Microsoft

Microsoft Azure Virtual Training Day: Develop Your Own Custom Copilots with Azure AI

Introduction to the Azure AI Studio

Microsoft

Learning Objectives

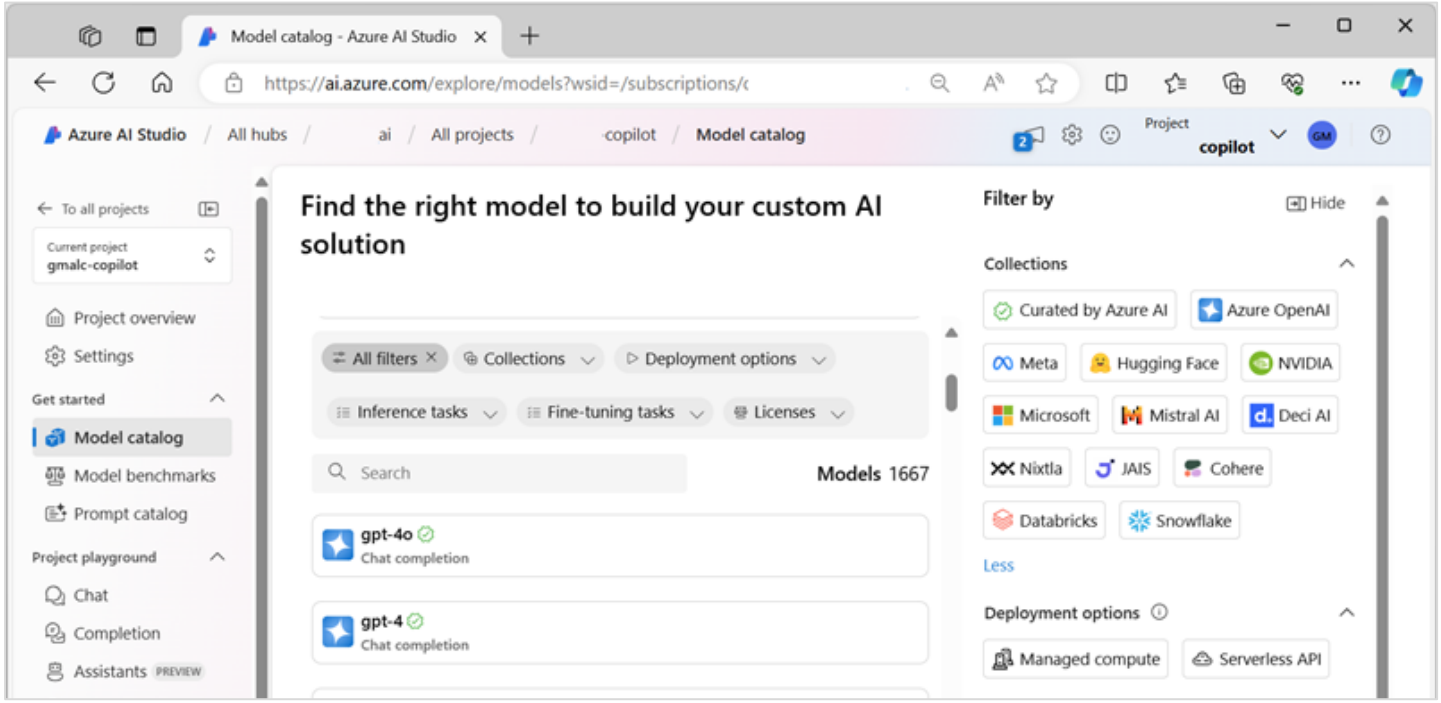
- Get familiar with AI Studio

Microsoft

Learning Objective: Get familiar with AI Studio

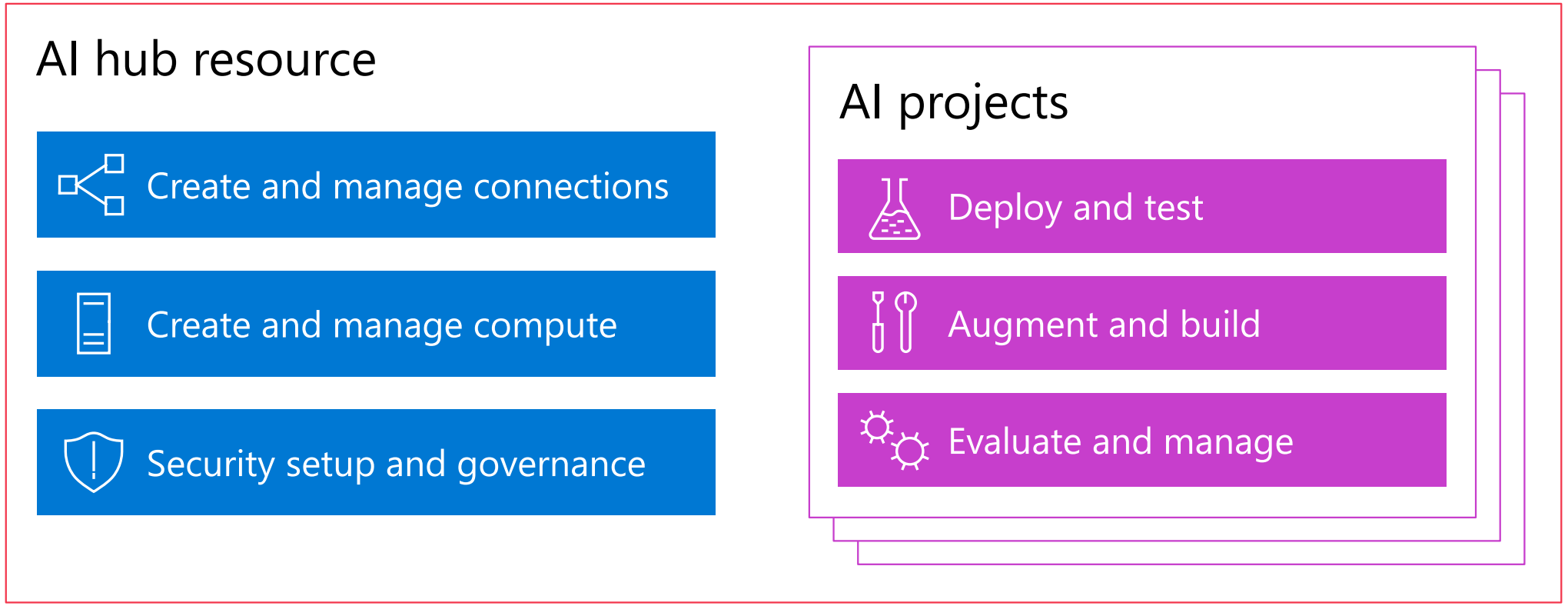
What is the Azure AI Studio?

- Pro-code development with full catalog of models and fine-tuning capabilities.
- PaaS services with full control over cloud infrastructure.
- Prompt and model orchestration.
- Evaluations engine to test performance, reliability, scalability, and responsible AI safety.
- Deploy as an endpoint in Azure for use in custom apps and services.



Configure your environment with hubs and projects

Azure AI Studio is the platform for developing generative AI solutions and custom copilots.



Create connections to external resources

Connections in Azure AI Studio are a way to authenticate and consume both Microsoft and non-Microsoft resources within your AI Studio projects.

- Connect to Azure AI Services like Azure OpenAI and Azure AI Search.
- Connect to non-Microsoft services (API key or custom connection).
- Connect to datastores to access external data.

Secrets associated with connections are securely persisted in the corresponding Azure Key Vault, adhering to robust security and compliance standards.

Demo

- Explore Azure AI Studio
- Create Azure AI Hub & Project
- Create Connections to External Resources

Plan a responsible generative AI solution

The Microsoft guidance for responsible generative AI is designed to be practical and actionable. It defines a four stage process to develop and implement a plan for responsible AI when using generative models. The four stages in the process are:

Identify potential harms that are relevant to your planned solution.

Measure the presence of these harms in the outputs generated by your solution.

Mitigate the harms at multiple layers in your solution to minimize their presence and impact, and ensure transparent communication about potential risks to users.

Operate the solution responsibly by defining and following a deployment and operational readiness plan.

Operate a responsible generative AI solution

- Complete prerelease reviews
- Release and operate the solution
- Utilize **Azure AI Content Safety**:

Feature	Functionality
Prompt shields	Scans for the risk of user input attacks on language models
Groundedness detection	Detects if text responses are grounded in a user's source content
Protected material detection	Scans for known copyrighted content
Custom categories	Define custom categories for any new or emerging patterns

Demo

- Responsible AI Toolbox

Manage access for collaboration

Use the Azure built-in roles

With **role-based access control** (**RBAC**) you can assign roles to users to give them access on the AI hub or project level.

- Use the Azure built-in roles, available by default:

Role	Hub	Project
Owner	Full access to the hub, including the ability to manage and create new hubs and assign permissions. This role is automatically assigned to the hub creator.	Full access to the project, including the ability to assign permissions to project users.
Contributor	User has full access to the hub, including the ability to create new hubs, but isn't able to manage hub permissions on the existing resource.	User has full access to the project but can't assign permissions to project users.
Reader	Read only access to the hub. This role is automatically assigned to all project members within the hub.	Read only access to the project.

Manage access for collaboration

Use Azure AI specific roles

With **role-based access control** (**RBAC**) you can assign roles to users to give them access on the AI hub or project level.

- Use the Azure built-in roles, available by default :

Role	Hub	Project
Azure AI Developer	Perform all actions except create new hubs and manage the hub permissions. For example, users can create projects, compute, and connections. Users can assign permissions within their project. Users can interact with existing Azure AI resources such as Azure OpenAI, Azure AI Search, and Azure AI services.	User can perform most actions, including create deployments, but can't assign permissions to project users.
Azure AI Inference Deployment Operator	Perform all actions required to create a resource deployment within a resource group.	Perform all actions required to create a resource deployment within a resource group.

Manage access for collaboration

Create a custom role

With **role-based access control** (**RBAC**) you can assign roles to users to give them access on the AI hub or project level.

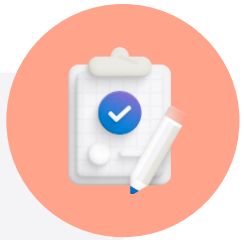
- Use the Azure built-in roles, available by default.
- Or create a custom role:

1. Determine the permissions you need.
2. Decide how you want to create the custom role.
3. Create the custom role.
4. Test the custom role.

Demo

- Manage Access for Azure AI Hub Collaboration

Summary



Key learning points

- What is the Azure AI Studio?
- Configure your environment with hubs and projects.
- Create connections to connect to external resources.
- Manage access for collaboration.

Explore, deploy, and chat with language models

Microsoft

Learning Objectives

- Understand and use the model catalog

Microsoft

Learning Objective: Understand and use the model catalog

Select a model from the Model catalog

Foundation models available through the Model catalog are already pretrained. You can deploy a foundation model to an endpoint or fine-tune a model to make it perform better in a specialized task and on domain-specific knowledge.

Your selected model will depend on your use case and deployment preferences.

Model	Description
BERT (Bidirectional Encoder Representations from Transformers)	Focused on encoding information by using context from before and after a token (bidirectional). Commonly used when you want to fine-tune a model to perform a specific task like *text classification* and *question answering*.
GPT (Generative Pretrained Transformer)	Trained to create coherent and contextually relevant text and is most commonly used for tasks like *text generation* and *chat completions*.
LLaMA (Large Language Model Meta AI)	A family of models created by Meta. When training LLaMA models, the focus has been on providing more training data than increasing the complexity of the models. You can use LLaMA models for *text generation* and *chat completions*.
Phi-3-mini (3.8B parameters variation of phi models)	A lightweight, state-of-the-art model optimized for resource-constrained environments and local inference (like on a phone), supporting long-context prompts up to 128K tokens. It is developed with a focus on safety, alignment, and reinforcement learning from human feedback.

Deploy a model to an endpoint

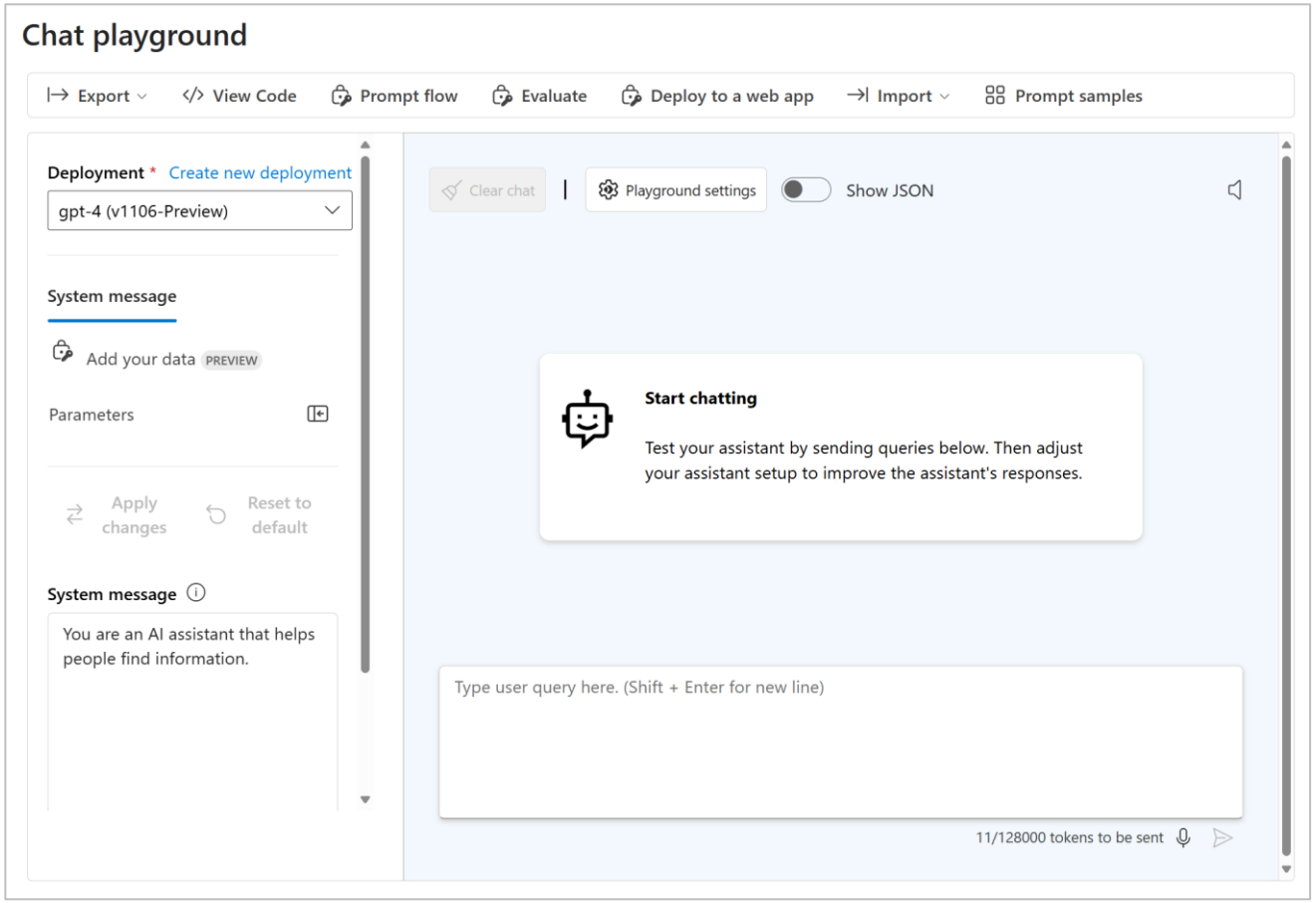
Before you can integrate a model from the Model catalog in your applications, you have to **deploy the model** to generate an **API endpoint** for your applications to consume.

The type of deployment depends on the model you want to deploy:

	Azure OpenAI models	Models deployed as Serverless APIs (pay-as-you-go)	Models deployed with user-managed compute
Deploy the model	No, you aren't billed for deploying an Azure OpenAI model to your project.	Yes, you're billed minimally per the infrastructure of the endpoint.	Yes, you're billed for the infrastructure hosting the model per minute.
Call the model endpoint	Yes, you're billed based on your token usage.	Yes, you're billed based on your token usage.	None.

Chat with model in playground

Experiment and improve your model performance quickly and easily by chatting with your model in the **chat playground**.



Demo

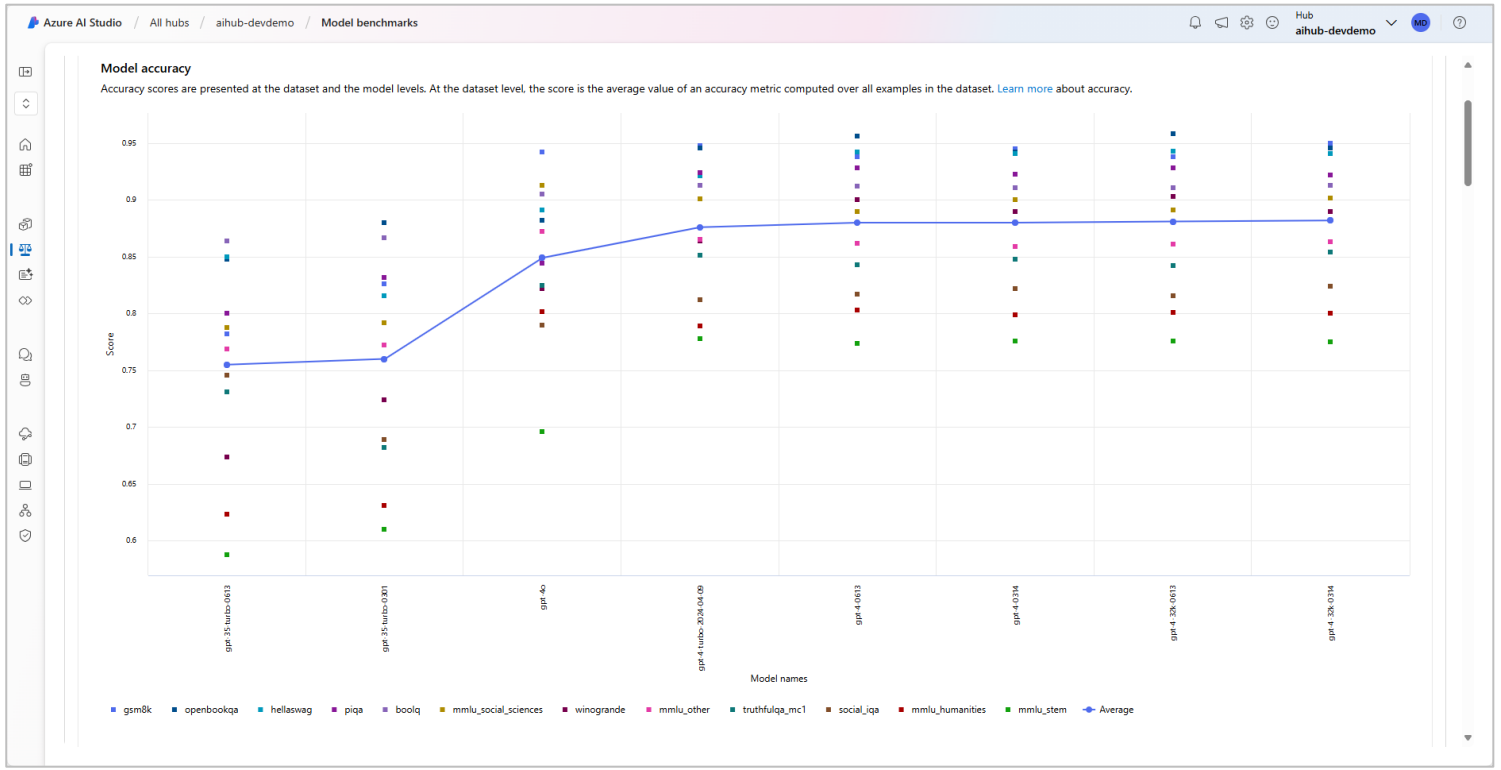
- Explore, deploy, and chat with language models in the Azure AI Studio

Compare benchmarks across models

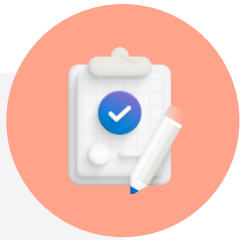
Model benchmarks are used to assess the quality of language models before deploying and integrating them.

Most commonly used metrics:

- **Accuracy**
- **Coherence**
- **Fluency**
- **GPTSimilarity**
- **Groundedness**
- **Relevance**



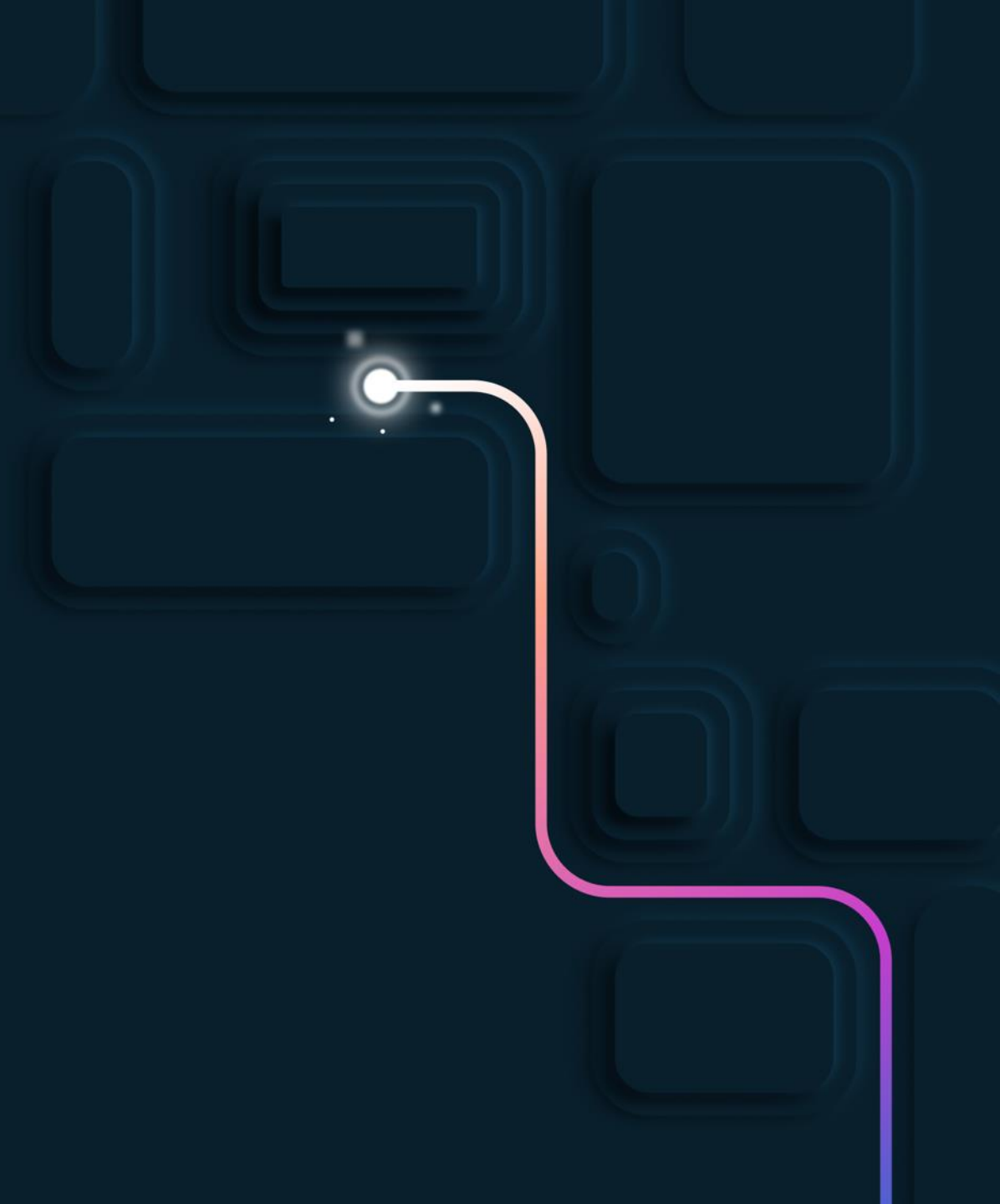
Summary



Key learning points

- Select a model from the Model catalog
- Compare models with Model benchmarks
- Deploy a model to an endpoint
- Chat with a model in the playground

Compare model optimization strategies



Microsoft

Learning Objectives

- Learn when to use fine tuning, RAG, or prompt engineering

Microsoft

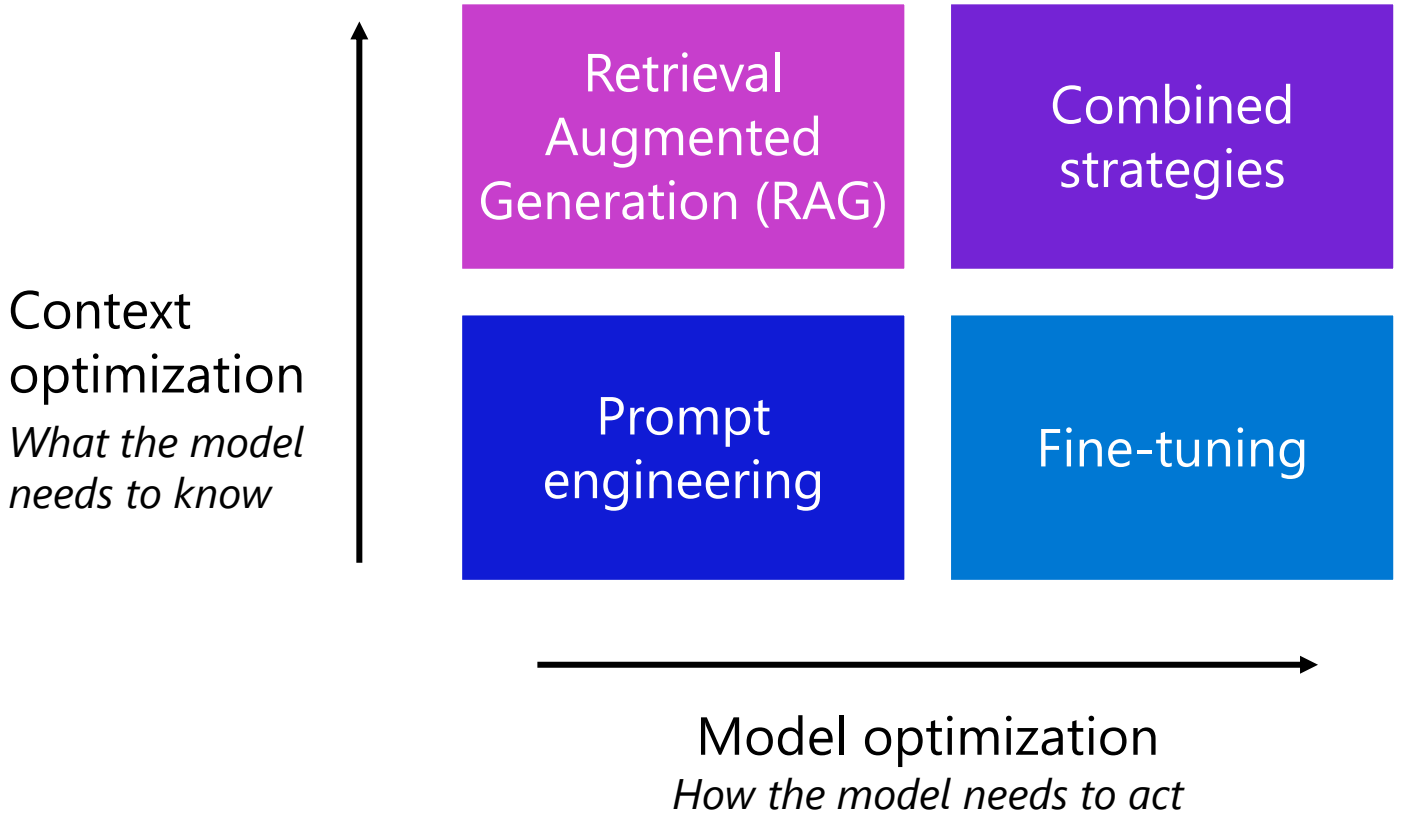
Learning Objective: Learn when to use fine tuning, RAG, or prompt engineering

Improve the model performance

Optimization strategy will depend on your requirements:

Optimize for context when the model lacks contextual knowledge, and you want to **maximize response accuracy**.

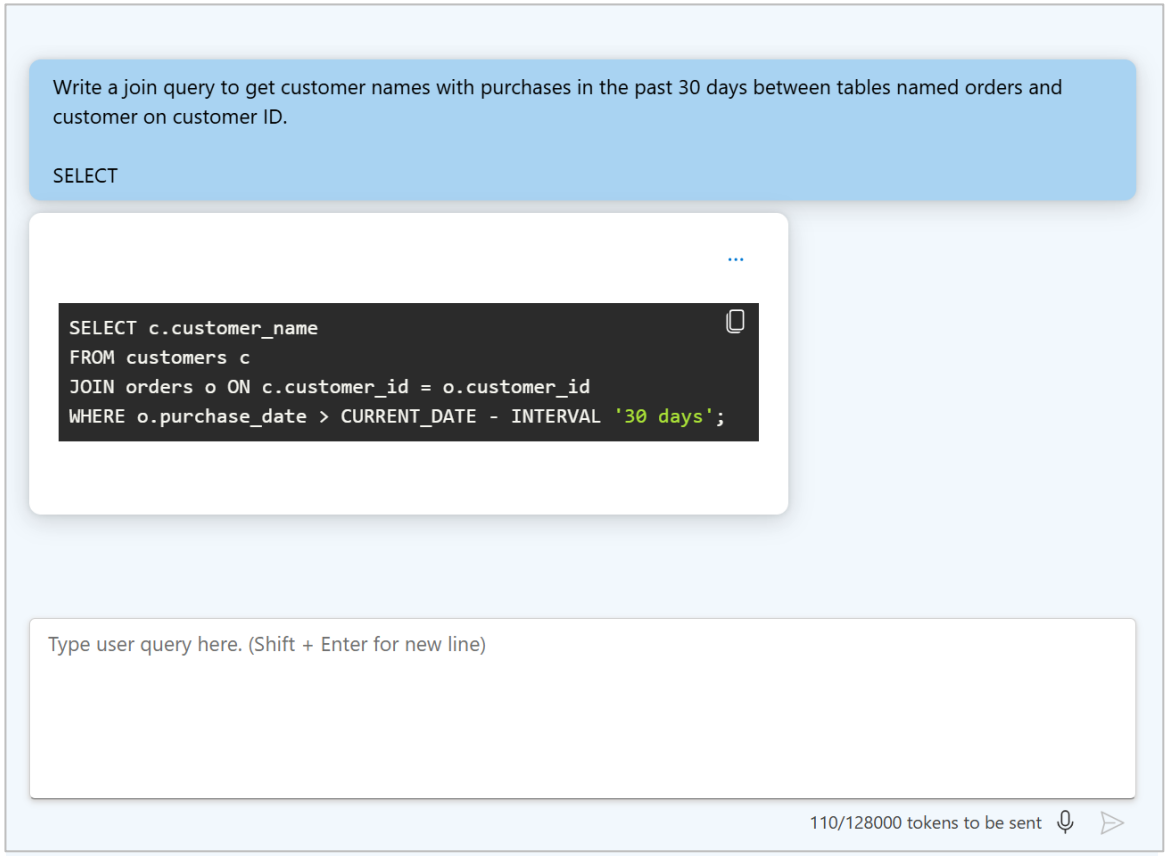
Optimize the model when you want to improve the response format, style, or speech by **maximizing consistency of behavior**.



Apply prompt engineering

To improve the model's output as a user, you can apply **prompt engineering**:

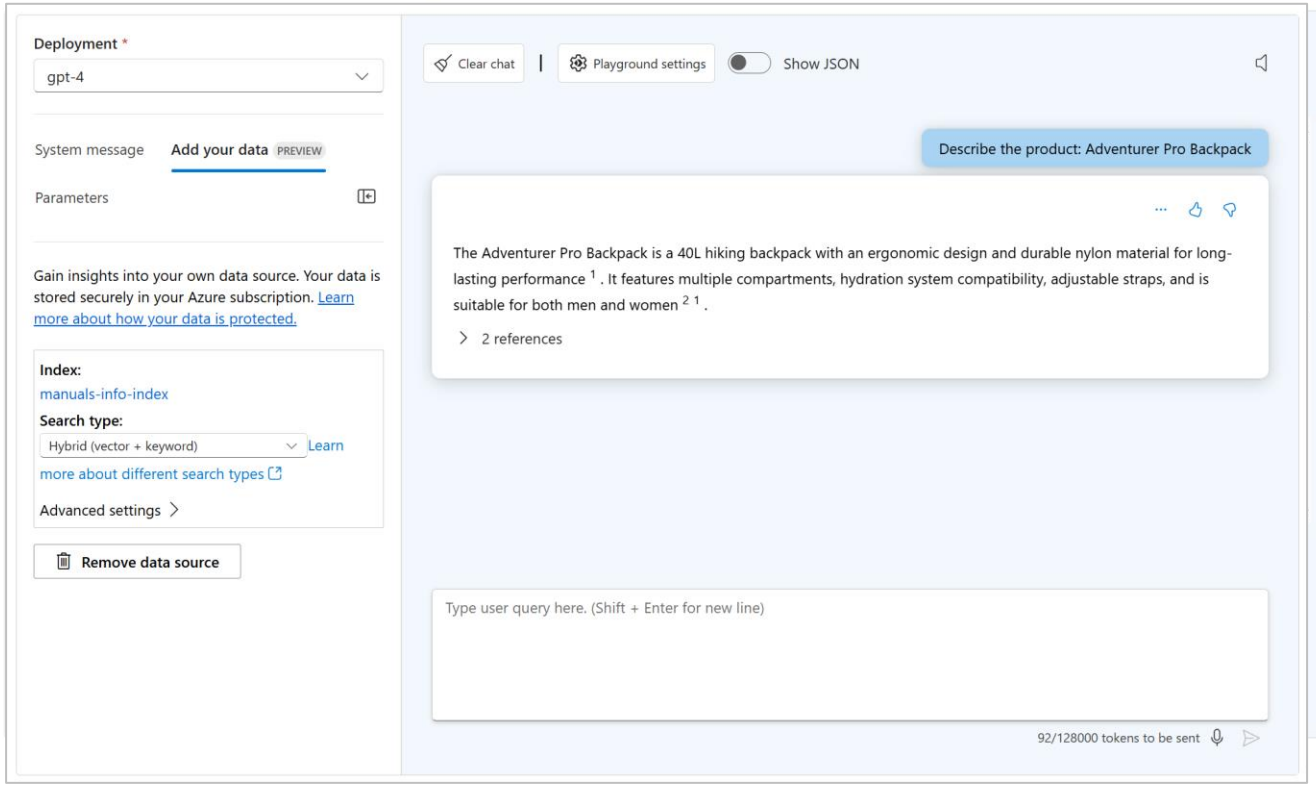
- Provide clear instructions
- Format your instructions
- Use cues



Update the system message

To improve the model's output as a developer, you can update the **system message**:

- Use one/few shot(s)
- Use chain-of-thought
- Add context

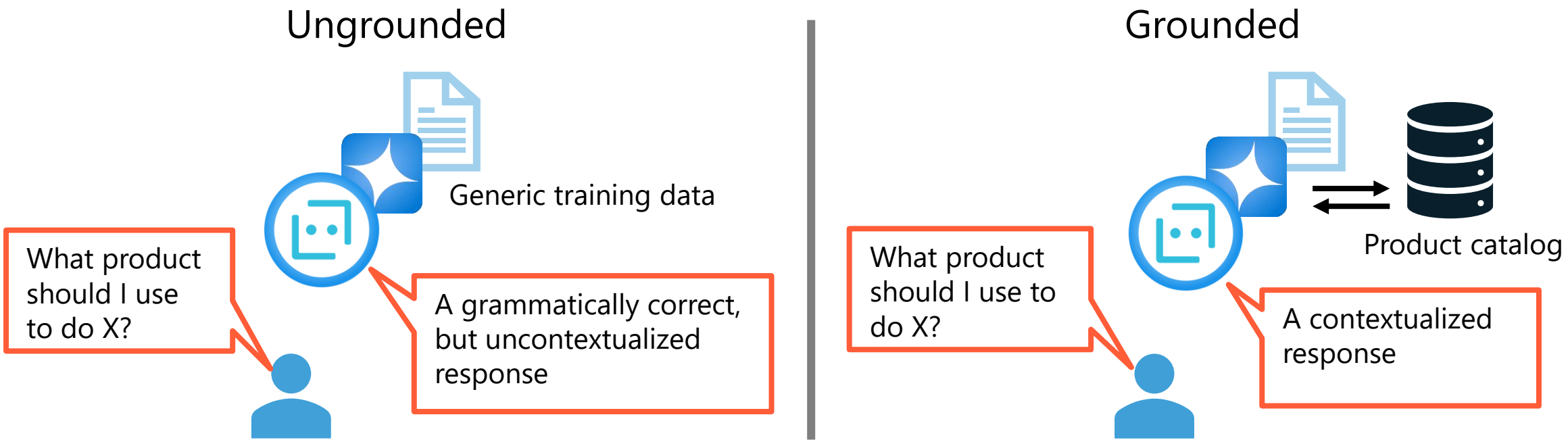


Demo

- Prompt Engineering
- Updating Chat System Message

Grounding a copilot with your own data

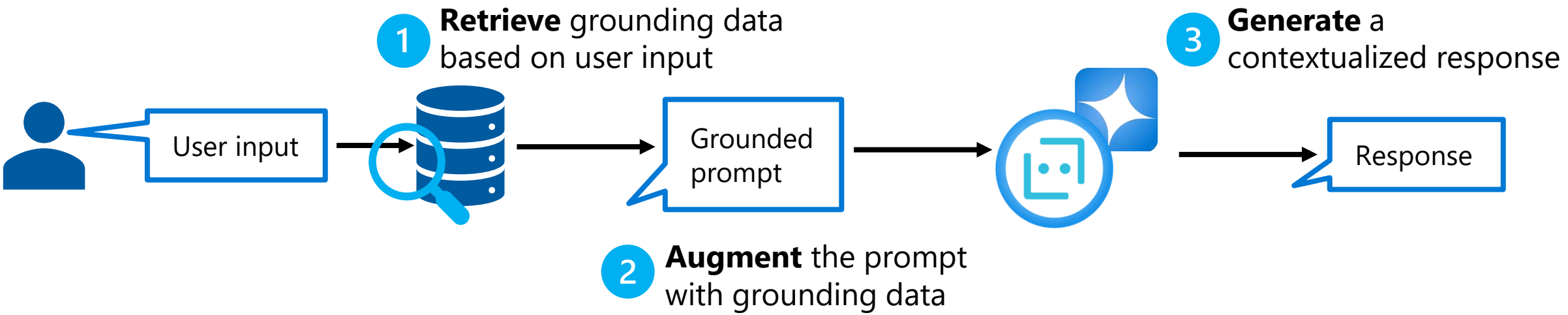
- Language models create coherent answers to questions, but what are those answers based on?
- **Grounding** provides specific context to the model to provide accurate, relevant responses.



Retrieval Augmented Generation (RAG)

RAG gathers relevant data to include with a prompt for the language model to use for grounding context.

A **RAG pattern** is an architectural design for RAG by including retrieved relevant data in a prompt:



Understand when to fine-tune a model

Fine-tuning will improve the model when:

- You have a clear specific use case.
- You need the model's output to be in a specific customized style.
- You want to have a more consistent performance by providing more few shots than can fit in your prompt.

You can fine-tune a model by selecting a **base model** and training it on **your own training data**. Once fine-tuning is complete, you can deploy your fine-tuned model to an **endpoint**.

Prepare your data to fine-tune a chat completion model

The data must be formatted as a JSON Lines (JSONL) document.
For fine-tuning on a chat completion task your data should include:

System message:
"You are an Xbox customer support agent whose primary goal is to help users with issues they are experiencing with their Xbox devices. You are friendly and concise. You only provide factual answers to queries, and do not provide answers that are not related to Xbox."

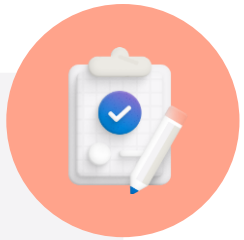
User message:
"Is Xbox better than PlayStation?"

Assistant answer:
"I apologize, but I cannot provide personal opinions. My primary job is to assist you with any issues related to your Xbox device. Do you have any Xbox-related issues that need addressing?"

Demo

- Fine-Tuning a Model

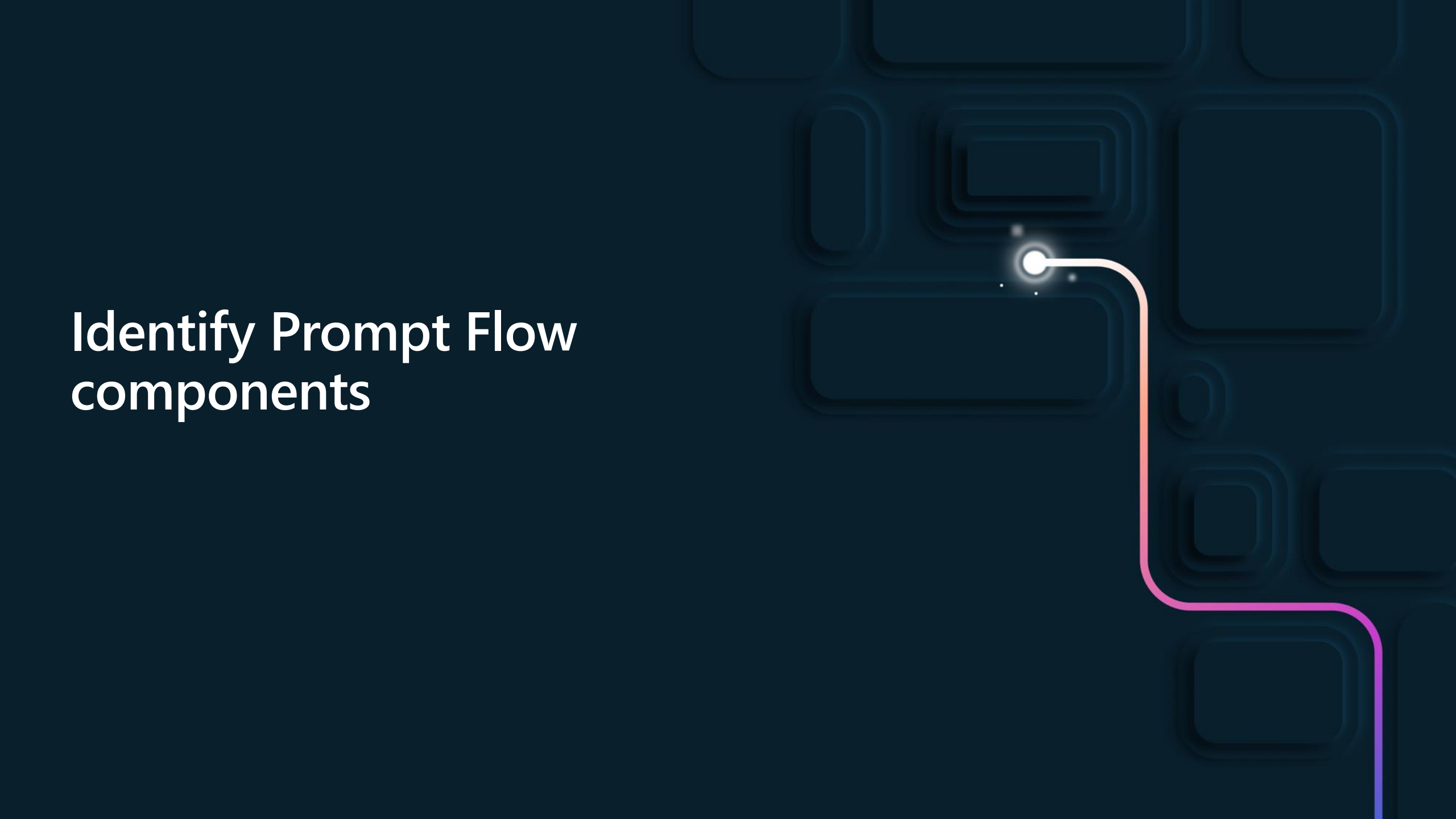
Summary



Key learning points

- Apply prompt engineering
- Update the system message
- Ground a copilot with your own data (RAG)
- Fine-tune a model

Identify Prompt Flow components



Microsoft

Learning Objectives

- Use prompt flow to build a copilot

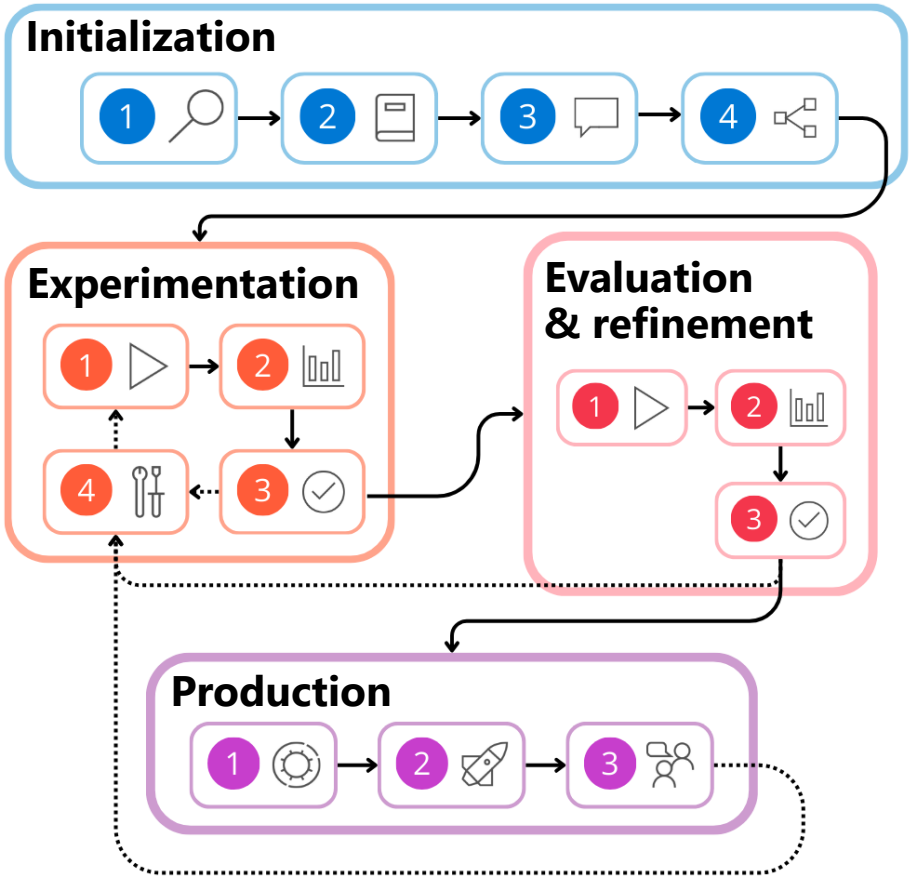
Microsoft

Learning Objective: Use prompt flow to build a copilot

Explore the language app development lifecycle

The lifecycle consists of the following stages:

- **Initialization:**
  Define the use case and design the solution.
- **Experimentation:**
  Develop a flow and test with a small dataset.
- **Evaluation and refinement:**
  Assess the flow with a larger dataset.
- **Production:**
  Deploy and monitor the flow and application.

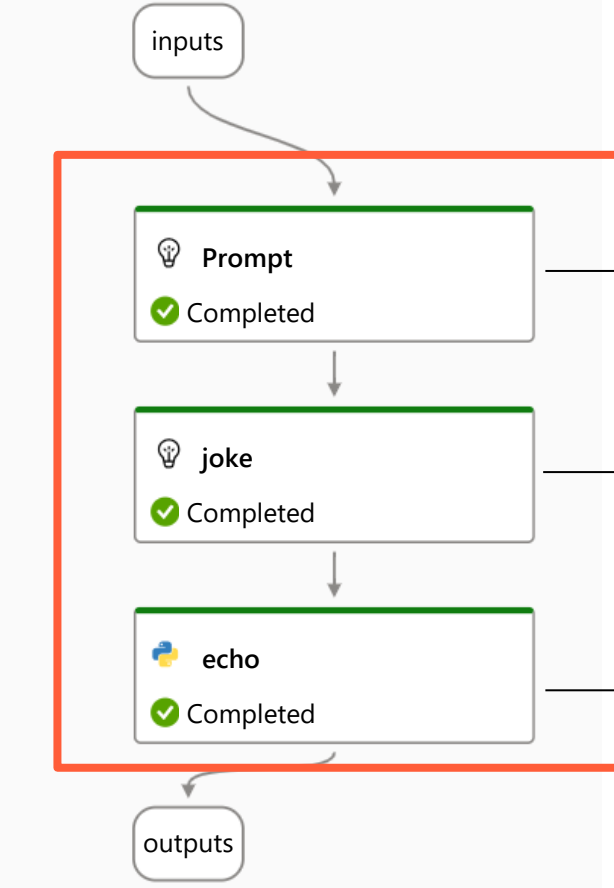


Understand a flow and its components

Inputs: Represent data passed into the flow. Can be different data types like strings, integers, or boolean.

Nodes: Represent *tools* that perform data processing, task execution, or algorithmic operations.

Outputs: Represent the data produced by the flow.



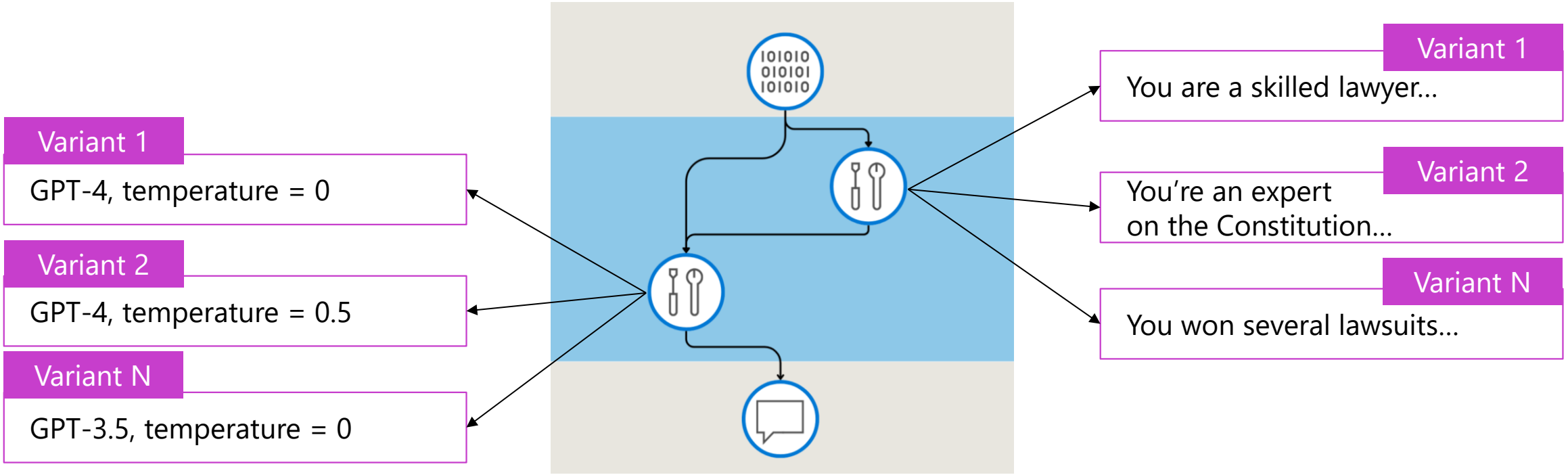
Prompt tool: Prepares prompts as strings for complex scenarios or integration with other tools.

LLM tool: Enables custom prompt creation utilizing Large Language Models.

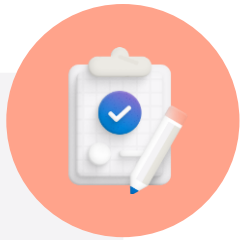
Python tool: Allows the execution of custom Python scripts.

Optimize your flow with variants

Create **variants** for your prompt or LLM nodes to easily compare variations when invoking your language model. You can vary your system message, prompt, and parameters.



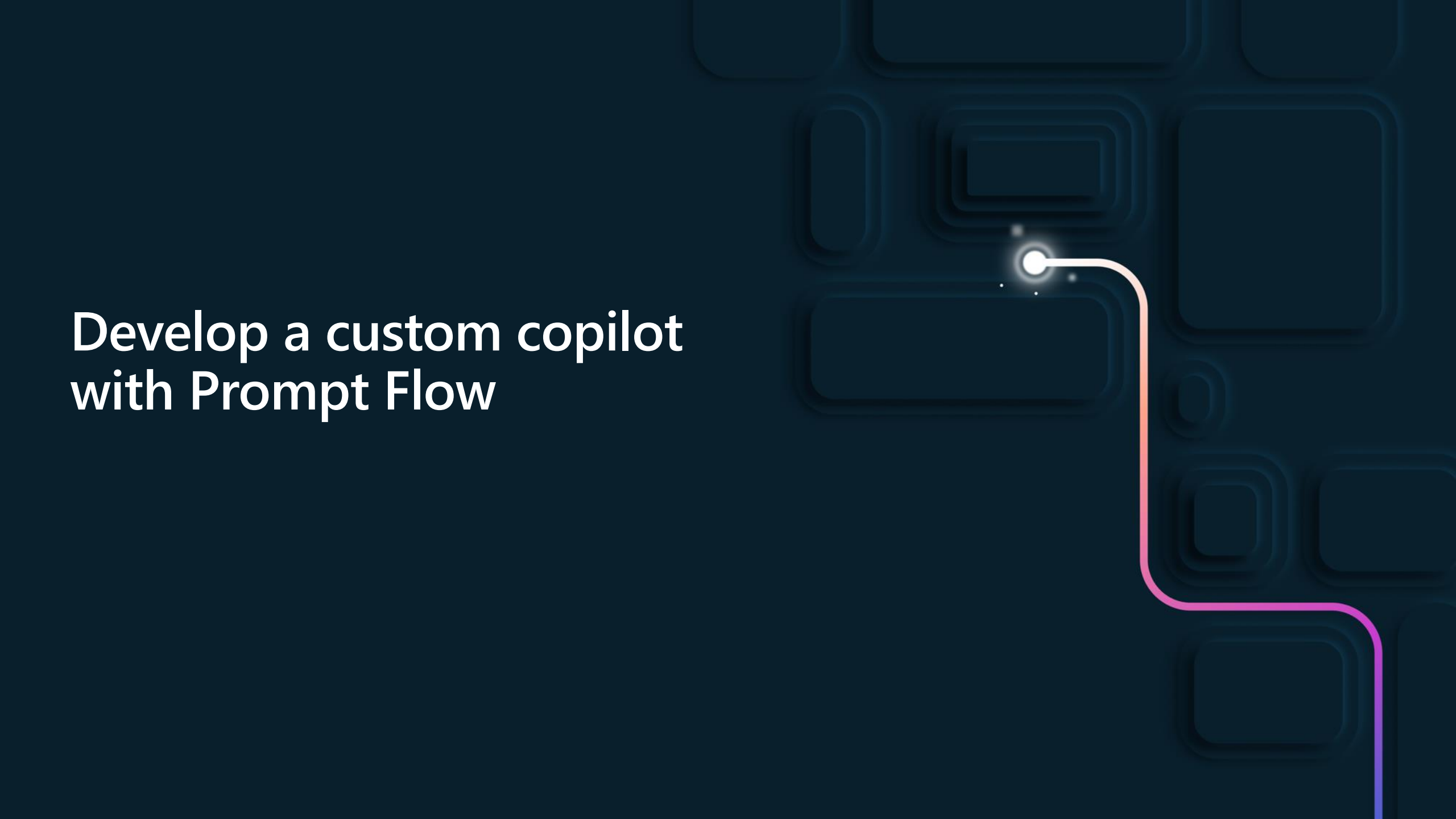
Summary



Key learning points

- Explore the language app development lifecycle
- Understand a flow and its components
- Optimize your flow with variants

Develop a custom copilot with Prompt Flow



Microsoft

Learning Objectives

- Use RAG in your custom copilot

Microsoft

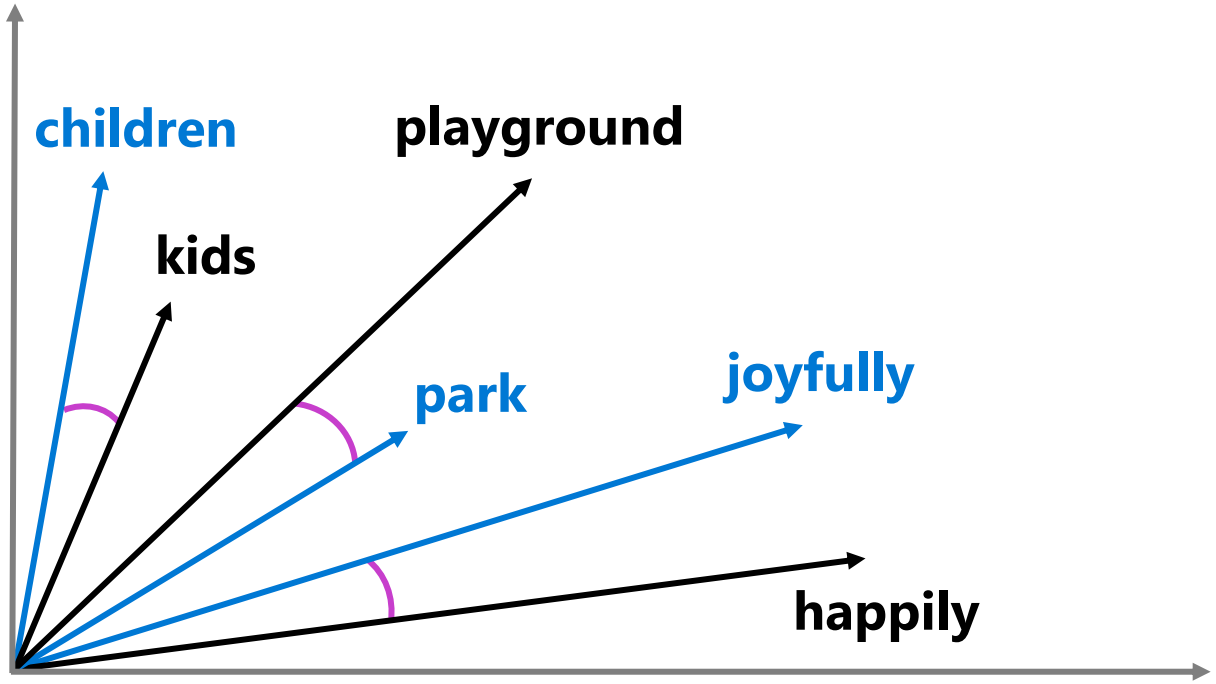
Learning Objective: Use RAG in your custom copilot

Retrieval Augmented Generation (RAG)

Data is stored as embeddings in a **vector-based search index**.

Embeddings allow better data retrieval for *semantic similarities* in the search results.

Document 1
The **children** played **joyfully** in the **park**



Document 2
Kids **happily** ran around the **playground**

Demo

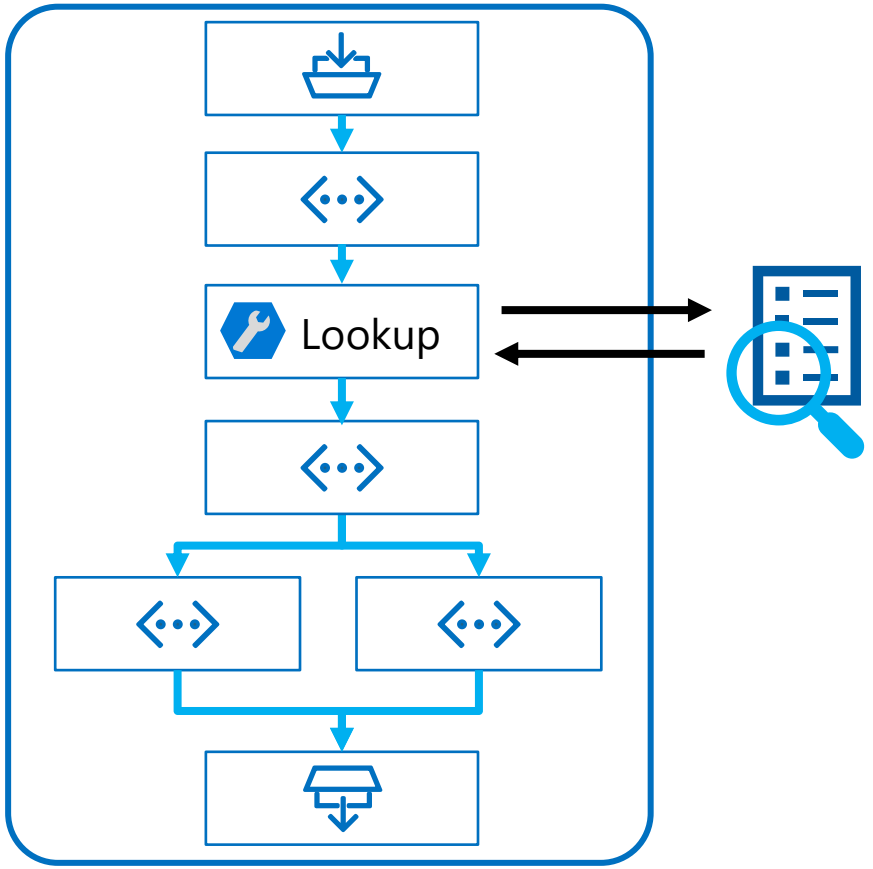
- Develop a custom copilot with Prompt Flow

Add your data in a prompt flow

To ground your language model's responses, add your own data source to retrieve relevant context:

1. Add your **data** to an Azure AI project.
2. Index your data with **Azure AI Search**.
3. Query your indexed data in a prompt flow with the **Index Lookup** tool.
4. Reference the retrieved **context** in a prompt.
5. Send the prompt with context to an **LLM**.

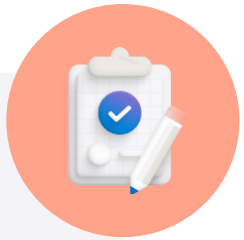
The generated response will be grounded by the retrieved context.



Demo

- Create a custom copilot that uses your own data
- Explore Content Filters

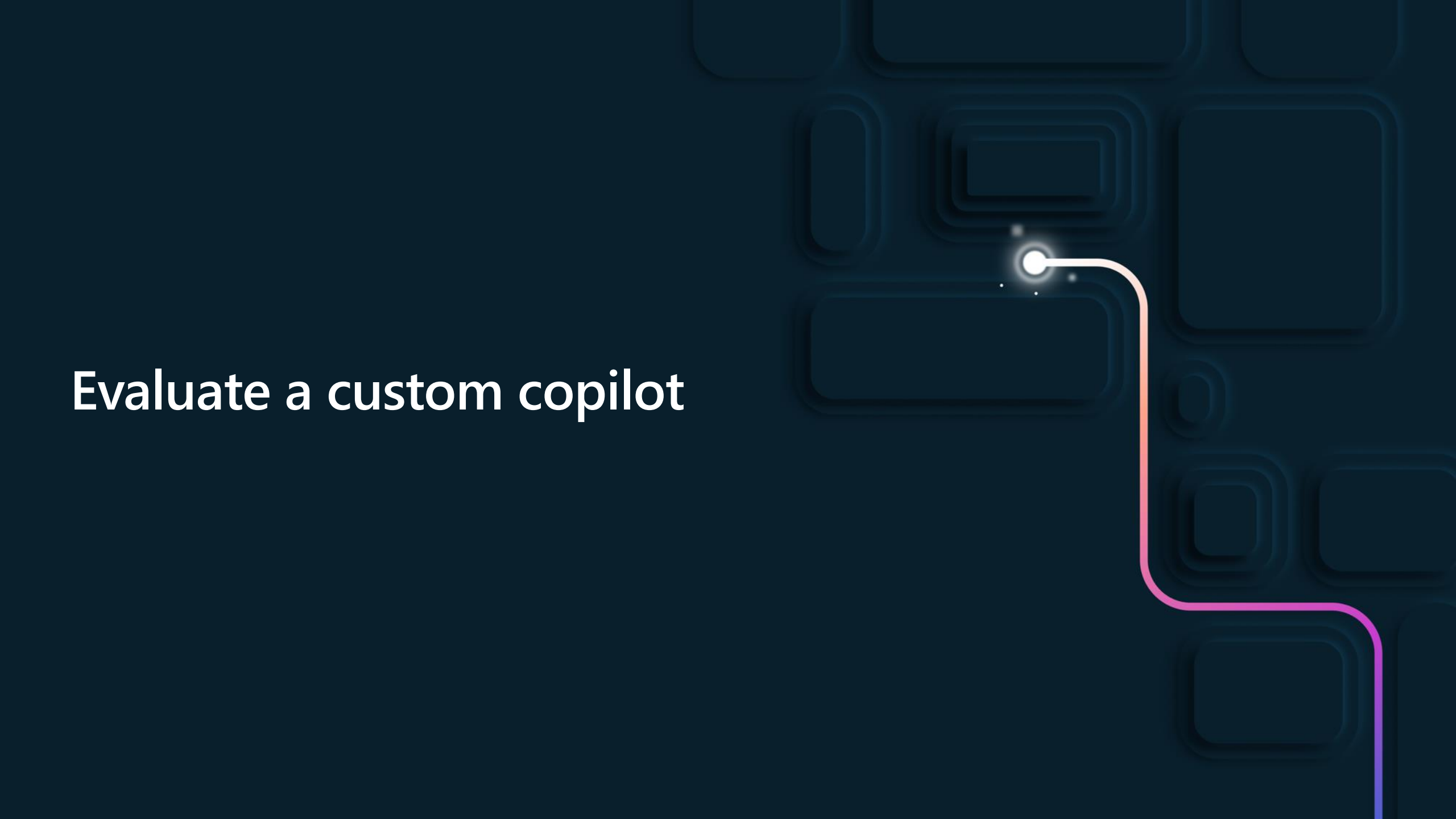
Summary



Key learning points

- Implement RAG in a chat flow
- Develop a custom copilot with Prompt Flow
- Integrate language models in a chat flow

Evaluate a custom copilot



Microsoft

Learning Objectives

- Evaluate your prompt flow
- Use Code-First Development approach

Microsoft

Learning Objective: Evaluate your prompt flow

Assess the model performance

Evaluate your model performance at different phases, using a variety of evaluation approaches:

Model benchmarks

- Compare publicly available metrics across models and dataset

Manual evaluations

- Rate your model responses

Traditional machine learning metrics

- Measure ratio of number of shared words between generated and ground truth answers

AI-assisted metrics

- Risk and safety metrics
- Generation quality metrics

Understand model benchmarks

Datasets are publicly available to calculate individual benchmarks and compare across models. Some commonly used benchmarks are:

Accuracy:
Compares model generated text with correct answer according to the dataset. Result is one if generated text matches the answer exactly, and zero otherwise.

Coherence:
Measures whether the model output flows smoothly, reads naturally, and resembles human-like language

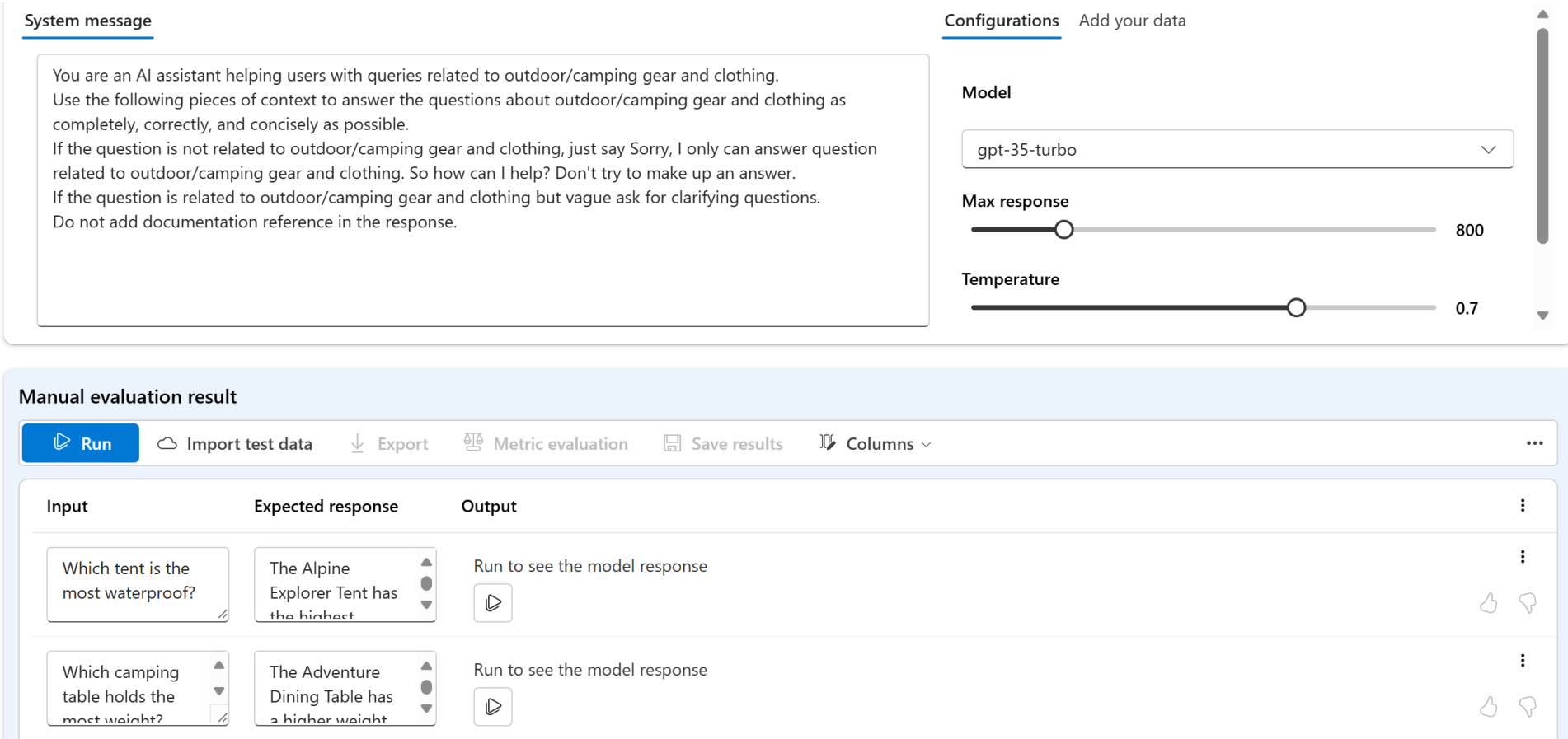
Fluency:
Assesses how well the generated text adheres to grammatical rules, syntactic structures, and appropriate usage of vocabulary, resulting in linguistically correct and natural-sounding responses.

GPT Similarity:
Quantifies the semantic similarity between a ground truth sentence (or document) and the prediction sentence generated by an AI model

Demo

- Compare Benchmarks across Models

Explore manual evaluations



Assess your flow with built-in metrics

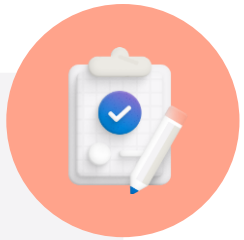
To evaluate chat applications you built with prompt flow:

1. Create a test dataset.
2. Create a new **automated** evaluation.
3. Select a flow or a dataset with model generated outputs.
4. Select the metrics you want to evaluate on.
5. Run the evaluation flow.
6. Review the results.

Demo

Evaluate the performance of your custom copilot in the Azure AI Studio

Summary



Key learning points

- Explore the evaluation of custom copilots
- Explore manual evaluations
- Assess your copilot with built-in metrics

Microsoft

Learning Objective: Use Code-First Development approach

Custom Copilot Code-First Development

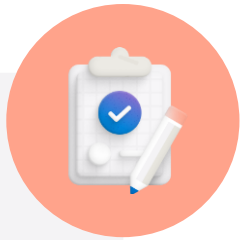
Variety of tools and frameworks available within the Azure AI ecosystem:

- Azure AI Studio
- Azure AI SDKs
- Jupyter Notebooks
- Prompt Flow
- VS Code, Visual Studio, GitHub CodeSpaces
- Semantic Kernel and LangChain
- Azure OpenAI Service
- Azure AI Large Language Models
- Azure Cognitive Search and Vector Search

Demo

Custom Copilot Code-First Development

Summary



Key learning points

- Custom Copilot Code-First Development
- Azure AI Tools and Frameworks