Free checklist

Azure LLM production checklist

45 checks for taking a language model app from prototype to production on Azure, in 9 areas: scope, model endpoints, security, data, safety, reliability, monitoring, cost and releases.

1 Scope and success

- [ ] **Write down the one job the app does, and what a good answer looks like.**
  Without it, nobody can say whether a change made the app better or worse.
- [ ] **Build an evaluation set of 50 to 200 real questions with the answers you expect.**
  It's how you test every prompt, model and search change before it reaches users.
- [ ] **Decide what happens when the app can't answer: say so, or hand over to a person.**
  A confident wrong answer costs more trust than "I don't know."
- [ ] **Set targets for answer quality, response time (95th percentile) and cost per conversation.**
  Targets turn "is it ready?" into a yes or no.
- [ ] **Name an owner for the app in production.**
  Someone has to watch quality, cost and incidents after launch.

2 Model endpoints

- [ ] **Pin the model version in each deployment, and note its retirement date.**
  Azure retires model versions on a schedule; an automatic upgrade can change answers without warning.
- [ ] **Choose the deployment type on purpose: Standard, Global Standard, Data Zone or Provisioned throughput.**
  It decides where your data is processed, how much you can send and how you pay.
- [ ] **Request quota (tokens per minute) for your expected peak, with headroom, in every region you use.**
  Running out of quota means users get errors at your busiest hour.
- [ ] **Keep a second deployment, in another region or of another model, to fail over to.**
  A regional outage or a full quota shouldn't take the app down.
- [ ] **Keep development, test and production deployments separate, each with its own quota.**
  A load test or a runaway script in development can't starve production.

3 Identity, secrets and network

- [ ] **Call the model with a managed identity (Microsoft Entra ID), not an API key, and turn off key access where you can.**
  There's no key to leak, rotate or find in a log.
- [ ] **Give each part only the role it needs, such as Cognitive Services OpenAI User for the app.**
  A compromised component can then do little harm.
- [ ] **Keep any remaining secrets in Azure Key Vault, never in code or plain app settings.**
  Secrets in code end up in Git history and in every copy of the repository.
- [ ] **Put the model and search services behind private endpoints, with public network access turned off.**
  Only your app's network can reach them.
- [ ] **Deploy from CI with OpenID Connect (federated credentials) instead of a stored cloud secret.**
  No long-lived credential sits in GitHub waiting to be stolen.

4 Data and retrieval

- [ ] **Index your documents with a repeatable pipeline: splitting, embeddings and a refresh schedule.**
  Answers drift out of date when the index is updated by hand.
- [ ] **Filter search results by each user's permissions (security trimming).**
  Otherwise the app can quote documents the user isn't allowed to read.
- [ ] **Use hybrid search (keywords and vectors) and measure it on your evaluation set.**
  Most wrong answers start with the wrong passages, not the wrong model.
- [ ] **Show the sources behind each answer.**
  Users can check the answer, and you can trace a bad one back to its passage.
- [ ] **Remove or mask personal data you don't need before it's indexed.**
  Data that isn't in the index can't leak through an answer.

5 Safety

- [ ] **Keep content filtering on, with levels set for your users.**
  It blocks harmful requests and answers before they reach anyone.
- [ ] **Turn on Prompt Shields to catch prompt injection in user messages and in retrieved documents.**
  A sentence hidden in a document can try to give the model new instructions.
- [ ] **Treat model output as untrusted: never run it as code or SQL, or pass it to a tool unchecked.**
  A model can be talked into writing something harmful.
- [ ] **Tell users they're talking to AI, and how to reach a person.**
  It sets expectations and gives them a way out.
- [ ] **Check what Azure keeps for abuse monitoring, and whether your data needs modified abuse monitoring.**
  Some contracts and regulations don't allow prompts to be stored, even briefly.

6 Reliability and scale

- [ ] **Put Azure API Management in front of the model as an AI gateway: one endpoint, token limits per client, load balancing and failover.**
  One busy client can't use up everyone's quota, and a failing deployment is skipped.
- [ ] **Retry rate-limit (429) and server errors with exponential backoff, and respect the retry-after header.**
  Most of these errors pass in seconds; hammering makes them last longer.
- [ ] **Set timeouts, and stream long answers so users see progress.**
  A request that hangs looks like a broken app.
- [ ] **Load-test at your expected peak before launch, and check you stay within quota.**
  It's cheaper to find the limit in a test than in front of customers.
- [ ] **Cache answers to common questions where that's safe.**
  Repeated questions cost nothing and come back instantly.

7 Monitoring

- [ ] **Send logs, metrics and traces to Application Insights, with one request ID across every service.**
  You can follow a single bad answer from the user's click to the model call.
- [ ] **Record tokens used per request and per feature.**
  You can see what each feature costs and spot a prompt that suddenly grew.
- [ ] **Alert on error rate, rate limits, 95th-percentile response time and daily cost.**
  You hear about a problem before your users do.
- [ ] **Don't log full prompts and answers with personal data by default; if you must, mask them and keep them briefly.**
  Logs are copied widely and kept for a long time.
- [ ] **Collect user feedback (thumbs up or down) and review the worst answers every week.**
  Real failures are the best additions to your evaluation set.

8 Cost

- [ ] **Set budgets and alerts in Azure Cost Management for each environment.**
  A loop that calls the model all night shows up as an alert, not a surprise bill.
- [ ] **Work out the cost per conversation from real token counts, and recheck it after every change.**
  A longer prompt or a new model can double the bill quietly.
- [ ] **Use a smaller model for simple steps, such as routing or classifying, and the larger one only where it helps.**
  Small models are often a tenth of the price and faster.
- [ ] **Compare pay-as-you-go with Provisioned throughput once your traffic is steady.**
  Reserved capacity can cost less at a steady high volume, and more at a low one.
- [ ] **Scale down development and test resources when they're not in use.**
  Idle environments can cost as much as production.

9 Releases and change

- [ ] **Define all infrastructure as code (Bicep or Terraform), reviewed in pull requests.**
  You can rebuild any environment, and every change has a record.
- [ ] **Keep prompts, model versions and search settings in version control, and release them like code.**
  A prompt change is a code change: it can break things just as easily.
- [ ] **Run the evaluation set in CI on every change to prompts, models or search, and stop the release if quality drops.**
  Quality regressions are caught before users see them.
- [ ] **Roll out gradually, to a pilot group or a share of traffic first, with a quick way back.**
  A bad release reaches a few people for a few minutes, not everyone.
- [ ] **Write a short runbook: who's on call, how to roll back and how to switch to the fallback deployment.**
  At 2 a.m. nobody should have to work it out from scratch.

Want it checked for you?

The one-week AI infrastructure audit goes through your setup against this list and more, and gives you a report with what to fix first and how. It starts from $500 (₹25,000 in India).

See it at sayhellomodel.com/services/

Last reviewed November 2026. Azure renames services and changes limits often, so check names and numbers against Microsoft's documentation before you rely on them. From sayhellomodel.com/services/llm-production-checklist/