

Guia de preparação AI-200

Tudo o que a prova cobra, organizado por domínio: conceitos-chave, comparações que confundem, pegadinhas reais, gatilhos do enunciado, questões comentadas e um plano para chegar pronto ao exame.

4 capítulos por domínio	**28** questões comentadas	**70%** nota de corte
6 provas completas no curso		

ANTES DE COMEÇAR

Como usar este guia

Este guia foi feito para ser estudado junto com as 6 provas completas do curso AI-200: o guia explica o que cai e onde estão as armadilhas; as provas treinam o ritmo e mostram o que você ainda não domina.

Quando	O que fazer
Semana 1	Faça o teste diagnóstico (página seguinte) e leia os capítulos 1 e 2. No fim da semana, faça a Prova 1 do curso no tempo do exame.
Semana 2	Leia os demais capítulos, começando pelos domínios em que você errou mais no diagnóstico. Faça as provas 2 e 3.
Semana 3	Releia os alertas de pegadinha e as tabelas de comparação. Faça as provas 4 e 5 e revise cada erro pela explicação.
Semana 4	Estude a folha de cola, faça a Prova 6 e refaça as provas abaixo de 80%. Agende o exame quando estiver acima de 80% em todas (a nota de corte é 70%).

Mapa do exame

Cada prova do curso tem 50 questões e 100 minutos, no formato do exame. Referência de aprovação: 70%.

Domínio (capítulo)	Peso	Neste guia
1. Conectar e consumir serviços do Azure	20-25%	18 conceitos · 14 pegadinhas
2. Desenvolver soluções de IA com serviços de dados do Azure	25-30%	18 conceitos · 14 pegadinhas
3. Desenvolver soluções em contêineres no Azure	20-25%	18 conceitos · 14 pegadinhas
4. Proteger, monitorar e solucionar problemas de soluções do Azure	20-25%	18 conceitos · 14 pegadinhas

Como ler os capítulos: os **tópicos-chave** são o que mais cai; a **tabela de memorização** separa o que a prova costuma confundir; os **alertas de prova** mostram as armadilhas das alternativas erradas; os **gatilhos** ligam palavras do enunciado à resposta; e o **checklist** diz se você está pronto para o domínio.

Estude um pouco todo dia, de graça

Instagram @strecht.simulados

Um post de prova por dia: questão comentada, pegadinhas e o "não confunda" de cada certificação.

Seguir no Instagram →

YouTube Strecht Simulados

Shorts diários com pegadinhas de prova e, aos sábados, um simulado comentado.

Inscrever-se no canal →

DIAGNÓSTICO

Você já sabe isso?

Responda às 12 questões sem consultar nada. As respostas, com a explicação e o capítulo para revisar, estão no fim do guia. Comece o estudo pelos capítulos em que errou.

D1 · Conectar e consumir serviços do Azure

Um aplicativo publica eventos próprios em um tópico personalizado do Event Grid e deve autenticar sem chave de acesso. O que configurar?

A. Chave de acesso no código
B. Assinatura com webhook anônimo
C. Acesso público sem autenticação
D. Função EventGrid Data Sender

D2 · Conectar e consumir serviços do Azure

Uma mensagem do Service Bus não pode ser perdida se o consumidor falhar durante o processamento. Qual modo de recebimento usar?

A. Recebimento em lote sem conclusão
B. PeekLock com a conclusão após processar
C. ReceiveAndDelete com leitura rápida
D. Leitura com peek sem bloqueio

D3 · Conectar e consumir serviços do Azure

Eventos de um mesmo dispositivo precisam ser processados em ordem no Event Hubs. Como enviar os eventos?

A. ID do dispositivo como chave de partição
B. Sem chave, distribuídos entre as partições
C. Com a captura habilitada no namespace
D. Com um grupo de consumidores por evento

D4 · Desenvolver soluções de IA com serviços de dados do Azure

As respostas do modelo estão sendo cortadas no meio do texto. O que ajustar na chamada?

A. A região do recurso
B. A temperatura da chamada
C. O limite de tokens de saída
D. O nome da implantação

D5 · Desenvolver soluções de IA com serviços de dados do Azure

A indexação no Azure AI Search falha porque o tamanho dos vetores não confere com o campo do índice. O que corrigir?

A. O analisador de idioma do campo
B. A camada do serviço de busca
C. A chave de API de consulta
D. A dimensão do campo vetorial

D6 · Desenvolver soluções de IA com serviços de dados do Azure

Um contêiner do Cosmos DB terá milhões de vetores e precisa de buscas rápidas com baixo consumo de RUs. Qual tipo de índice vetorial usar?

A. Índice flat
B. Sem índice nos vetores
C. Índice diskANN
D. Índice de intervalo em texto

D7 · Desenvolver soluções em contêineres no Azure

Em um ambiente do Container Apps configurado com Log Analytics, onde ficam os logs de console dos aplicativos?

A. No registro de contêineres
B. No log de atividades da assinatura
C. Na conta de armazenamento do app
D. Tabela ContainerAppConsoleLogs

D8 · Desenvolver soluções em contêineres no Azure

Variáveis sensíveis passadas a contêineres do Azure Container Instances não podem aparecer nas propriedades do recurso. O que usar?

A. Rótulos no grupo de contêineres
B. Variáveis de ambiente comuns
C. Porta extra exposta no grupo
D. Variáveis de ambiente seguras

D9 · Desenvolver soluções em contêineres no Azure

Qual comando implanta uma nova imagem em um Container App já existente?

A. az acr login no registro
B. az containerapp update
C. az acr build com --image
D. az containerapp env create

D10 · Proteger, monitorar e solucionar problemas de soluções do Azure

As permissões de acesso ao Key Vault precisam existir antes da criação do Container App, para que ele já inicie funcionando. Que identidade usar?

A. Conta de usuário do desenvolvedor
B. Identidade atribuída pelo sistema
C. Identidade atribuída pelo usuário
D. Chave de acesso do cofre

D11 · Proteger, monitorar e solucionar problemas de soluções do Azure

Um aplicativo precisa ler e gravar blobs em um único contêiner, com o menor privilégio. Qual função atribuir à identidade?

A. Storage Blob Data Reader na conta
B. Colaborador da conta de armazenamento
C. Proprietário na conta de armazenamento
D. Storage Blob Data Contributor no contêiner

D12 · Proteger, monitorar e solucionar problemas de soluções do Azure

Um app em ambiente do Container Apps com VNet não resolve o nome do ponto privado do banco de dados. O que falta?

A. Ativar a afinidade de sessão
B. Criar um domínio personalizado
C. Vincular a zona DNS privada à VNet
D. Aumentar as réplicas do app

01

CAPÍTULO 1 · 20-25% DA PROVA

Conectar e consumir serviços do Azure

Aqui eu preciso escolher o serviço de mensageria certo (Event Grid, Event Hubs, Service Bus, filas do Armazenamento) e conhecer seus detalhes operacionais, além de Azure Functions (gatilhos, associações, Durable Functions) e API Management, inclusive as políticas de IA para o Azure OpenAI. A prova descreve um requisito como ordem, não perder mensagem, processar no futuro, reprocessar ou limitar tokens por cliente, e pede o recurso exato. Onde mais erro é misturar sessões do Service Bus com chave de partição do Event Hubs, PeekLock com ReceiveAndDelete e versão com revisão do APIM. A captura do Event Hubs e o grupo de consumidores aparecem como distratores em quase todo cenário de ordem e posição. Também caem padrões do Durable Functions e a regra de que o orquestrador precisa ser determinístico.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Azure Event Grid

Entrega eventos discretos de mudança de estado por push a vários assinantes, com filtros de tipo de evento, de assunto (prefixo e sufixo) e avançados. Aceita o esquema próprio e o CloudEvents 1.0, recomendado para interoperabilidade. Manipuladores incluem Service Bus, filas do Armazenamento, Event Hubs, Functions e webhooks; publicar sem chave usa a função EventGrid Data Sender.

Validação de webhook do Event Grid

Ao criar uma assinatura para um webhook próprio, o ponto de extremidade precisa responder ao evento de validação com o código recebido ou pela URL de validação; sem isso, a criação falha. O gatilho do Event Grid no Functions faz essa validação automaticamente e aceita os esquemas do Event Grid e CloudEvents.

Partições e chave de partição do Event Hubs

A ordem é garantida dentro de uma partição, e a chave de partição (como o ID do dispositivo) mantém eventos relacionados juntos. O número de partições define o paralelismo máximo de leitores em um grupo de consumidores e, em camadas como Standard, não pode ser reduzido depois. O envio em lote com EventDataBatch reduz chamadas de rede respeitando o tamanho máximo.

Grupos de consumidores, checkpoints e retenção

Grupos de consumidores são visões independentes do fluxo: um por aplicativo para cada um ler no seu ritmo. O EventProcessorClient usa armazenamento de checkpoints para retomar de onde parou e dividir partições entre instâncias; checkpoints em lote aceleram, com risco de reprocessamento. Eventos ficam retidos pelo período configurado e podem ser relidos a partir de uma posição.

Capacidade e esquemas do Event Hubs

O nível Standard usa unidades de taxa de transferência, com expansão automática opcional; ingestão limitada em picos pede mais unidades. O Registro de Esquemas fica no namespace e aceita Avro e JSON Schema com regras de compatibilidade. A captura grava arquivos (Avro) em blobs e não controla ordem nem consumidores.

Modos de recebimento do Service Bus

No PeekLock, a mensagem é bloqueada e só sai da fila após a conclusão; se o bloqueio expirar ou ela for abandonada, volta à fila e pode ir para a DLQ após várias tentativas. No ReceiveAndDelete, ela é removida ao ser lida. A operação de peek só inspeciona, sem remover nem bloquear; processamento longo pede renovação automática do bloqueio.

Fila de mensagens mortas (DLQ)

Mensagens que passaram do número máximo de entregas vão para a subfila de mensagens mortas, que pode ser lida e reenviada após corrigir o problema. Com TTL e a opção de mensagens mortas na expiração, mensagens não processadas no prazo vão para análise; sem essa opção, as expiradas são descartadas.

Sessões, tópicos e filtros do Service Bus

Sessões com o mesmo SessionId bloqueiam um grupo de mensagens para um consumidor e mantêm a ordem FIFO. Assinaturas de tópicos usam filtros SQL e de correlação sobre propriedades; uma assinatura nova recebe tudo por causa da regra padrão, que deve ser removida para filtrar. O nível Basic não oferece tópicos.

Recursos avançados do Service Bus

Mensagens agendadas (ScheduledEnqueueTime) só ficam disponíveis no horário definido e podem ser canceladas antes. Transações tornam atômicas operações como concluir uma mensagem e enviar outra. Encaminhamento automático move mensagens entre entidades do mesmo namespace, pré-busca aumenta a vazão com mensagens pequenas e detecção de duplicatas remove repetidas.

Filas do Armazenamento

O limite de mensagem é 64 KB; conteúdo maior pede Service Bus ou o padrão Claim-Check com blob. O tempo limite de visibilidade esconde a mensagem durante o processamento e pode ser estendido. No gatilho do Functions, a mensagem que falha repetidamente vai para a fila -poison, com número máximo de tentativas configurável no host.json.

Gatilhos do Azure Functions

O gatilho de blob com origem no Event Grid tem menor latência e é o recomendado no Flex Consumption. O temporizador usa expressões NCRONTAB com seis campos, incluindo segundos. O gatilho do Cosmos DB usa o feed de alterações e um contêiner de concessões; o de fila tem o batchSize no host.json para reduzir paralelismo.

Gatilho HTTP do Functions

O nível de autorização function exige a chave da função na chamada, enquanto anonymous não exige chave; para usuários, prefira autenticação do App Service ou API Management. O streaming HTTP envia a resposta do modelo em partes e está disponível em modelos de programação como .NET isolado, Python e Node.js.

Associações e conexões baseadas em identidade

Associações de entrada e saída (Cosmos DB, blob, fila, SignalR) reduzem código de conexão; a saída do SignalR Service no modo sem servidor envia notificações em tempo real. Conexões baseadas em identidade usam configurações como __serviceUri ou __fullyQualifiedNamespace, sem chave.

Escala e resiliência do Functions

Os planos Flex Consumption e Premium permitem limitar instâncias para proteger dependências. Políticas de repetição valem para gatilhos como Event Hubs e Cosmos DB, repetindo falhas transitórias antes de avançar o checkpoint. Como a entrega costuma ser pelo menos uma vez, manipuladores devem ser idempotentes.

Durable Functions

Padrões: encadeamento (etapas em sequência com estado), fan-out/fan-in (paralelo), API HTTP assíncrona (resposta imediata e URL de status), monitoramento (temporizadores duráveis que não consomem recursos na espera) e interação humana (WaitForExternalEvent com temporizador durável); entidades guardam estado de objetos. O orquestrador reproduz o histórico e precisa ser determinístico: E/S, números aleatórios e hora atual vão para atividades ou usam a hora da orquestração.

API Management: produtos, assinaturas, versões e importação

Produtos agrupam APIs e definem assinatura, aprovação e políticas como quota e limite de taxa; a chave vai no cabeçalho Ocp-Apim-Subscription-Key ou no parâmetro subscription-key. Versões separam mudanças incompatíveis; revisões testam mudanças compatíveis. O APIM importa OpenAPI, WSDL, GraphQL e APIs do Azure, como Functions, Container Apps e Azure OpenAI.

Políticas do API Management

set-header com valor nomeado que referencia o Key Vault envia segredo ao back-end sem expor ao cliente. Cache (interno ou externo compatível com Redis), retry (contagem, intervalo e condição como 429 ou 503), mock-response (exemplos da especificação) e transformações json-to-xml, xml-to-json, set-body e find-and-replace são as mais cobradas.

Políticas de IA no API Management

authentication-managed-identity chama o Azure OpenAI sem chave, com função de usuário no recurso. llm-token-limit (ou azure-openai-token-limit) limita tokens por chave e retorna 429; emit-token-metric envia consumo por cliente ao Application Insights. Pools de back-ends aceitam round-robin, peso e prioridade, com disjuntor para 429, permitindo transbordo de PTU para Standard.

Macetes para memorizar

💡 Mensageria

Grid é notícia (evento discreto por push); Hubs é rio (fluxo de telemetria); Bus é carta registrada (comando de alto valor).

💡 Modos do Service Bus

PeekLock espia, trava e só apaga quando você confirma; ReceiveAndDelete leu, sumiu; peek só olha.

💡 Event Hubs

Partição dá ordem e paralelismo; grupo dá independência; checkpoint dá memória; captura só grava arquivo.

💡 APIM versão x revisão

Versão quebra contrato; revisão conserta sem quebrar.

💡 Durable Functions

Corrente (encadeamento), leque (fan-out/fan-in), recibo (API HTTP assíncrona), ronda (monitoramento) e assinatura (interação humana).

💡 Filas do Armazenamento

64 KB, visibilidade e -poison: os três números e nomes que caem.

💡 Políticas de IA no APIM

Identidade para entrar, limit para frear, metric para medir, pool para distribuir.

Tabela de memorização: não confunda

Quando é...	...e quando é
PeekLock A mensagem não pode ser perdida se o consumidor falhar; conclua só depois de processar.	**ReceiveAndDelete** Leitura rápida em que perder a mensagem em falha é aceitável; ela é removida ao ser lida.

Sessões do Service Bus Mensagens de um mesmo pedido em ordem, por um consumidor de cada vez, com SessionId.	**Chave de partição do Event Hubs** Eventos de um mesmo dispositivo em ordem em um fluxo de telemetria.
Azure Event Grid Eventos discretos de mudança de estado entregues por push a vários assinantes com filtros.	**Azure Service Bus** Comandos de alto valor, como pagamentos, que exigem entrega confiável, DLQ e transações.
Nova versão da API Mudança incompatível publicada mantendo a versão antiga para os clientes atuais.	**Nova revisão da API** Mudança compatível testada antes de virar a revisão atual.
Grupo de consumidores Aplicativos independentes leem os mesmos eventos, cada um no seu ritmo.	**Partições** Definir ordem por chave e o paralelismo máximo de leitores dentro de um grupo.
Mensagem agendada Colocar na fila agora para ficar disponível só no futuro, como daqui a 24 horas.	**TTL com mensagens mortas na expiração** Mensagens não processadas dentro do prazo devem ir para análise em vez de ficar paradas.
Monitoramento (Durable) Verificar periodicamente o status de um trabalho externo até concluir.	**Interação humana (Durable)** Esperar uma aprovação com prazo e seguir outro caminho se ela não vier.

Números que caem

Item	Valor
Tamanho máximo de mensagem na fila do Armazenamento	**64 KB**
Janela de conclusão da API de lote do Azure OpenAI	**até 24 horas**
Campos da expressão NCRONTAB do gatilho de temporizador	**6 (inclui segundos)**
Versão do esquema CloudEvents aceito pelo Event Grid	**1.0**
Código retornado pela política de limite de tokens ao estourar	**429**
Padrões do Durable Functions	**5: encadeamento, fan-out/fan-in, API HTTP assíncrona, monitoramento e interação humana**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Usar ReceiveAndDelete com leitura rápida para não perder mensagens.
Como não cair: A mensagem é removida ao ser lida e se perde na falha. Use PeekLock e conclua após processar.

⚠ **Alerta de prova:** Receber com PeekLock só para inspecionar mensagens.
Como não cair: PeekLock bloqueia a mensagem. Para inspecionar sem remover nem bloquear, use peek.

⚠ **Alerta de prova:** Habilitar a captura do Event Hubs para manter ordem, separar leitores ou retomar o consumo.
Como não cair: A captura só grava arquivos em blobs. Ordem é chave de partição, leitores independentes são grupos de consumidores e retomar é EventProcessorClient com checkpoints.

⚠ **Alerta de prova:** Criar um grupo de consumidores por evento para garantir ordem ou guardar posição.
Como não cair: Grupos não definem ordem nem guardam posição; são visões independentes do fluxo, uma por aplicativo.

⚠ **Alerta de prova:** Criar mais filas do Armazenamento ou mudar a conta para GRS para caber mensagens de 200 KB.
Como não cair: O limite de 64 KB é por mensagem. Use Service Bus ou Claim-Check com blob.

⚠ **Alerta de prova:** Publicar uma mudança incompatível como nova revisão da mesma versão.
Como não cair: Revisões são para mudanças compatíveis. Mudança incompatível é nova versão.

⚠ **Alerta de prova:** Criar uma fila separada por região para entregar só pedidos da região Sul.
Como não cair: Isso obriga o remetente a escolher a fila. Use filtro na assinatura do tópico pela propriedade da mensagem.

⚠ **Alerta de prova:** Filtrar no código da função os eventos que não são .pdf.
Como não cair: Funciona, mas gera execuções desnecessárias. Use filtro de assunto com sufixo .pdf no Event Grid.

⚠ **Alerta de prova:** Usar filtro de sufixo no assunto para receber só criação de blobs.
Como não cair: Sufixo filtra nomes, não tipos. Use filtro de tipo de evento BlobCreated.

⚠ **Alerta de prova:** Manter uma função HTTP aguardando o fim de um processo longo ou de uma aprovação.
Como não cair: Ela estoura o tempo limite. Use API HTTP assíncrona do Durable ou evento externo com temporizador durável.

⚠ **Alerta de prova:** Usar data e hora atual, números aleatórios ou chamadas HTTP diretas no orquestrador.
Como não cair: O orquestrador reproduz o histórico e precisa ser determinístico. Leve E/S para atividades e use a hora da orquestração.

⚠ **Alerta de prova:** Desligar as repetições de entrega para evitar eventos duplicados.
Como não cair: Isso pode perder eventos. Torne o manipulador idempotente.

⚠ **Alerta de prova:** Limitar tokens por cliente com restrição por IP, cache ou produto sem chave de assinatura.
Como não cair: Nenhum deles conta tokens. Use a política llm-token-limit (ou azure-openai-token-limit).

⚠ **Alerta de prova:** Aumentar max_tokens para a resposta aparecer aos poucos na interface.
Como não cair: max_tokens só permite respostas maiores. Habilite stream na chamada do chat.

Como cai na prova

📝 **Como cai:** Escolher entre Event Grid, Event Hubs, Service Bus e filas do Armazenamento.
Como resolver: Pergunte o que trafega: evento discreto por push vai ao Event Grid; fluxo de telemetria em volume, ao Event Hubs; comando de negócio com DLQ, sessões e transações, ao Service Bus; mensagem simples até 64 KB, à fila do Armazenamento.

📝 **Como cai:** Problema operacional no Service Bus: perda, ordem, prazo, bloqueio, vazão.
Como resolver: Perda: PeekLock. Ordem: sessões. Futuro: mensagem agendada. Prazo vencido para análise: TTL com mensagens mortas na expiração. Bloqueio expira: renovação automática. Vazão com mensagens pequenas: pré-busca. Atomicidade: transação.

📝 **Como cai:** Fluxo com estado no Durable Functions.
Como resolver: Etapas dependentes em sequência: encadeamento. Paralelo: fan-out/fan-in. Processo longo por HTTP: API assíncrona. Verificação periódica: monitoramento. Aprovação com prazo: evento externo com temporizador. Lembre que o orquestrador é determinístico.

📝 **Como cai:** Requisito no API Management.
Como resolver: Mapeie o verbo do enunciado para a política: autenticar no back-end sem chave (authentication-managed-identity), esconder segredo (set-header com valor nomeado), reaproveitar (cache), repetir (retry), simular (mock-response), converter (json-to-xml), limitar tokens (llm-token-limit), medir tokens (emit-token-metric), distribuir (pool de back-ends).

📝 **Como cai:** Escolher gatilho ou associação do Functions.
Como resolver: Blob com baixa latência: gatilho de blob com origem no Event Grid. Horário fixo: temporizador NCRONTAB. Mudança no Cosmos DB: gatilho com contêiner de concessões. Gravar sem código de conexão: associação de saída. Tempo real para clientes web: saída do SignalR.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
eventos discretos, por push, para vários assinantes com filtros	**Azure Event Grid**
entrega confiável, fila de mensagens mortas e transações	**Azure Service Bus**
eventos de um mesmo dispositivo em ordem no Event Hubs	**ID do dispositivo como chave de partição**
mensagens de um mesmo pedido em ordem, um consumidor por vez	**Sessões com SessionId**
colocar na fila agora e processar só daqui a 24 horas	**Mensagem agendada**
concluir a mensagem recebida e enviar outra de forma atômica	**Transação do Service Bus**
processamento de cerca de 10 minutos e o bloqueio expira antes	**Renovar o bloqueio automaticamente**
mensagem da fila do Armazenamento reaparece para outro consumidor durante o processamento	**Tempo limite de visibilidade**
esperar aprovação do gestor por até 72 horas e seguir outro caminho	**Evento externo com temporizador durável**
verificar a cada cinco minutos o status até concluir, sem servidor	**Padrão de monitoramento do Durable**
responder logo ao cliente e permitir consultar o status depois	**API HTTP assíncrona (Durable)**
parceiros com chaves e cotas diferentes para as mesmas APIs	**Produtos com assinaturas próprias**
um milhão de solicitações ao modelo sem urgência e com menor custo	**API de lote do Azure OpenAI**
usar primeiro a implantação provisionada e transbordar para Standard	**Pool de back-ends com prioridades e disjuntor**

🎯 **Se você lembrar de só uma coisa deste capítulo...**

- Ordem no Event Hubs é por partição; no Service Bus, por sessão.
- Captura do Event Hubs só grava arquivos; não controla ordem, posição nem leitores.
- Uma assinatura nova de tópico recebe tudo por causa da regra padrão.
- O orquestrador do Durable Functions precisa ser determinístico.

Essenciais para a prova (checklist)

☐ Sei escolher entre Event Grid, Event Hubs, Service Bus e filas do Armazenamento pelo tipo de carga.

☐ Sei configurar filtros de tipo, assunto e avançados no Event Grid e por que a validação de webhook falha.

☐ Sei usar chave de partição, grupos de consumidores, checkpoints, retenção e unidades de taxa no Event Hubs.

☐ Sei a diferença entre PeekLock, ReceiveAndDelete e peek e quando a mensagem vai para a DLQ.

☐ Sei usar sessões, filtros de assinatura, mensagens agendadas, TTL, transações, encaminhamento e pré-busca.

☐ Sei escolher o gatilho e as associações do Functions e configurar conexões baseadas em identidade.

☐ Sei identificar cada padrão do Durable Functions e as restrições do orquestrador.

☐ Sei usar produtos, assinaturas, versões, revisões e as principais políticas do API Management.

☐ Sei aplicar as políticas de IA do APIM: identidade gerenciada, limite e métrica de tokens, cache semântico e pool de back-ends.

☐ Sei consumir o modelo com streaming, chamadas paralelas de função, embeddings em lista e API de lote.

Questões de revisão do capítulo

R1.1 · Conectar e consumir serviços do Azure

Uma função deve receber eventos do Event Grid com a validação da assinatura feita automaticamente. Qual gatilho usar?

A. Gatilho de temporizador
B. Gatilho do Event Grid na função
C. Associação de saída de blob
D. Gatilho de fila do Armazenamento

R1.2 · Conectar e consumir serviços do Azure

Os eventos publicados pela empresa devem seguir uma especificação aberta, para facilitar a integração com outras plataformas. Qual esquema usar no Event Grid?

A. Formato CSV simples
B. Esquema de log de atividades
C. Mensagens AMQP brutas
D. CloudEvents 1.0

R1.3 · Conectar e consumir serviços do Azure

Qual cabeçalho padrão envia a chave de assinatura ao API Management?

A. Authorization: Basic
B. Ocp-Apim-Subscription-Key
C. Content-Type
D. X-Api-Version

R1.4 · Conectar e consumir serviços do Azure

Um lembrete deve ser colocado na fila agora, mas só ficar disponível para processamento daqui a 24 horas. O que usar no Service Bus?

A. Fila de mensagens mortas
B. Sessão por usuário
C. Mensagem agendada
D. Detecção de duplicatas

Respostas comentadas

R1.1. B) Gatilho do Event Grid na função

Por quê: O gatilho do Event Grid aceita os esquemas do Event Grid e CloudEvents.

✗ A) Incorreta. Não recebe eventos.
✗ C) Incorreta. Grava arquivos.
✗ D) Incorreta. Recebe mensagens de fila.

> **Dica de prova:** Eventos na função: gatilho do Event Grid.

R1.2. D) CloudEvents 1.0

Por quê: O Event Grid aceita o esquema próprio e o CloudEvents, recomendado para interoperabilidade.

✗ A) Incorreta. Não é esquema de eventos.

✗ B) Incorreta. É específico do Azure Monitor.

✗ C) Incorreta. É protocolo, não esquema de eventos.

> **Dica de prova:** Eventos em formato aberto: CloudEvents.

R1.3. B) Ocp-Apim-Subscription-Key

Por quê: Também é possível enviar a chave pelo parâmetro de consulta subscription-key.

✗ A) Incorreta. É usado em autenticação básica.

✗ C) Incorreta. Indica o formato do corpo.

✗ D) Incorreta. Não é o cabeçalho da chave.

> **Dica de prova:** Chave do APIM: Ocp-Apim-Subscription-Key.

R1.4. C) Mensagem agendada

Por quê: Mensagens agendadas usam ScheduledEnqueueTime e podem ser canceladas antes da hora.

✗ A) Incorreta. Guarda falhas.

✗ B) Incorreta. Trata ordem.

✗ D) Incorreta. Remove repetidas.

> **Dica de prova:** Processar no futuro: mensagem agendada.

02

CAPÍTULO 2 · 25-30% DA PROVA

Desenvolver soluções de IA com serviços de dados do Azure

Este é o domínio de maior peso, e eu preciso dominar o RAG inteiro: divisão em blocos, embeddings, índice do Azure AI Search, busca híbrida, classificador semântico, filtros e avaliação. Também caem bancos com busca vetorial (Cosmos DB for NoSQL, PostgreSQL com pgvector e Banco de Dados SQL com o tipo vector), a forma de chamar o modelo (nome da implantação, tipos de implantação, temperatura, tokens, saídas estruturadas, chamada de função) e serviços especializados como Document Intelligence, Translator e Fala. A prova descreve um problema de qualidade, custo ou formato e pede a técnica certa, e quase sempre há uma alternativa com 'aumentar a temperatura' que nunca resolve conhecimento, busca nem formato. Onde mais erro é em diagnosticar se o problema está na recuperação ou na geração, e em lembrar que documentos e consultas precisam do mesmo modelo de embeddings. Também preciso separar RAG de ajuste fino e saber qual avaliador mede o quê.

Neste capítulo
18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Geração aumentada por recuperação (RAG)

Os documentos são indexados e, a cada pergunta, os trechos mais relevantes são recuperados e enviados como contexto ao modelo. É o padrão para responder com dados próprios que mudam com frequência, sem treinar o modelo de novo. Modelos têm data de corte de conhecimento, então fatos recentes devem vir no contexto.

Embeddings

Vetores gerados por um modelo de embeddings de texto, como text-embedding-3-small e text-embedding-3-large; modelos de chat não geram vetores. O mesmo modelo e a mesma dimensão devem ser usados para indexar e consultar: trocar o modelo só na consulta, ou aposentar o modelo, exige reindexar. O parâmetro dimensions dos modelos text-embedding-3 gera vetores menores com pequena perda de qualidade.

Divisão em blocos (chunking)

Documentos longos são divididos antes de gerar os vetores, com tamanho fixo e sobreposição ou por estrutura (títulos e parágrafos). Trechos muito pequenos perdem contexto e muito grandes diluem a relevância, então o tamanho deve ser testado com avaliações. A habilidade de layout de documento usa o Document Intelligence e gera trechos em Markdown por seção, preservando títulos e tabelas.

Busca híbrida e classificador semântico

A busca híbrida combina palavras-chave e similaridade vetorial usando Reciprocal Rank Fusion. O classificador semântico reordena os melhores resultados com modelos de linguagem da Microsoft e gera legendas e respostas semânticas. A configuração semântica define quais campos (título, conteúdo) o classificador prioriza, pois ele tem limites de tamanho de entrada.

Atributos de campo do índice

searchable, filterable, sortable, facetable e retrievable definem o uso de cada campo. O texto do trecho deve ser recuperável para o app montar o prompt, enquanto campos vetoriais normalmente não precisam ser. Um campo de categoria usado só em filtros fica filtrável e não pesquisável.

Vetorização integrada e vetorizador

A vetorização integrada divide o texto em blocos e chama um modelo de embeddings na indexação e na consulta, sem código próprio. O vetorizador no perfil vetorial permite enviar a consulta como texto e o serviço gera o vetor; pode usar Azure OpenAI, modelos do Foundry ou APIs personalizadas. A dimensão do campo vetorial precisa coincidir com a do modelo.

Indexadores, conjuntos de habilidades e push

Indexadores de blobs com agendamento detectam documentos novos e alterados pela data de modificação; o cache de enriquecimento evita reprocessar o que não mudou. Conjuntos de habilidades enriquecem com OCR, divisão e embeddings. A API de push envia documentos na hora, e o padrão evento de blob, função e push dá indexação em minutos com código próprio de divisão.

Quantização de vetores

Quantização escalar ou binária reduz a memória do índice vetorial com pouca perda de qualidade. Pode ser combinada com reclassificação usando os vetores completos. Mais réplicas aumentam capacidade de consulta, mas não reduzem memória.

Filtros, pontuação e sinônimos no Azure AI Search

Campos filtráveis com metadados (data, tipo, vigência) filtram antes ou depois da busca vetorial, com impacto em desempenho e recall. Filtro de segurança por IDs de grupos e campo filtrável com o ID do cliente isolam resultados no mesmo índice; limite de similaridade descarta trechos fracos. Perfil de pontuação prioriza, por exemplo, documentos recentes, e mapa de sinônimos faz NF encontrar nota fiscal.

Busca vetorial no Cosmos DB for NoSQL

O contêiner precisa de política de embeddings e índice vetorial: flat, quantizedFlat ou diskANN, com diferentes equilíbrios de custo e precisão. Flat faz busca exata e é indicado para poucos vetores; diskANN atende milhões de vetores com busca rápida e baixo consumo de RUs. Consultas usam ORDER BY VectorDistance com TOP N. Colocar o cliente ou o sessionId na chave de partição deixa as consultas dentro de uma partição, mais baratas e rápidas.

Feed de alterações do Cosmos DB

O gatilho do Cosmos DB no Functions usa o feed de alterações e um contêiner de concessões para reagir a mudanças, como regenerar o embedding de um produto alterado. Evita consultas completas periódicas que gastam RUs e têm atraso.

PostgreSQL e Banco de Dados SQL com vetores

No PostgreSQL servidor flexível, o pgvector oferece índices HNSW (melhor equilíbrio entre velocidade e recall) e IVFFlat, e a extensão azure_ai gera embeddings chamando o Azure OpenAI direto do SQL. No Banco de Dados SQL do Azure, o tipo vector e a função VECTOR_DISTANCE calculam similaridade em T-SQL. Similaridade de cosseno é a métrica usual para embeddings de texto.

Implantações do Azure OpenAI

Cada modelo é implantado com um nome, e o código usa esse nome de implantação no parâmetro model ou na rota. Os tipos incluem Global, Data Zone e Regional, com variações Standard, Provisioned e Batch: Data Zone mantém dados em uma zona como a União Europeia; provisionada (PTU) dá capacidade reservada e latência estável. Fixar a versão do modelo evita mudança sem testes.

Parâmetros e tokens

Temperatura e top_p controlam a aleatoriedade: temperatura baixa, perto de zero, dá respostas consistentes. Resposta cortada com finish_reason length pede aumento do limite de tokens de saída, e o total deve caber na janela de contexto. Contar tokens (como com tiktoken) estima custo; o cache de prompts aproveita prefixos longos e idênticos colocados no início do prompt.

Engenharia de prompts

A mensagem de sistema define regras de todas as conversas (escopo, idioma, tom). Delimitadores separam instruções de dados; exemplos few-shot corrigem formato sem ajuste fino; instruir o modelo a dizer que não sabe reduz invenção; enviar trechos com IDs e pedir citações mostra a fonte. Histórico longo deve ser resumido, e documentos enormes são resumidos por partes (map-reduce).

Saídas estruturadas e chamada de função

Saídas estruturadas com esquema JSON garantem o esquema; o modo JSON simples garante só JSON válido. Na chamada de função (tools), o modelo devolve nome e argumentos e o app executa e devolve o resultado, ideal para consultar APIs como estoque. Argumentos errados se corrigem melhorando a descrição dos parâmetros, tipos, enumerações e campos obrigatórios.

Escolha do modelo e ajuste fino

Use o menor modelo que atinge a qualidade exigida, medida com avaliações. Modelos de raciocínio servem para matemática, código e planejamento com várias etapas, aceitando mais latência; modelos com visão recebem imagem por URL ou base64. Ajuste fino é para estilo, formato e tarefas específicas; RAG é para conhecimento que muda.

Avaliação de aplicativos de IA

Avaliadores de RAG incluem fundamentação (resposta apoiada no contexto), relevância, recuperação e completude; fluência mede a linguagem, coerência a organização e similaridade compara com resposta de referência. Avaliar a recuperação separada da geração mostra onde está o problema. O Microsoft Foundry roda avaliações locais ou na nuvem, avaliações de segurança adversariais e usa conjuntos com respostas esperadas para comparar versões.

Macetes para memorizar

💡 RAG x ajuste fino

RAG ensina O QUE dizer (conhecimento que muda); ajuste fino ensina COMO dizer (estilo e formato).

💡 Temperatura

Temperatura nunca é a resposta, exceto quando o pedido é consistência: aí é baixa, perto de zero.

💡 Índices vetoriais do Cosmos DB

Flat é foto exata de poucos; diskANN é o disco que aguenta milhões.

💡 Atributos de campo

Filtrar é filterable, devolver é retrievable, pesquisar texto é searchable.

💡 Avaliadores

Fundamentação pergunta 'está no contexto?'; similaridade, 'bate com o gabarito?'; fluência, 'soa bem?'; coerência, 'está organizado?'.

💡 Indexação

Indexador puxa no horário; push empurra na hora.

💡 Cache de prompts

Fixo na frente, variável no fim.

Tabela de memorização: não confunda

Quando é...	...e quando é
RAG Responder com documentos internos que mudam toda semana ou com fatos recentes, sem treinar o modelo.	**Ajuste fino** Fixar um estilo de escrita ou formato muito específico em milhares de respostas quando o prompt ficou longo e caro.

Saídas estruturadas com esquema JSON O JSON precisa seguir exatamente um esquema definido pelo aplicativo.	**Modo JSON ou prompt pedindo JSON** Basta um JSON válido qualquer; não há garantia de esquema.
Indexador com agendamento Novos documentos de um contêiner de blobs devem entrar no índice periodicamente, sem ação manual.	**API de push** Um documento alterado deve aparecer no índice em segundos, sem esperar o indexador.
Cache semântico de respostas Perguntas muito parecidas geram chamadas repetidas; o API Management reaproveita respostas por similaridade com embeddings e cache Redis compatível.	**Cache de prompts do serviço** Muitas chamadas repetem o mesmo início longo de instruções; deixe o conteúdo fixo no começo do prompt.
Índice flat Poucos vetores e busca exata no Cosmos DB.	**Índice diskANN** Milhões de vetores no Cosmos DB, com busca rápida e baixo consumo de RUs.
Implantação provisionada (PTU) Carga alta e previsível que exige capacidade reservada e latência estável.	**Implantação Standard** Uso variável sem reserva; com cota baixa sofre limite de taxa em picos.
Fundamentação Medir se a resposta se baseia no contexto recuperado, sem inventar.	**Similaridade** Comparar a resposta com uma resposta de referência esperada.

Números que caem

Item	Valor
Tipos de índice vetorial do Cosmos DB	**3: flat, quantizedFlat e diskANN**
Índices do pgvector	**2: HNSW e IVFFlat**
Temperatura para respostas consistentes	**baixa, perto de zero**
Motivo de término quando o limite de tokens é atingido	**length**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Aumentar a temperatura para trazer conhecimento, melhorar a busca, corrigir formato ou caber no contexto.
Como não cair: Temperatura só muda a variação. Para consistência, ela deve ser baixa; para os demais problemas, procure a técnica específica.

⚠ **Alerta de prova:** Colocar todos os documentos na mensagem de sistema.
Como não cair: Isso estoura o limite de contexto. Use RAG e envie só os trechos relevantes.

⚠ **Alerta de prova:** Fazer ajuste fino semanal para acompanhar documentos que mudam sempre.
Como não cair: É caro e lento para dados que mudam. Conhecimento atual vem por RAG; ajuste fino é para estilo e formato.

⚠ **Alerta de prova:** Usar RAG com documentos de estilo para fixar o jeito de escrever.
Como não cair: RAG recupera fatos, mas não fixa estilo tão bem. Estilo consistente em escala pede ajuste fino com exemplos.

⚠ **Alerta de prova:** Trocar o modelo de embeddings só no código de consulta.
Como não cair: Vetores ficam incompatíveis. Documentos e consultas devem usar o mesmo modelo e a mesma dimensão; reindexe.

⚠ **Alerta de prova:** Escolher o índice flat para milhões de vetores no Cosmos DB.
Como não cair: Flat faz busca exata e é indicado para poucos vetores. Use diskANN.

⚠ **Alerta de prova:** Usar cache exato por texto da pergunta para perguntas parecidas.
Como não cair: Só acerta perguntas idênticas. Use cache semântico.

⚠ **Alerta de prova:** Aumentar o número de trechos quando aparecem documentos desatualizados ou trechos pouco relacionados.
Como não cair: Isso traz ainda mais ruído. Filtre por vigência ou defina um limite mínimo de similaridade; aumentar o top k só ajuda quando a informação está fora dos primeiros resultados.

⚠ **Alerta de prova:** Aplicar filtro por idioma quando usuários pesquisam em inglês e os documentos estão em português.
Como não cair: O filtro excluiria os documentos. Use embeddings multilíngues, que favorecem busca entre idiomas.

⚠ **Alerta de prova:** Criar um índice por usuário ou um serviço de busca por cliente para isolar documentos.
Como não cair: É caro e difícil de manter. Use filtro de segurança por grupos ou campo filtrável com o ID do cliente.

⚠ **Alerta de prova:** Usar o classificador semântico para restringir resultados por permissão.
Como não cair: Ele só melhora a ordem. Restrição por permissão é filtro de segurança.

⚠ **Alerta de prova:** Usar OCR nas imagens do catálogo para buscar imagens por descrição em texto.
Como não cair: OCR extrai texto, não o conteúdo visual. Use embeddings multimodais do Azure AI Vision.

⚠ **Alerta de prova:** Escolher Global Standard quando os dados devem ficar em uma zona geográfica como a União Europeia.
Como não cair: Global Standard pode processar em qualquer região. Use implantação Data Zone.

⚠ **Alerta de prova:** Estimar custo e contexto medindo megabytes ou contando palavras.
Como não cair: Bytes e palavras não equivalem a tokens. Conte os tokens do prompt.

Como cai na prova

📝 **Como cai:** Problema de qualidade no RAG: informação faltando, conteúdo desatualizado, trechos fracos, invenção, pergunta de acompanhamento.
Como resolver: Primeiro decida se a falha é da recuperação ou da geração. Recuperação: aumentar top k, busca híbrida, filtros por vigência, limite de similaridade, reescrita da consulta. Geração: instrução para dizer que não sabe, citações, orçamento de tokens. Descarte temperatura e região.

📝 **Como cai:** Escolher onde guardar e buscar vetores.
Como resolver: Busca completa com texto, filtros, facetas e reclassificação: Azure AI Search. Dados operacionais já no Cosmos DB: política de embeddings, índice vetorial e VectorDistance. PostgreSQL: pgvector com HNSW e azure_ai. SQL: tipo vector e VECTOR_DISTANCE.

📝 **Como cai:** Ajuste na chamada ao modelo: consistência, formato, tamanho, ferramentas.
Como resolver: Consistência: temperatura baixa. Formato: few-shot ou saídas estruturadas com esquema. Tamanho: limite de tokens de saída. Ferramentas: chamada de função com parâmetros bem descritos. Regras para todas as conversas: mensagem de sistema.

📝 **Como cai:** Medir qualidade ou segurança antes de publicar.
Como resolver: Qualidade do RAG: avaliações no Foundry com fundamentação e relevância. Comparar versões: conjunto com respostas de referência. Segurança: avaliações adversariais. Métricas de CPU, custo ou teste de carga nunca medem qualidade.

📝 **Como cai:** Custo, capacidade e residência da implantação.
Como resolver: Tarefa simples e barata: menor modelo validado. Carga alta previsível: PTU. Residência em zona geográfica: Data Zone. Versão sob controle: fixar a versão. Perguntas parecidas: cache semântico; prefixo repetido: cache de prompts.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
documentos internos que mudam toda semana, sem treinar o modelo	**RAG**
combinar palavras-chave e similaridade vetorial e reordenar por relevância	**Busca híbrida com classificador semântico**
mostrar o trecho que responde à pergunta, destacado	**Legendas e respostas semânticas**
embeddings gerados durante a indexação de blobs, sem código próprio	**Vetorização integrada no indexador**
a consulta envia texto e o próprio serviço gera o vetor	**Vetorizador no perfil**
PDFs digitalizados como imagem precisam de texto pesquisável	**Habilidade de OCR no indexador**
dividir preservando títulos, seções e tabelas	**Habilidade de layout de documento**
busca pela sigla deve encontrar o termo por extenso	**Mapa de sinônimos no campo**
na busca por palavras-chave, documentos recentes primeiro	**Perfil de pontuação por atualidade**
respostas cortadas no meio do texto	**Aumentar o limite de tokens de saída (finish_reason length)**
o assistente deve consultar uma API da empresa	**Chamada de função (tools)**
dados processados devem ficar em uma zona geográfica com mais capacidade que uma regional	**Implantação Data Zone**
pergunta de acompanhamento como 'e o preço dele?' não acha os trechos	**Reescrever a consulta com o histórico**
extrair total e data de recibos digitalizados	**Document Intelligence (modelo de recibos)**

🎯 Se você lembrar de só uma coisa deste capítulo...

- Documentos e consultas usam o mesmo modelo e a mesma dimensão de embeddings; trocou, reindexa.
- O código chama o modelo pelo nome da implantação (parâmetro model ou rota), não pelo ID da assinatura, pela região ou pelo grupo de recursos.
- Saídas estruturadas garantem o esquema; o modo JSON garante só JSON válido.
- Dados do cliente enviados ao Azure OpenAI não treinam os modelos.

Essenciais para a prova (checklist)

☐ Sei montar um pipeline de RAG: divisão em blocos, embeddings, índice, recuperação e prompt com citações.

☐ Sei a diferença entre vetorização integrada, vetorizador, indexador agendado e API de push.

☐ Sei configurar atributos de campo, configuração semântica, perfil de pontuação e mapa de sinônimos.

☐ Sei isolar resultados por grupo ou cliente com filtros no mesmo índice.

☐ Sei configurar busca vetorial no Cosmos DB, no PostgreSQL e no Banco de Dados SQL.

☐ Sei escolher entre Global, Data Zone, Regional, Standard, PTU e Batch.

☐ Sei ajustar temperatura, limite de tokens de saída e usar cache de prompts.

☐ Sei quando usar few-shot, saídas estruturadas, chamada de função e mensagem de sistema.

☐ Sei qual avaliador mede fundamentação, relevância, recuperação, fluência, coerência e similaridade.

☐ Sei quando usar Document Intelligence, Translator, Fala, embeddings multimodais e Foundry Agent Service.

Questões de revisão do capítulo

R2.1 · Desenvolver soluções de IA com serviços de dados do Azure

Documentos em vários idiomas devem ser traduzidos para português, mantendo o formato original dos arquivos. O que usar?

A. Mapa de sinônimos
B. Habilidade de OCR
C. Modelo de embeddings multilíngue
D. Tradução de documentos do Translator

R2.2 · Desenvolver soluções de IA com serviços de dados do Azure

Os dados processados pelo modelo devem ficar em uma zona geográfica, como a União Europeia, com mais capacidade que uma implantação regional. O que usar?

A. Implantação sem região definida
B. Implantação do tipo Data Zone
C. Implantação Global Standard
D. Implantação em uma VM própria

R2.3 · Desenvolver soluções de IA com serviços de dados do Azure

A equipe quer criar um agente que use busca em arquivos, interpretador de código e ferramentas próprias, com conversas gerenciadas pelo serviço. O que usar?

A. Uma fila do Service Bus
B. Um índice sem campos vetoriais
C. Um registro de contêiner
D. Foundry Agent Service

R2.4 · Desenvolver soluções de IA com serviços de dados do Azure

Qual função é usada em consultas do Cosmos DB for NoSQL para ordenar resultados por similaridade vetorial?

A. STARTSWITH
B. VectorDistance
C. COUNT
D. ARRAY_CONTAINS

Respostas comentadas

R2.1. D) Tradução de documentos do Translator

Por quê: O Azure AI Translator traduz textos e documentos inteiros, com glossários personalizados.

✗ A) Incorreta. Expande termos.
✗ B) Incorreta. Extrai texto.
✗ C) Incorreta. Gera vetores, não traduções.

> **Dica de prova:** Traduzir arquivos mantendo formato: tradução de documentos.

R2.2. B) Implantação do tipo Data Zone

Por quê: Os tipos de implantação incluem Global, Data Zone e Regional, com variações Standard, Provisioned e Batch.

✗ A) Incorreta. Não existe.
✗ C) Incorreta. Pode processar em qualquer região.
✗ D) Incorreta. Não é um tipo de implantação do serviço.

> **Dica de prova:** Residência por zona de dados: Data Zone.

R2.3. D) Foundry Agent Service

Por quê: O Foundry Agent Service oferece ferramentas como busca de arquivos, Azure AI Search, interpretador de código e OpenAPI.

✗ A) Incorreta. Não cria agentes.
✗ B) Incorreta. Não cria agentes.
✗ C) Incorreta. Não cria agentes.

> **Dica de prova:** Agente com ferramentas gerenciado: Foundry Agent Service.

R2.4. B) VectorDistance

Por quê: Consultas vetoriais costumam usar ORDER BY VectorDistance com TOP N para limitar os resultados.

✗ A) Incorreta. Compara prefixos de texto.
✗ C) Incorreta. Conta itens.
✗ D) Incorreta. Verifica elementos em matrizes.

> **Dica de prova:** Similaridade no Cosmos DB: VectorDistance.

03

CAPÍTULO 3 · 20-25% DA PROVA

Desenvolver soluções em contêineres no Azure

Aqui eu preciso dominar o Azure Container Apps de ponta a ponta: revisões, ingresso, escala com KEDA, sondas, segredos, volumes, Dapr e jobs, além do Azure Container Registry (build, tarefas, replicação, tokens) e o básico de Dockerfile. A prova descreve um requisito ou sintoma curto e pede a configuração exata, então o segredo é ligar cada requisito à peça certa: réplicas mínimas x máximas, prontidão x atividade, rótulo x divisão de tráfego, Azure Files x volume efêmero. Onde mais erro é nas alternativas parecidas, como regra de CPU no lugar de regra HTTP, réplicas mínimas fixas no lugar de escala cron e recursos do ACR que exigem a camada Premium. Também caem comandos da CLI (az containerapp update, up, exec, logs show --follow, secret set, job start e az acr build), sempre trocados entre si nas alternativas. Fecho o capítulo com quando usar ACI, App Service com contêiner, jobs e Functions no Container Apps.

Neste capítulo
18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Revisões e modos de revisão

Cada atualização de modelo do app, como trocar a imagem com az containerapp update, gera uma nova revisão. No modo de revisão única, a nova revisão assume o tráfego sozinha quando fica pronta. No modo de várias revisões, revisões ficam ativas ao mesmo tempo e o tráfego é dividido por pesos; manter a revisão anterior ativa com peso zero permite reverter mandando o tráfego de volta, sem nova implantação.

Rótulo de revisão

Dá uma URL dedicada a uma revisão para testá-la antes de ela receber tráfego de produção. Combinado com a divisão de tráfego, viabiliza implantações azul-verde e canário no Container Apps.

Regras de escala (KEDA)

O Container Apps escala com regras baseadas em KEDA: HTTP (concurrentRequests por réplica), TCP, eventos como filas e tópicos e escalonadores personalizados como cron (com fuso horário) e azure-queue (nome da fila e tamanho alvo). Os escalonadores autenticam por segredo ou por identidade gerenciada, o que dispensa cadeia de conexão no Service Bus.

Réplicas mínimas e máximas

O mínimo e o máximo valem para todas as regras de escala do app. Mínimo 0 permite escala até zero e causa atraso de inicialização a frio; mínimo 1 deixa a primeira chamada previsível, com custo contínuo. O máximo é o teto que controla custo e protege dependências; um mínimo fixo alto mantém essas réplicas ligadas sempre.

Sondas de integridade

Existem sondas de inicialização, prontidão e atividade. A de prontidão (readiness) só libera tráfego quando a réplica está pronta, como depois de carregar um modelo; a de atividade (liveness) reinicia a réplica cujo processo travou. Implantação sem parada combina sonda de prontidão com mais de uma réplica mínima.

Segredos do Container App

Segredos são configuração do app e não criam revisão: depois de alterar, é preciso reiniciar a revisão ativa, pois réplicas em execução mantêm o valor antigo. Podem referenciar o Key Vault e ser expostos como variáveis de ambiente (secretRef) ou volumes. Pela CLI, use az containerapp secret set.

Ingresso

A porta de destino precisa ser a porta em que o processo escuta dentro do contêiner. Com allowInsecure desativado, HTTP é redirecionado para HTTPS. Restrições de IP usam listas de permissão ou negação com CIDR; afinidade de sessão devolve o cliente à mesma réplica; ingresso TCP expõe protocolos não HTTP em ambiente com VNet própria.

Domínio personalizado e certificado gerenciado

O Container App pode responder em um domínio próprio com HTTPS usando certificado gerenciado, sem comprar certificado. A validação do domínio é feita por registro CNAME ou TXT.

Ambiente e rede do Container Apps

A integração com VNet é definida na criação do ambiente e exige uma sub-rede dedicada. Apps com ingresso no mesmo ambiente se chamam pelo nome do app ou pelo FQDN interno, nunca por IP de réplica. O mTLS do ambiente criptografa o tráfego entre apps com certificados gerenciados pela plataforma.

Volumes

O Container Apps aceita volumes do Azure Files, efêmeros e de segredos. Azure Files persiste e compartilha arquivos entre réplicas, como um modelo de 2 GB fora da imagem. O volume efêmero (EmptyDir) serve para arquivos temporários entre contêineres da mesma réplica e não guarda nada depois.

Sidecar e contêiner de inicialização

Um Container App pode ter vários contêineres. O sidecar roda junto da API em cada réplica, compartilhando a rede local. O contêiner de inicialização roda antes do principal e pode preparar arquivos em um volume compartilhado.

Dapr gerenciado

Oferece invocação de serviço (descoberta, mTLS e repetições sem código próprio), pub/sub com agente trocável (Service Bus, Event Hubs, Kafka) e armazenamento de estado, como Cosmos DB sem o SDK no código. Componentes ficam no ambiente e aceitam identidade gerenciada; o app configura app-id, app-port e protocolo.

Jobs do Container Apps

Jobs são para tarefas finitas e podem ser manuais, agendados ou orientados a eventos com escalonadores do KEDA. Um job orientado a eventos roda só quando há mensagens e termina ao concluir. Execução manual: az containerapp job start; acompanhamento: az containerapp job execution list.

Perfis de carga de trabalho, GPU e sessões dinâmicas

CPU e memória são definidas por contêiner, dentro das combinações do perfil. Para mais memória por réplica do que o consumo oferece, use perfil dedicado; GPUs sem servidor atendem inferência com escala até zero e cobrança por segundo. Sessões dinâmicas usam pools com isolamento Hyper-V para executar código gerado por IA.

ACR Tasks

O az acr build compila a imagem na nuvem, sem Docker local, inclusive para várias plataformas. Tarefas podem ser disparadas por commit, por atualização da imagem base ou por agenda; tarefas de várias etapas executam build, cmd e push em sequência ou em paralelo. O acr purge em tarefa agendada remove imagens antigas por repositório, tag e idade.

Recursos Premium e acesso ao ACR

Replicação geográfica (um único nome de registro), ponto privado (zona privatelink.azurecr.io) e política de retenção de manifestos sem tag exigem a camada Premium. Pull sem senha usa identidade gerenciada com AcrPull; acesso a um único repositório usa token com mapa de escopo (content/read). O ACR guarda artefatos OCI, como gráficos Helm e assinaturas.

Tags e digest

Tags são mutáveis e latest serve só para conveniência; repositório sem tag usa latest por padrão. Referenciar pelo digest sha256 garante a imagem exata testada, e o ACR permite bloquear tags contra sobrescrita. Tag com versão e hash do commit permite rastrear a origem de cada build.

Boas práticas de Dockerfile

Build em várias etapas separa SDK e runtime e gera imagem menor; imagens chiseled não têm shell nem gerenciador de pacotes e rodam sem root. ENTRYPOINT define o executável principal e CMD os argumentos padrão; USER tira o root, e sem root a porta deve ficar acima de 1024. Copiar e instalar dependências antes do código aproveita o cache de camadas, e o .dockerignore tira .git e .env do contexto.

Macetes para memorizar

💡 Sondas

Prontidão é o porteiro (só deixa o tráfego entrar quando está pronto); atividade é o desfibrilador (reinicia quem travou).

💡 Réplicas

Mínimo é piso (sempre ligado, custa sempre); máximo é teto (protege custo e dependências).

💡 ACR Premium

GPR é Premium: Geo-replicação, Ponto privado e Retenção de manifestos sem tag.

💡 Volumes

Files fica e é de todos; efêmero some com a réplica.

💡 Tag x digest

Tag é apelido (pode mudar de dono); digest é impressão digital (nunca muda).

💡 Comandos do Container Apps

up cria tudo do código; update implanta nova imagem; exec entra no contêiner; logs show --follow assiste ao vivo; secret set guarda segredo; job start dispara job.

💡 Contêineres extras

Init prepara antes; sidecar acompanha durante.

Tabela de memorização: não confunda

Quando é...	...e quando é
Sonda de prontidão (readiness) A réplica não pode receber tráfego antes de terminar de iniciar, como ao carregar um modelo por um minuto; ela só tira do tráfego, sem reiniciar.	**Sonda de atividade (liveness)** O processo trava sem terminar e a réplica deve ser reiniciada automaticamente.

Réplicas mínimas Evitar inicialização a frio (mínimo 1) ou manter disponibilidade durante implantações (mais de uma).	**Réplicas máximas** Colocar um teto em picos para controlar custo e proteger dependências; mínimo alto manteria réplicas sempre ligadas.
Volume do Azure Files Arquivos que precisam persistir e ser compartilhados entre réplicas, como um modelo grande fora da imagem.	**Volume efêmero** Arquivos temporários compartilhados entre contêineres da mesma réplica, sem guardar nada depois.
Contêiner sidecar Um auxiliar precisa rodar junto da API durante toda a execução, em cada réplica, na mesma rede local.	**Contêiner de inicialização** Um script precisa rodar antes do contêiner principal, como baixar configurações para um volume compartilhado.
Modo de revisão única A nova revisão deve assumir o tráfego automaticamente assim que ficar pronta.	**Modo de várias revisões** É preciso manter revisões ativas, dividir tráfego por pesos (canário, azul-verde) e reverter rápido mandando tráfego para a anterior.
az acr build Gerar a imagem a partir do Dockerfile do repositório, sem Docker instalado no pipeline.	**az acr import** Copiar imagens prontas de outro registro, como imagens base do Docker Hub, para fugir do limite de downloads.
Tarefa agendada com acr purge Excluir imagens por idade, repositório e tag, como imagens de testes com mais de 30 dias; atende outras camadas.	**Política de retenção de manifestos sem tag** Remover automaticamente manifestos sem tag acumulados; disponível na camada Premium.

Números que caem

Item	Valor
Porta para contêiner que roda sem root	**acima de 1024**
Porta padrão das imagens recentes do .NET	**8080**
Réplicas mínimas para evitar inicialização a frio	**1 (mínimo acima de zero)**
Réplicas mínimas para implantação sem parada	**mais de uma, junto com sonda de prontidão**
Camada do ACR para replicação geográfica, ponto privado e retenção de manifestos	**Premium**
Zona DNS privada do ponto privado do ACR	**privatelink.azurecr.io**
Delegação da sub-rede do ACI em VNet	**Microsoft.ContainerInstance/containerGroups**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Escalar por requisições simultâneas com uma regra de CPU com limite de 50%.
Como não cair: CPU não mede requisições simultâneas. Use a regra HTTP com concurrentRequests no valor pedido.

⚠ **Alerta de prova:** Fixar réplicas mínimas em três para ter três réplicas em horário comercial.
Como não cair: Mínimo fixo mantém as réplicas sempre ligadas, inclusive fora do horário. Escala por horário é o escalonador cron do KEDA.

⚠ **Alerta de prova:** Usar a sonda de atividade para segurar o tráfego enquanto o modelo carrega.
Como não cair: A atividade reinicia réplicas travadas; quem segura o tráfego até a réplica ficar pronta é a sonda de prontidão.

⚠ **Alerta de prova:** Achar que, depois de alterar um segredo, as réplicas passam a usar o novo valor na hora.
Como não cair: Segredos não criam revisão e réplicas em execução mantêm o valor antigo. Reinicie a revisão ativa.

⚠ **Alerta de prova:** Aumentar as réplicas quando o ingresso aponta para uma porta diferente da que o processo escuta (por exemplo, 80 no ingresso e 8080 no contêiner).
Como não cair: Mais réplicas continuariam na porta errada. Ajuste a porta de destino do ingresso para a porta em que o processo escuta.

⚠ **Alerta de prova:** Aumentar o número de réplicas quando o contêiner reinicia por falta de memória ou precisa de mais memória do que o perfil de consumo oferece.
Como não cair: Cada réplica continua com pouca memória. Aumente a memória do contêiner; se o consumo não oferecer o necessário por réplica, use um perfil de carga de trabalho dedicado.

⚠ **Alerta de prova:** Acreditar que remover com uma instrução posterior um segredo definido com ENV apaga o valor da imagem.
Como não cair: O valor fica nas camadas ou nos metadados para qualquer pessoa com pull. Use segredos de build e configure segredos em tempo de execução.

⚠ **Alerta de prova:** Reduzir a imagem compactando com gzip ou trocando a tag da imagem base do SDK.
Como não cair: O registro já compacta as camadas e a tag não muda o tamanho. Separe compilação e runtime com build em várias etapas.

⚠ **Alerta de prova:** Dar AcrPull no registro inteiro ou o usuário administrador para um parceiro que só deve ver um repositório.
Como não cair: Essas opções liberam todo o registro. Use token com mapa de escopo do repositório.

⚠ **Alerta de prova:** Resolver acesso só pela rede privada com token de mapa de escopo ou replicação para outra região.
Como não cair: Token controla permissões e replicação copia imagens; nenhum restringe a rede. Use ponto privado no ACR Premium.

⚠ **Alerta de prova:** Recompilar quando a base recebe correção usando webhook do registro chamando um script manual.
Como não cair: Isso exige manutenção e ação externa. Configure uma tarefa do ACR disparada por atualização da imagem base.

⚠ **Alerta de prova:** Criar outro Container App no mesmo ambiente para um auxiliar que deve rodar junto da API.
Como não cair: Outro app teria réplicas separadas. O auxiliar na mesma réplica e na mesma rede local é um sidecar.

⚠ **Alerta de prova:** Referenciar a imagem pelo nome do repositório sem tag para garantir a versão testada.
Como não cair: Sem tag, vale latest por padrão, e tags são mutáveis. Referencie pelo digest sha256.

⚠ **Alerta de prova:** Rodar inferência com GPU em Functions no plano de consumo clássico ou App Service Basic.
Como não cair: Nenhum dos dois oferece GPU. Para GPU sem gerenciar cluster e com escala até zero, use GPUs sem servidor do Container Apps.

Como cai na prova

📝 **Como cai:** Requisito de escala: 'deve ganhar réplicas quando...', 'em horário comercial', 'conforme mensagens na fila'.
Como resolver: Identifique o sinal que dispara a escala (requisições, horário, mensagens) e escolha a regra correspondente: HTTP com concurrentRequests, cron ou azure-queue. Descarte réplicas mínimas fixas, que deixam tudo ligado sempre, e regra de CPU, que não mede o sinal pedido.

📝 **Como cai:** Sintoma de falha: reinício em laço, erro de conexão no ingresso, falha ao baixar a imagem, erro de falta de memória.
Como resolver: Ligue o sintoma à causa: reinício, olhar logs de console; porta, ajustar a porta de destino; download, conferir nome da imagem (registro, repositório, tag) e permissão da identidade; memória, aumentar a alocação do contêiner. Alternativas com 'mais réplicas' ou 'novo ambiente' quase nunca resolvem.

📝 **Como cai:** Cenário do ACR: build, acesso, rede, limpeza ou região.
Como resolver: Classifique o pedido: build sem Docker ou por gatilho é ACR Tasks; acesso granular é token com mapa de escopo; pull sem senha é identidade gerenciada com AcrPull; rede privada, várias regiões e retenção exigem Premium; limpeza por idade é acr purge.

📝 **Como cai:** Dockerfile com problema de tamanho, segurança ou velocidade de build.
Como resolver: Tamanho: build em várias etapas e imagem de runtime mínima. Segurança: USER sem root, porta acima de 1024, .dockerignore e nada de segredo em ENV. Velocidade: copiar e instalar dependências antes do código para aproveitar o cache.

📝 **Como cai:** Escolher onde hospedar o contêiner.
Como resolver: Slots de troca: App Service com contêiner. Tarefa finita ou por mensagens: job do Container Apps. Functions junto dos microsserviços na mesma rede e logs: Functions no Container Apps. Contêiner avulso em VNet: ACI com sub-rede delegada. Microsserviços com escala e revisões: Container Apps.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
descoberta de serviço, mTLS e repetições sem escrever esse código	**Dapr com a API de invocação de serviço**
agente de mensagens trocável sem alterar o código	**Componente de pub/sub do Dapr**
testar a nova revisão por endereço próprio antes do tráfego de produção	**Rótulo de revisão com URL dedicada**
sem atraso de inicialização a frio na primeira chamada	**Réplicas mínimas iguais a 1**
pipeline sem Docker instalado precisa gerar a imagem	**az acr build**
Container Apps em duas regiões baixando imagens do registro local	**Replicação geográfica do ACR (Premium)**
executar código Python gerado pelo modelo, isolado e descartável por usuário	**Sessões dinâmicas do Container Apps**
contêiner com GPU, escala até zero, sem gerenciar cluster	**GPUs sem servidor no Container Apps**

slots de implantação com troca entre preparo e produção	**App Service com contêiner**
contêineres que só rodam quando há mensagens e terminam ao concluir	**Job orientado a eventos**
variável sensível no ACI não pode aparecer nas propriedades do recurso	**Variáveis de ambiente seguras (secureValue)**
contêiner do ACI precisa acessar banco privado na VNet	**Sub-rede delegada da VNet**
mesma imagem em x64 e ARM64 sem tags diferentes	**Imagem de várias plataformas**
pipeline falha por limite de downloads do Docker Hub	**Importar as imagens base para o ACR**

🎯 Se você lembrar de só uma coisa deste capítulo...

- Segredo alterado no Container App não cria revisão: reinicie a revisão ativa.
- Réplicas mínimas e máximas valem para todas as regras de escala; mínimo fixo não substitui regra de escala.
- Replicação geográfica, ponto privado e retenção de manifestos sem tag exigem ACR Premium.
- A porta de destino do ingresso deve ser a porta em que o processo escuta no contêiner.
- Tag é mutável; para a imagem exata, use o digest sha256.

Essenciais para a prova (checklist)

☐ Sei escolher entre regra HTTP, cron, azure-queue e réplicas mínimas/máximas para cada requisito de escala.

☐ Sei a diferença entre sondas de inicialização, prontidão e atividade.

☐ Sei como funcionam revisão única, várias revisões, rótulos e reversão por divisão de tráfego.

☐ Sei usar segredos com referência ao Key Vault e secretRef, e que mudar segredo exige reiniciar a revisão.

☐ Sei quando usar Azure Files, volume efêmero, sidecar e contêiner de inicialização.

☐ Sei o que o Dapr oferece: invocação de serviço, pub/sub, estado e a configuração app-port.

☐ Sei quais recursos do ACR exigem Premium e como dar acesso por identidade gerenciada ou token com mapa de escopo.

☐ Sei os gatilhos e tipos de tarefa do ACR Tasks e quando usar acr purge.

☐ Sei aplicar build em várias etapas, USER, .dockerignore, ENTRYPOINT/CMD e ordem de camadas no Dockerfile.

☐ Sei associar cada comando (up, update, exec, logs show --follow, secret set, job start, az acr build, az container logs) à sua função.

Questões de revisão do capítulo

R3.1 · Desenvolver soluções em contêineres no Azure

Funções do Azure Functions devem rodar no mesmo ambiente do Container Apps, junto com os microsserviços, com a mesma rede e logs. O que usar?

A. Uma VM com o runtime
B. Functions no plano de consumo clássico
C. Functions no Container Apps
D. Um job agendado sem gatilhos

R3.2 · Desenvolver soluções em contêineres no Azure

As imagens do aplicativo devem ser recompiladas automaticamente sempre que a imagem base receber uma correção de segurança. O que configurar?

A. Tarefa do ACR por atualização da imagem base
B. Webhook do registro chamando um script manual
C. Replicação geográfica do registro
D. Política de retenção de manifestos

R3.3 · Desenvolver soluções em contêineres no Azure

Uma nova revisão de um Container App deve ser testada por um endereço próprio antes de receber tráfego de produção. O que usar?

A. Escala até zero na revisão ativa
B. Réplica mínima igual a dez
C. Novo ambiente para cada revisão
D. Rótulo de revisão com URL dedicada

R3.4 · Desenvolver soluções em contêineres no Azure

Container Apps em duas regiões devem baixar as imagens de um registro na própria região, com baixa latência. O que configurar no ACR?

A. Webhook para cada envio
B. Tarefa agendada de limpeza
C. Replicação geográfica do registro
D. Token com mapa de escopo

Respostas comentadas

R3.1. C) Functions no Container Apps

Por quê: Essa hospedagem mantém o modelo de programação do Functions com os recursos do Container Apps.

✗ A) Incorreta. Exige gerenciar a VM.
✗ B) Incorreta. Fica fora do ambiente.
✗ D) Incorreta. Não oferece gatilhos do Functions.

> **Dica de prova:** Functions junto dos microsserviços: hospedagem no Container Apps.

R3.2. A) Tarefa do ACR por atualização da imagem base

Por quê: Tarefas do ACR podem ser disparadas por commit, por atualização de imagem base ou por agenda.

✗ B) Incorreta. Exige manutenção e ação externa.
✗ C) Incorreta. Replica imagens, sem recompilar.
✗ D) Incorreta. Remove manifestos antigos.

> **Dica de prova:** Recompilar quando a base muda: ACR Tasks.

R3.3. D) Rótulo de revisão com URL dedicada

Por quê: Com rótulos e divisão de tráfego, é possível fazer implantações azul-verde e canário no Container Apps.

✗ A) Incorreta. Não cria endereço de teste.
✗ B) Incorreta. Não cria endereço de teste.
✗ C) Incorreta. É desnecessário para o teste.

> **Dica de prova:** Testar revisão antes do tráfego: rótulo com URL.

R3.4. C) Replicação geográfica do registro

Por quê: A replicação geográfica usa um único nome de registro e exige a camada Premium.

✗ A) Incorreta. Avisa sobre envios.
✗ B) Incorreta. Remove imagens antigas.
✗ D) Incorreta. Controla acesso, não a região.

> **Dica de prova:** Pull local em várias regiões: replicação geográfica.

04

CAPÍTULO 4 · 20-25% DA PROVA

Proteger, monitorar e solucionar problemas de soluções do Azure

Aqui eu preciso dominar autenticação sem chave (identidades gerenciadas, DefaultAzureCredential e funções de dados do RBAC), Key Vault, App Configuration, rede privada, Microsoft Entra ID para aplicativos e a segurança específica de IA com Content Safety. A outra metade é observabilidade: Application Insights com OpenTelemetry, KQL, métricas e logs do Azure Monitor e solução de problemas no Container Apps. A prova pede quase sempre o menor privilégio no menor escopo e a opção que elimina segredos, e as alternativas erradas trazem funções de plano de controle (Leitor, Colaborador), chaves guardadas no Key Vault ou o log de atividades. Onde mais erro é separar o que cada ferramenta do Content Safety detecta e qual código HTTP indica cada falha do Azure OpenAI. Também preciso saber qual recurso do Application Insights responde cada pergunta de diagnóstico.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Identidades gerenciadas

A atribuída pelo sistema só existe depois que o recurso é criado e acompanha o ciclo de vida dele. A atribuída pelo usuário permite pré-autorização, antes de criar o app, e reutilização entre recursos. Com várias identidades no app, informe o client ID no DefaultAzureCredential ou no ManagedIdentityCredential; identidades gerenciadas só existem no Azure.

DefaultAzureCredential

Tenta em sequência variáveis de ambiente, identidade gerenciada, CLI do Azure, PowerShell e outras fontes, permitindo o mesmo código local e no Azure. Localmente, o desenvolvedor precisa entrar com az login; em produção, é possível usar ManagedIdentityCredential direto para evitar tentativas desnecessárias. AzureNamedKeyCredential usa chave, UsernamePasswordCredential não suporta MFA e ClientSecretCredential exige segredo.

Key Vault com Azure RBAC

RBAC é o modelo recomendado para cofres novos; políticas de acesso são o modelo legado, e o Key Vault não usa chave de acesso. Há funções separadas para segredos, chaves e certificados: Secrets Officer gerencia segredos, Secrets User lê segredos, Reader só lê metadados e Administrator acessa tudo. No código, SecretClient com DefaultAzureCredential lê segredos.

Chaves e certificados no Key Vault

Com o CryptographyClient, operações como encrypt, decrypt, sign, verify, wrapKey e unwrapKey usam a chave sem ela sair do cofre. Certificados no Key Vault podem ser renovados automaticamente e referenciados por serviços como App Service e Application Gateway.

Operação do Key Vault

Com exclusão temporária, recriar um segredo com nome recém-excluído dá conflito: recupere ou limpe; com proteção contra limpeza, só recuperar. O cofre tem limites de taxa, então guarde segredos em cache com atualização periódica. Eventos como SecretNearExpiry vão ao Event Grid e acionam uma função para rotação automática.

Funções de plano de dados

Funções de plano de controle não são o caminho para dados: Leitor não permite ler dados nem fazer inferência; Colaborador e Proprietário gerenciam o recurso, podem obter chaves e dão muito mais acesso que o necessário. Para blobs, Storage Blob Data Contributor no contêiner; para Cosmos DB, funções de dados atribuídas por CLI ou modelos com escopo de conta ou contêiner; para IA, Azure AI User ou a função de usuário do OpenAI. Token válido com 403 indica falta de função de dados.

Autenticação sem chave em recursos de IA

O app usa token do Entra ID com uma função de usuário de IA, e a autenticação por chave pode ser desativada com disableLocalAuth, exigível por Azure Policy em toda a organização. Quando ainda houver chaves, rotacione alternando entre as duas e guarde-as no Key Vault.

Microsoft Entra ID para aplicativos

SPA obtém tokens com fluxo de código de autorização com PKCE e MSAL.js; o fluxo implícito não é mais recomendado. Permissões delegadas exigem usuário conectado, e permissões de aplicativo servem a serviços sem usuário. Em Expor uma API, define-se o URI do ID do aplicativo e os escopos; a autenticação interna (Easy Auth) do App Service, Functions e Container Apps exige login sem mudar o código.

Federação de identidade com OIDC

Um fluxo do GitHub Actions implanta no Azure sem guardar segredo de cliente no repositório. A credencial federada é configurada na identidade gerenciada ou no registro de aplicativo.

App Configuration e segredos de desenvolvimento

Referências do Key Vault deixam no App Configuration só a referência, e o provedor lê o segredo com a credencial do app. Atualização com chave sentinela (ou eventos do Event Grid) recarrega configurações sem reiniciar; sinalizadores de recurso têm filtros de percentual, público e janela de tempo. Localmente em .NET, valores sensíveis ficam em user-secrets.

Rede privada para IA e Container Apps

Ponto privado sem acesso público mantém as chamadas na rede privada, com a zona privatelink.openai.azure.com; o nome privado só resolve com a zona DNS privada vinculada à VNet. Azure Policy com definições internas nega criação de recursos de IA com acesso público. Restrições de IP no ingresso limitam a origem, e CORS libera as origens permitidas do front-end.

Azure AI Content Safety

Filtros de conteúdo avaliam entradas e saídas para ódio, sexual, violência e automutilação, com níveis configuráveis por implantação. Prompt Shields detecta jailbreak nas entradas e ataques indiretos vindos de documentos; detecção de material protegido acha textos (e código) protegidos na saída; detecção de fundamentação acha respostas sem apoio no contexto. Listas de bloqueio barram termos específicos.

Erros do Azure OpenAI

400 com código content_filter indica entrada barrada pelo filtro (respostas filtradas podem ter finish_reason content_filter); 401 ou 403, falta de permissão; 404, implantação inexistente; 429, cota de tokens por minuto. Para 429, repita com espera e ajuste a cota; filas e balanceamento entre implantações também ajudam.

Application Insights com OpenTelemetry

A distribuição OpenTelemetry do Azure Monitor existe para .NET, Java, Node.js e Python e é configurada pela cadeia de conexão, pois a ingestão só por chave de instrumentação foi descontinuada. Contadores, histogramas e medidores geram métricas personalizadas; logs estruturados viram propriedades em customDimensions. Processadores de spans e amostragem filtram ruído, como requisições de sondas de integridade.

Ferramentas de diagnóstico do Application Insights

Dependências na visão de transação e no mapa do aplicativo mostram a latência de chamadas externas, como ao Azure OpenAI. O operation_Id, com W3C Trace Context, segue uma requisição por vários microsserviços. Live Metrics mostra taxa de requisições, falhas e CPU em tempo real, o Depurador de Instantâneos (.NET) mostra variáveis no momento da exceção e a detecção inteligente avisa sobre falhas anormais sem limite fixo.

Consultas KQL no Application Insights

As tabelas principais são requests, dependencies, exceptions, traces, customEvents e customMetrics. Requisições com falha: requests | where success == false. Exceções por tipo: exceptions | summarize count() by type.

Monitoramento de recursos de IA e Container Apps

Tokens processados e latência de uma implantação ficam nas métricas do Azure Monitor; logs de diagnóstico enviam solicitações ao Log Analytics para auditoria com KQL. O log de atividades mostra só operações de gerenciamento. No Container Apps, logs de console mostram a saída do processo, alertas de métrica cobrem reinícios e 5xx (dividindo por código de status e revisão) e az containerapp exec abre um shell.

Proteção de IA em produção e boas práticas

O Defender for Cloud para IA detecta jailbreak e vazamento de dados sensíveis, com alertas no Defender XDR e no Microsoft Sentinel. Logs não devem ter chaves, tokens nem dados pessoais sem necessidade; consultas parametrizadas evitam injeção de SQL. Contra injeção de prompt, combine Prompt Shields, delimitação do conteúdo não confiável e menor privilégio para ferramentas.

Macetes para memorizar

Identidades

Sistema nasce e morre com o app; usuário existe antes e é reaproveitada.

Funções do Key Vault

Officer gerencia, User usa, Reader só vê o rótulo (metadados), Administrator abre tudo.

Códigos do Azure OpenAI

400 filtro, 401/403 permissão, 404 implantação, 429 cota.

Content Safety

Shields guarda a porta de entrada; material protegido e fundamentação vigiam a saída; filtro de conteúdo vigia os dois lados.

Plano de controle x dados

Leitor e Colaborador mexem no recurso; Data e User mexem no conteúdo.

Onde olhar

Log de atividades conta quem mexeu no Azure; Application Insights conta o que o app fez.

KQL

requests para falhas, exceptions por type, dependencies para latência externa, traces para mensagens de log.

Tabela de memorização: não confunda

Quando é...	...e quando é
Identidade atribuída pelo sistema Uma identidade exclusiva que nasce e morre com o recurso, sem necessidade de permissão prévia.	**Identidade atribuída pelo usuário** As permissões precisam existir antes de criar o app, ou a identidade será reutilizada entre recursos.
Key Vault Secrets Officer Um pipeline precisa criar e atualizar segredos, sem permissão sobre chaves e certificados.	**Key Vault Secrets User** O aplicativo só precisa ler segredos.
Permissões delegadas O aplicativo age em nome de um usuário conectado.	**Permissões de aplicativo** Um serviço em segundo plano chama o Microsoft Graph sem usuário conectado; conceda só o necessário.
Prompt Shields Detectar jailbreak nas entradas e instruções maliciosas escondidas em documentos antes de chamar o modelo.	**Filtro de conteúdo** Bloquear ódio, sexual, violência e automutilação acima de um nível de severidade em entradas e saídas.
Detecção de fundamentação A resposta não se apoia nos documentos fornecidos como contexto (alucinação em RAG e resumos).	**Detecção de material protegido** A resposta reproduz textos protegidos, como letras de música, ou código.
Métricas no Azure Monitor e logs de diagnóstico Acompanhar tokens e latência da implantação ou auditar chamadas ao recurso de IA com KQL.	**Log de atividades** Ver operações de gerenciamento do Azure, como criação e exclusão de recursos.
Alerta de métrica com limite Avisar quando um valor definido for ultrapassado, como muitos 5xx ou reinícios frequentes.	**Detecção inteligente de anomalias de falha** Avisar quando a taxa de falhas subir de forma anormal sem definir limite fixo.

Números que caem

Item	Valor
Código do Azure OpenAI para entrada barrada pelo filtro	**400 (content_filter)**
Códigos para falta de permissão	**401 ou 403**
Código para implantação inexistente	**404**
Código para cota de tokens esgotada	**429**

Linguagens da distribuição OpenTelemetry do Azure Monitor	**4: .NET, Java, Node.js e Python**
Zona DNS privada do Azure OpenAI	**privatelink.openai.azure.com**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Escolher identidade atribuída pelo sistema quando a permissão no Key Vault precisa existir antes do app.
Como não cair: Ela só existe depois que o app é criado. Use identidade atribuída pelo usuário.

⚠ **Alerta de prova:** Usar políticas de acesso do cofre em um Key Vault novo.
Como não cair: É o modelo legado. O recomendado é Azure RBAC.

⚠ **Alerta de prova:** Procurar uma chave de acesso do cofre para o SecretClient.
Como não cair: O Key Vault não usa chave de acesso. Autentique com DefaultAzureCredential e uma função de dados.

⚠ **Alerta de prova:** Dar Colaborador na conta de armazenamento ou no recurso de IA para ler dados ou chamar o modelo.
Como não cair: Colaborador gerencia o recurso, pode obter chaves e dá mais acesso que o necessário. Use a função de dados no menor escopo.

⚠ **Alerta de prova:** Atribuir Leitor no grupo de recursos para ler o Cosmos DB ou chamar o Azure OpenAI.
Como não cair: Leitor atua no plano de controle e gera 403 mesmo com token válido. Atribua a função de dados do serviço.

⚠ **Alerta de prova:** Guardar as chaves no Key Vault ou rotacioná-las toda semana para permitir só Entra ID.
Como não cair: As chaves continuariam válidas. Desative a autenticação local (disableLocalAuth).

⚠ **Alerta de prova:** Regenerar as duas chaves de uma vez para rotacionar.
Como não cair: Isso interrompe os aplicativos. Alterne entre as duas chaves.

⚠ **Alerta de prova:** Criar uma identidade gerenciada local quando o DefaultAzureCredential falha no computador do desenvolvedor.
Como não cair: Identidades gerenciadas só existem no Azure. Entre com az login.

⚠ **Alerta de prova:** Usar fluxo implícito com token na URL em uma SPA.
Como não cair: Não é mais recomendado. Use código de autorização com PKCE e MSAL.

⚠ **Alerta de prova:** Guardar o segredo de cliente nos segredos do GitHub para implantar sem segredo no repositório.
Como não cair: Ainda é um segredo. Use federação de identidade com OIDC.

⚠ **Alerta de prova:** Configurar a telemetria com a chave de instrumentação sozinha.
Como não cair: A ingestão só por chave foi descontinuada. Use a cadeia de conexão.

⚠ **Alerta de prova:** Procurar erros do app, telemetria ou chamadas de modelo no log de atividades da assinatura.
Como não cair: Ele mostra operações de gerenciamento do Azure. Use logs de console, Application Insights, métricas ou logs de diagnóstico.

⚠ **Alerta de prova:** Repetir de imediato em laço quando o Azure OpenAI retorna 429.
Como não cair: Isso agrava o limite de taxa. Repita com espera e ajuste a cota.

⚠ **Alerta de prova:** Usar Prompt Shields para detectar texto protegido na resposta.
Como não cair: Prompt Shields detecta ataques nas entradas. Texto protegido na saída é detecção de material protegido.

Como cai na prova

📝 **Como cai:** Acessar um recurso sem chave e com menor privilégio.
Como resolver: Identifique o recurso e a operação (ler, gravar, inferir). Escolha identidade gerenciada mais a função de plano de dados exata no menor escopo (contêiner, cofre, recurso). Descarte Proprietário, Colaborador, Leitor e qualquer alternativa que guarde chave.

📝 **Como cai:** Código de erro ou sintoma em uma chamada ao Azure OpenAI.
Como resolver: Mapeie o código: 400 content_filter é filtro; 401/403 é permissão (com token válido, falta função de dados); 404 é implantação; 429 é cota, resolvida com repetição com espera e ajuste de cota.

📝 **Como cai:** Onde ver ou como monitorar algo.
Como resolver: Separe três camadas: operações do Azure no log de atividades; telemetria do app no Application Insights (dependências, operation_Id, Live Metrics, Depurador de Instantâneos, KQL); uso e auditoria do recurso nas métricas do Azure Monitor e nos logs de diagnóstico.

📝 **Como cai:** Ameaça ou risco específico de IA.
Como resolver: Jailbreak ou instrução escondida em documento: Prompt Shields. Conteúdo nocivo: filtro de conteúdo e mensagem de sistema. Texto protegido: material protegido. Resposta sem fonte: fundamentação. Termos sigilosos: lista de bloqueio. Ameaças em produção com alertas ao SOC: Defender for Cloud para IA.

📝 **Como cai:** Gestão de segredos e configuração.
Como resolver: Segredo em produção: Key Vault com identidade gerenciada. Segredo via configuração: referência do Key Vault no App Configuration. Recarregar sem reiniciar: chave sentinela. Liberar parcialmente: sinalizador de recurso. Local em .NET: user-secrets.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
permissões precisam existir antes da criação do app	**Identidade atribuída pelo usuário**
app com várias identidades autentica com a errada	**Informar o client ID**
mesmo código com identidade gerenciada no Azure e credenciais do desenvolvedor localmente	**DefaultAzureCredential**
exigir login do Entra ID sem alterar o código	**Autenticação interna (Easy Auth)**
aplicativo de página única obtendo tokens	**Código de autorização com PKCE e MSAL**
serviço em segundo plano chama o Microsoft Graph sem usuário	**Permissões de aplicativo**
GitHub Actions implanta sem segredo de cliente	**Federação de identidade com OIDC**
403 com token do Entra ID válido	**Falta função de dados à identidade**
erro 400 com código content_filter	**Entrada barrada pelo filtro de conteúdo**
seguir uma requisição por vários microsserviços	**Filtrar pelo operation_Id**
ver o valor das variáveis quando a exceção ocorre em produção	**Depurador de Instantâneos**
segredo rotacionado automaticamente perto de expirar	**Evento do Key Vault (SecretNearExpiry) com uma função**
recarregar configurações sem reiniciar quando uma chave muda	**Atualização com chave sentinela**
ativar recurso para parte dos usuários sem nova implantação	**Sinalizador de recurso por percentual**

🎯 **Se você lembrar de só uma coisa deste capítulo...**

- Leitor não permite inferência (403 com token válido); Colaborador e Proprietário excedem o menor privilégio. Para dados, use a função de plano de dados no menor escopo.
- O Key Vault não tem chave de acesso; RBAC é o modelo recomendado.
- Configure o Application Insights pela cadeia de conexão, não pela chave de instrumentação sozinha.
- Desativar a autenticação local é o que realmente impede o uso de chaves.

Essenciais para a prova (checklist)

☐ Sei quando usar identidade atribuída pelo sistema ou pelo usuário e como escolher pelo client ID.

☐ Sei a ordem e o uso do DefaultAzureCredential e por que az login resolve o erro local.

☐ Sei as funções RBAC do Key Vault e as operações de SecretClient e CryptographyClient.

☐ Sei resolver conflito de nome por exclusão temporária, limitação do cofre e rotação por evento.

☐ Sei escolher a função de plano de dados certa para blobs, Cosmos DB e Azure OpenAI.

☐ Sei aplicar PKCE em SPA, permissões delegadas x de aplicativo, Expor uma API, Easy Auth e OIDC.

☐ Sei usar App Configuration com referências do Key Vault, chave sentinela e sinalizadores de recurso.

☐ Sei o que detectam filtros de conteúdo, Prompt Shields, material protegido, fundamentação e listas de bloqueio.

☐ Sei instrumentar com OpenTelemetry e usar dependências, operation_Id, Live Metrics e Depurador de Instantâneos.

☐ Sei escrever as consultas KQL básicas sobre requests e exceptions.

Questões de revisão do capítulo

R4.1 · Proteger, monitorar e solucionar problemas de soluções do Azure

Novos recursos de IA da empresa não podem ser criados com acesso de rede público. O que usar para garantir isso?

A. Filtro de conteúdo nas implantações
B. Política que nega acesso público
C. Revisão manual de cada recurso
D. Alerta de custo na assinatura

R4.2 · Proteger, monitorar e solucionar problemas de soluções do Azure

Um app web no Container Apps deve exigir login do Microsoft Entra ID sem alterar o código do aplicativo. O que configurar?

A. Um NSG na sub-rede do ambiente do app
B. Autenticação interna do Entra
C. Uma regra de escala HTTP
D. Um volume do Azure Files

R4.3 · Proteger, monitorar e solucionar problemas de soluções do Azure

Um pipeline precisa criar e atualizar segredos no Key Vault, sem permissão sobre chaves e certificados. Qual função atribuir?

A. Key Vault Secrets Officer
B. Key Vault Administrator
C. Key Vault Reader
D. Key Vault Secrets User

R4.4 · Proteger, monitorar e solucionar problemas de soluções do Azure

Ao recriar um segredo com o mesmo nome de um segredo excluído há pouco, o Key Vault retorna conflito. O que fazer?

A. Criar um novo cofre na mesma região
B. Aumentar a cota do cofre
C. Trocar a chave de criptografia
D. Recuperar ou limpar o segredo

Respostas comentadas

R4.1. B) Política que nega acesso público

Por quê: Há definições internas para serviços de IA que exigem Link Privado e desativam o acesso público.

✗ A) Incorreta. Não trata rede.
✗ C) Incorreta. Não impede a criação.
✗ D) Incorreta. Não trata rede.

> **Dica de prova:** Garantir recursos de IA privados: Azure Policy.

R4.2. B) Autenticação interna do Entra

Por quê: A autenticação interna (Easy Auth) é oferecida pelo App Service, Functions e Container Apps.

✗ A) Incorreta. Não exige login.
✗ C) Incorreta. Não exige login.
✗ D) Incorreta. Não exige login.

> **Dica de prova:** Login sem mudar o código: autenticação interna.

R4.3. A) Key Vault Secrets Officer

Por quê: O modelo RBAC do Key Vault tem funções separadas para segredos, chaves e certificados.

✗ B) Incorreta. Dá acesso a todos os objetos.
✗ C) Incorreta. Só lê metadados.
✗ D) Incorreta. Só lê segredos.

> **Dica de prova:** Gerenciar só segredos: Secrets Officer.

R4.4. D) Recuperar ou limpar o segredo

Por quê: Com proteção contra limpeza, não é possível limpar antes do prazo; a solução é recuperar o segredo.

✗ A) Incorreta. Não resolve o nome no cofre atual.
✗ B) Incorreta. Não há cota envolvida.
✗ C) Incorreta. Não afeta o nome.

> **Dica de prova:** Conflito de nome no Key Vault: recuperar ou limpar.

DIA DA PROVA

Estratégia de prova

O AI-200 (Developing AI Cloud Solutions on Azure) substituiu o AZ-204, aposentado em 31/07/2026, e é uma prova de desenvolvedor: quase todo enunciado descreve um requisito curto e pede a configuração, função, política ou comando exato. As restrições do enunciado decidem tudo: sem chave, menor privilégio, sem alterar o código, sem nova implantação, sem gerenciar servidor ou cluster, em segundos, menor custo. As alternativas erradas costumam ser recursos reais do Azure que resolvem outro problema, como aumentar réplicas, trocar região, subir a temperatura ou olhar o log de atividades. A Microsoft favorece identidade gerenciada com RBAC no menor escopo, Key Vault para segredos e recursos gerenciados no lugar de código próprio. O formato tem múltipla escolha e múltipla resposta; nas de múltipla resposta, o enunciado diz quantas opções marcar, como 'Escolha DUAS'.

Como responder

✓ Com cerca de 50 questões em 100 minutos (a Microsoft não divulga o número exato, que fica entre 40 e 60), conto com uns 2 minutos por questão: leio primeiro a última frase para saber o que pedem e depois caço o requisito-chave no enunciado.

✓ Sublinho mentalmente as restrições (sem chave, menor privilégio, sem alterar o código, sem nova implantação, sem cluster) e elimino toda alternativa que viole qualquer uma delas.

✓ Descarto de saída as alternativas coringa que não resolvem nada: aumentar a temperatura, trocar a região, log de atividades, relatório de custos e recomendações do Advisor. 'Mais réplicas' só vale quando o requisito é disponibilidade em implantações ou evitar inicialização a frio; para memória, porta, DNS ou erro do app, descarto.

✓ Em 'Escolha DUAS', julgo cada alternativa sozinha como verdadeira ou falsa para o cenário; as duas certas costumam ser complementares, como sonda de prontidão mais réplicas mínimas ou Key Vault mais identidade gerenciada.

✓ Quando duas opções parecem certas, fico com a mais específica e gerenciada pela plataforma (política pronta, função de dados exata, recurso nativo) em vez de código próprio ou solução manual.

✓ Separo sempre plano de controle (Leitor, Colaborador, log de atividades) de plano de dados (funções Data/User, telemetria do app); essa distinção resolve boa parte das questões de segurança e monitoramento.

✓ Distribuo o estudo pelos pesos: soluções de IA com serviços de dados vale 25-30% e é o maior bloco; contêineres, conectar e consumir serviços e proteger/monitorar valem 20-25% cada.

✓ A aprovação exige 700 em uma escala de 1 a 1000; marco as dúvidas para revisar no fim e não deixo nenhuma sem resposta.

Erros que derrubam candidatos

✗ Escolher uma função de plano de controle para acessar dados ou chamar o modelo: Leitor resulta em 403; Colaborador e Proprietário dão mais acesso que o necessário e ferem o menor privilégio.

✗ Confundir sonda de prontidão com sonda de atividade, ou réplicas mínimas fixas com regra de escala.

✗ Achar que guardar chaves no Key Vault ou rotacioná-las impede o uso de chaves, quando o certo é desativar a autenticação local.

✗ Trocar o modelo de embeddings só nas consultas ou responder problemas de RAG com temperatura em vez de ajustar a recuperação.

✗ Misturar sessões do Service Bus com chave de partição do Event Hubs, ou usar a captura do Event Hubs como solução de ordem e posição.

✗ Procurar erros e telemetria do aplicativo no log de atividades da assinatura.

✗ Esquecer que replicação geográfica, ponto privado e retenção de manifestos do ACR exigem a camada Premium.

Na semana da prova

- ☐ Refazer as provas práticas e, em cada erro, reler por que cada alternativa errada não atende ao requisito, que é de onde vêm as pegadinhas.
- ☐ Revisar as tabelas de mapeamento rápido: código HTTP do Azure OpenAI, funções do Key Vault, padrões do Durable Functions, políticas do API Management e ferramentas do Content Safety.
- ☐ Reforçar o domínio de soluções de IA com serviços de dados (25-30%): RAG, Azure AI Search, vetores no Cosmos DB e parâmetros do modelo.
- ☐ Treinar os comandos da CLI do Container Apps e do ACR até associar cada um à sua função sem hesitar.

- [ ] Na véspera, revisar só os blocos de lembrar e os macetes de cada capítulo e descansar.

REVISÃO FINAL

Folha de cola

Leia na véspera e na manhã da prova: os gatilhos e as pegadinhas mais importantes de cada domínio.

1. Conectar e consumir serviços do Azure

‣ eventos discretos, por push, para vários assinantes com filtros → **Azure Event Grid**

‣ entrega confiável, fila de mensagens mortas e transações → **Azure Service Bus**

‣ eventos de um mesmo dispositivo em ordem no Event Hubs → **ID do dispositivo como chave de partição**

‣ mensagens de um mesmo pedido em ordem, um consumidor por vez → **Sessões com SessionId**

‣ colocar na fila agora e processar só daqui a 24 horas → **Mensagem agendada**

⚠ A mensagem é removida ao ser lida e se perde na falha. Use PeekLock e conclua após processar.

⚠ PeekLock bloqueia a mensagem. Para inspecionar sem remover nem bloquear, use peek.

⚠ A captura só grava arquivos em blobs. Ordem é chave de partição, leitores independentes são grupos de consumidores e retomar é EventProcessorClient com checkpoints.

2. Desenvolver soluções de IA com serviços de dados do Azure

‣ documentos internos que mudam toda semana, sem treinar o modelo → **RAG**

‣ combinar palavras-chave e similaridade vetorial e reordenar por relevância → **Busca híbrida com classificador semântico**

‣ mostrar o trecho que responde à pergunta, destacado → **Legendas e respostas semânticas**

‣ embeddings gerados durante a indexação de blobs, sem código próprio → **Vetorização integrada no indexador**

‣ a consulta envia texto e o próprio serviço gera o vetor → **Vetorizador no perfil**

⚠ Temperatura só muda a variação. Para consistência, ela deve ser baixa; para os demais problemas, procure a técnica específica.

⚠ Isso estoura o limite de contexto. Use RAG e envie só os trechos relevantes.

⚠ É caro e lento para dados que mudam. Conhecimento atual vem por RAG; ajuste fino é para estilo e formato.

3. Desenvolver soluções em contêineres no Azure

‣ descoberta de serviço, mTLS e repetições sem escrever esse código → **Dapr com a API de invocação de serviço**

‣ agente de mensagens trocável sem alterar o código → **Componente de pub/sub do Dapr**

‣ testar a nova revisão por endereço próprio antes do tráfego de produção → **Rótulo de revisão com URL dedicada**

‣ sem atraso de inicialização a frio na primeira chamada → **Réplicas mínimas iguais a 1**

‣ pipeline sem Docker instalado precisa gerar a imagem → **az acr build**

⚠ CPU não mede requisições simultâneas. Use a regra HTTP com concurrentRequests no valor pedido.

⚠ Mínimo fixo mantém as réplicas sempre ligadas, inclusive fora do horário. Escala por horário é o escalonador cron do KEDA.

⚠ A atividade reinicia réplicas travadas; quem segura o tráfego até a réplica ficar pronta é a sonda de prontidão.

4. Proteger, monitorar e solucionar problemas de soluções do Azure

‣ permissões precisam existir antes da criação do app → **Identidade atribuída pelo usuário**

‣ app com várias identidades autentica com a errada → **Informar o client ID**

‣ mesmo código com identidade gerenciada no Azure e credenciais do desenvolvedor localmente → **DefaultAzureCredential**

‣ exigir login do Entra ID sem alterar o código → **Autenticação interna (Easy Auth)**

‣ aplicativo de página única obtendo tokens → **Código de autorização com PKCE e MSAL**

⚠ Ela só existe depois que o app é criado. Use identidade atribuída pelo usuário.

⚠ É o modelo legado. O recomendado é Azure RBAC.

⚠ O Key Vault não usa chave de acesso. Autentique com DefaultAzureCredential e uma função de dados.

GABARITO

Respostas do diagnóstico

D1. D) Função EventGrid Data Sender → revise o capítulo 1

Por quê: O Event Grid aceita autenticação do Entra ID para publicar, e as chaves podem ser desativadas.

✗ A) Incorreta. Usa chave.
✗ B) Incorreta. Trata entrega, não publicação.
✗ C) Incorreta. Não é permitido.

> **Dica de prova:** Publicar sem chave: EventGrid Data Sender.

D2. B) PeekLock com a conclusão após processar → revise o capítulo 1

Por quê: No PeekLock, se o bloqueio expirar ou o consumidor abandonar a mensagem, ela volta à fila e pode ir para a DLQ após várias tentativas.

✗ A) Incorreta. As mensagens voltariam sem controle.
✗ C) Incorreta. A mensagem é removida ao ser lida.
✗ D) Incorreta. Só inspeciona, sem processar.

> **Dica de prova:** Sem perder mensagens: PeekLock.

D3. A) ID do dispositivo como chave de partição → revise o capítulo 1

Por quê: A ordem é garantida dentro de uma partição. A chave de partição mantém eventos relacionados juntos.

✗ B) Incorreta. Eventos do dispositivo ficariam em partições diferentes.
✗ C) Incorreta. A captura grava arquivos, sem ordem por dispositivo.
✗ D) Incorreta. Grupos de consumidores não definem a ordem.

> **Dica de prova:** Ordem por dispositivo: chave de partição.

D4. C) O limite de tokens de saída → revise o capítulo 2

Por quê: O motivo de término length indica que o limite de tokens foi atingido. O total deve caber na janela de contexto do modelo.

✗ A) Incorreta. Não muda o limite.
✗ B) Incorreta. Muda a variação, não o tamanho.
✗ D) Incorreta. Não muda o limite.

> **Dica de prova:** Resposta cortada: aumentar o limite de tokens de saída.

D5. D) A dimensão do campo vetorial → revise o capítulo 2

Por quê: Cada modelo de embeddings gera vetores de dimensão fixa, que devem coincidir com o campo do índice.

✗ A) Incorreta. O analisador vale para texto.
✗ B) Incorreta. A camada não define a dimensão.
✗ C) Incorreta. Não afeta a indexação.

> **Dica de prova:** Erro de dimensão: alinhar campo e modelo.

D6. C) Índice diskANN → revise o capítulo 2

Por quê: Os índices vetoriais do Cosmos DB são flat, quantizedFlat e diskANN, com diferentes equilíbrios de custo e precisão.

✗ A) Incorreta. Faz busca exata e é indicado para poucos vetores.
✗ B) Incorreta. Força varredura completa.
✗ D) Incorreta. Não é índice vetorial.

> **Dica de prova:** Muitos vetores no Cosmos DB: diskANN.

D7. D) Tabela ContainerAppConsoleLogs → revise o capítulo 3

Por quê: Os logs de sistema ficam em ContainerAppSystemLogs. Nomes com sufixo _CL aparecem em configurações antigas.

✗ A) Incorreta. Guarda imagens.
✗ B) Incorreta. Registra operações do Azure.
✗ C) Incorreta. Não é o destino padrão.

> **Dica de prova:** Logs de console no Log Analytics: ContainerAppConsoleLogs.

D8. D) Variáveis de ambiente seguras → revise o capítulo 3

Por quê: Variáveis seguras usam secureValue. Para segredos de longo prazo, prefira o Key Vault com identidade gerenciada.

✗ A) Incorreta. Não guardam segredos.
✗ B) Incorreta. O valor aparece nas propriedades.
✗ C) Incorreta. Não guarda segredos.

> **Dica de prova:** Segredo no ACI: variável de ambiente segura.

D9. B) az containerapp update → revise o capítulo 3

Por quê: Cada atualização de modelo, como a imagem, gera uma nova revisão do app.

✗ A) Incorreta. Autentica o Docker local.
✗ C) Incorreta. Compila a imagem, sem implantar.
✗ D) Incorreta. Cria um ambiente.

> **Dica de prova:** Nova imagem no app: az containerapp update.

D10. C) Identidade atribuída pelo usuário → revise o capítulo 4

Por quê: Identidades atribuídas pelo usuário permitem pré-autorização e reutilização entre recursos.

✗ A) Incorreta. Não deve ser usada por aplicativos.
✗ B) Incorreta. Só existe depois que o app é criado.
✗ D) Incorreta. O Key Vault não usa chave de acesso.

> **Dica de prova:** Permissão antes do app existir: identidade atribuída pelo usuário.

D11. D) Storage Blob Data Contributor no contêiner → revise o capítulo 4

Por quê: Funções de dados podem ser atribuídas no escopo do contêiner para limitar o acesso.

✗ A) Incorreta. Só lê, e em toda a conta.
✗ B) Incorreta. Gerencia a conta e pode obter chaves.
✗ C) Incorreta. Concede muito mais.

> **Dica de prova:** Ler e gravar blobs com menor privilégio: Data Contributor no contêiner.

D12. C) Vincular a zona DNS privada à VNet → revise o capítulo 4

Por quê: Zonas privatelink devem estar vinculadas à VNet do ambiente ou ao DNS personalizado usado por ela.

✗ A) Incorreta. Não afeta DNS.
✗ B) Incorreta. É para entrada do app.
✗ D) Incorreta. Não afeta DNS.

> **Dica de prova:** Nome privado não resolve: zona DNS privada.

Agora, treine no formato real

Faça as 6 provas completas do curso AI-200: 300 questões inéditas, tempo do exame, explicação de cada alternativa e link da documentação para revisar.

Ver as 6 provas completas e o cupom do mês →

Até o dia da prova, uma revisão por dia no seu feed

Instagram @strecht.simulados

Um post de prova por dia: questão comentada, pegadinhas e o "não confunda" de cada certificação.

Seguir no Instagram →

YouTube Strecht Simulados

Shorts diários com pegadinhas de prova e, aos sábados, um simulado comentado.

Inscrever-se no canal →

Material independente de preparação, sem afiliação ou endosso de Microsoft. As questões são inéditas e não reproduzem a prova real.