Step-by-Step Guide – Azure AI Foundry → PHP Chatbot → Render

SBM Project • Technical Tutorial

This document provides a complete walkthrough on how to create an AI Assistant in Azure AI Foundry, connect it to a PHP-based chatbot, and finally deploy it online using Render.

Important Warning – Assistant vs. Agent

There is a key difference between an Agent and an Assistant in Azure AI Foundry.

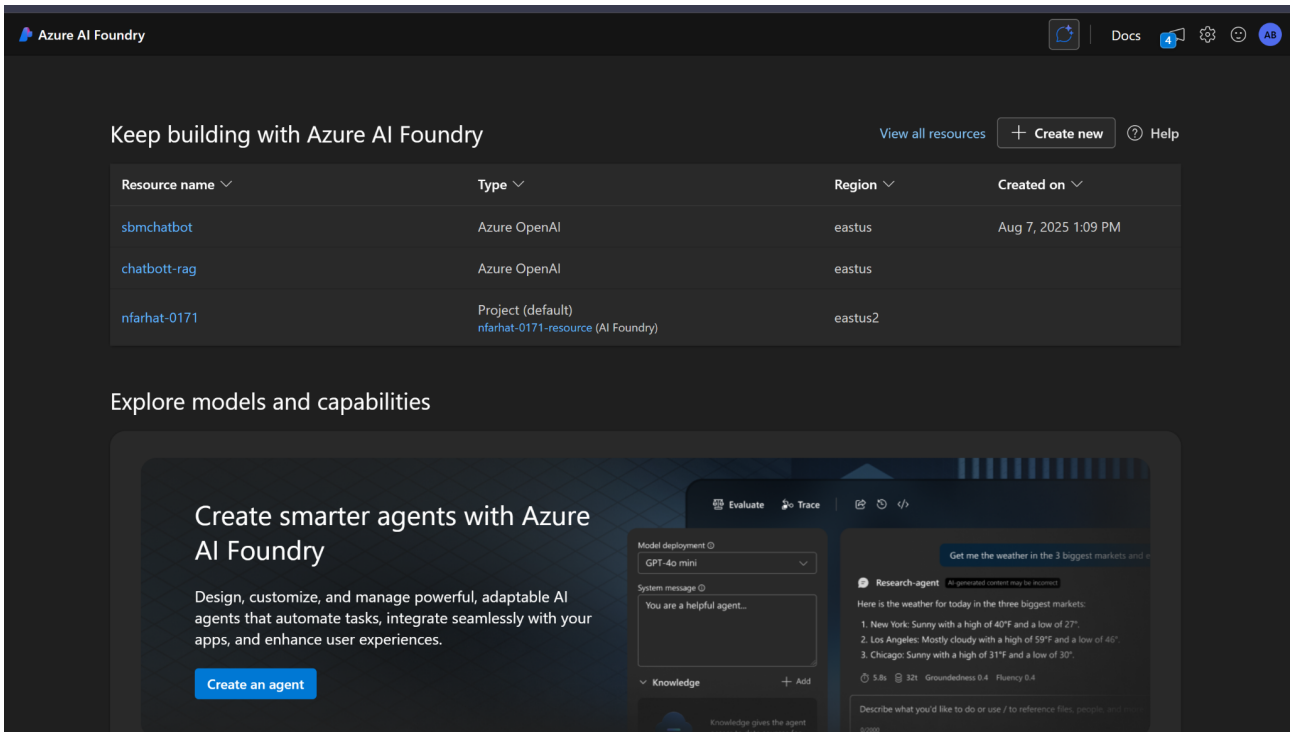
An Agent is a higher-level concept used mainly inside Azure AI Foundry for testing and internal workflows. It cannot be directly integrated or deployed to an external website.

An Assistant, on the other hand, provides the necessary API access (Assistant ID, Deployment, Endpoint, API Key) that allows you to connect it to your own application (for example, a PHP chatbot) and host it online (via Render or any other platform).

Content: 1) Azure (creation & setup), 2) GitHub (PHP chatbot code), 3) Render (deployment to the cloud).

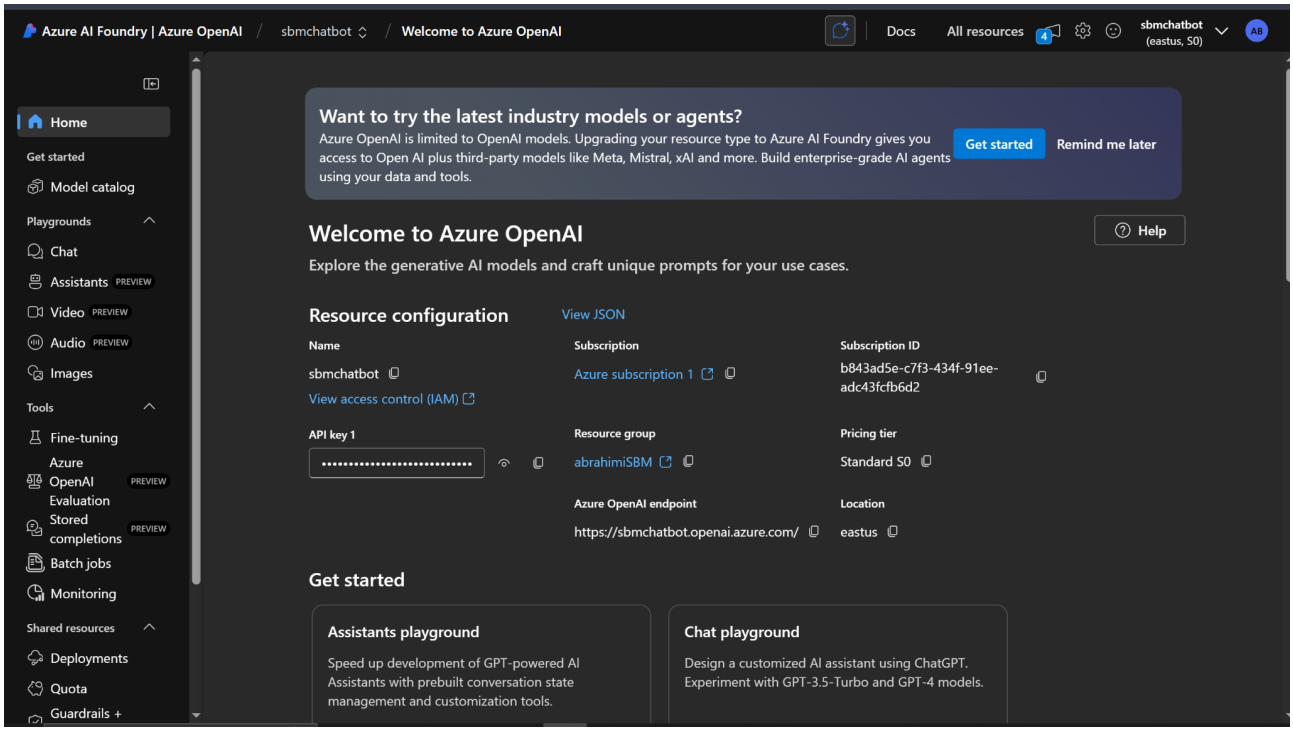
1. Azure AI Foundry – Creating & Configuring the Assistant

Azure AI Foundry is the platform where we create and manage our AI Assistant. This is the 'brain' of the chatbot: it hosts the AI model, the knowledge files, and provides the secure API endpoint. Think of it as the server that does all the heavy lifting, while our PHP code is just the interface.



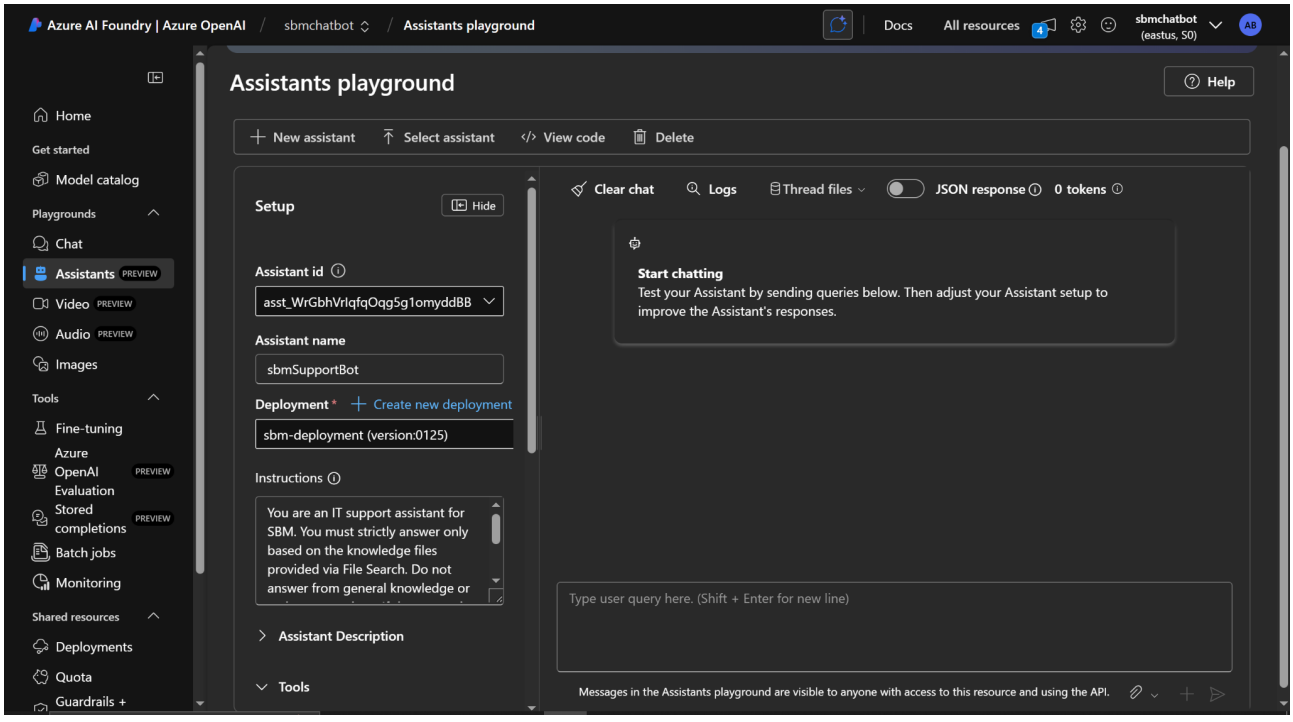
Azure AI Foundry – List of resources.

Here you see the resources available. Our main resource is named sbmchatbot. This resource gives us access to Azure OpenAI services in the East US region.



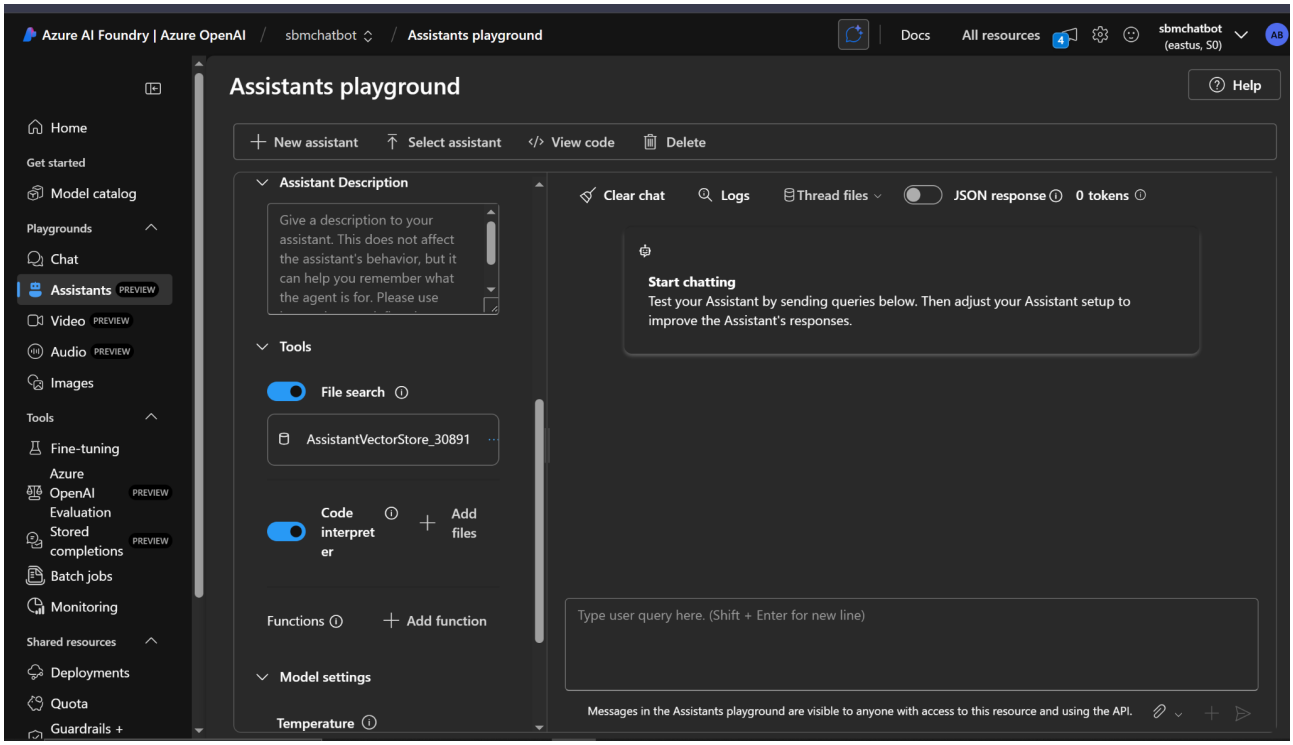
Azure OpenAI – Resource configuration page.

Important fields to note: the API key (acts like a password for apps), the endpoint (URL to send requests), and the subscription details. We will later use the API key and endpoint in our PHP code and Render deployment.



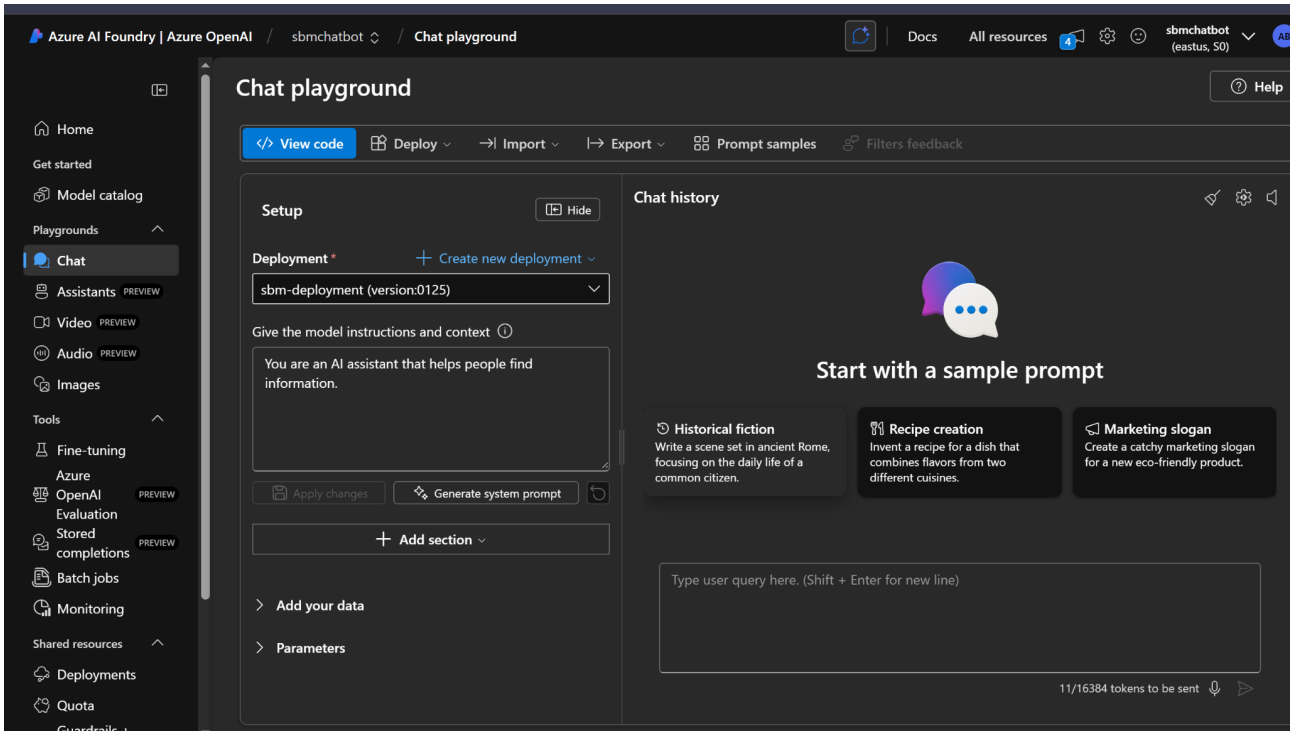
Assistant playground – Setup of our Assistant.

This screen shows the Assistant ID (a unique identifier for our bot), the Assistant name, and the Deployment. The Deployment links to a specific AI model version, so you can control exactly which model powers your chatbot.



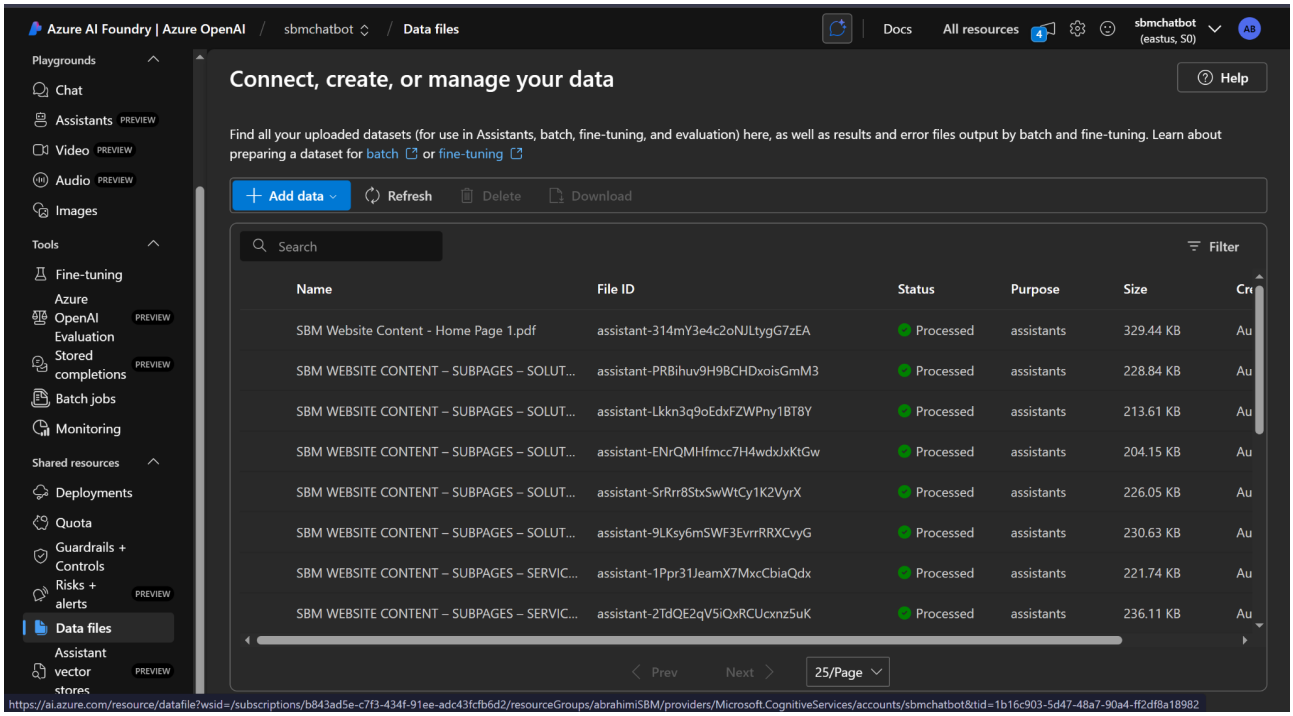
Assistant playground – Tools configuration.

Here we enabled File Search, which connects our Assistant to specific documents (like SBM manuals). This means the chatbot will only answer using company-approved documents, avoiding generic responses.



Chat playground – Quick model test.

This is a sandbox to try out the AI model before final integration. It helps you confirm the Deployment is working correctly.

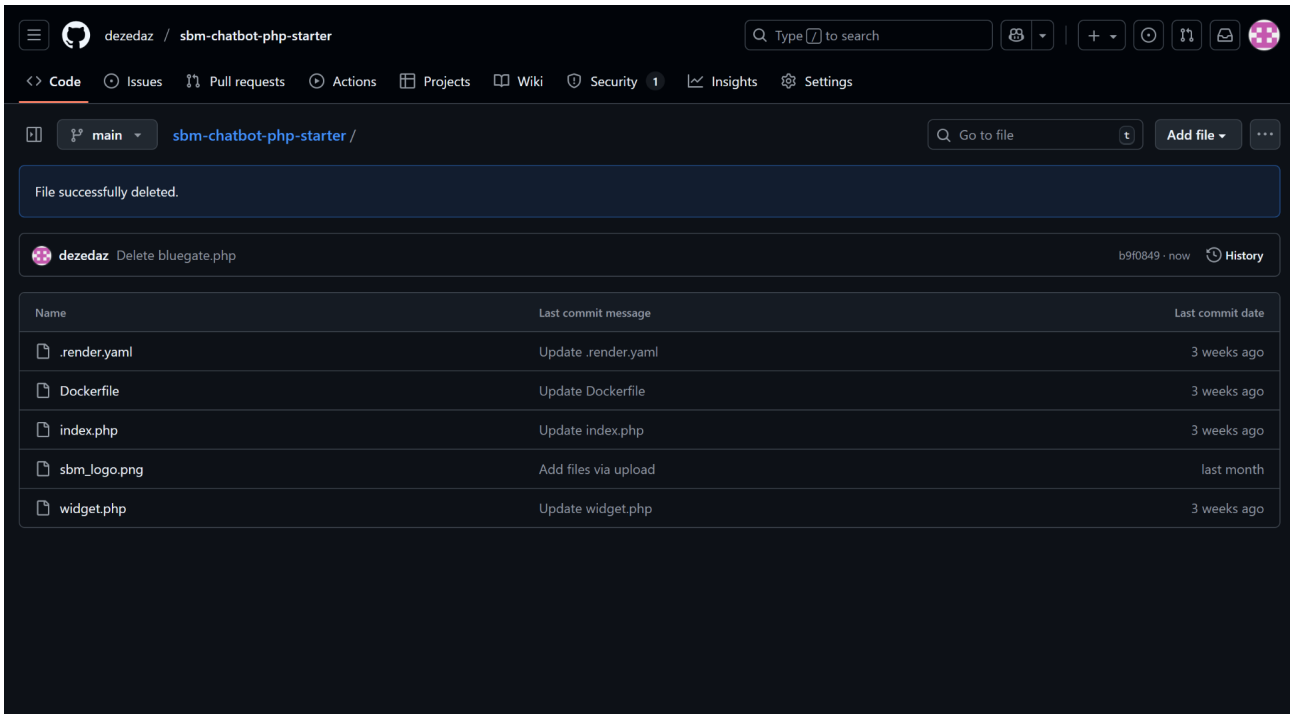


Data files – Uploaded SBM documents.

Each document has been processed and indexed. Notice the status 'Processed'. This indexing is what makes File Search possible: the AI can now look up answers in these files.

2. GitHub – Hosting the PHP Chatbot Code

GitHub is where we keep the chatbot's source code. Think of it as a shared folder with version control: you can update, track changes, and connect it directly to Render for automatic deployment.



GitHub repository – Project structure.

Files included: index.php (main chatbot UI), widget.php (extra widget option), sbm_logo.png (branding), Dockerfile (instructions to build an image with PHP), and .render.yaml (Render configuration).

Each of these files plays a role:

- index.php: Displays the chatbot interface,handles sessions (chat history), reset, dark mode, etc.

- widget.php: Optional component to embed the chatbot in another site.

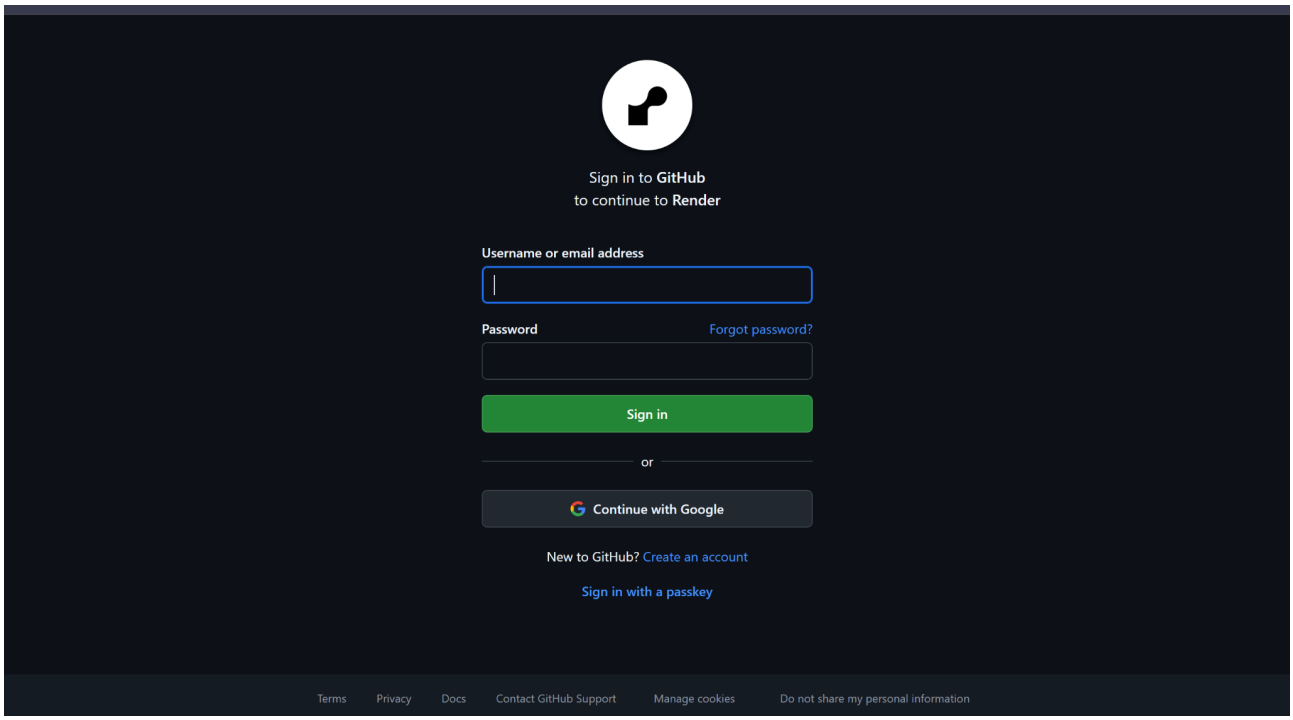
- sbm_logo.png: The company logo.

- Dockerfile: Ensures Render knows how to build a container with PHP +cURL (needed for API calls).

- .render.yaml: Defines how Render should deploy this app (service type, port, environment variables).

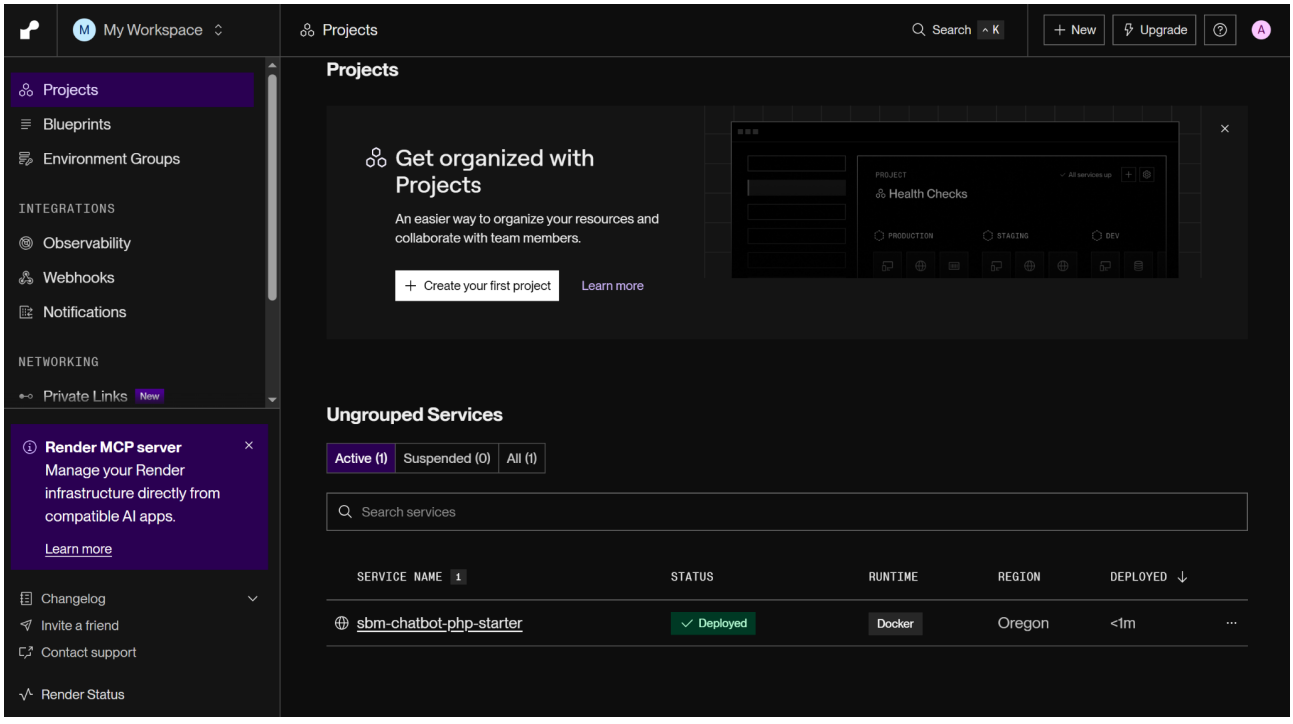
3. Render – Deploying the Chatbot Online

Render is the hosting service. It builds the app from GitHub, runs it inside a Docker container, and exposes it to the web with a public URL. This is the step that makes your chatbot accessible to anyone online.



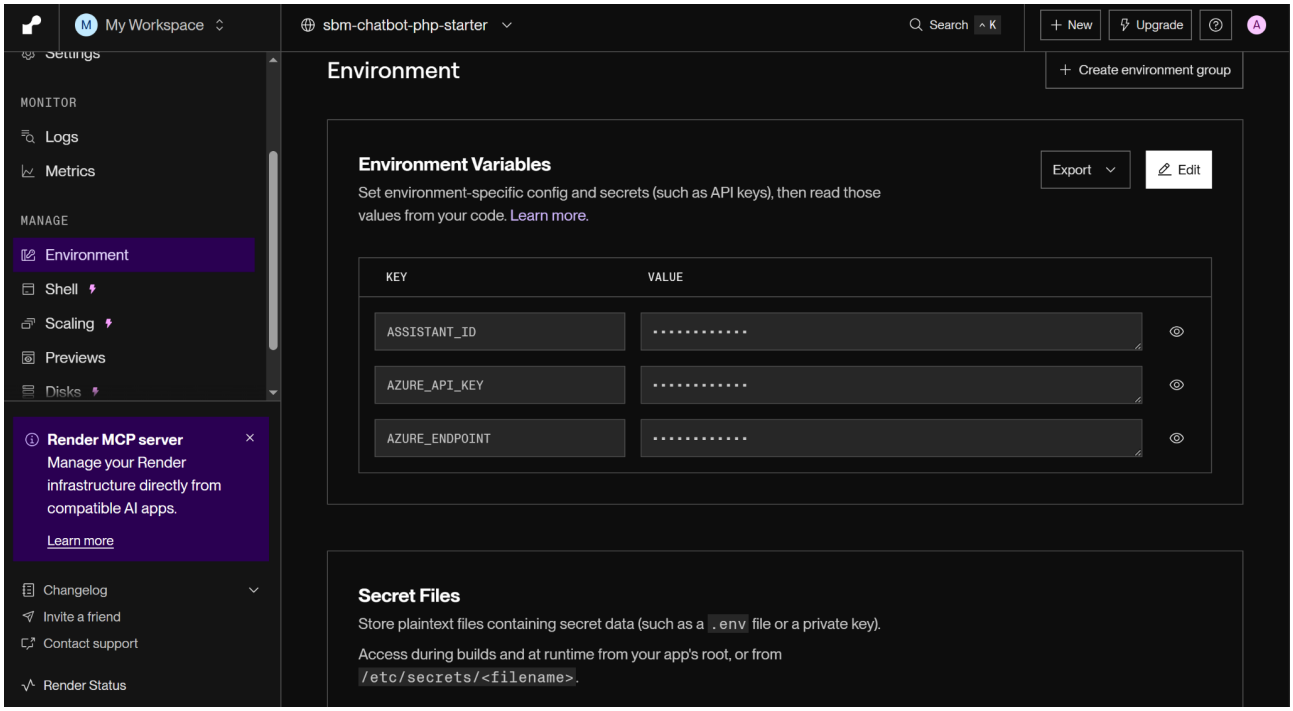
Render login – Sign in with GitHub.

Render can connect directly to your GitHub account. This way, each time you push changes to GitHub, you can redeploy easily.



Render dashboard – Active service.

Here you see the deployed service 'sbm-chatbot-php-starter'. Status is green (Deployed). The region and runtime (Docker) are also displayed.



Render – Environment variables.

Notice the keys: ASSISTANT_ID, AZURE_API_KEY, and AZURE_ENDPOINT. These values replace sensitive information in code. By using environment variables, we keep the code clean and secure.

Once configured, Render builds the Docker image, launches the service, and provides you with a URL. Testing is simple: open the URL and start chatting. The assistant should answer based only on the indexed SBM documents.

Troubleshooting & Common Issues

- Authentication errors (401/403): Check your API key and endpoint. - No response or timeout: Check logs in Render, confirm the Assistant ID. - Irrelevant answers: Ensure File Search is enabled and documents are linked. - Build failures: Check the Dockerfile for PHP + cURL installation and versions.

The next step: You will receive the PHP files with detailed inline comments. Those comments explain the code line by line, making it easier for you to understand and extend the project.