Chatoso's ability to handle complex queries that require contextual understanding and comprehensive data synthesis. Without the ability to retrieve relevant benefits information dynamically, the chatbot fails to provide nuanced and tailored responses, which are essential for addressing diverse employee needs. This deficiency not only affects employee satisfaction but also places additional strain on human support teams, who must compensate for the chatbot's shortcomings by handling routine queries that could otherwise be automated.

Integrating Retrieval Augmented Generation capabilities into Chatoso is imperative for overcoming these technical challenges. By enabling the chatbot to access and utilize Contoso's extensive employee benefits catalog, Chatoso can deliver accurate, up-to-date, and contextually relevant responses, thereby enhancing employee experience and operational efficiency. This integration will position Chatoso as a more effective tool in Contoso's employee support strategy, aligning with the company's commitment to leveraging cutting-edge technology for superior service delivery.

Goals

To address these challenges - Contoso has chosen you and your team - to help with a series of challenges. The goal of which is to upgrade an existing Contoso chat application, called Chatoso, to enable it with Retrieval Augmented Generation (RAG) with an example set of Contoso's employee benefits catalog.

Contoso has provided a code repository for deployment of a development environment that is a replica of the Chatoso production environment. Deploy this environment and enhance Chatoso with RAG capabilities leveraging Azure technologies to enable product catalog understanding for Contoso.

Challenges

Challenge 0: Environment Setup

Overview

We will set up the initial environment for you to build on top of during your Microhack. This comprehensive setup includes configuring essential Azure services and ensuring access to all necessary resources. Participants will familiarize themselves with the architecture, gaining insights into how various components interact to create a cohesive solution. With the foundational environment in place, the focus will shift seamlessly to the first Microhack Challenge endeavor.

Prerequisites

- Azure Developer CLI: [Download azd for Windows](#), [Other OS's](#).
- Powershell 7+ with AZ module (Windows only): [Powershell](#), [AZ Module](#)
- Git: [Download Git](#)
- Node.js 16+ [windows/mac linux/wsl](#)
- Python 3.11: [Download Python](#)
- Initiate an [Azure AI services creation](#) and agree to the Responsible AI terms **
 - ** If you have not created an Azure AI service resource in the subscription before

Related Technologies

[Azure Developer CLI](#)

- The Azure Developer CLI (azd) is an open-source tool that accelerates provisioning and deploying app resources on Azure. azd provides best practice, developer-friendly commands that map to key stages in your development workflow, whether you're working in the terminal, an integrated development environment (IDE), or through CI/CD (continuous integration/continuous deployment) pipelines.
- azd uses [extensible blueprint templates](#) that include everything you need to get an application up and running on Azure. These templates include:
 - Reusable infrastructure as code assets to provision cloud resources services using Bicep or Terraform.
 - Proof-of-concept or starter app code that can be customized or replaced with your own app code.

- Configuration files to handle deploying your app to the provisioned resources.
- Optionally, pipeline workflow files for GitHub Actions or Azure Pipelines to enable CI/CD integrations.

[Azure Bicep](#)

- Bicep is a domain-specific language that uses declarative syntax to deploy Azure resources. In a Bicep file, you define the infrastructure you want to deploy to Azure and then use that file throughout the development lifecycle to repeatedly deploy that infrastructure. Your resources are deployed in a consistent manner.
- Bicep provides concise syntax, reliable type safety, and support for reusing code. Bicep offers a first-class authoring experience for your [infrastructure-as-code](#) solutions in Azure.
- Support for all resource types and API versions: Bicep immediately supports all preview and GA versions for Azure services. As soon as a resource provider introduces new resource types and API versions, you can use them in your Bicep file. You don't need to wait for tools to be updated before using the new services.
- Simple syntax: When compared to the equivalent JSON template, Bicep files are more concise and easier to read. Bicep doesn't require prior knowledge of programming languages. Bicep syntax is declarative and specifies which resources and resource properties you want to deploy.

Steps

1. [Download](https://github.com/Boykai/octo-microhack-rag-ai-and-your-data) the GitHub repository - <https://github.com/Boykai/octo-microhack-rag-ai-and-your-data>
2. Open a terminal window.
3. Navigate to the downloaded directory `octo-microhack-rag-ai-and-your-data/` and enter:

`azd init`

4. Give the environment a unique name. This will be used to create your resources. For example, `cowboy-hats` would create resource group `rg-cowboy-hats`. See [Define your naming convention - Cloud Adoption Framework | Microsoft Learn](#) for more details.

a. Enter a new environment name: `some-name-here`

5. Login to Azure (complete both logins below):

a. Azure Developer CLI:

`azd auth login`

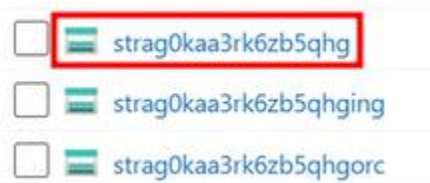
b. Azure CLI:

`az login`

6. Select your Azure Subscription from the list.
7. Start building the infrastructure and components deployment:

`azd up`

8. Select your Azure Subscription from the list.
9. Select Azure region. *Recommended: East US 2 (eastus2)*. See [Regional Selection](#).
10. Confirm successful deployment via [Azure Portal](#)
11. [Upload your documents](#) to the 'documents' folder located in the storage account. The name of this account should start with 'strag'. This is the default storage account, as shown in the sample image below.

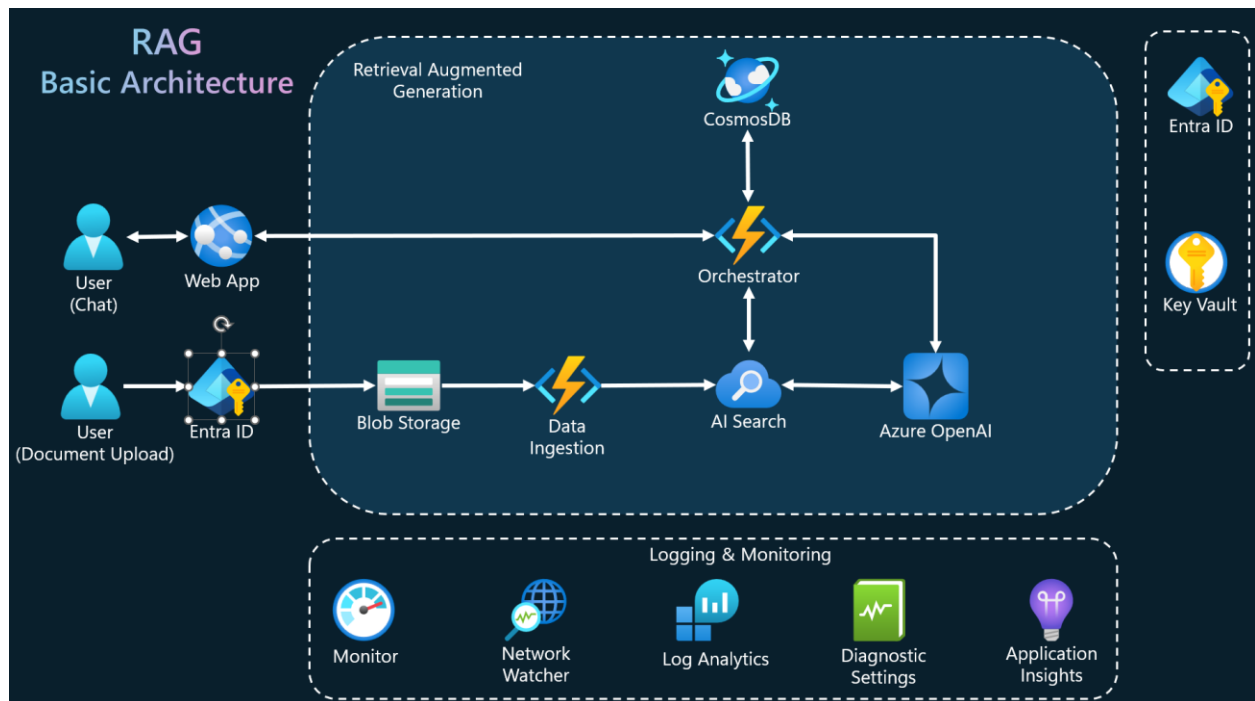


12. Sample documents are in `samples/`. Upload all documents.
13. Done!

Architecture

Basic Architecture

Azure Search AI not deployed with Azure Developer CLI



Success Criteria

- ☐ Successfully completed all Challenge 0 Steps without error
- ☐ Successfully deployed all resources in architecture, excluding Azure AI Search
- ☐ Understanding of architecture and its components
- ☐ Understanding GitHub code repository and its components

Useful Resources

- [Cloning a repository - GitHub Docs](#)
- [Azure Developer CLI \(azd\) | Microsoft Learn](#)
- [Get started using Azure Developer CLI | Microsoft Learn](#)
- [Install Azure PowerShell on Windows | Microsoft Learn](#)
- [Azure Command-Line Interface \(CLI\) - Overview | Microsoft Learn](#)
- [Bicep documentation | Microsoft Learn](#)
- [Azure AI Search documentation | Microsoft Learn](#)
- [Azure OpenAI Service documentation | Microsoft Learn](#)
- [Azure Functions documentation | Microsoft Learn](#)
- [Azure App Service documentation | Microsoft Learn](#)
- [Azure Cosmos DB | Microsoft Learn](#)
- [Azure Key Vault documentation | Microsoft Learn](#)
- [Azure Blob Storage documentation | Microsoft Learn](#)
- [Azure Monitor documentation | Microsoft Learn](#)
- [Network Watcher documentation | Microsoft Learn](#)
- [Diagnostic Settings documentation | Microsoft Learn](#)
- [Application Insights documentation | Microsoft Learn](#)
- [Microsoft Entra documentation | Microsoft Learn](#)
- [Azure/gpt-rag-ingestion](#)
- [Azure/gpt-rag-orchestrator](#)
- [Azure/gpt-rag-frontend](#)

Challenge 1: Deploying Search Service

Overview

Contoso, a leading enterprise in the technology sector, is embarking on a transformative project to enhance its data retrieval capabilities by deploying Azure AI Search. This initiative aims to streamline the vast amounts of information housed within Contoso's expansive digital ecosystem, providing faster and more accurate search results for its users. By leveraging Azure AI Search, Contoso hopes to improve customer experience, drive operational efficiency, and unlock actionable insights from their data repositories.

Your Challenge is to deploy an Azure AI Search service into the Microhack environment to help Contoso achieve its business goals.

Related Technologies

[Azure AI Search](#)

- Azure AI Search ([formerly known as "Azure Cognitive Search"](#)) is an enterprise-ready search and retrieval system, with a comprehensive set of advanced search technology, built for high-performance applications at any scale.
- Azure AI Search is the primary recommended retrieval system when building RAG-based applications on Azure, with native LLM integrations between Azure OpenAI Service and Azure Machine Learning.
- Azure AI Search can be used in both traditional and GenAI scenarios. Common use cases include knowledge base insights (catalog or document search), information discovery (data exploration), retrieval-augmented generation (RAG), and automation.
- When you create a search service, you work with the following capabilities:
 - A search engine for [vector search](#) and [full text](#) and [hybrid search](#) over a search index
 - Rich indexing with [integrated data chunking and vectorization](#), [lexical analysis](#) for text, and [optional applied AI](#) for content extraction and transformation
 - Rich query syntax for [vector queries](#), text search, [hybrid queries](#), fuzzy search, autocomplete, geo-search and others
 - Relevance and query performance tuning with [semantic ranking](#), [scoring profiles](#), [quantization for vector queries](#), and parameters for controlling query behaviors at runtime

- Azure scale, security, and reach
- Azure integration at the data layer, machine learning layer, Azure AI services and Azure OpenAI

Steps

1. In a web browser, open [Azure Portal](#).
2. Follow the steps to deploy an Azure Search Service - [Create a search service in the portal - Azure AI Search | Microsoft Learn](#).

Setting	Recommended Value
Subscription	<i>value from Challenge 0</i>
Resource Group	<i>value from Challenge 0</i>
Service name	search0-{resource-token}
Location	East US 2
Pricing tier	Standard
Replicas	1
Partitions	1
Endpoint connectivity (data)	Public

Resource-token is the randomized value created for resources during deployment.

For example, kv0-dqid8jff is created for your Key Vault resource:

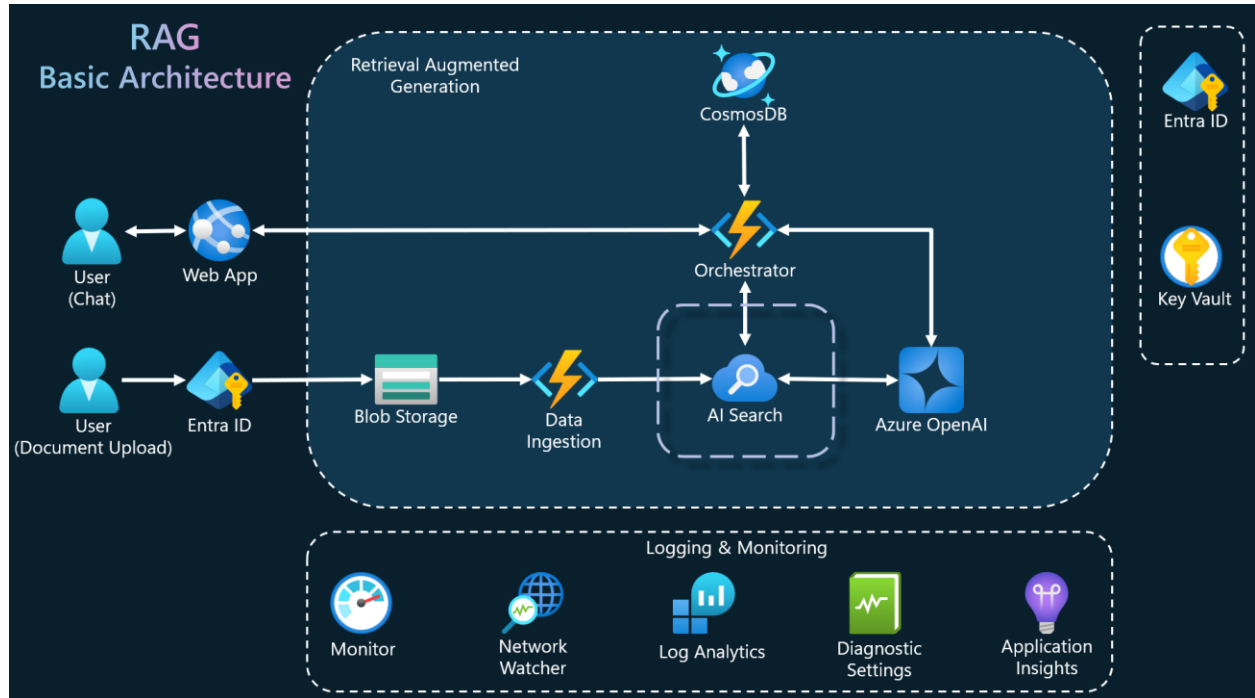
resource-token = dqid8jff

3. Done!

Architecture

Basic Architecture

Azure Search AI deployed with this Challenge



Success Criteria

- ☐ Successfully completed all Challenge 1 Steps without error
- ☐ Successfully deployed Azure AI Search into existing Azure environment
- ☐ Understanding of architecture and its components – and how Azure AI Search fits
- ☐ Understanding Azure AI Search deployment configuration options

Useful Resources

- [Create a search service in the portal - Azure AI Search | Microsoft Learn](#)
- [Azure AI Search documentation | Microsoft Learn](#)
- [Pricing - Azure AI Search | Microsoft Azure](#)
- [Azure AI Search - Generative Search | Microsoft Azure](#)
- [Selecting Subscription - Azure AI Search | Microsoft Learn](#)
- [Selecting Resource Group - Azure AI Search | Microsoft Learn](#)
- [Selecting Service name - Azure AI Search | Microsoft Learn](#)
- [Selecting Region - Azure AI Search | Microsoft Learn](#)
- [Selecting Pricing tier - Azure AI Search | Microsoft Learn](#)
- [Estimate capacity for query and index workloads - Azure AI Search | Microsoft Learn](#)
- [Configure network access - Azure AI Search | Microsoft Learn](#)
- [Enable role-based access control - Azure AI Search | Microsoft Learn](#)

Challenge 2: Indexing Data

Overview

Contoso had been dealing with the challenge of searching through their contents in a way that rightly matches the user's intents. The team at Contoso has decided to use the “Vector Index” to allow for searching the contents that are semantically related in the meaning and are searchable for given query in the natural language (NLP). This initiative aims to streamline the vast amounts of information housed within Contoso's expansive digital ecosystem, providing faster and more accurate search results for its users. Now that we have created Azure AI search in the previous challenge, let's build a vector index to help with searching semantically related contents at ease.

Your Challenge is to build a “vector” index in the previously provisioned Azure AI Search service into the Microhack environment.

Related Technologies

[Azure OpenAI Embeddings](#)

- An embedding is a special format of data representation that machine learning models and algorithms can easily use. The embedding is an information dense representation of the semantic meaning of a piece of text. Each embedding is a vector of floating-point numbers, such that the distance between two embeddings in the vector space is correlated with semantic similarity between two inputs in the original format.
- For example, if two texts are similar, then their vector representations should also be similar. Embeddings power vector similarity search in retrieval systems such as [Azure AI Search](#) (recommended) and in Azure databases such as [Azure Cosmos DB for MongoDB vCore](#) , [Azure SQL Database](#), and [Azure Database for PostgreSQL - Flexible Server](#).
- Azure OpenAI embeddings often rely on cosine similarity to compute similarity between documents and a query. From a mathematical perspective, cosine similarity measures the cosine of the angle between two vectors projected in a multidimensional space. This measurement is beneficial, because if two documents are far apart by Euclidean distance because of size, they could still have a smaller angle between them and therefore higher cosine similarity. For more information about cosine similarity equations, see [Cosine Similarity](#).

Vector Indexes

- Integrated vectorization is an extension of the indexing and query pipelines in Azure AI Search. It adds the following capabilities:
 - Data chunking during indexing
 - Text-to-vector conversion during indexing
 - Text-to-vector conversion during queries
- Data chunking isn't a hard requirement, but unless your raw documents are small, chunking is necessary for meeting the token input requirements of embedding models.
- Vector conversions are one-way: text-to-vector. There's no vector-to-text conversion for queries or results (for example, you can't convert a vector result to a human-readable string).
- Integrated data chunking and vectorization speeds up the development and minimizes maintenance tasks during data ingestion and query time because there are fewer external components to configure and manage. This capability is now generally available.

Steps

1. In a web browser, open [Azure Portal](#).
2. Navigate to the previously created resource group in the step above.
3. You will be creating a vector index in the Azure AI Studio. Navigate to the previously created Azure Open AI resource and then click on

Explore Azure AI Studio

- a. This will bring up the Azure AI studio UI. Click *Continue* if asked.
4. Navigate to *Chat Playground*.
5. Click *Add your data*.
6. Click *Add Data Source*. This will bring up a dialog box to enter the data source information
7. Next select *Azure Blob Storage* as the DataSource. Provide all information including location of storage account.
 - a. This is where raw documents are located/uploaded in the step earlier.
8. Also provide information about the previously created Azure AI Search resource.
9. In addition, Check *Add vector search to this resource* check box. Below is the screen shot

- Data source
- Data management
- Data connection
- Review and finish

Select or add data source

Your data source is used to ground the generated results with your data. Select an existing data source or create a new data connection with Azure Blob Storage, databases, search, URLs, or local files as the source the grounding data will be built from.

[Learn more about data privacy and security in Azure AI.](#)

Select data source *

Azure Blob Storage (preview) ▼

Subscription *

MCAPS-Hybrid-REQ-51005-2023... ▼

Select Azure Blob storage resource ⓘ *

strag0gqam4cgvqrnhk ▼ ↻

Select storage container ⓘ *

documents ▼

Select Azure AI Search resource ⓘ *

lp-micro-hack-search ▼ ↻

[Create a new Azure AI Search resource](#)

Enter the index name ⓘ *

new-vector-index-for-micro-hack

Using Azure AI Search will incur usage to your account. [View Pricing](#)

Indexer schedule ⓘ *

Once ▼

The indexer schedule can be changed in the Indexer settings for this search resource in the Azure portal.

☒ Add vector search to this search resource. ⓘ

Embedding model


Select an embedding model ⓘ *

Azure OpenAI - text-embedding-ada-002 ▼

Next

Cancel

10. Click *Next*.
11. Select Search type as *Hybrid (vector + keyword)*.
12. Select Chunk Size as *1024*.
13. Select *API key* on the *Next* screen.
14. Click *Save and Close*. A newly created vector index is seen created on the screen as below screen shot

Gain insights into your own data source. Your data is stored securely in your Azure subscription. [Learn more about how your data is protected.](#) 

Data source:	Search Resource:
Azure Blob Storage	lp-micro-hack-
	search
Index:	
new-vector-index-for-micro-hack	Chunk Size:
	1024
Advanced settings >	

15. At this point you have successfully created a vector index.

new-vector-index-for-micro-hack ...

[illegible]

16. Done!

Success Criteria

- ☐ Successfully completed all Challenge 2 Steps without error
- ☐ A quick test in search explorer, should return the results with vectors
- ☐ Successfully deployed Azure AI Search into existing Azure environment
- ☐ Understanding of vector indexes
- ☐ Understanding Azure AI Search vector configuration options

Useful Resources

- [Create an index - Azure AI Search | Microsoft Learn](#)
- [Create an indexer - Azure AI Search | Microsoft Learn](#)
- [Data sources gallery - Azure AI Search | Microsoft Learn](#)
- [Full-text query how-to - Azure AI Search | Microsoft Learn](#)
- [How to shape search results - Azure AI Search | Microsoft Learn](#)
- [Create a vector index - Azure AI Search | Microsoft Learn](#)
- [Vector query how-to - Azure AI Search | Microsoft Learn](#)
- [Return a semantic answer - Azure AI Search | Microsoft Learn](#)
- [Enable or disable semantic ranker - Azure AI Search | Microsoft Learn](#)
- [Azure OpenAI Service embeddings | Microsoft Learn](#)

Challenge 3: Retrieving Data

Overview

Your Challenge is to understand and validate different data retrieval techniques using Azure AI Search service. You will be leveraging the Azure AI Search service deployed in the previous challenge; you also have Contoso business documents indexed in the AI Search service that you must use to confirm the search service is able to return relevant business documents using various search options.

By the end of this challenge, you will have hands-on experience with Azure AI Search, a solid understanding of vectors, and the ability to implement and optimize various search options to retrieve relevant documents effectively.

You will be using Python Notebooks to connect to Azure AI Search and Azure OpenAI services to execute various search queries to get a deeper understanding of various search options (basic, semantic, vector & hybrid search).

Related Technologies

[Azure AI Search Python SDK](#)

- [Azure AI Search](#) (formerly known as "Azure Cognitive Search") is an AI-powered information retrieval platform that helps developers build rich search experiences and generative AI apps that combine large language models with enterprise data.
- Azure AI Search is well suited for the following application scenarios:
 - Consolidate varied content types into a single searchable index. To populate an index, you can push JSON documents that contain your content, or if your data is already in Azure, create an indexer to pull in data automatically.
 - Attach skillsets to an indexer to create searchable content from images and unstructured documents. A skillset leverages APIs from Azure AI Services for built-in OCR, entity recognition, key phrase extraction, language detection, text translation, and sentiment analysis. You can also add custom skills to integrate external processing of your content during data ingestion.
 - In a search client application, implement query logic and user experiences similar to commercial web search engines and chat-style apps.

- Use the `Azure.Search.Documents` client library to:
 - Submit queries using vector, keyword, and hybrid query forms.
 - Implement filtered queries for metadata, geospatial search, faceted navigation, or to narrow results based on filter criteria.
 - Create and manage search indexes.
 - Upload and update documents in the search index.
 - Create and manage indexers that pull data from Azure into an index.
 - Create and manage skillsets that add AI enrichment to data ingestion.
 - Create and manage analyzers for advanced text analysis or multi-lingual content.
 - Optimize results through semantic ranking and scoring profiles to factor in business logic or freshness.

[Azure AI Search Hybrid Search](#)

- Hybrid search is a combination of full text and vector queries that execute against a search index containing both searchable plain text content and generated embeddings. For query purposes, hybrid search is:
 - A single query request that includes both search and vectors query parameters
 - Executing in parallel
 - With merged results in the query response, scored using [Reciprocal Rank Fusion \(RRF\)](#)
- Watch this [embedded video](#) for an explanation and short demos of how hybrid retrieval contributes to high quality chat-style and copilot apps.

Prerequisites

- Python 3.11: [Download Python](#)
- Install Python SDK for AI Search and Azure OpenAI. Run below commands on terminal window:
 - `pip install azure-search-documents`
 - `pip install openai`

Steps

1. Locate Python Notebooks for Challenge 3:
 - Get *retrieve-data.ipynb* & *ai_search_sample.env* from the GitHub repo for this Microhack. These files are located under the *code/challenge-3-retrieve-data/* directory.
2. Connect to Azure AI Search Service:
 - We need to establish a connection with the Azure AI Search & Azure OpenAI service from the Python-based client application. Follow steps below to do that:
3. Rename *ai_search_sample.env* file to *ai_search.env*.
4. Update *ai_search.env* file with your AI Search & Azure OpenAI service details.
 - Here is an example of *ai_search.env* file.

```
AI_SEARCH_ENDPOINT="https://<search-service-name>.search.windows.net"
AI_SEARCH_KEY="YOUR SEARCH KEY WITHIN DOUBLE QUOTES"
AI_SEARCH_INDEX_NAME="ragindex"
AI_SEARCH_SEMANTIC_CONFIG_NAME="my-semantic-config"
AOAI_ENDPOINT="https://<aoai-service-name>.openai.azure.com"
AOAI_KEY="YOUR AOAI KEY WITHIN DOUBLE QUOTES"
AOAI_EMBEDDING_DEPLOYMENT_NAME="text-embedding-ada-002"
```

- You can get values for the above parameters using Azure portal.
5. Open Python Notebook.
 6. Using VS Code, Jupyter Notebook (or other compatible tools), open *retrieve-data.ipynb* and complete following tasks as per the [instructions provided in the notebook](#).
 - Perform Basic Search using Azure AI Search service
 - Perform Semantic Search using Azure AI Search service
 - Perform Vector Search using Azure AI Search service
 7. Done!

Success Criteria

- ☐ Successfully completed all Challenge 3 Steps without error
- ☐ Successfully connected to Azure AI Search service from Python client application
- ☐ Successfully performed Basic, Semantic and Vector search; validated the search results
- ☐ Successfully connected to Azure OpenAI Service and used the embedding model to generate embedding vectors to perform vector search
- ☐ Understanding various search options (basic, semantic, vector & hybrid search)

Useful Resources

- [Query types - Azure AI Search | Microsoft Learn](#)
- [Semantic ranking - Azure AI Search | Microsoft Learn](#)
- [How to generate embeddings with Azure OpenAI Service - Azure OpenAI | Microsoft Learn](#)
- [Examples of full Lucene query syntax - Azure AI Search | Microsoft Learn](#)
- [Vector query how-to - Azure AI Search | Microsoft Learn](#)
- [How to generate embeddings with Azure OpenAI Service - Azure OpenAI | Microsoft Learn](#)

Challenge 4: Integrating RAG with Application

Overview

Integrating Retrieval Augmented Generation (RAG) capabilities into Chatoso, an existing chat application, presents an exciting opportunity to enhance user interaction and information retrieval. By embedding RAG, Chatoso can leverage advanced AI techniques to provide more contextually relevant responses to user queries. This integration will not only improve the accuracy and relevance of the information shared but also significantly enrich the overall user experience. Enabling users to retrieve precise data on demand while maintaining a conversational flow aligns with the growing expectations for intelligent, responsive digital platforms.

The addition of RAG capabilities to Chatoso involves incorporating sophisticated AI models that can generate coherent, context-aware responses. This approach ensures that users receive answers that are not only accurate but also tailored to the specific context of their queries. By combining the strengths of retrieval and generation, Chatoso can transition from a simple query-response system to a dynamic, knowledge-rich conversational agent.

Related Technologies

[Semantic Kernel](#)

- Semantic Kernel is a lightweight, open-source development kit that lets you easily build AI agents and integrate the latest AI models into your C#, Python, or Java codebase. It serves as an efficient middleware that enables rapid delivery of enterprise-grade solutions.
- The kernel is the central component of Semantic Kernel. At its simplest, the kernel is a Dependency Injection container that manages all of the services and plugins necessary to run your AI application. If you provide all of your services and plugins to the kernel, they will then be seamlessly used by the AI as needed.
- Because the kernel has all of the services and plugins necessary to run both native code and AI services, it is used by nearly every component within the Semantic Kernel SDK to power your agents. This means that if you run any prompt or code in Semantic Kernel, the kernel will always be available to retrieve the necessary services and plugins.

Azure Application Service

- Azure App Service is an HTTP-based service for hosting web applications, REST APIs, and mobile back ends. You can develop in your favorite language, be it .NET, .NET Core, Java, Node.js, PHP, or Python. Applications run and scale with ease on both Windows and [Linux](#)-based environments.
- App Service adds the power of Microsoft Azure to your application, including improved security, load balancing, autoscaling, and automated management. Additionally, you can take advantage of its DevOps capabilities, such as continuous deployment from Azure DevOps, GitHub, Docker Hub, and other sources, package management, staging environments, custom domains, and TLS/SSL certificates.
- With App Service, you pay for the Azure compute resources you use. The compute resources you use are determined by the App Service plan that you run your apps on. For more information, see [Azure App Service plans overview](#).

Azure Key Vault

- Secrets Management - Azure Key Vault can be used to Securely store and tightly control access to tokens, passwords, certificates, API keys, and other secrets
- Key Management - Azure Key Vault can be used as a Key Management solution. Azure Key Vault makes it easy to create and control the encryption keys used to encrypt your data.
- Certificate Management - Azure Key Vault lets you easily provision, manage, and deploy public and private Transport Layer Security/Secure Sockets Layer (TLS/SSL) certificates for use with Azure and your internal connected resources.

Microsoft Entra

- Microsoft Entra is a family of identity and network access products. It enables organizations to implement a [Zero Trust](#) security strategy and create a [trust fabric](#) that verifies identities, validates access conditions, checks permissions, encrypts connection channels, and monitors for compromise.

Steps

1. In a web browser, open [Azure Portal](#).
2. Navigate to Azure Function resource for the Orchestration codebase.
 - a. `fnorch0-<your-resource-token>`
3. Select Settings → Environment variables.
4. Update the following:

Name	Value
AZURE_DB_CONVERSATIONS_CONTAINER_NAME	conversations
AZURE_DB_ID	dbgpt0-<your-resource-token>
AZURE_DB_MODELS_CONTAINER_NAME	models
AZURE_DB_NAME	db0-<your-resource-token>
AZURE_OPENAI_API_VERSION	2024-07-01-preview
AZURE_OPENAI_CHATGPT_DEPLOYMENT	chat
AZURE_OPENAI_CHATGPT_LLM_MONITORING	false
AZURE_OPENAI_CHATGPT_MODEL	gpt-4o
AZURE_OPENAI_EMBEDDING_DEPLOYMENT	text-embedding-ada-002
AZURE_OPENAI_EMBEDDING_MODEL	text-embedding-ada-002
AZURE_OPENAI_LOAD_BALANCING	false
AZURE_OPENAI_RESOURCE	oai0-<your-resource-token>
AZURE_OPENAI_STREAM	false
AZURE_SEARCH_API_KEY	<your-azure-search-api-key>
AZURE_SEARCH_API_VERSION	2024-07-01
AZURE_SEARCH_APPROACH	hybrid
AZURE_SEARCH_INDEX	<your-search-index-name>
AZURE_SEARCH_SERVICE	search0-<your-resource-token>
AZURE_SEARCH_SEMANTIC_SEARCH_CONFIG	<your-search-semantic-configuration-name>
AZURE_SEARCH_TRIMMING	false
AZURE_SEARCH_USE_SEMANTIC	true
FUNCTIONS_WORKER_RUNTIME	python
ORCHESTRATOR_MESSAGES_LANGUAGE	En
RETRIEVAL_PRIORITY	search

5. Select Apply.
6. Navigate to your Azure Key Vault resource.
7. Select Objects → Secrets → Generate/Import
8. Add the following secret:

Name	Value
azureSearchKey	<your-azure-ai-search-api-key>

9. Select Create.
10. Navigate to Azure AI Search service.
11. Select Access Control (IAM).
12. Select Add → Add Role Assignment.
13. Select *Search Index Data Contributor*. Select Next.
14. Select *Managed Identity*.
15. Select +*Select members*.
16. Select *Managed Identity = Function App*.
17. Select *fnorch0-<your-resource-token>* (Azure Function App with Orchestrator code).
18. Select Review + assign.
19. Repeat Steps 11 – 18.
 - a. Instead of *Index Data Contributor*, use *Search Service Contributor*.
20. Navigate to your Application Service.
 - a. <https://webgpt0-<your-resource-token>.azurewebsites.net>
21. Validate that you can return results from the RAG document knowledgebase.
22. Done!

Success Criteria

- ☐ Successfully completed all Challenge 4 Steps without error
- ☐ Successfully integrated Azure AI Search RAG into existing web application
- ☐ Understanding of Environment Variables, Secrets, and Managed Identities – and how Azure AI Search fits
- ☐ Understanding Azure integration configuration considerations

Useful Resources

- [Azure App Service documentation - Azure App Service | Microsoft Learn](#)
- [Azure Key Vault documentation | Microsoft Learn](#)
- [Microsoft Entra fundamentals documentation - Microsoft Entra | Microsoft Learn](#)
- [Environment variables and app settings reference - Azure App Service | Microsoft Learn](#)
- [What is identity and access management \(IAM\)? - Microsoft Entra | Microsoft Learn](#)
- [Managed identities for Azure resources | Microsoft Learn](#)

Advanced Challenges

Challenge 5: Content Safety

Overview

Contoso needs to update its AI chatbot by integrating advanced AI content filtering mechanisms to ensure that all interactions align with the company's code of conduct policies. This involves implementing comprehensive responsible AI practices, which include monitoring and moderating content in real-time to prevent the dissemination of inappropriate, harmful, or misleading information. By applying Azure AI Search's content filtering categories, Contoso can effectively identify and block content that violates company policies, thereby maintaining a safe and respectful environment for users.

To achieve this, Contoso must configure its AI chatbot with both standard and advanced content filtering options, ensuring that all forms of communication, whether textual or multimedia, are scrutinized for compliance. This includes applying for unrestricted content filters that allow for a more nuanced and thorough analysis of potentially problematic content. Additionally, the company should continuously update and refine these filters to adapt to emerging threats and changing user behaviors, thus safeguarding against new forms of content violations.

By embedding these AI-driven content safety measures into the chatbot, Contoso not only adheres to its code of conduct but also enhances user trust and engagement. This proactive approach to content moderation demonstrates the company's commitment to ethical AI deployment and sets a higher standard for digital interactions within its platform.

Azd automatically creates content filters profile with default severity threshold (*Medium*) for all content harms categories (*Hate, Violence, Sexual, Self-Harm*) and assigns it to provisioned AOAI model through post deployment script. However, if you want to customize them to be more or less restrictive, you can make changes to *scripts/rai/raipolicies.json* file.

Related Technologies

[Azure AI Content Safety](#)

- Azure AI Content Safety is an AI service that detects harmful user-generated and AI-generated content in applications and services. Azure AI Content Safety includes text and image APIs that allow you to detect material that is harmful. The interactive Content Safety Studio allows you to view, explore, and try out sample code for detecting harmful content across different modalities.

- Prompt shields: scans text for the risk of a User input attack on a Large Language Model.
- Groundedness detection: Detects whether the text responses of large language models (LLMs) are grounded in the source materials provided by the users.
- Protected material text detection: Scans AI-generated text for known text content (for example, song lyrics, articles, recipes, selected web content).
- Custom categories (standard) API: Lets you create and train your own custom content categories and scan text for matches.
- Custom categories (rapid) API: Lets you define emerging harmful content patterns and scan text and images for matches.
- Analyze text API: Scans text for sexual content, violence, hate, and self-harm with multi-severity levels.
- Analyze image API: Scans images for sexual content, violence, hate, and self-harm with multi-severity levels.

Steps

1. Navigate to code repository.
2. Open the following files and gain an understanding of their purpose within the application – as it relates to Content Safety:
 - `code/gpt-rag-orchestrator/orc/orchestrator.py`
 - `code/gpt-rag-orchestrator/orc/code_orchestration.py`
 - `code/gpt-rag-orchestrator/orc/plugins/ResponsibleAI/`
 - `scripts/rai/raipolicies.json`
3. Open the `scripts/rai/raipolicies.json` file to edit the Content Safety configuration.
4. Decrease Content Safety settings to the lowest and least restrictive. Save changes.
5. Open a terminal window.
6. To push changes to the Azure environment, navigate to the downloaded directory 'octo-microhack-rag-ai-and-your-data/' and enter:

azd provision

7. Experiment with chatbot and note changes in Content Safety restrictions.
8. Increase Content Safety settings to the highest and most restrictive. Save changes.
9. Open a terminal window.
10. To push changes to the Azure environment, navigate to the downloaded directory 'octo-microhack-rag-ai-and-your-data/' and enter:

azd provision

11. Experiment with chatbot and note changes in Content Safety restrictions.
12. View `docs/CUSTOMIZATIONS_CONTENT_FILTERING.md` for more details on Content Safety customizations with this codebase.
13. Done!

Success Criteria

- ☐ Successfully completed all Challenge 5 Steps without error
- ☐ Successfully updated deployment to increase Content Safety for RAG application
- ☐ Understanding of architecture, code, and its components
- ☐ Running examples of Content Safety in RAG application and analysis of results

Useful Resources

- [Overview of Responsible AI practices for AOAI models](#)
- [AOAI Content filtering categories](#)
- [Apply for unrestricted content filters via this form](#)
- [Azure AI Content Safety documentation | Microsoft Learn](#)

Challenge 6: Load Testing

Overview

Contoso wants to conduct load testing on their RAG-based (Retrieval-Augmented Generation) application to ascertain its capacity to manage a high volume of requests efficiently. Load testing is crucial for Contoso to ensure that the application can handle peak traffic volumes without performance degradation or service interruptions.

By simulating a variety of user load scenarios, Contoso aims to identify potential bottlenecks and understand the limits of the application's request-handling capabilities. This process will provide valuable insights into the infrastructure's robustness, enabling the company to make informed decisions about scaling and optimizing the application to meet growing demand.

Ultimately, successful load testing will help Contoso guarantee a seamless and reliable user experience, even during periods of intense usage, thereby maintaining user satisfaction and trust.

This test is structured to emulate a fundamental scenario, showcasing how you can construct your own test scenario utilizing Azure Load Testing. This test scenario is adaptable and can be tailored to more accurately meet your specific requirements.

To understand the details of the test scenario, it's crucial to know that Azure Load Testing employs the JMeter standard for defining your test scenario. This involves the use of two key files: *loadtest/loadtest.jmx* and *loadtest/user.properties*. The *loadtest.jmx* file outlines the JMeter test plan, while *user.properties* configures properties like virtual users and test duration.

Upon examining these two files, you'll see that our test simulates 10 virtual testers, each sending a question to the orchestrator component. There's a 10-second delay before the test starts, then it takes 30 seconds for all the testers to be active during the ramp-up period. The test is designed to run for a maximum of 360 seconds. To simulate real users, there's a gap between requests, often referred to as "Think Time" of 1 second, followed by a random delay or "Pause", which ranges between 0 and 1 second.

In simple terms, the test is conducted by sending HTTP requests to a specific endpoint in the orchestrator function. These requests include a user-defined variable *udv_functionKey* that acts as the API Key. During the test, Azure Load Testing fetches this key from the Key Vault. The HTTP request is a POST method aimed at the */api/orc* endpoint of the function app. Each request includes a JSON object from this file, which contains a unique *conversation_id* and a question obtained from a *dataset.csv* file.

This file is important because it's where the questions used by the virtual testers are stored.

Related Technologies

[Azure Load Testing](#)

Azure Load Testing is a fully managed load testing service that enables you to generate high-scale load. The service simulates traffic for your applications, regardless of where they're hosted. Developers, testers, and quality assurance (QA) engineers can use it to optimize application performance, scalability, or capacity.

Quickly [create a load test for your web application by using a URL](#), and without prior knowledge of testing tools. Azure Load Testing abstracts the complexity and infrastructure to run your load test at scale.

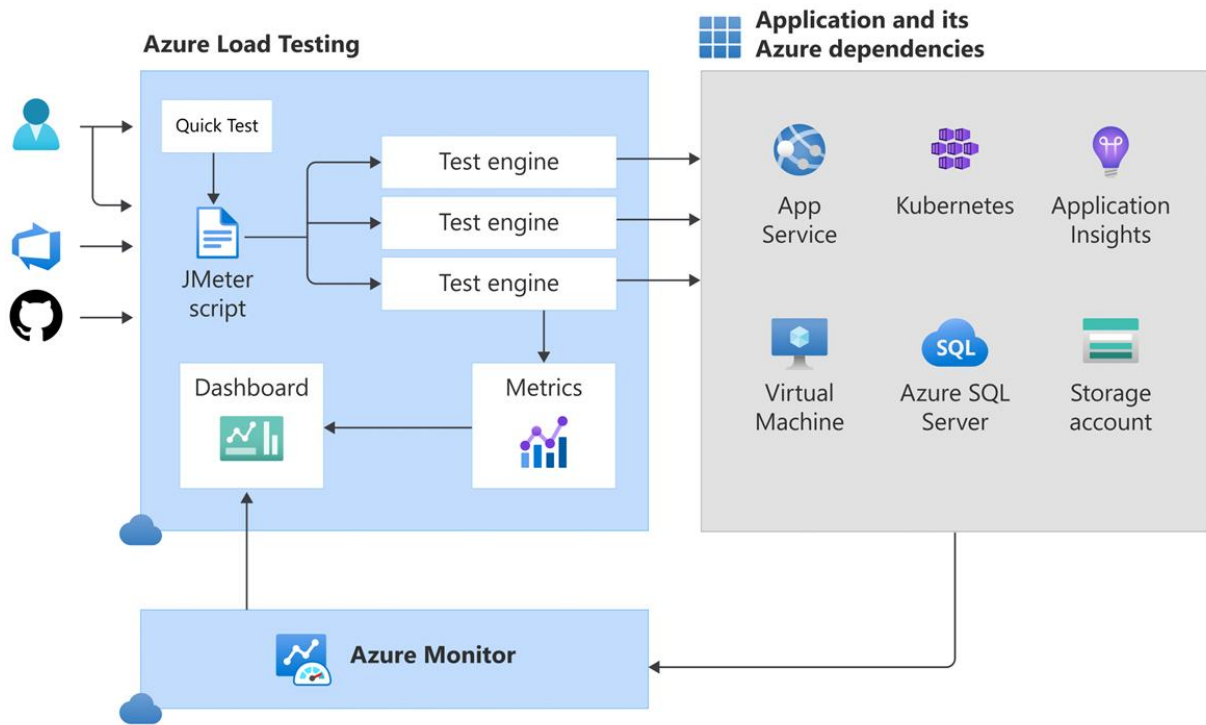
For more advanced load testing scenarios, you can also create a load test by uploading a test script. Azure Load Testing supports running [Apache JMeter-based tests](#) or [Locust-based tests](#). For example, your test plan might consist of multiple application requests, you want to call non-HTTP endpoints, or you're using input data and parameters to make the test more dynamic.

If your application is hosted on Azure, Azure Load Testing collects detailed resource metrics to help you [identify performance bottlenecks](#) across your Azure application components.

To capture application performance regressions early, add your load test in your [continuous integration and continuous deployment \(CI/CD\) workflow](#). Leverage test fail criteria to define and validate your application quality requirements.

Azure Load Testing enables you to test private application endpoints or applications that you host on-premises. For more information, see the [scenarios for deploying Azure Load Testing in a virtual network](#).

The following diagram shows an architecture overview of Azure Load Testing.

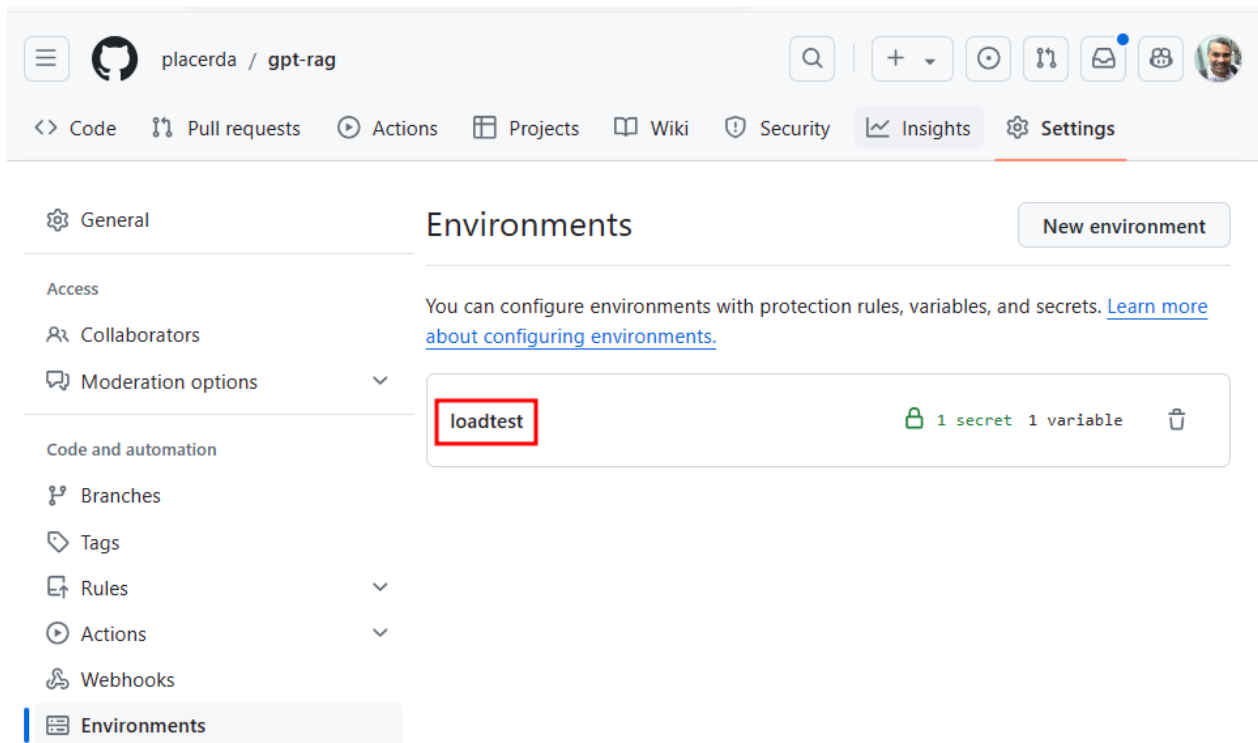


Steps

1. Make sure you have your own copy of the Microhack repository in your GitHub account. You can do this by either forking the original repository or creating a new repository and copying over the existing files. This is a crucial step for setting up the GitHub workflow to execute performance tests in your Azure environment.
2. Copy. azure/ used in previous challenges into the local directory for the newly created repository.
3. Open a terminal window.
4. Navigate to the newly created repository directory 'octo-microhack-rag-ai-and-your-data/' and enter:

```
azd env set AZURE_PROVISION_LOAD_TESTING=true
```

5. Create GitHub Repo Environment: Start by navigating to your new Microhack repository. In the repository's Settings, create a new environment named loadtest. This environment will be used to store the secret and the variable needed for the load testing.



6. Create an Azure Service Principal: The next step involves creating an Azure Service Principal. This is a security identity used by user-created apps, services, and

automation tools to access specific Azure resources. You'll use it to execute the load testing on Azure. Follow steps 1-4 in this [guide](#) to create your Service Principal. When naming your Service Principal, consider using a unique format like <your-initials>-gptrag-load-test-cicd to prevent name conflicts within the same tenant.

- a. Note: In the third step of the guide, you'll be asked to add a Secret. This Secret should not be added as a general GitHub repository secret. Instead, it should be added as an environment secret specifically within the **loadtest** environment you created earlier. This method ensures that the secret is securely isolated within the **loadtest** environment, enhancing the security of your load testing setup.

7. Create an ENV_NAME variable: The last step is to create a variable named ENV_NAME in the GitHub loadtest environment. This variable should hold the same value as the name of the environment in Azure Developer CLI (AZD) that you used when deploying the application. By doing this, you ensure that the load test targets the correct instance of your application, and that there's consistency between the environments used for deployment and load testing.

Environments

Codespaces

Pages

Security

Code security and analysis

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Email notifications

☒ Allow administrators to bypass configured protection rules

Save protection rules

Deployment branches and tags

Limit which branches and tags can deploy to this environment based on rules or naming patterns.

No restriction

Environment secrets

Add environment secret

Secrets are encrypted environment variables. They are accessible only by GitHub Actions in the context of this environment by using the [secrets context](#).

Name	Last updated
AZURE_CREDENTIALS	yesterday

Environment variables

Add environment variable

Variables are used for non-sensitive configuration data. They are accessible only by GitHub Actions in the context of this environment by using the [variables context](#).

Name	Value	Last updated
ENV_NAME	gptrag	yesterday

- Executing the tests is a straightforward process. Simply navigate to the Actions tab in your GitHub repository, select the Load Testing workflow, and finally, click on the "Run workflow" button.

The screenshot shows the GitHub Actions interface for a repository named 'placerta / gpt-rag'. The 'Actions' tab is selected, and the 'Load Testing' workflow is chosen from the left sidebar. The main area displays '3 workflow runs' for the 'Load Testing' workflow. A modal dialog is open, showing the 'Run workflow' button and a dropdown menu for 'Use workflow from' with 'Branch: main' selected. The workflow runs are listed in a table with columns for Event, Status, Branch, and Actor.

Event	Status	Branch	Actor
This workflow has a <code>workflow_dispatch</code> event trigger. Run workflow			
Load Testing	Success	main	placerta
Load Testing	Success	main	placerta
Load Testing	Failure	main	placerta

- The load test will then begin.

The screenshot shows the GitHub Actions interface for a workflow named "Load Testing #20". The workflow is currently "In progress". The left sidebar shows the workflow file "loadtest.yaml" and a job "test_run" which is "Deploying to loadtest". The main area shows the workflow details, including the status "In progress", total duration, and artifacts.

← Load Testing

Load Testing #20 Cancel workflow ...

Summary

Jobs

test_run

Run details

Usage

Workflow file

Manually triggered 1 minute ago

Status: **In progress**

Total duration: —

Artifacts: —

loadtest.yaml

on: workflow_dispatch

test_run 1m 18s

Deploying to loadtest

10. Once the test concludes, you can access the results on the Load Testing resource page in the Azure Portal. Simply select "Enterprise RAG Load Test"

The screenshot shows the Azure Portal interface for the "loadtest0-mbfnsxflh7res" resource. The "Tests" tab is selected, showing a table of test runs. The table has columns for Name, Description, Test type, and Update. The first test run is "Enterprise RAG Load Test" with a test type of "JMX" and an update time of "5/8/20".

Microsoft Azure Search resources, services, and docs (G+/)

Home > loadtest0-mbfnsxflh7res

loadtest0-mbfnsxflh7res | Tests ☆ ...

Azure Load Testing

Search Create Edit Set up CI/CD Refresh Delete test

Time range: Last one month

Name	Description	Test type	Update
<input type="checkbox"/> Enterprise RAG Load Test		JMX	5/8/20

Overview

Activity log

Access control (IAM)

Tags

Tests

Settings

Monitoring

Help

11. Next, select the most recent test run, as shown in the following figure. Note that in this example, the test run passed.

- Note: The test in the above example passed because the p95 response time (response_time_ms) was less than 10000ms. The failure criteria for the test are defined in the *loadtest/config.yaml* file.

Microsoft Azure

Search resources, services, and docs (G+/)

Copilot

paulolacerda@microsoft...
MICROSOFT NON-PRODUCTION

Home > loadtest0-mbfnsxfh7res | Tests > Enterprise RAG Load Test >

TestRun_5/8/2024

Initiated on : 5/8/2024

View all test runs

Stop

Refresh

Rerun

Compare

App components

Configure metrics

Download

Copy artifacts

Test run details

Start time	End time	Engine instances	Test result	Status
5/8/2024	5/8/2024	1	Passed	Done

Load test results

Engine health

Additional insights

Get more detailed insights for the App service resource fnorch0-mbfnsxfh7res by clicking [here](#). Please note it may take up to 45 mins after the test ends for data to be available.

Statistics

Load

10

Total requests

Duration

28 secs

Response time

4.32 secs

90th percentile response time

Error percentage

0 %

Aggregate requests which failed

Throughput

0.36 /s

Request rate

12. In addition to the client metrics available on the test page, you can also add App Components such as the Orchestrator Function App and the Azure OpenAI service to view server metrics on the same page, just click on the "App components" in your test run page.

Page | 45

Home

Test

Initial

⊕

⌵

Load

Add

Sta

U

T

Test

Configure app components

This configuration will only apply to this test run. It will not affect new test runs.

Filter for any field...

Subscription equals **mcaps-paulolacerda (from active filter)**

⊕

Add filter

⌵

More (3)

☐ Show hidden types ⓘ

No grouping ⌵

<input checked="" type="checkbox"/> Name ↑↓	Type ↑↓	Resource group ↑↓	Location ↑↓	Subscription ↑↓
<input type="checkbox"/> appins0-mbfnsxfih7res	Application Insights	rg-gptrag	East US	mcaps-paulolacerda
<input type="checkbox"/> appplan0-mbfnsxfih7res	App Service plan	rg-gptrag	East US	mcaps-paulolacerda
<input type="checkbox"/> cs0-mbfnsxfih7res	Azure AI services multi-s...	rg-gptrag	East US	mcaps-paulolacerda
<input type="checkbox"/> dbgpt0-mbfnsxfih7res	Azure Cosmos DB account	rg-gptrag	East US	mcaps-paulolacerda
<input type="checkbox"/> fninges0-mbfnsxfih7res	Function App	rg-gptrag	East US	mcaps-paulolacerda
<input checked="" type="checkbox"/> fnorch0-mbfnsxfih7res	Function App	rg-gptrag	East US	mcaps-paulolacerda
<input type="checkbox"/> kv0-mbfnsxfih7res	Key vault	rg-gptrag	East US	mcaps-paulolacerda
<input checked="" type="checkbox"/> oai0-mbfnsxfih7res	Azure OpenAI	rg-gptrag	East US	mcaps-paulolacerda
<input type="checkbox"/> strag0mbfnsxfih7res	Storage account	rg-gptrag	East US	mcaps-paulolacerda
<input type="checkbox"/> strag0mbfnsxfih7resing	Storage account	rg-gptrag	East US	mcaps-paulolacerda
<input type="checkbox"/> strag0mbfnsxfih7resorc	Storage account	rg-gptrag	East US	mcaps-paulolacerda
<input type="checkbox"/> webgpt0-mbfnsxfih7res	App Service	rg-gptrag	East US	mcaps-paulolacerda

Apply

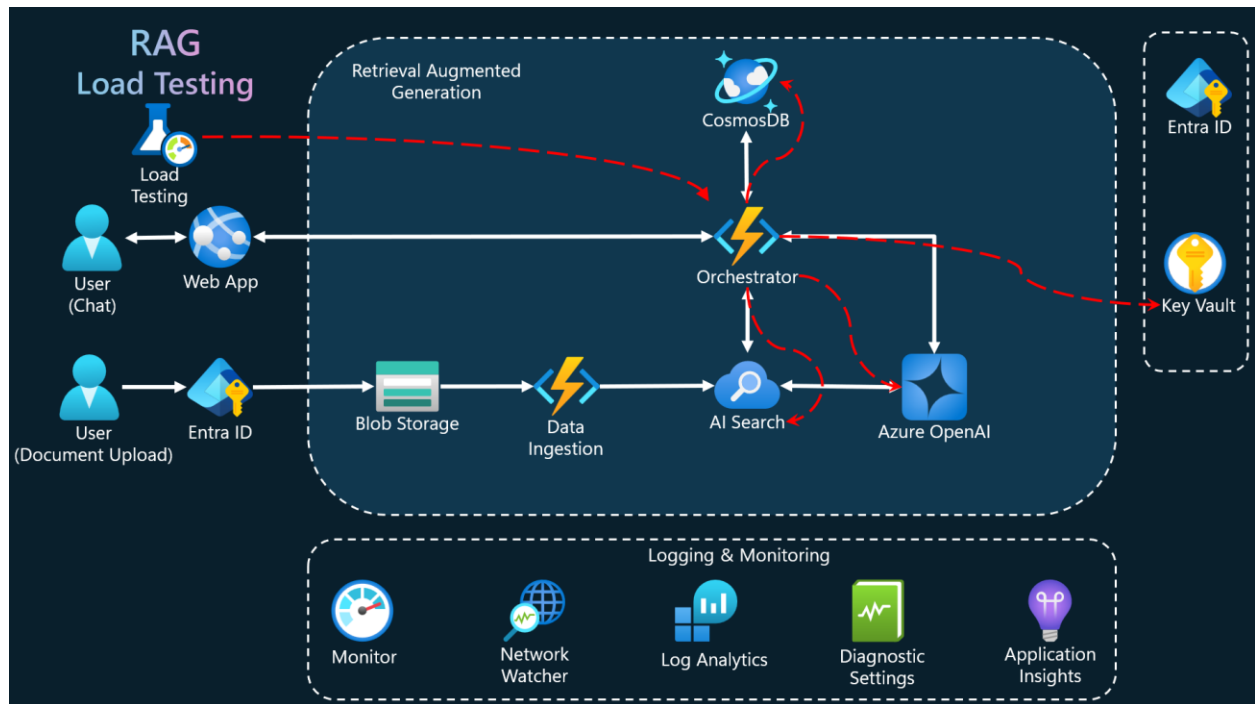
Cancel

13. Done!

Page | 46

Architecture

Load Testing Architecture



Success Criteria

- ☐ Successfully completed all Challenge 6 Steps without error.
- ☐ Successfully updated deployment to match Load Testing Architecture.
- ☐ Understanding of architecture and its components.
- ☐ Running load testing workflow and analysis of results.

Useful Resources

- [Load Testing RAG based Generative AI Applications | Microsoft Community Hub](#)
- [Azure Load Testing documentation | Microsoft Learn](#)
- [Performance efficiency quick links - Microsoft Azure Well-Architected Framework | Microsoft Learn](#)

Challenge 7: Security in AI

Overview

Company Contoso is rapidly expanding its service offerings and client base, driving the need to enhance the security of its Chatoso environment. In the face of increasing cyber threats and compliance requirements, the company must prioritize robust network security and the implementation of private endpoints to safeguard sensitive data.

The business requirements for this upgrade include the establishment of secure, encrypted communication channels across all network layers, and the deployment of private endpoints to ensure that internal resources are accessible only within the corporate network. By focusing on these key areas, Contoso aims to protect against unauthorized access, data breaches, and ensure regulatory compliance, thereby maintaining the trust of their clients and stakeholders.

Related Technologies

[Azure Private Link](#)

- Azure Private Link enables you to access Azure PaaS Services (for example, Azure Storage and SQL Database) and Azure hosted customer-owned/partner services over a [private endpoint](#) in your virtual network.
- Traffic between your virtual network and the service travels the Microsoft backbone network. Exposing your service to the public internet is no longer necessary. You can create your own [private link service](#) in your virtual network and deliver it to your customers. Setup and consumption using Azure Private Link is consistent across Azure PaaS, customer-owned, and shared partner services.
- Privately access services on the Azure platform: Connect your virtual network using private endpoints to all services that can be used as application components in Azure. Service providers can render their services in their own virtual network and consumers can access those services in their local virtual network. The Private Link platform will handle the connectivity between the consumer and services over the Azure backbone network.
- On-premises and peered networks: Access services running in Azure from on-premises over ExpressRoute private peering, VPN tunnels, and peered virtual networks using private endpoints. There's no need to configure ExpressRoute

Microsoft peering or traverse the internet to reach the service. Private Link provides a secure way to migrate workloads to Azure.

- Protection against data leakage: A private endpoint is mapped to an instance of a PaaS resource instead of the entire service. Consumers can only connect to the specific resource. Access to any other resource in the service is blocked. This mechanism provides protection against data leakage risks.
- Global reach: Connect privately to services running in other regions. The consumer's virtual network could be in region A and it can connect to services behind Private Link in region B.
- Extend to your own services: Enable the same experience and functionality to render your service privately to consumers in Azure. By placing your service behind a standard Azure Load Balancer, you can enable it for Private Link. The consumer can then connect directly to your service using a private endpoint in their own virtual network. You can manage the connection requests using an approval call flow. Azure Private Link works for consumers and services belonging to different Microsoft Entra tenants.

[Azure Virtual Network](#)

- Azure Virtual Network is a service that provides the fundamental building block for your private network in Azure. An instance of the service (a virtual network) enables many types of Azure resources to securely communicate with each other, the internet, and on-premises networks. These Azure resources include virtual machines (VMs).
- A virtual network is similar to a traditional network that you'd operate in your own datacenter. But it brings extra benefits of the Azure infrastructure, such as scale, availability, and isolation.

Steps

1. Navigate to code repository.
2. Open `'octo-microhack-rag-ai-and-your-data/infra/main.bicep'`
3. Update the following:

```
param searchServiceName string = ''  
to  
param searchServiceName string = '<your-search-service-name>'
```

4. Remove `'//'` to uncomment lines #976 – 983.

```
module orchestratorSearchAccess './core/security/search-access.bicep'
```

5. Save file `'octo-microhack-rag-ai-and-your-data/infra/main.bicep'`.
6. Open a terminal window.
7. Navigate to the downloaded directory `'octo-microhack-rag-ai-and-your-data/'` and enter:

```
azd env set AZURE_NETWORK_ISOLATION true
```

8. Next, in the terminal, enter:

```
azd provision --preview
```

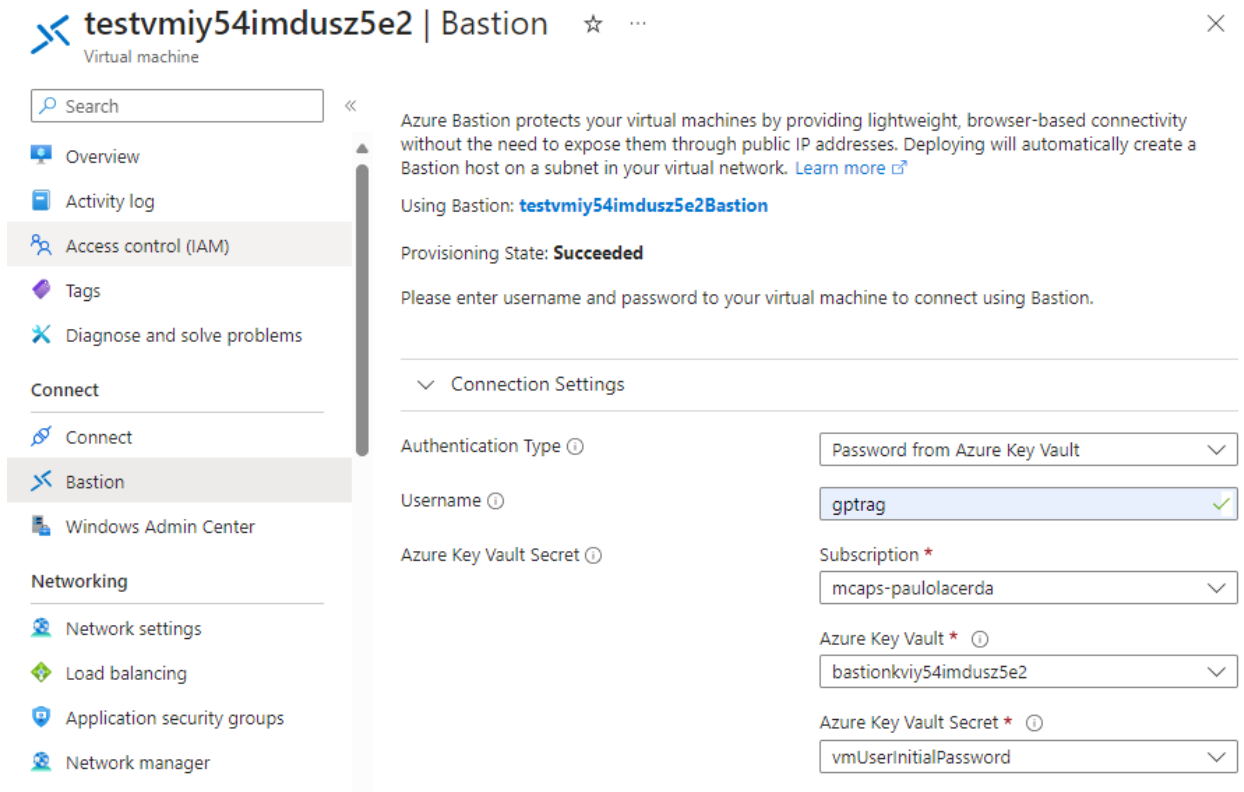
9. Expect to observe an output similar to the following:

Resources:

Skip	: Resource group	: rg-microhack-rag-10
Modify	: Azure AI Services	: ai0-cw37x3hzukpk4
Modify	: Azure AI Services	: oai0-cw37x3hzukpk4
Modify	: Azure AI Services Model Deployment	: chat
Modify	: Azure AI Services Model Deployment	: text-embedding-ada-002
Modify	: Azure Cosmos DB	: dbgpt0-cw37x3hzukpk4
Modify	: Application Insights	: appins0-cw37x3hzukpk4
Modify	: Key Vault	: kv0-cw37x3hzukpk4
Create	: Virtual Network	: aivnet0-cw37x3hzukpk4
Skip	: Search service	: search0-cw37x3hzukpk4
Modify	: Storage account	: strag0cw37x3hzukpk4
Skip	: Storage account	: strag0cw37x3hzukpk4ing
Skip	: Storage account	: strag0cw37x3hzukpk4orc
Skip	: App Service plan	: appplan0-cw37x3hzukpk4
Skip	: Web App	: fninges0-cw37x3hzukpk4
Skip	: Web App	: fnorch0-cw37x3hzukpk4
Skip	: Web App	: webgpt0-cw37x3hzukpk4

SUCCESS: Generated provisioning preview in 22 seconds.

10. Make required updates to Azure environment to accomplish the architecture listed below.
 - a. Hint: Resources labeled 'Create' and 'Modify' will need to be created and/or updated.
 - b. Hint: If stuck, please refer to *docs/MANUAL_INSTALLATION.md*
11. Next, you will use the Virtual Machine with the Bastion connection to continue the deployment.
12. Log into the created VM with the user **gptrag** and authenticate with the password stored in the keyvault, similar to the figure below:



13. Upon accessing Windows, install [Powershell](#), as the other prerequisites are already installed on the VM.

14. Open the command prompt and run the following command to update azd to the latest version.

```
choco upgrade azd
```

15. After updating azd, simply close and reopen the terminal.

16. Create a new directory, for example: `deploy`, then enter the created directory.

```
mkdir deploy
cd deploy
```

17. To finalize the procedure, execute the subsequent commands in the command prompt to successfully complete the deployment:

- Note: when running the `azd init ...` and `azd env refresh`, use the same environment name, subscription, and region used in the initial provisioning of the infrastructure.

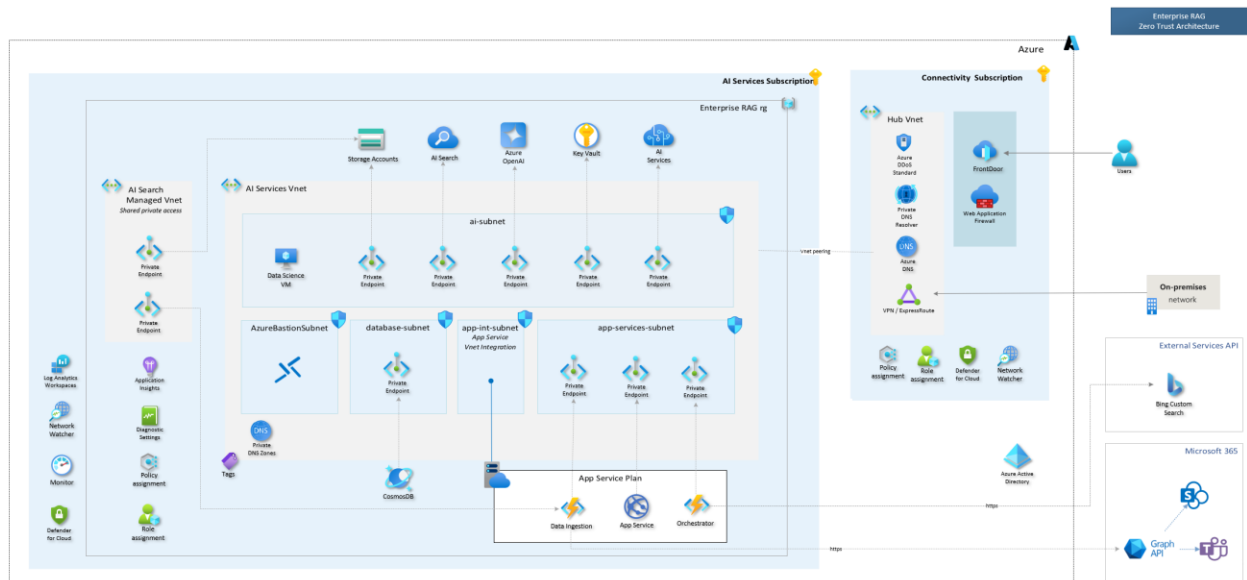
```
azd init -t azure/gpt-rag
azd auth login
```

```
azd env refresh
azd package
azd deploy
```

18. Done!

Architecture

Zero-Trust Architecture



Success Criteria

- ☐ Successfully completed all Challenge 7 Steps without error
- ☐ Successfully updated deployment to match Zero-Trust Architecture
- ☐ Understanding of architecture and its components
- ☐ Running `azd provision --preview` results in *Skip for all* resource outputs

Useful Resources

- [Zero Trust Guidance Center | Microsoft Learn](#)
- [Azure Virtual Network Documentation | Microsoft Learn](#)
- [Private Link Documentation | Microsoft Learn](#)
- [Using your data with Azure OpenAI securely - Azure OpenAI | Microsoft Learn](#)
- [Create a private endpoint for a secure connection - Azure AI Search | Microsoft Learn](#)
- [Use private endpoints to integrate Azure Functions with a virtual network | Microsoft Learn](#)
- [Configure Azure Private Link for Azure Cosmos DB | Microsoft Learn](#)
- [Connect privately to an App Service apps using private endpoint - Azure App Service | Microsoft Learn](#)
- [Integrate Key Vault with Azure Private Link | Microsoft Learn](#)
- [Use private endpoints - Azure Storage | Microsoft Learn](#)
- [Connect to a VM using Bastion - Windows native client - Azure Bastion | Microsoft Learn](#)

Congratulations!

Cleaning Up Environment

1. Navigate to the downloaded directory *octo-microhack-rag-ai-and-your-data/* and enter:

```
azd down
```

2. Confirm successful resource group deletion via [Azure Portal](#).

Ideas for Enhancements

- **API Management Integration**
 - It is becoming increasingly important to leverage an application gateway solution like Azure API Management for serving LLMs for other applications. To discover how to add Azure API Management to this deployment, follow the guide located at *docs/API_MANAGEMENT_INTEGRATION.md*.
- **Agentic Deployment**
 - The default deployment utilizes [Semantic Kernel](#) to orchestrate the model functionality. The Agentic deployment utilizes [AutoGen](#) for an agentic pattern.

```
azd init -t azure/gpt-rag -b agentic
```

- **Customize Your Deployment**
 - The standard deployment process sets up Azure resources and deploys the accelerator components with a standard configuration. To tailor the deployment to your specific needs, follow the steps in the *docs/CUSTOMIZATIONS.md* section for further customization options.
- **Integrate with Additional Data Sources**
 - Expand your data retrieval capabilities by integrating new data sources such as Bing Custom Search, SQL Server, and Teradata. For detailed instructions, refer to the *docs/AI_INTEGRATION_HUB.md* page.
- **Multi-Environment Deployment**
 - Once you have successfully deployed the GPT-RAG solution as a proof of concept and you're ready to formalize the deployment using a proper CI/CD process to accelerate your deployment to production, refer to the multi-environment deployment guides for either *docs/AZDO-setup.md* or *docs/GH-SETUP.md*.

