



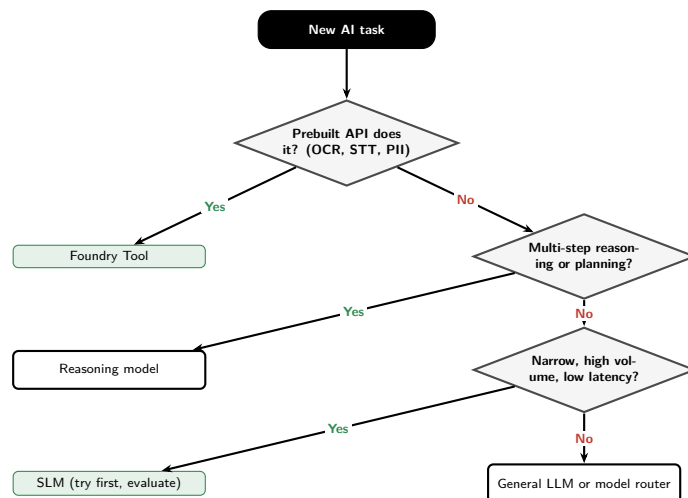
AI-103 D1: Choose Foundry Services and Set Up

AI-103 skills measured, Plan and manage an Azure AI solution (25-30 %), as of April 2026 | Choose models, Foundry services, retrieval, memory and tools; design the infrastructure; deployment types and configuration; Foundry projects in CI/CD

AI-103 Domain 1 Microsoft Foundry Deployment types CI/CD

Choose the model

Need	Model family (as of 2026, check the catalog)	Why
Chat, summarise, RAG answers at scale	GPT-4.1 / GPT-5 mini class	good quality per token, long context
Hard multi-step logic, planning, code review	reasoning models (o-series, GPT-5 with reasoning effort)	accuracy over latency; set reasoning_effort
Classification, extraction, routing, edge	SLMs: Phi-4-mini, gpt-4.1-nano / GPT-5 nano class	10x cheaper, fast, fine-tunable
Code generation and agents that edit code	code-tuned models (codex class)	trained for repositories and tools
Images and audio in the prompt	multimodal GPT-4o / GPT-4.1 / GPT-5, gpt-realtime for live voice	one model reads text, image, audio
Create images or video	gpt-image-1 family, Sora (video)	generation and editing (sheet 07)
Vectors for search and memory	text-embedding-3-small / -large	dimension vs cost trade-off
Open weights, data residency, fine-tune freedom	Llama, Mistral, DeepSeek, Phi via Foundry Models	serverless API or managed compute
Fixed task with a stable API (OCR, STT, translation, PII)	Foundry Tools	deterministic output, per-call price, no prompt



EXAM TIP

Least cost wording ("minimize cost", "high volume", "simple classification") → the smallest model that passes evaluation, or a Foundry Tool. "Complex reasoning", "multi-step math" → reasoning model. "Image in the prompt" → multimodal deployment, not Azure Vision.

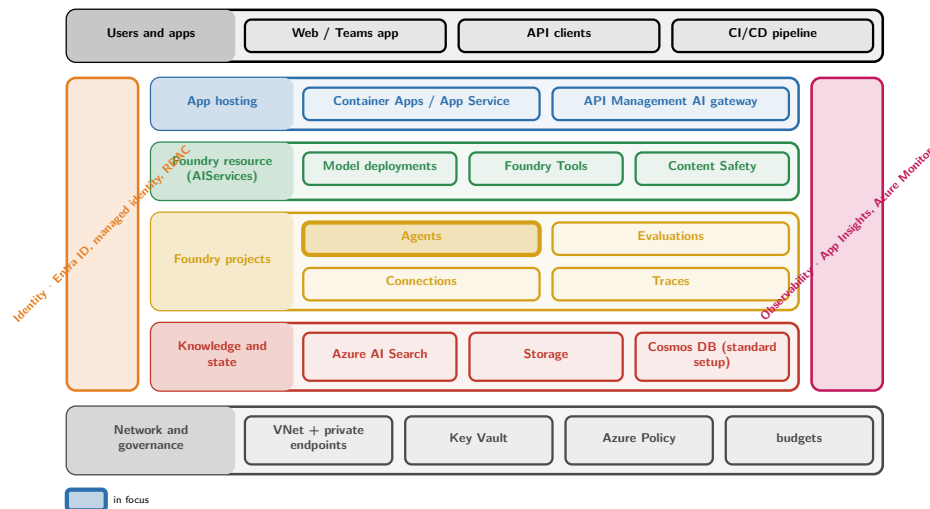
Choose services for grounding, retrieval, memory and tools

Need	Service	Pick it when
Enterprise knowledge, millions of chunks	Azure AI Search (hybrid + semantic ranker); Foundry IQ knowledge bases on top	security trimming, filters, many sources, agentic retrieval
A few files attached to one agent	file search tool (managed vector store)	no index to run; per-agent or per-conversation
Vectors next to operational data	Cosmos DB vector search, PostgreSQL pgvector	app data and embeddings in one store
Public, current facts	Grounding with Bing Search tool	news, prices; data leaves the compliance boundary
Microsoft 365 or Fabric data	SharePoint tool, Fabric data agent	respect user permissions on source
Multi-turn state	Foundry conversations (service-stored history)	you do not want to store chat history yourself
Long-term user memory	Foundry Agent Service memory (preview as of 2026) or your Cosmos DB store	preferences and facts across sessions
Call your code or APIs	function tool, OpenAPI tool, Azure Functions, MCP server, Logic Apps connectors	MCP to reuse one tool server across agents
Agent to agent across platforms	A2A (Agent2Agent) tool / endpoint	agent built elsewhere must be called
Documents, images, audio, video to fields	Azure Content Understanding	markdown + schema fields for RAG and agents

- Retrieval method keyword (BM25) for codes and names; vector for paraphrases; **hybrid + semantic ranker** as the default for RAG; agentic retrieval (LLM plans sub-queries) for complex questions over several knowledge sources
- Indexing method pull: indexer + skillset with integrated vectorization (chunk with Text Split, embed with Azure OpenAI embedding skill); push: your code calls the index API (any source, real time)

- **Chunking** start at about 512 tokens with 10-25 % overlap; layout-aware chunks (Content Understanding or Document Layout skill) keep tables and headings whole

Design the infrastructure



Resource vs project	one Foundry resource per environment and region holds deployments, quota and network settings; projects separate teams or apps, each with its own RBAC, agents, files and evaluations
Agent setup	basic (Microsoft-managed storage, fastest) vs standard (your Cosmos DB, Storage, AI Search; add VNet injection for private agents); choose standard when data must stay in your subscription
Hub-based project	older model on an Azure Machine Learning hub; needed only for features not yet on Foundry projects (as of 2026, check); new work uses Foundry projects
AI gateway	API Management in front of several deployments: azure-openai-token-limit policy per subscription key, backend pool with priority and circuit breaker for 429 failover, token metrics to App Insights
Regions	pick the region by model availability and quota; Data Zone or regional deployments when processing must stay in the EU or US

Deployment options and configuration

	Global / Data Zone Standard	Regional Standard	Recommended Provisioned (Global, Data Zone, Regional)	Global / Data Zone Batch	Serverless API / managed compute
Billing	per 1K tokens	per 1K tokens	PTU per hour, reservations	discounted per token	per token / per VM hour
Capacity	TPM quota, shared	TPM quota, lower	reserved	separate enqueued-token quota	per model
Latency	variable	variable	predictable	within 24 h target	variable
Data processing	any region / data zone	deployment region	per scope	per scope	region
Best for	default, spiky traffic	strict residency	steady high volume, SLA	offline bulk jobs	partner and open models

```
deploy.bicep: a chat deployment (capacity unit = 1,000 TPM)
resource dep 'Microsoft.CognitiveServices/accounts/deployments@2025-06-01' = {
  parent: foundry // kind: 'AIServices'
  name: 'chat'
  sku: { name: 'GlobalStandard', capacity: 110 }
  properties: {
    model: { format: 'OpenAI', name: 'gpt-4.1-mini', version: '2025-04-14' }
    versionUpgradeOption: 'OnceNewDefaultVersionAvailable'
    raiPolicyName: 'Microsoft.DefaultV2' // content filter
  }
}
```

Setting	Where	What to choose
Capacity (TPM)	deployment sku.capacity	peak RPM × (prompt + max_tokens); limited by regional quota per model
Version upgrade	versionUpgradeOption	OnceNewDefaultVersionAvailable, OnceCurrentVersionExpired, NoAutoUpgrade (pin for regulated apps, re-evaluate before moving)
Content filter	raiPolicyName	default or a custom guardrail policy (sheet 03)
Dynamic quota	deployment property	lets Standard deployments use spare capacity above quota when available
Agent version	agent definition	each change creates a version; publish a version as an agent application with a stable endpoint
Generation defaults	agent or request	temperature, top_p, max output tokens, tool_choice, reasoning effort

Y

A support chat peaks at 40 requests per minute. Prompts average 1,800 tokens and you set max_tokens = 800. How much TPM must the deployment get, and what RPM limit follows?

Given: R = 40 requests/min at peak; P = 1,800 prompt tokens per request; M = 800 max_tokens per request; RPM ratio: 6 requests/min per 1,000 TPM (chat models, as of 2026)

Y

Step 1: How many tokens does the limiter count per request?

$P + M = 1,800 + 800 = 2,600$
the rate limiter reserves max_tokens up front, not the real output

T

Y

Step 2: So what TPM do I need at peak?

$R \times 2,600 = 40 \times 2,600 = 104,000$ TPM
tokens per minute at peak

T

Y

Step 3: Which capacity do I set?

capacity = 110 (x 1,000 TPM)
about 5 % headroom

T

Y

Step 4: And the RPM limit?

$110 \times 6 = 660$ RPM
far above 40, tokens are the binding limit

T

Y

Step 5: What if I lower max_tokens to 400?

$40 \times 2,200 = 88,000$ TPM
same traffic, 15 % less quota

T

About 110K TPM (capacity 110), RPM limit 660; trim max_tokens to save quota

T

Check: If 429 errors appear below 104K real tokens, max_tokens is the hidden culprit

⑤ Cost of a commitment

WORKLOAD

85 % of 730 h = 620 h / month

ON-DEMAND

20.00 USD / h → 12,410 USD / month

CHEAPEST

PTU yearly reservation:
6,570.00 USD / month

Option	Terms	Monthly	Over 12 mo	Saving	Break-even
Standard (pay per token)	pay per hour	12,410 USD	148,920 USD	0 %	--
PTU hourly, no reservation	commit 100 % of hours	14,600 USD	175,200 USD	-18 %	100 %
PTU monthly reservation	—30 %, commit 100 % of hours, 1 mo	10,220 USD	122,640 USD	+18 %	70 %
PTU yearly reservation	—55 %, commit 100 % of hours, 12 mo	6,570.00 USD	78,840 USD	+47 %	45 %

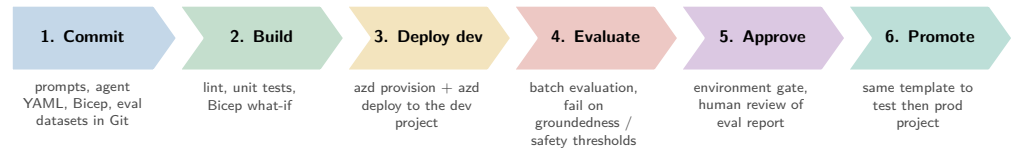
Monthly = rate × (1 − discount) × committed hours + rate × hours above the commitment + upfront ÷ term; pay as you go = rate × (1 − discount) × used hours ÷ (1 − interruption). Break-even utilisation = (1 − discount) × commitment + upfront ÷ (term × rate × hours): above it the commitment beats on-demand. Illustrative prices: check the provider's current price list.

Illustrative rates, not a quote: a workload worth 20 USD/h in tokens, busy 85 % of the month; reservations pay off only above their break-even utilisation

💡 Spillover

Combine a provisioned deployment for the base load with a Standard deployment for peaks: Foundry spillover or an APIM backend pool sends overflow (429) traffic to the Standard deployment instead of failing.

🔗 Foundry projects in CI/CD



```

.github/workflows/agent.yml (excerpt): OIDC, no secret
permissions: { id-token: write, contents: read }
steps:
- uses: actions/checkout@v4
- uses: azure/login@v2
  with: { client-id: ${vars.AZURE_CLIENT_ID},
        tenant-id: ${vars.AZURE_TENANT_ID},
        subscription-id: ${vars.AZURE_SUBSCRIPTION_ID} }
- run: azd provision --no-prompt && azd deploy --no-prompt
- run: python evals/run_eval.py --min-groundedness 4
  
```

MYTH

❌ One Foundry project for dev and prod is simpler

❌ Global Standard sends my stored data abroad

❌ PTU is always cheaper at scale

❌ Store the API key in a GitHub secret

FACT

✅ **Separate projects (often resources) per environment**
Different RBAC, quotas, network rules and data; promote by template, not by clicking.

✅ **Only processing may be global**
Data at rest stays in the resource geography; use Data Zone or regional types when processing must stay in a zone.

✅ **Only above break-even utilisation**
An idle PTU deployment bills every hour; spiky traffic fits Standard.

✅ **Use OIDC federation and a managed identity**
Workload identity federation removes long-lived secrets from pipelines.

Recap and self-test

Tick a box only when you could explain the point to someone else

☐ **Smallest model that passes evaluation**
Foundry Tool if a prebuilt API fits

☐ **Resource per environment, project per app or team**
RBAC at project scope

☐ **Deployment type = billing + data processing scope**
Standard, Provisioned, Batch; Global, Data Zone, Regional

☐ **Ship by template**
azd + Bicep + OIDC + evaluation gate

☐ **Hybrid + semantic ranker**
default retrieval for RAG

☐ **Standard agent setup**
when data must stay in your Cosmos DB, Storage, Search

☐ **TPM = RPM x (prompt + max_tokens)**
max_tokens counts against quota

Q1 Offline scoring of 2 million prompts by tomorrow at the lowest cost?

A Global (or Data Zone) Batch deployment: discounted, separate quota, 24-hour target.

Q2 Predictable latency for a steady 24/7 workload?

A Provisioned throughput (PTU), with a reservation if utilisation stays above break-even.

Q3 Processing must stay inside the EU but you want high quota?

A Data Zone Standard (EU data zone).

Q4 Which deployment property pins a model version?

A versionUpgradeOption: NoAutoUpgrade (re-evaluate before you change it).

Q5 Agent threads must be stored in your own Cosmos DB. Which setup?

A Standard agent setup (bring your own Cosmos DB, Storage and AI Search).

Q6 Reuse the same set of tools across many agents and frameworks?

A Expose them on an MCP server and add the MCP tool to each agent.

Q7 What does one capacity unit of a Standard deployment equal?

A 1,000 tokens per minute (TPM).

Q8 Tool that deploys a Foundry template's infrastructure and code in one command?

A Azure Developer CLI: azd up.

Q9 How does a GitHub workflow sign in to Azure without a secret?

A OIDC workload identity federation with azure/login and id-token: write.