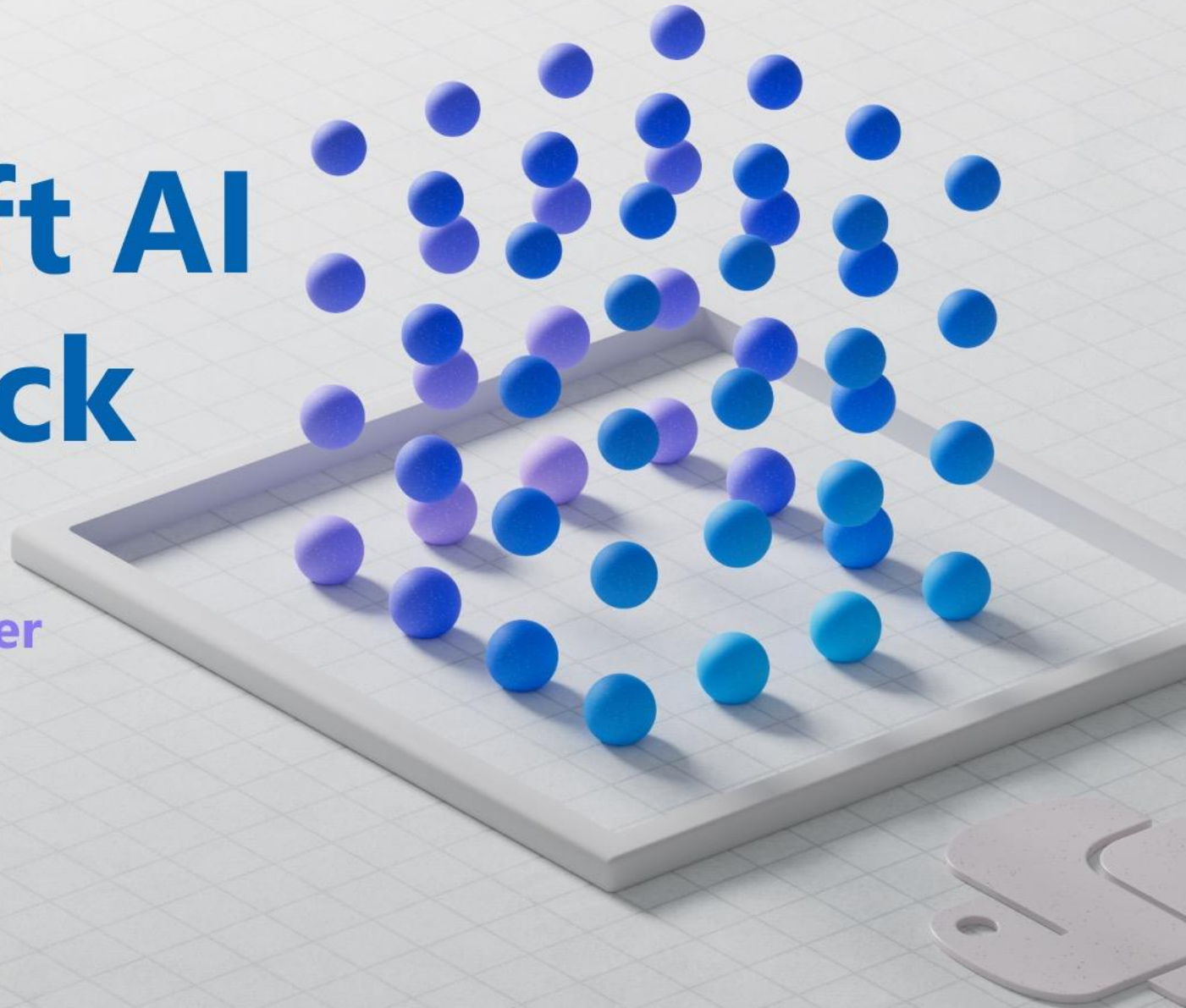


January 29th - February 12th

The Microsoft AI Chat App Hack

Build, innovate, and **#HackTogether**
aka.ms/hacktogether/chatapp



The AI Chat App Hack

January 29th - February 12th

29th



Building a RAG Chat App in Python

30th



Customizing your RAG Chat App

31st



Azure AI Search Best Practices

1st



GPT-4 with Vision

2nd

HACK

HACK

3rd

HACK

HACK

4th

HACK

HACK

HACK

5th



AM: RAG Chat Web Components

PM: Access Control in RAG Chat Apps

6th



Evaluating a RAG Chat App

7th



RAG Chat Special Topic

8th



Continuous Deployment of your Chat App

9th

HACK

10th

HACK

11th

HACK

HACK

12th

SUBMIT YOUR PROJECT

Build, innovate, and [#HackTogether](#)

RAG Chat App in JavaScript and Web Components

Prerequisites: RAG Chat App

This talk will be about the JS variant of the RAG Chat App:

<https://github.com/Azure-Samples/azure-search-openai-javascript>

We'll also use this app you previously deployed:

<https://github.com/Azure-Samples/azure-search-openai-demo>

If you've not deployed it yet, you must deploy all the resources.

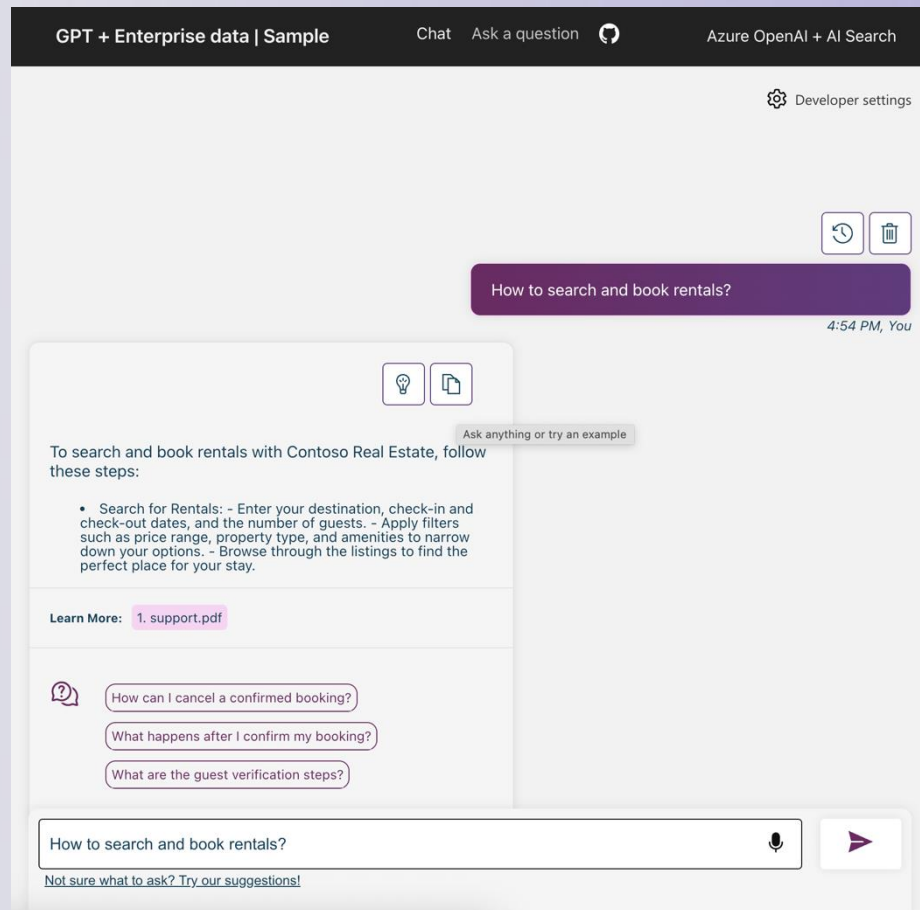
See this doc for tips on deploying for free:

 aka.ms/ragchat-free

(Keep watching either way, you can always rewatch after you deploy!)

RAG chat app in JavaScript

Open-source RAG chat app solution



Differences with the initial solution:

- All services implemented in TypeScript
- Reusable Web Components built with Lit
- Hosting with Azure Container Apps
- Dedicated data ingestion service
- Support for PDFs and MD documents
- LangChain.js integration
- Different source data set

Compatible chat backend:

- /chat API is compatible between solutions
- Backend can be swapped

<https://github.com/Azure-Samples/azure-search-openai-javascript/>

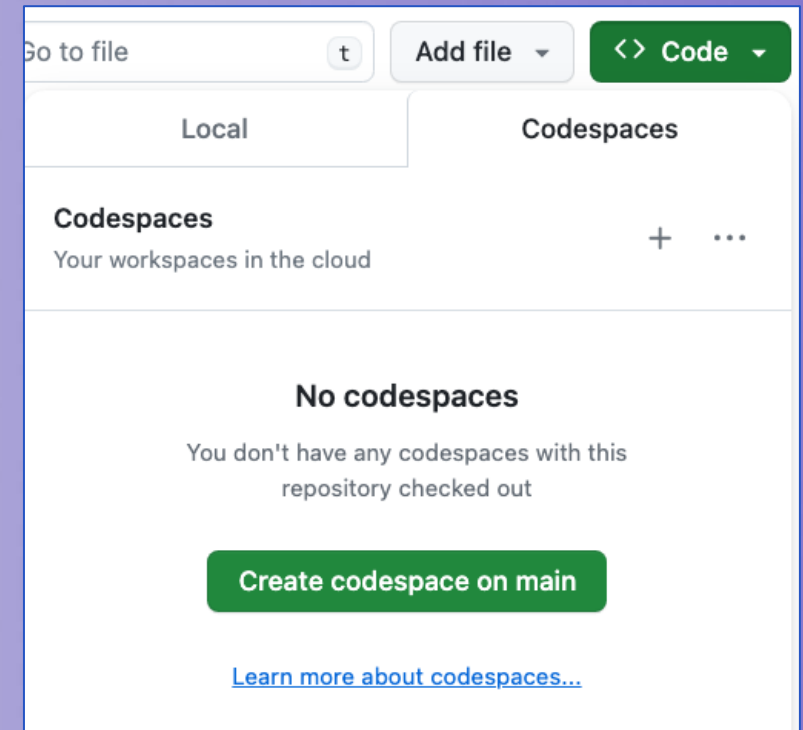
Prerequisites

- Azure account and subscription
 - A free account can be used, but will have limitations.
- Access to Azure OpenAI or an openai.com account
 - Request access to Azure OpenAI today!
<https://aka.ms/oaiapply>

<https://github.com/Azure-Samples/azure-search-openai-javascript/#azure-account-prerequisites>

Opening the project: 3 options

- GitHub Codespaces →
- VS Code with Dev Containers extension
- Your Local Environment
 - Node 18+
 - Azure Developer CLI



<https://github.com/Azure-Samples/azure-search-openai-javascript/#project-setup>

Deploying with the Azure Developer CLI

Login to your Azure account:

```
azd auth login
```

Create a new azd environment: (to track deployment parameters)

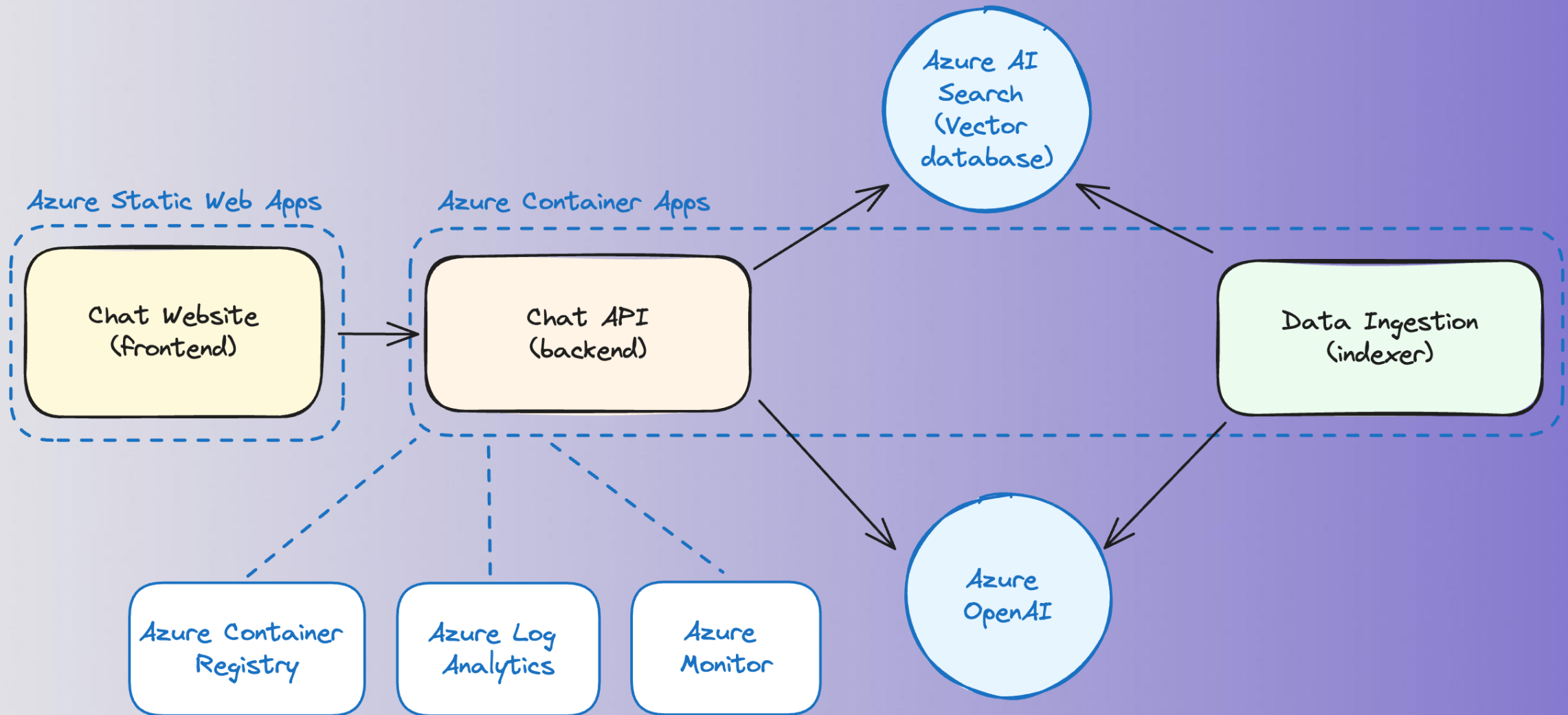
```
azd env new
```

Provision resources and deploy app:

```
azd up
```

azd up is a combination of azd provision and azd deploy

Application architecture on Azure



Code walkthrough

TypeScript frontend

(React + Lit Web Components)

```
chat.tsx  
<chat-component>
```

```
chat-component.ts  
getApiResponse()
```

```
http/index.ts  
callHttpApi()
```

TypeScript backend

(Fastify)

```
root.ts  
fastify.post('/chat')
```

```
chat-read-retrieve-read.ts  
run()
```

```
// get search query  
// compute text embedding  
// search  
// get messages from history  
chat.completions.create()
```

Web Components



Why Web Components?

- Reusability
- Integration with any existing app
Supported in React, Angular, Vue, Svelte...
- Allow branding customization: logo, colors

Code walkthrough

TypeScript frontend

(React + Lit Web Components)

```
chat.tsx  
<chat-component>
```



```
chat-component.ts  
getApiResponse()
```



```
http/index.ts  
callHttpApi()
```

TypeScript backend

(Fastify)

```
root.ts  
fastify.post('/chat')
```




```
chat-read-retrieve-read.ts  
run()
```



```
// get search query  
// compute text embedding  
// search  
// get messages from history  
chat.completions.create()
```

JavaScript app with Python backend

The background features a light gray grid pattern. A cluster of semi-transparent blue and purple spheres is arranged in a roughly circular pattern on the right side. A white, three-dimensional rectangular block is positioned in the lower right, partially overlapping the grid and the sphere cluster. The overall aesthetic is clean and modern.

Switch the backends

Prerequisite: deploy the Python app and the JavaScript app.

Set the Python backend to allow the JS frontend calls (CORS):

```
azd env set ALLOWED_ORIGIN <your_frontend_url> # and redeploy Python app
```

Set the JS frontend to use the Python backend:

```
azd env set BACKEND_URI <your_backend_url> # and redeploy JS app
```

Next steps

- Register for the hackathon → aka.ms/hacktogether/chatapp
- Introduce yourself in our discussion forum
- Deploy the repo with the sample data
 - See steps on low cost deployment → aka.ms/ragchat-free
- Post in forum if you have any questions or issues deploying.
- Join this afternoon session: Access Control in RAG Chat Apps!

Thank You

