



Hands-On Demo

Azure PaaS App

App Url: <https://salespoc.azurewebsites.net>

GitHub Url: <https://github.com/csdmichael/SalesPOC.UI>

Read Me: [Frontend UI for Sales POC App](#)

Prepared by:

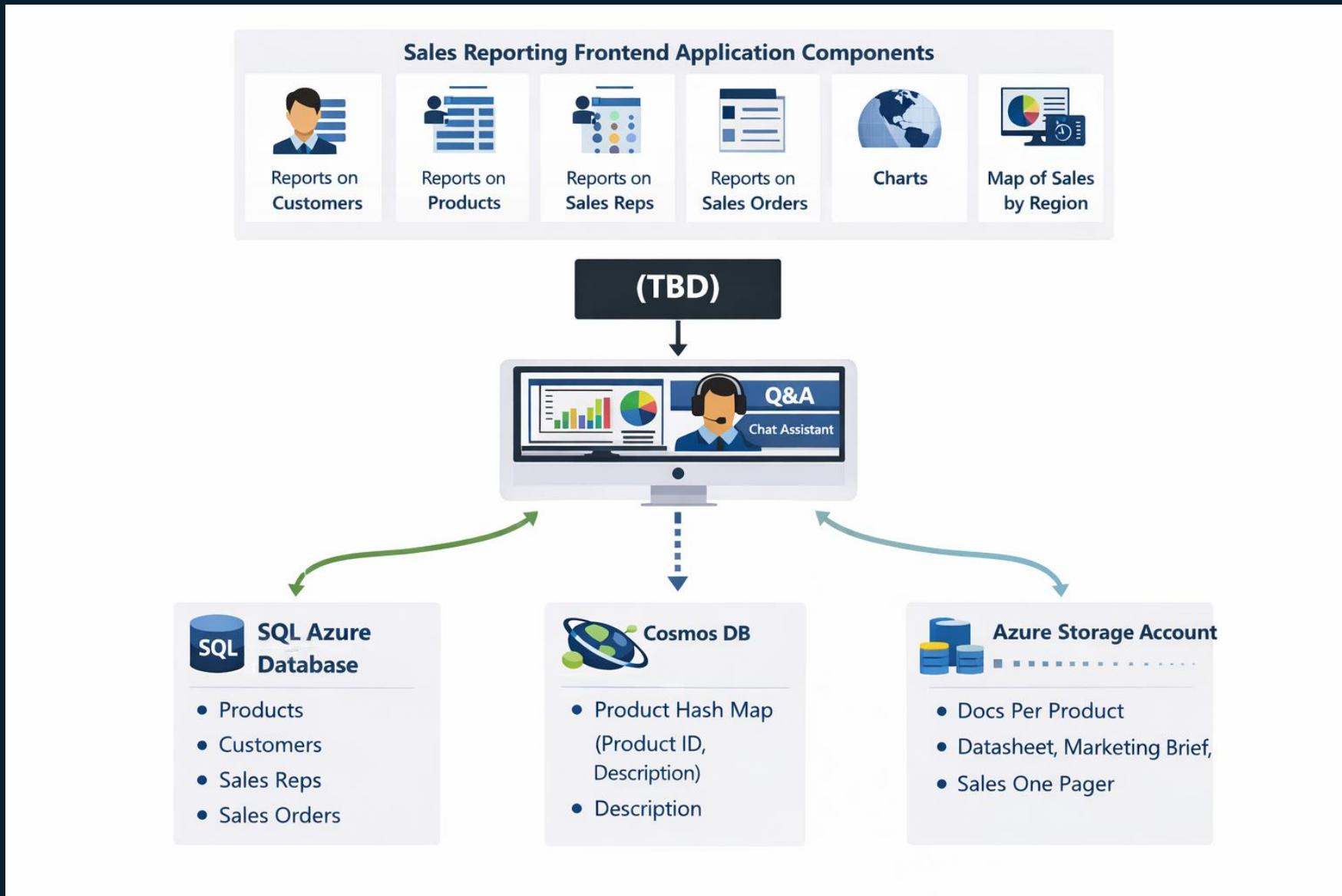
Michael Yaacoub
Sr. Solution Engineer – AI Applications
Microsoft

Steps for building Sales POC



- 0. High Level Requirements & UX Mockups**
- 1. Identify functional Requirements and non-functional Requirements**
- 2. Generate Architecture Diagram to satisfy requirements**
- 3. Build APIs and publish to Azure Container App**
- 4. Create new API Management Instance in Azure and Import Sales-API**
- 5. Create MCP Server in APIM**
- 6. Go to API Center -> Register APIM Instances for Dev, Testing, Prod for API discovery & Linting**
- 7. Go to AI Foundry -> Build an agent that uses MCP Server (exposed in APIM) as a tool**
- 8. Build Frontend Application and Publish to Azure App Service**
- 9. Test the Application & Sales Chat AI Assistant**

0. High Level Requirements



0. UX Mockups



Sales POC

Customers Total: 1000 records

ID	Name	Type	Industry	Country	State	City	Annual Revenue (USD)
1	Customer 1	Distributor	Semiconductors	USA	CA	San Jose	\$476,385,208.00
2	Customer 2	Foundry	Semiconductors	USA	CA	San Jose	\$400,317,597.00
3	Customer 3	Fabless	Semiconductors	USA	CA	San Jose	\$274,165,439.00
4	Customer 4	Research	Semiconductors	USA	CA	San Jose	\$64,610,403.00
5	Customer 5	OEM	Semiconductors	USA	CA	San Jose	\$12,672,534.00
6	Customer 6	Distributor	Semiconductors	USA	CA	San Jose	\$268,438,184.00
7	Customer 7	Foundry	Semiconductors	USA	CA	San Jose	\$11,983,267.00
8	Customer 8	Fabless	Semiconductors	USA	CA	San Jose	\$501,041,045.00
9	Customer 9	Research	Semiconductors	USA	CA	San Jose	\$348,010,605.00
10	Customer 10	OEM	Semiconductors	USA	CA	San Jose	\$288,451,090.00
11	Customer 11	Distributor	Semiconductors	USA	CA	San Jose	\$214,094,691.00
12	Customer 12	Foundry	Semiconductors	USA	CA	San Jose	\$412,678,643.00
13	Customer 13	Fabless	Semiconductors	USA	CA	San Jose	\$249,942,464.00
14	Customer 14	Research	Semiconductors	USA	CA	San Jose	\$329,872,4
15	Customer 15	OEM	Semiconductors	USA	CA	San Jose	\$412,947,454.00

Sales POC

Products Total: 200 records

ID	Name	Category	Process Node (nm)	Package Type	Unit Price (USD)	Lifecycle Status
1	Chip-1	ASIC	7	QFN	\$455.00	Active
2	Chip-2	FPGA	14	WLCSP	\$324.00	Active
3	Chip-3	Power IC	28	DIP	\$84.00	Active
4	Chip-4	Sensor	40	BGA	\$356.00	Active
5	Chip-5	RF	85	QFN	\$308.00	Active

Product Description
Chip-2 is a FPGA semiconductor device built on a 14nm process node. It is optimized for IoT applications requiring high performance, power efficiency, and long-term reliability. Chip-2 supports industrial-grade temperature ranges and is designed for scalable system architectures and long lifecycle deployments.

Product Documents

- Engineering Datasheet
- Marketing Brief
- Sales One Pager

Sales POC

Sales Facts Total: 30054 records

CUSTOMER	PRODUCT	CATEGORY	REGION	SALES REP
Customer 259	Chip-29	RF	2487	\$267.00
Customer 259	Chip-192	MCU	2168	\$232.00
Customer 259	Chip-134	FPGA	744	\$138.00
Customer 259	Chip-97	ASIC	1423	\$201.00
Customer 218	Chip-66	MCU	607	\$178.00
Customer 218	Chip-13	ASIC	1952	\$464.00
Customer 218	Chip-100	Sensor	176	\$166.00
Customer 218	Chip-18	MCU	2956	\$178.00
Customer 218	Chip-150	MCU	4817	\$326.00
Customer 519	Chip-70	Sensor	4337	\$67.00
Customer 519	Chip-19	ASIC	2891	\$359.00
Customer 519	Chip-199	ASIC	2429	\$392.00

Sales Orders Total: 9 records

ORDER ID	CUSTOMER	SALES REP
226	Customer #10	Rep #23
374	Customer #10	Rep #16
3500	Customer #10	Rep #4
4923	Customer #10	Rep #3
5013	Customer #10	Rep #18
5412	Customer #10	Rep #12
8353	Customer #10	Rep #15
8480	Customer #10	Rep #20
9176	Customer #10	Rep #17

Sales Assistant
Powered by Azure AI Foundry

Hii I'm the Sales Assistant. Ask me anything about sales data, customers, products, or orders.

What is product description for Chip-50? Also retrieve Sales One Pager, Datasheet and market brief documentation. Finally check its unit price and status of its life cycle

Product Description for Chip-50
Chip-50 is a FPGA semiconductor device built on a 14nm process node. It is optimized for automotive applications requiring high performance, power efficiency, and long-term reliability. Chip-50 supports industrial-grade temperature ranges and is designed for scalable system architectures and long lifecycle deployments.

Key Product Documents for Chip-50

- Sales One Pager: Download Chip-50 Sales One Pager
- Datasheet:

Ask a question about sales data...

1. Identify functional Requirements and non-functional Requirements



Functional Requirements (*What the system does for the user*)

1. Reports to display and filter data

Provide interactive reports and dashboards for **Customers** (segmentation), **Products** (performance), and **Sales Reps** (quota attainment)

2. Real-Time Order Tracking

Real-time visibility into order status, fulfillment pipelines, and historical transaction logs

3. AI-Powered Natural Language Query (NLQ)

An integrated chatbot that allows users to ask questions like "*What are the top 10 products by price?*" and receive instant data visualizations.

Next Phase

1. Role-Based Access Control (RBAC)

Non-Functional Requirements

How the system operates and its quality constraints.

1. Intuitive UX/UI

A "zero-training" interface designed for mobile and desktop, ensuring high adoption rates among non-technical sales staff.

2. Low Latency Performance

Chatbot responses and dashboard renders must occur in **under 5 seconds** to maintain user engagement.

3- Reusability of APIs across organization

Reuse same APIs in Sales Transactional System (Consumer Application) and make them available for internal teams

Next Phase

1. Scalability – Low priority as it is internal App

2. High Availability – Low priority as it is internal App

2. Data Security & Compliance

2. Generate Architecture Diagram to satisfy requirements



- Account for big Picture & Integrations & Reusability in other systems
- Estimate the system capacity in short and long term to determine the proper Services, Service Tiers, SKUs
- Choose relevant Azure Services to implement a reliable / scalable / cost effective Architecture

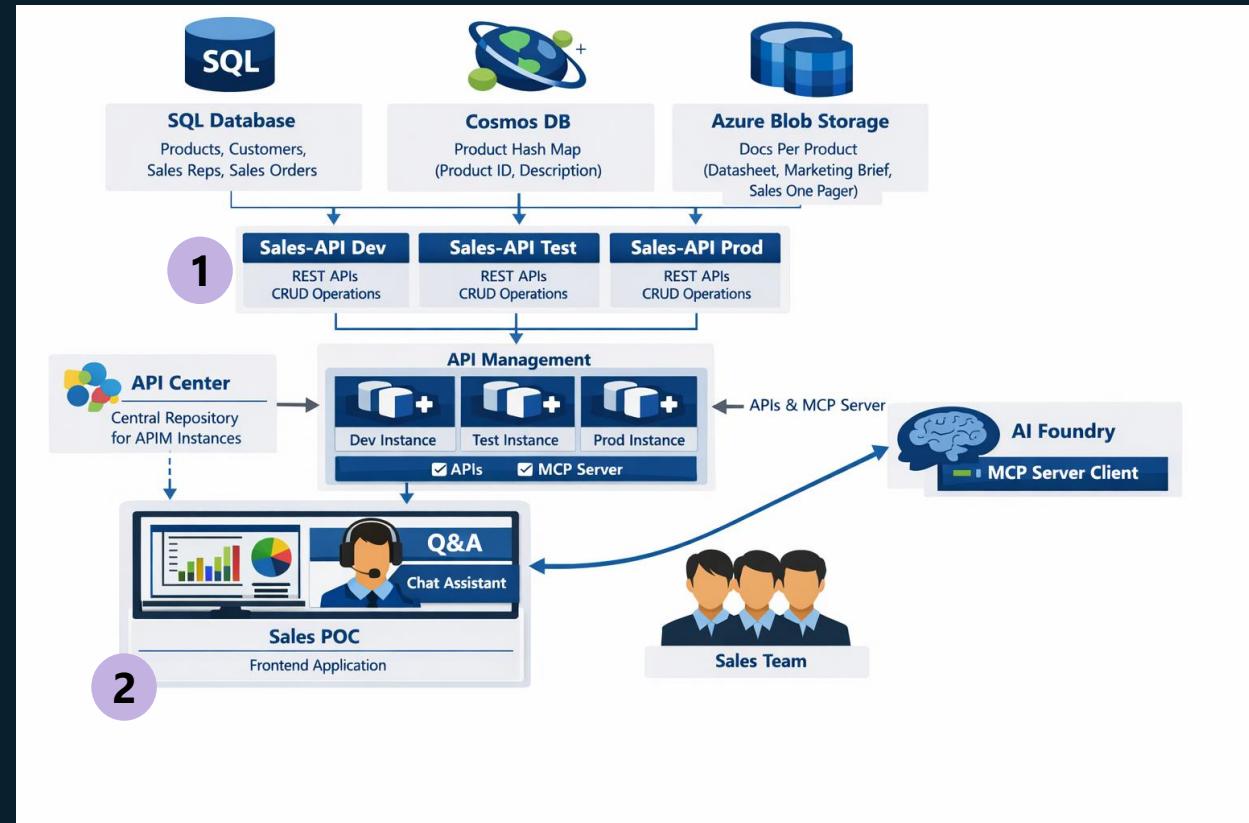
Decisions to make

1. Which Service to use for Sales-API?

➤ We need to review big picture

2. Which Service to use for Frontend Internal Reporting Web App?

➤ Azure App Service (Internal Sales Team – Small number of users)



2. Account for big picture - Sales Transactional System

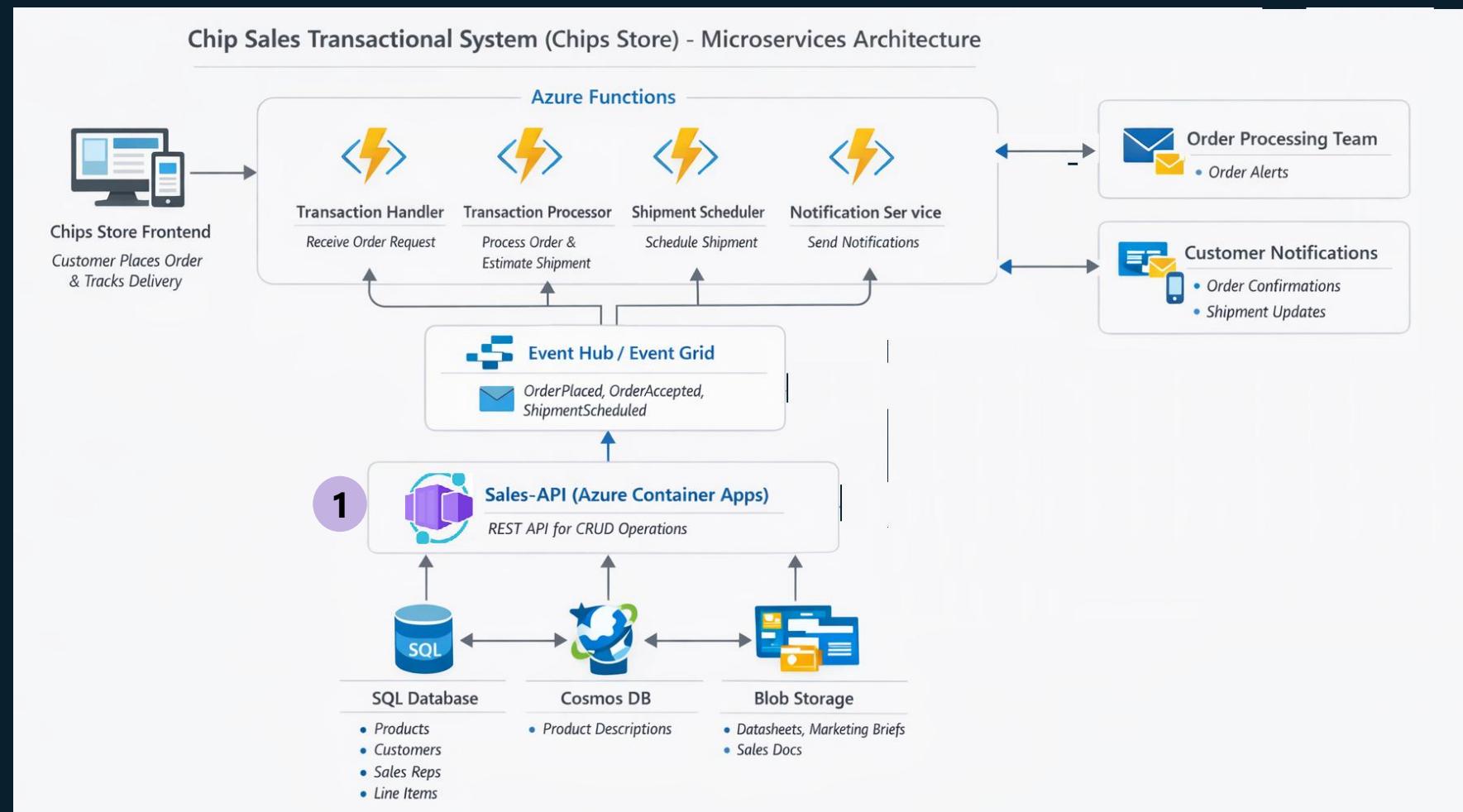


Big Picture helps understand scale

Sales-API can use "Azure Container Apps"

Why?

Shared service in internal Report System and Sales Transactional System used by external Consumers



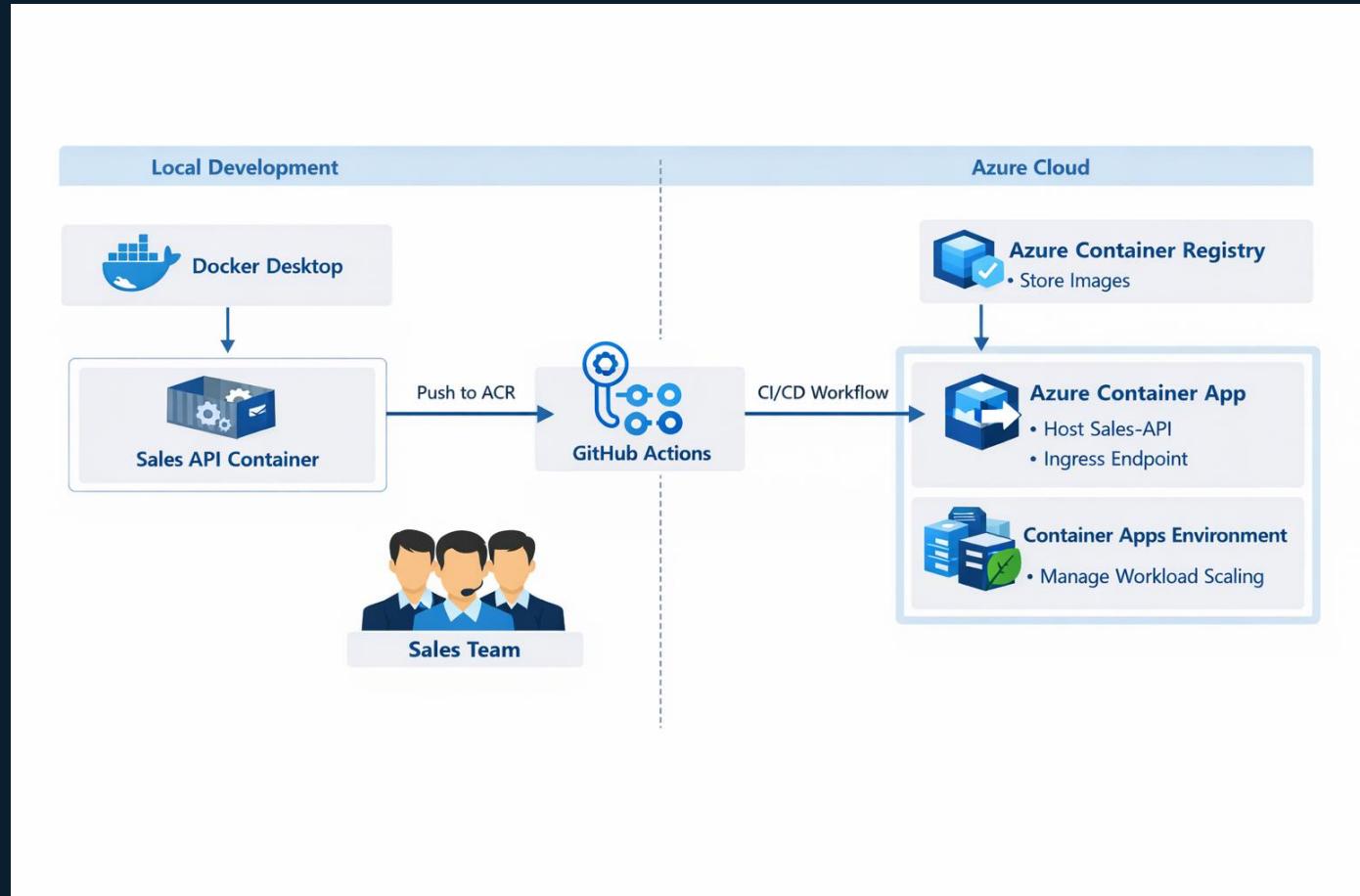
3. Build APIs – Azure Container Apps



Flow to create Azure Container App

Components Needed

- **Docker Desktop**
- **GitHub Action** to push docker container (e.g. Sales-API) to **ACR**
- **ACR** stores Container Images
- Deploy Sales-API from **ACR** to **ACA** (Azure Container App)
- **ACA** hosts Sales-API
- **Container Apps Environment** manages Workload Scaling



- **ACR:** Azure Container Registry
- **ACA:** Azure Container Apps

3. Build APIs – Azure Container Apps



1. Go to Azure Portal
 - a) Create "**Azure Container Registry**" and fetch Uri (Identifier)
 - b) Create "**Azure Container App**" and fetch Uri (Identifier)
2. Fetch Connection Strings for all Data Sources that you need to connect to in your App Service
3. Create a GitHub Repository (e.g. SalesPOC.API) and clone it to your computer
4. Open Visual Studio Code (or Visual Studio) and leverage "Copilot Coding Agent" to generate the source code.
Example Prompt: "Generate a .NET 10 REST APIs to implement CRUD operations for the following data sources: {Paste list of data source names and connection strings / access keys} and generate GitHub actions workflow file to deploy the code to Azure Container Registry and Azure Container App {Paste Uri / identifier for ACR, ACA}. Add an ingress https endpoint for Sales-API in Azure Container App".
5. Run Code locally in browser to ensure that it works as expected or re-prompt Copilot to fix issues
6. Ask Copilot Coding Agent to generate a swagger documentation and / or OpenAI Specs to import in APIM later
7. Go to GitHub Repo online and ensure that GitHub Actions ran successfully and deployed App
8. Test APIs

4. Create new API Management Instance in Azure and Import Sales-API



Microsoft Azure

Home > apim-poc-my | APIs

Create an AI API

- Language Model API
- Microsoft Foundry

Define a new API

- HTTP
- WebSocket
- GraphQL
- gRPC

Create from definition

- OpenAPI
- WSDL
- WSDL
- OData

Create from Azure resource

- App Service
- Function App
- Container App
- Logic App

Developer portal Send us your feedback

Search resources, services, and docs (G+)

Microsoft Azure

apim-poc-my | APIs

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Resource visualizer

Events

APIs

MCP Servers

Products

Subscriptions

Named values

Backends

Policy fragments

API Tags

Schemas

Credential manager

API Center

Power Platform

Developer portal

Portal overview

Settings

Users

Groups

Identities

Delegation

OAuth 2.0 + OpenID Connect

Monitoring

Analytics

Analytics (classic)

Application Insights

Alerts

Metrics

Diagnostic settings

Logs

Advisor

Microsoft Azure

Home > apim-poc-my | APIs

API Management service

Search resources, services, and docs (G+)

Developer portal Send us your feedback

Search APIs Filter by tags Group by tag

All APIs

Echo API

SalesAPI

v1

salespoc-ac-a-api

Swagger Petstore

REVISION 3 CREATED Feb 10, 2026, 4:10:00 PM

Design Settings Test Revisions (3) Change log

POST Create an order item

DEL Delete a customer

DEL Delete a product

DEL Delete a sales order

DEL Delete a sales representative

DEL Delete an order item

GET Get a customer by ID

GET Get a product by ID

GET Get a sales order by ID

GET Get a sales rep by ID

GET Get all customers

GET Get all customers

GET Get all order items

GET Get all order items

GET Get all products

GET Get all products

GET Get all sales facts

GET Get all sales facts

GET Get all sales orders

GET Get all sales orders

GET Get all sales representatives

GET Get all sales reps

GET Get an order item by ID

SalesAPI > Get all products > Console

Headers

+ Add header

NAME	VALUE	TYPE	DESCRIPTION
Product	None		

Apply product scope

Request URL

HTTP request

GET https://apim-poc-my.azure-api.net/v1/api/Products HTTP/1.1
Host: apim-poc-my.azure-api.net

HTTP response

Message Trace

HTTP/1.1 200 OK
content-encoding: gzip
content-type: application/json; charset=utf-8
date: Wed, 11 Feb 2026 23:48:52 GMT
request-context: appId=cid-v1:604f8a5d-e448-413e-93cc-6059076cc3ec
transfer-encoding: chunked
vary: Accept-Encoding,Origin
x-powered-by: ASP.NET

```
[{"productid": 1, "productname": "Chip-1", "productcategory": "ASIC", "processnodeid": 7, "packagetype": "QFN", "unitpriceusd": 455.00, "lifecyclestatus": "Active", "createdate": "2026-01-26T23:56:53.1254873", "orderitems": []}, {"productid": 2, "productname": "Chip-2", "productcategory": "FPGA", "processnodeid": 14, "packagetype": "MLCC", "unitpriceusd": 324.00, "lifecyclestatus": "Active", "createdate": "2026-01-26T23:56:53.1254873", "orderitems": []}, {"productid": 3,
```

Send Trace Bypass CORS proxy Timeout (in second)

5. Create MCP Server in APIM



- Expose Tools (API operations)
- Define policies for AI Agents to consume MCP Server

The screenshot shows the Microsoft Azure Policies blade for the 'sales-api-mcp' API Management service. The left sidebar includes links for Overview, MCP, Tools, Policies (which is selected), and Settings. The main area displays the following XML policy code:

```
1 <!--
2   - Policies are applied in the order they appear.
3   - Position <base/> inside a section to inherit policies from the outer scope.
4   - Comments within policies are not preserved.
5 -->
6 <!-- Add policies as children to the <inbound>, <outbound>, <backend>, and <on-error> elements -->
7 <policies>
8   <!-- Throttle, authorize, validate, cache, or transform the requests -->
9   <inbound>
10  </inbound>
11  <!-- Control if and how the requests are forwarded to services -->
12  <backend>
13    | <base />
14    </backend>
15    <!-- Customize the responses -->
16    <outbound>
17    </outbound>
18    <!-- Handle exceptions and customize error responses -->
19    <on-error>
20      | <base />
21    </on-error>
22 </policies>
```

The screenshot shows the Microsoft Azure MCP Servers blade for the 'sales-api-mcp' API Management service. The left sidebar includes links for Overview, MCP (selected), Tools, Policies, and Settings. The main area lists various API operations with their HTTP methods, URLs, and descriptions:

Operation	HTTP Method	URL	Description
Create a new customer	POST	/Customers	Create a new customer
Create a new order item	POST	/OrderItems	Create a new order item
Create a new product	POST	/Products	Create a new product
Create a new sales order	POST	/SalesOrders	Create a new sales order
Create a new sales representative	POST	/SalesReps	Create a new sales representative
Create a product	POST	/api/Products	Creates a new product record.
Create a sales order	POST	/api/SalesOrders	Creates a new sales order.
Create a sales rep	POST	/api/SalesReps	Creates a new sales representative record.
Create an order item	POST	/api/OrderItems	Creates a new order item.
Delete a customer	DELETE	/Customers/{id}	Delete a customer
Delete a product	DELETE	/Products/{id}	Delete a product
Delete a sales order	DELETE	/SalesOrders/{id}	Delete a sales order
Delete a sales representative	DELETE	/SalesReps/{id}	Delete a sales representative
Delete an order item	DELETE	/OrderItems/{id}	Delete an order item
Get a customer by ID	GET	/api/Customers/{id}	Retrieves a single customer by their unique identifier.
Get a product by ID	GET	/api/Products/{id}	Retrieves a single product by its unique identifier.
Get a sales order by ID	GET	/api/SalesOrders/{id}	Retrieves a single sales order by its unique identifier.
Get a sales rep by ID	GET	/api/SalesReps/{id}	Retrieves a single sales representative by their unique identifier.
Get all customers	GET	/Customers	Get all customers
Get all customers	GET	/api/Customers	Retrieves a list of all customers.
Get all order items	GET	/api/OrderItems	Retrieves a list of all order items.
Get all order items	GET	/OrderItems	Get all order items
Get all products	GET	/api/Products	Retrieves a list of all products.
Get all products	GET	/Products	Get all products
Get all sales facts	GET	/api/SalesFacts	Retrieves all aggregated sales fact records.
Get all sales facts	GET	/SalesFacts	Get all sales facts
Get all sales orders	GET	/api/SalesOrders	Retrieves a list of all sales orders.
Get all sales orders	GET	/SalesOrders	Get all sales orders
Get all sales representatives	GET	/SalesReps	Get all sales representatives
Get all sales reps	GET	/api/SalesReps	Retrieves a list of all sales representatives.
Get an order item by ID	GET	/api/OrderItems/{id}	Retrieves a single order item by its unique identifier.
Get customer by ID	GET	/Customers/{id}	Get customer by ID
Get order item by ID	GET	/OrderItems/{id}	Get order item by ID
Get order items by order ID	GET	/api/OrderItems/order/{orderId}	Retrieves all order items for a given order.
Get order items by order ID	GET	/OrderItems/order/{orderId}	Get order items by order ID
Get product by ID	GET	/Products/{id}	Get product by ID
Get products by category	GET	/api/Products/category/{category}	Retrieves all products in a given category.

6. API Center – Register APIM Instance



- Register APIM Instances for Dev, Test, Prod
- Review Portal to discover APIs & MCP Servers

- API Analysis / Linting

Microsoft Azure

Home > api-center-poc-my

api-center-poc-my | Environments

An environment represents a location where an API runtime is deployed, such as an API management platform or a Kubernetes cluster. You can

Search Overview Activity log Access control (IAM) Tags Resource visualizer Events Inventory Metadata Assets Environments (preview) Discovery MCP Platforms Integrations Copilot Studio (preview) Governance Authorization (preview) API Analysis API Center portal (preview) Settings Security Managed identities Automation Help

New environment Refresh

Environment title	Environment type	Server type
APIM Prod	Production	Azure API Management
APIM Dev	Development	Azure API Management

Microsoft Azure

API Analysis Report - Default

Title: SalesAPI API version: v1 Definition: Default

Lint issue summary: 190 total issues

Error (77): Action required. The error must be fixed for successful operation.

Warning (113): Recommendation. Consider revising for optimal performance and readability.

Information (0): Contains suggestions for improvement and notes on code practices not classified as errors or warnings.

```
1 {  
2   "openapi": "3.0.1",  
3   "info": {  
4     "title": "SalesAPI",  
5     "description": "Sales API for managing customers, products, orders, and sales representatives",  
6     "version": "v1"  
7   },  
8   "servers": [  
9     {  
10       "url": "http://apim-poc-my.azure-api.net/v1"  
11     },  
12     {  
13       "url": "https://apim-poc-my.azure-api.net/v1"  
14     }  
15   ],  
16   "paths": {  
17     "/customers": {  
18       "get": {  
19         "tags": [  
20           "customers"  
21         ],  
22         "summary": "Get all customers",  
23         "description": "Get all customers",  
24         "operationId": "Customers_GetAll",  
25         "responses": {  
26           "200": {  
27             "description": "Success",  
28             "content": {  
29               "application/json": {  
30                 "schema": {  
31                   "$ref": "#/components/schemas/CustomerArray"  
32                 }  
33               },  
34             },  
35             "example": [  
36               {  
37                 "customerId": 0,  
38                 "customerName": "string",  
39                 "customerType": "string",  
40                 "industry": "string",  
41                 "code": "string",  
42                 "state": "string",  
43                 "city": "string",  
44                 "annualRevenueLkd": 0,  
45                 "createdDate": "string"  
46               }  
47             ]  
48           }  
49         }  
50       }  
51     }  
52   }  
53 }
```

All issues (190) Error (77) Warning (113) Information (0)

Description	Line	Rule	Analyzer
Info object must have "contact" object.	3	info-contact	Spectral
Info object must have "license" object.	3	info-license	Spectral
License object must include "uri".	3	license-url	Spectral
* "createdDate" property must match format "date-time"	43	oas3-valid-media-example	Spectral
* "createdDate" property must match format "date-time"	73	oas3-valid-media-example	Spectral
* "createdDate" property must match format "date-time"	95	oas3-valid-media-example	Spectral
Parameter objects must have "description".	112	oas3-parameter-description	Spectral
* "createdDate" property must match format "date-time"	139	oas3-valid-media-example	Spectral
Parameter objects must have "description".	160	oas3-parameter-description	Spectral

7. AI Foundry – Build Sales Agent using MCP Server



- Create a new agent
- Add MCP Server as a Tool under Tools
- Provide clear instructions
- Add Guardrails (Responsible AI)

The screenshot shows the Microsoft Foundry interface for building an AI sales agent. The left sidebar lists various tools: Agents, Workflows, Models, Fine-tune, Tools, Knowledge, Data, Evaluations, and Guardrails. The main workspace is titled "arrow-sales-agent" and contains sections for "Playground", "Traces", "Monitor", and "Evaluation". In the "Playground" section, the model "gpt-4.1" is selected. The "Instructions" section provides the AI with instructions to act as an expert sales analytics assistant, with access to a SQL Server database containing sales data. The "Tools" section shows a connection to "sales-mcp" with the URL "https://apim-poc-my.azure-api.net/sales-api-mcp/mcp". The "Knowledge", "Memory", and "Guardrails" sections are also visible. On the right side, there is a "Chat" tab, "YAML" tab, and "Code" tab. A bar chart titled "Bar Chart (text representation)" displays the top 10 products by unit price, with data as follows:

Rank	Product Name	Unit Price (USD)
1	Chip-70	\$503.00
2	Chip-127	\$501.00
3	Chip-11	\$496.00
4	Chip-193	\$494.00
5	Chip-60	\$493.00
6	Chip-108	\$493.00
7	Chip-74	\$491.00
8	Chip-90	\$486.00
9	Chip-39	\$485.00
10	Chip-21	\$482.00

Below the chart, a message states: "If you need a visual chart image or want this exported for use in Excel or another tool, let me know!" The bottom status bar shows "gpt-4.1 8.4s 15987t" and "sales_mcp".

8. Build Frontend web App – Azure App Service



1. Go to Azure Portal -> Create “Web App” Service and fetch Uri (Identifier)
2. Fetch API URL from APIM that will be used in the Frontend Web App
3. Create a GitHub Repository (e.g. SalesPOC.UI) and clone it to your computer
4. Open Visual Studio Code (or Visual Studio) and leverage “Copilot Coding Agent” to generate the source code.

Example Prompt:

“Generate a web application using Angular and Typescript to display and filter data using this API End-Point {Paste Sales-API URL from APIM}.

Create a separate screen for Customers, Products, Sales Reps, Sales Orders, Sales Facts.

For each product in Products screen, add a button to show the Product description and Product documentation which includes 3 documents per product (Datasheet, Marketing Brief, Sales One Pager).

Finally, add a Sales Chat Assistant that uses sales-agent from AI Foundry.

AI Foundry Project Endpoint is {Paste Foundry Project Endpoint} and API Key is {Paste API Key}”

5. Run Code locally in browser to ensure that it works as expected or re-prompt Copilot to fix issues
6. Go to GitHub Repo online and ensure that GitHub Actions ran successfully and deployed App
7. Test Application

Sales POC Alternative Architecture – Data Platform

